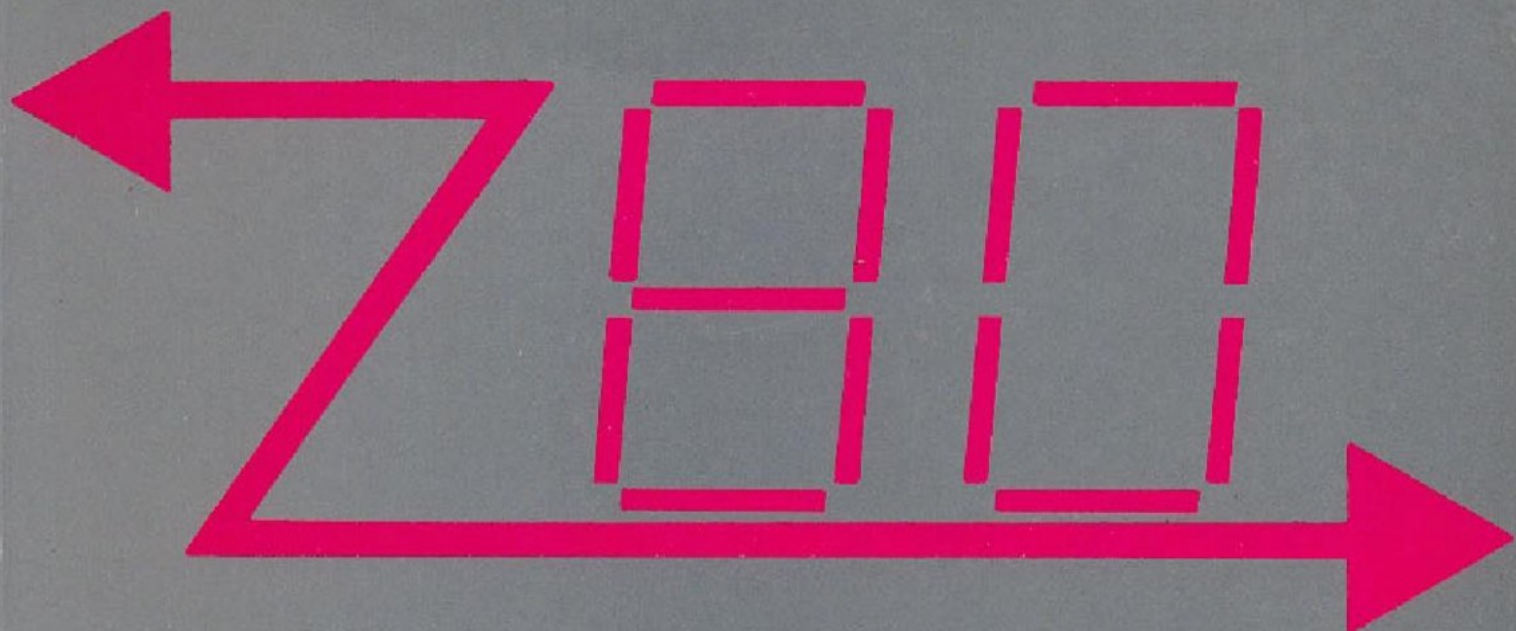


ENG^o LUIZ BENEDITO CYPRIANO



SOFTWARE

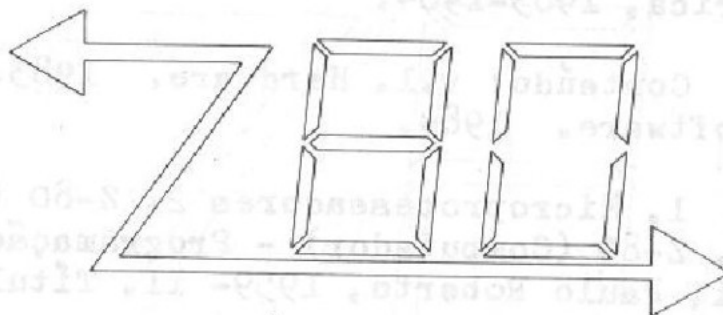
3.^a EDIÇÃO

VOL. II

ÉRICA

PH

MICROPROCESSADOR



SOFTWARE

VOL. 2

LIVRARIA OSORIO

RUA BARAO DO SERRO AZUL, 191

CENTRO, CURITIBA-PR.

FONE: (041) 224-3904/222-0852

livros@liuronet.com.br

www.liuronet.com.br

COMP

CYPRIANO, LUIZ BENEDITO

Z 80-SOFTWARE-VOL. II-3A. EDICAO

EDITORIA ERICA

156102

PH



CIP-Brasil. Catalogação-na-Publicação
Câmara Brasileira do Livro, SP

C523m
v.1-2

Cipriano, Luís Benedito, 1957-
Microprocessador Z-80 / Luiz Benedito Cypri-
ano, Paulo Roberto Cardinali. -- São Paulo :
Érica, 1983-1984.

Conteúdo: v.1. Hardware. 1983. -- v.2.
Software. 1984.

1. Microprocessadores 2. Z-80 (Computador)
3. Z-80 (Computador) - Programação I. Cardina-
li, Paulo Roberto, 1959- II. Título.

84-0796

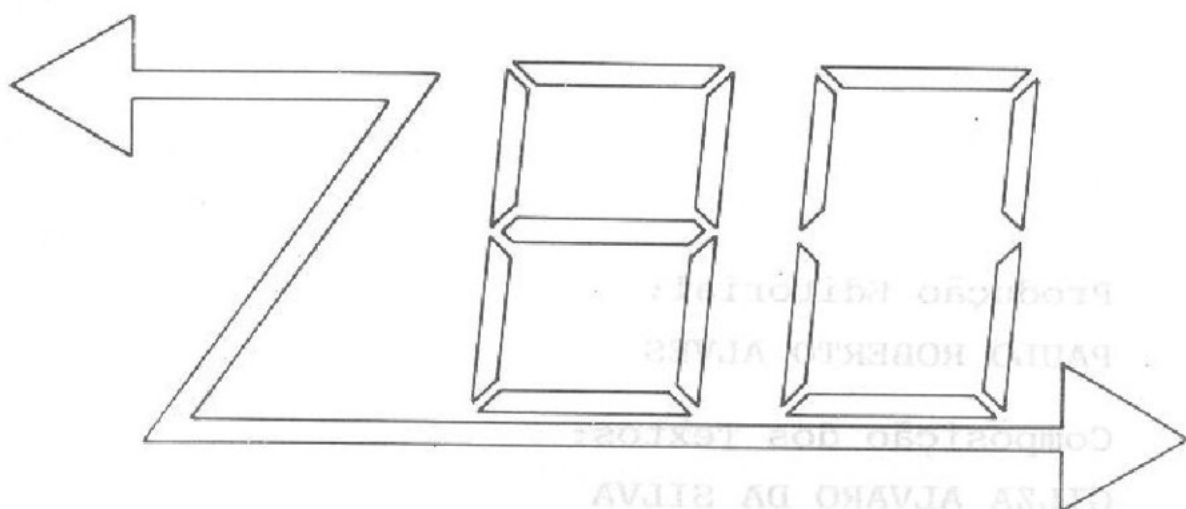
17. CDD-621.381958
18. -621.38195835
17. -651.8
18. -001.6425

Índices para catálogo sistemático:

1. Microprocessadores Z-80 : Engenharia eletrô-
nica 621.381958 (17.) 621.38195835 (18.)
2. Software : Z-80 : Computadores digitais :
Processamento de dados 651.8 (17.)
001.6425 (18.)
3. Z-80 : Computadores digitais : Engenharia
eletrônica 621.381958 (17.)
621.38195835 (18.)
4. Z-80 : Computadores digitais : Programação :
Processamento de dados 651.8 (17.)
001.6425 (18.)

ENG.º LUIZ BENEDITO CYPRIANO

MICROPROCESSADOR



SOFTWARE

VOL. 2

4.^a EDIÇÃO

LIVROS ÉRICA EDITORA LTDA.

Rua Jahn, 504 - Tatuapé - São Paulo
Fone: 294-8885 - C.G.C. 50.152.538/0001-38
Caixa Postal 15.817

Nenhuma parte desta publicação poderá ser reproduzida, guardada pelo sistema "retrieval" ou transmitida de qualquer modo ou por qualquer outro meio, seja este eletrônico, mecânico, de fotocópia, de gravação, ou outros, sem prévia autorização por escrito desta EDITORA

Produção Editorial:

PAULO ROBERTO ALVES

Composição dos Textos:

GILZA ALVARO DA SILVA

Desenhos:

DERNIVAL CORDEIRO DA SILVA

Revisão:

GUARACIABA MICHELETTI

LIVROS ÉRICA EDITORA LTDA.

Rua Jarinu, 594 - Tatuapé - São Paulo

Fone: 294-8686 - C.G.C. 50.268.838/0001-39

Caixa Postal 15.617

PREFÁCIO

Este livro é o resultado de uma longa e árdua luta, que se tornou necessária para a realização de um trabalho que, apesar de modesto, tem a intenção de contribuir para a cultura da nossa pátria.

Por isso, não hesitei em recorrer ao auxílio de alguns amigos, cujas sugestões e críticas foram muito úteis para a elaboração do presente trabalho.

Devo também agradecer ao Sr. Dr. João de Deus, por ter se interessado pelo trabalho e por ter me oferecido a oportunidade de publicá-lo.

Por fim, quero agradecer ao Sr. Dr. João de Deus, por ter me oferecido a oportunidade de publicá-lo.

Este livro é o resultado de uma longa e árdua luta, que se tornou necessária para a realização de um trabalho que, apesar de modesto, tem a intenção de contribuir para a cultura da nossa pátria.

Por isso, não hesitei em recorrer ao auxílio de alguns amigos, cujas sugestões e críticas foram muito úteis para a elaboração do presente trabalho.

Devo também agradecer ao Sr. Dr. João de Deus, por ter se interessado pelo trabalho e por ter me oferecido a oportunidade de publicá-lo.

Por fim, quero agradecer ao Sr. Dr. João de Deus, por ter me oferecido a oportunidade de publicá-lo.

Este livro é o resultado de uma longa e árdua luta, que se tornou necessária para a realização de um trabalho que, apesar de modesto, tem a intenção de contribuir para a cultura da nossa pátria.

A meus Pais.

estados, sobretudo no Rio de Janeiro, onde a situação não é muito diferente da dos outros estados. A situação geral é bastante ruim, com muitas dificuldades de ordem econômica e social.

Estado do Rio de Janeiro

Estado do Rio de Janeiro

Estado do Rio de Janeiro

Estado do Rio de Janeiro

Estado do Rio de Janeiro

Estado do Rio de Janeiro

Estado do Rio de Janeiro

Estado do Rio de Janeiro

Estado do Rio de Janeiro

LIVROS ERICA EDITORA LTDA

Rua Juruá, 594 - Tatuapé - São Paulo

Fone: 754 8886 - C.G.C. 50.768.325/0001-34

Caixa Postal 75.617

PREFÁCIO

CAPÍTULO A microcomputação é uma área que, para entendê-la, torna-se necessário, primeiramente, conhecer uma série de dados que caracterizam algumas informações específicas deste campo.

Para tornar o aprendizado mais suave e significativo, os volumes I e II apresentam uma divisão didática de grande utilidade, para aqueles que se interessam pelo campo da microcomputação.

No volume I, tem-se a oportunidade de entrar em contato com os passos necessários na confecção de circuitos, para diferentes tipos de microcomputadores.

O presente volume apresenta várias maneiras de se programar os diferentes circuitos, citados anteriormente, sendo que cada maneira carrega consigo certas propriedades características, para aquilo que se deseja da programação.

Tentando seguir o mesmo estilo do volume I, este se desenvolve através de uma linguagem de fácil compreensão e acessível a todos os níveis de conhecimento dentro desta área, e, ao mesmo tempo, apresenta um conteúdo altamente significativo.

Aqueles que, por motivos diversos, se interessam por este campo, têm reunidos nestes dois volumes todas as informações necessárias para compreendê-lo e nele trabalhar.

É constante a preocupação de tentar expor tais informações de maneira gradativa, isto é, uma informação torna-se pré-requisito na aquisição da próxima.

Assim sendo, quando chegar ao término da confecção e programação dos circuitos, o leitor terá garantido domínio suficiente para prosseguir num estudo mais profundo e detalhado da microcomputação.

Na esperança de ter atingido todos estes objetivos que, de certa forma, inspiraram este trabalho apresentado em dois volumes, desejo que estes sejam de grande valia para o leitor, seja ele iniciante ou dominador deste campo.

5.2 - PLANO DE Trabalho

5.3 - PLANO DE Trabalho

5.4 - PLANO DE Trabalho

5.5 - PLANOS A e B

O Autor

A microcomputação é uma nova forma de entender a realidade, tornando-se necessária, principalmente, com o advento das novas técnicas de dados que caracterizam algumas informações específicas deste campo. Para tornar a aprendizagem mais fácil e significativa, os volumes I e II apresentam uma série de gráficos de grande utilidade, para aqueles que se interessam pela área da microcomputação.

No volume I, tem-se a oportunidade de entrar em contato com os aspectos gerais da microcomputação, para os diferentes tipos de microcomputadores.

O presente volume apresenta várias maneiras de se trabalhar com diferentes tipos de dados, permitindo, assim, que cada maneira tenha suas próprias características, facilitando a aprendizagem.

Quando se utiliza o mesmo estilo de volume I, este se desenvolve através de uma linguagem de fácil compreensão e acesso a todos os níveis de conhecimento dentro desta área, e, ao mesmo tempo, apresenta um conteúdo altamente significativo. Apesar disso, por motivos diversos, se interessam por este campo, têm recebido dentro dos volumes todas as informações necessárias para compreenderem o que se trata.

É importante a preocupação de ter um texto claro e objetivo, de maneira gradativa, isto é, uma aprendizagem que se desenvolve na sequência da próxima.

Assim sendo, quando chegar ao término da coleção e programação dos cursos, o leitor terá garantido domínio suficiente para prosseguir sua leitura mais profunda e detalhada da microcomputação.

Na esperança de que esta obra seja útil para os objetivos que se tem em vista, apresentamos este trabalho, apresentando os dois volumes, de modo que cada um deles seja útil para o leitor, seja ele iniciante ou dominador deste campo.

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO.....	13
1.1 - SOFTWARE e HARDWARE.....	13
1.2 - Computador.....	14
1.3 - Microprocessador.....	14
CAPÍTULO 2 - TIPOS DE INFORMAÇÃO GUARDADAS NA MEMÓRIA....	17
2.1 - Código de Operação.....	17
2.2 - Registrador.....	20
CAPÍTULO 3 - INSTRUÇÃO.....	23
3.1 - Tipos de Instruções.....	23
3.2 - Número de BYTE de uma Instrução.....	28
CAPÍTULO 4 - ENDEREÇAMENTO.....	31
4.1 - Endereçamento Imediato.....	31
4.2 - Endereçamento por Registrador.....	32
4.3 - Endereçamento Imediato Estendido.....	33
4.4 - Endereçamento Indireto por Registrador.....	34
4.5 - Endereçamento Relativo.....	36
4.6 - Endereçamento Paginado Zero.....	37
4.7 - Endereçamento Implícito.....	38
4.8 - Endereçamento Indexado.....	38
4.9 - Endereçamento Estendido.....	40
4.10 - Endereçamento de BIT.....	40
CAPÍTULO 5 - FLAG.....	43
5.1 - FLAG de CARRY.....	43
5.2 - FLAG de Zero.....	44
5.3 - FLAG de Sinal.....	44
5.4 - FLAG de Paridade e OVERFLOW.....	45
5.5 - FLAGS H e N.....	47

CAPÍTULO 6 - INTERRUPÇÃO.....	49
6.1 - Introdução.....	49
6.2 - Interrupção Não Mascarada.....	49
6.3 - Interrupção Mascarada.....	51
CAPÍTULO 7 - REPERTÓRIO DAS INSTRUÇÕES DO MICROPROCESSADOR Z-80.....	59
CAPÍTULO 8 - LINGUAGEM.....	201
8.1 - Linguagem de Máquina.....	203
8.2 - Linguagem Assembly.....	205
8.3 - Pseudo-Instrução.....	208
CAPÍTULO 9 - PRÁTICA DE PROGRAMAÇÃO.....	215
9.1 - Introdução.....	215
9.2 - Método de Programação.....	215
9.3 - Aplicação Prática de Algumas Instruções.....	215
9.4 - Programa Simples.....	221
9.5 - LOOP.....	222
9.6 - Rotinas.....	230
9.7 - Algoritmo.....	233
9.8 - Tabelas.....	242
9.9 - Dispositivo I/O.....	250
CAPÍTULO 10 - CONCLUSÃO.....	263
APÊNDICE A - TABELA DE CONVERSÃO DOS SISTEMAS DE NUMERAÇÃO.....	265
APÊNDICE B - TABELA DE EXPONENCIAIS.....	273
APÊNDICE C - TABELA DE EQUIVALÊNCIA HEXADECIMAL E ASCII....	275
APÊNDICE D - ESTRUTURA DOS REGISTRADORES DO Z-80.....	279

APÊNDICE E -	INTERRUPÇÕES DO Z-80.....	281
APÊNDICE F -	CÓDIGO DE OPERAÇÃO POR ORDEM NUMÉRICA.....	283
APÊNDICE G -	CÓDIGO DE OPERAÇÃO POR ORDEM ALFABÉTICA.....	303
APÊNDICE H -	CORRESPONDÊNCIA DOS MNEMÔNICOS DO MICROPROCES SADOR 8080 com o Z-80.....	319

1.1 SOFTWARE E HARDWARE

O sistema de processamento de dados é composto por um hardware e um software. O hardware é a parte física do sistema, enquanto o software é a parte lógica. O hardware é dividido em dois tipos: o sistema de processamento e o sistema de armazenamento. O sistema de processamento é responsável por executar as operações aritméticas e lógicas, enquanto o sistema de armazenamento é responsável por armazenar os dados e os programas. O software é dividido em dois tipos: o sistema operacional e os aplicativos. O sistema operacional é responsável por gerenciar os recursos do hardware e fornecer uma interface para os aplicativos. Os aplicativos são programas que realizam tarefas específicas para o usuário.

O sistema de processamento de dados é composto por um hardware e um software. O hardware é a parte física do sistema, enquanto o software é a parte lógica. O hardware é dividido em dois tipos: o sistema de processamento e o sistema de armazenamento. O sistema de processamento é responsável por executar as operações aritméticas e lógicas, enquanto o sistema de armazenamento é responsável por armazenar os dados e os programas. O software é dividido em dois tipos: o sistema operacional e os aplicativos. O sistema operacional é responsável por gerenciar os recursos do hardware e fornecer uma interface para os aplicativos. Os aplicativos são programas que realizam tarefas específicas para o usuário.

CAPÍTULO 1 - INTRODUÇÃO

Esta nova filosofia que está sendo introduzida na vida de cada pessoa é a era da eletrônica digital.

Pode-se notar que, a cada momento que se passa, estamos sendo envolvidos por microprocessadores e computadores, tanto na vida profissional como na cotidiana.

Esta espantosa filosofia a que nos referimos, está sendo empregada para facilitar, divertir e até mesmo salvar vidas. É o que se pode chamar de "ERA DE AUTOMAÇÃO".

No volume "I" foram desenvolvidos circuitos, estruturas e arquiteturas de vários sistemas como: CPU, PIO, CTC, memórias e outros.

Neste volume será desenvolvida toda a parte de programação e aplicativos do microprocessador Z-80; e, também, serão apresentadas várias rotinas, tais como: divisão, multiplicação, aritmética de MULT-BYTE e outras.

Nosso principal intuito é desenvolver as técnicas de programação do microprocessador Z-80, e também desenvolver a linguagem Assembler.

1.1 - SOFTWARE e HARDWARE

O SOFTWARE representa a parte maleável que é a programação, podendo ser desenvolvida e modificada, sem alterar os circuitos e a estrutura da máquina. Já o HARDWARE corresponde ao circuito, parte que dificilmente é modificada, pois uma vez determinada a estrutura de uma máquina, para que ela possa realizar uma determinada função, não há necessidade de modificá-la.

Atualmente maiores esforços voltam-se ao desenvolvimento do SOFTWARE, pelo fato de apresentar vantagens inexistentes no desenvolvimento do HARDWARE. No primeiro, tem-se a disponibilidade de que uma mesma máquina executa várias funções, dependendo do SOFTWARE executado. No segundo, além da máquina apresentar-se com uma estrutura pré-estabelecida, qualquer modificação

que esta venha a sofrer, requer equipamentos sofisticados e de alto custo para a sua elaboração.

1.2 - Computador

Desde os primórdios de nossa era, o homem sempre se preocupou em desenvolver máquinas, fossem elas automáticas ou manuais. Tal preocupação tinha como objetivo minimizar esforços, o desgaste do agente humano no trabalho e aumentar a produção.

As primeiras máquinas eram formadas por engrenagens e correias e podem ser chamadas de computadores mecânicos.

O computador, de maneira geral, é um dispositivo que pode controlar e manipular eventos existentes na natureza.

Os computadores eletrônicos são subdivididos em duas partes: analógicos e digitais. Os analógicos têm a capacidade de lidar não apenas com dois níveis distintos de tensão, mas, com uma faixa de tensão, por exemplo, entre zero e quinze volts. O principal circuito utilizado pelo computador analógico é operacional. Um exemplo prático seria a divisão de dois números, 10 e 5. Para isso, na entrada do dividendo, injeta-se uma tensão que corresponde ao número "dez" e, na entrada do divisor, injeta-se ao número "cinco". Na saída, tem-se uma tensão que corresponde ao número "dois".

Os computadores digitais têm como sua principal característica manusear informações em código binário, onde o nível lógico "um" é representado pela tensão de cinco volts, na maioria dos circuitos, e o nível lógico "zero" é representado pela tensão de zero volts. Outras características do computador digital são a capacidade de efetuar operações aritméticas, controlar dispositivos externos e até efetuar decisões.

1.3 - Microprocessador

O microprocessador é um dispositivo digital baseado numa pastilha (CHIP). Esta pastilha, por si só, tem a

capacidade de controlar e manusear dispositivos externos como: memórias, porta de Entrada/Saída e outros circuitos. Um circuito digital que utilize microprocessador, tem 80% de sua potencialidade total ligada no mesmo, pois, é ele quem coordena toda a movimentação de dados, circuitos auxiliares e outros circuitos externos que, porventura, possam existir.

As pastilhas (CHIP) contêm elementos de semicondutores como transistores e diodos, são construídas a partir do material silício que com tratamento especial são implementados diversos semicondutores. Este ramo que desenvolve circuitos e dispositivos digitais, como microprocessadores, memórias e outros, é chamado de microeletrônica.

2.1 - Codificação Operacional

Para os computadores pessoais IBM, 8080 e outros, o primeiro BYTE de um endereço indica o tipo de operação.

De uma maneira geral, o código de operação representa

CAPÍTULO 2 - TIPOS DE INFORMAÇÃO GUARDADAS NA MEMÓRIA.

No volume I, pode-se ver a implementação de memórias. Neste capítulo serão estudados os tipos de informações guardadas numa memória.

Um BYTE, para uma memória, representa um conjunto de oito BITS, sendo cada BIT representado por um sinal: nível lógico "um" (em torno de "5" volts) ou nível lógico "zero" (em torno de "0" volts).

O conjunto de BYTES que formam uma memória é composto por informações, que terão significados diferentes para o microprocessador Z-80.

Todas estas informações são formadas por um BYTE (8 BITS) e o microprocessador Z-80 trabalha com seis tipos de informações, são elas:

CÓDIGOS DE OPERAÇÕES

BYTES DE DADOS

CÓDIGOS DE DISPOSITIVOS

BYTES DE ENDEREÇAMENTO MAIS SIGNIFICATIVOS (HI)

BYTES DE ENDEREÇAMENTO MENOS SIGNIFICATIVOS (LO)

BYTES DE DIRECIONAMENTO

Dependendo do tipo de instrução, os BYTES anteriormente descritos combinar-se-ão, formando instruções de um, dois, três e quatro BYTES, onde o microcomputador terá acesso e executará instrução por instrução.

Um conjunto de informações, às vezes, não representa uma instrução, mas apenas um conjunto de BYTES que pode ser usado como uma tabela. Por exemplo, o conjunto corresponde ao código ASCII.

2.1 - Código de Operação

Para os microprocessadores Z-80, 8080 e outros, o primeiro BYTE de uma instrução será o código de operação.

De uma maneira geral, o código de operação representa

a ação que ele deverá executar tais como: buscar outro código de operação para completar o primeiro; buscar o BYTE de endereçamento; BYTE de dado; BYTE dispositivo; BYTE de direcionamento. Em outras ocasiões, porém, este código de operação já é suficiente para que o microprocessador processe a respectiva instrução.

Cada código de operação representa uma instrução que o microprocessador irá executar, como operações lógicas e aritméticas, operações com STACK POINTER, transferência de dados, operação com dispositivo de entrada e saída e operação de controle.

O microprocessador Z-80 contém, no seu grupo de instruções, um, dois, ou três códigos de operações, isto é, para se definir uma instrução poderão existir até três códigos de operações.

Para resumir, o código de operação de uma instrução é o BYTE, que não pode ser modificado, pois, caso isto ocorra, a instrução será modificada, significando assim, outro tipo de instrução.

2.1.1.1 - BYTE de Dados

O BYTE de dados é um conjunto de 8 BITS, que será utilizado pelo microprocessador Z-80, principalmente nas instruções lógicas e aritméticas. Pode ser usado, também, para carregar um dado num determinado registrador; para carregar ou iniciar um determinado dispositivo de entrada e saída; para carregar certos dados em memória, servindo apenas como arquivo, que poderá ser utilizado no transcorrer do programa. O lugar da memória onde são guardados os dados para serem usados no transcorrer do programa, pode ser chamado de BUFFER.

A memória também poderá conter certos BYTES que são usados como tabelas, como por exemplo, para codificar decimal em ASCII, decimal em hexadecimal, binários e outros.

2.1.1.2 - BYTE Dispositivo

BYTE de dispositivo significa, para o microprocessador Z-80, o endereço que identifica o dispositivo de entrada e saída onde serão escritas ou gravadas certas informações, como por exemplo, um "Porto" (PIO), descrito no volume I.

O microprocessador Z-80 pode acessar apenas 2^8 ou 256 dispositivos de entrada e saída.

2.1.3 - BYTE de Endereçamento HI e LO

As linhas de endereçamento à memória do microprocessador Z-80 são formadas por 16 linhas (A0 a A15), sendo divididas em duas partes: linhas altas (A15 a A8), simbolizadas por (HI) e linhas baixas (A7 a A0), simbolizadas por (LO).

A necessidade desta divisão advém da memória possuir apenas 8 BITS e as linhas de endereço serem formadas por 16 BITS. Para que um determinado endereço, que simboliza uma posição de memória, possa ser carregado numa memória, necessita-se desta divisão.

Em todas as instruções que lidam com endereçamento, em primeiro lugar se carrega o BYTE menos significativo (LO) e, depois, o mais significativo (HI).

2.1.4 - BYTE de Direcionamento

O BYTE de Direcionamento é usado no grupo de instruções que envolvem endereçamento indexado.

Este BYTE aparece junto com o registrador de índice, formado por dois BYTES que, por sua vez, formam o endereçamento indexado.

O endereçamento indexado é uma técnica usada pelo microprocessador Z-80, para definir a posição real onde se localiza a informação desejada. Tem como função movimentar dados de uma certa posição para outra, com o objetivo de localizar dados em tabelas e outras técnicas, que serão mostradas no transcorrer deste capítulo.

No microprocessador 8080, esta técnica de endereça

mento indexado não é usado pelo fato deste não possuir regis
trador de índice.

2.2 - Registrador

O registrador é um circuito lógico que tem a finalidad
de de reter a curto prazo um conjunto de BITS. No microprocess
sador Z-80 existem dois grupos de registradores: os de 8 BITS (1
BYTE) e os de 16 BITS (2 BYTES). Cada um desses grupos são subd
divididos, tendo características diferentes para cada tipo de
aplicação.

De uma maneira geral, um registrador é utilizado como
se fosse uma memória para guardar informação a ser utilizada,
de acordo com a necessidade do programa. A grande vantagem de
um registrador sobre uma posição de memória é a sua versatilid
dade de movimentação de BITS. Por exemplo, em um registrador
formado por 8 BITS, de aspecto geral, tem-se a capacidade de
deslocar-se tanto para a direita como para a esquerda, entrada
e saída paralela, setar (colocar um BIT a nível lógico "um") e
resetar, (colocar um BIT a nível lógico "zero"). Na figura
2.1, pode-se ver a representação de um registrador geral, em
bora nem todos os registradores apresentem esta versatilidade.
Cada registrador apresenta, dependendo de sua aplicação, algum
mas dessas características.

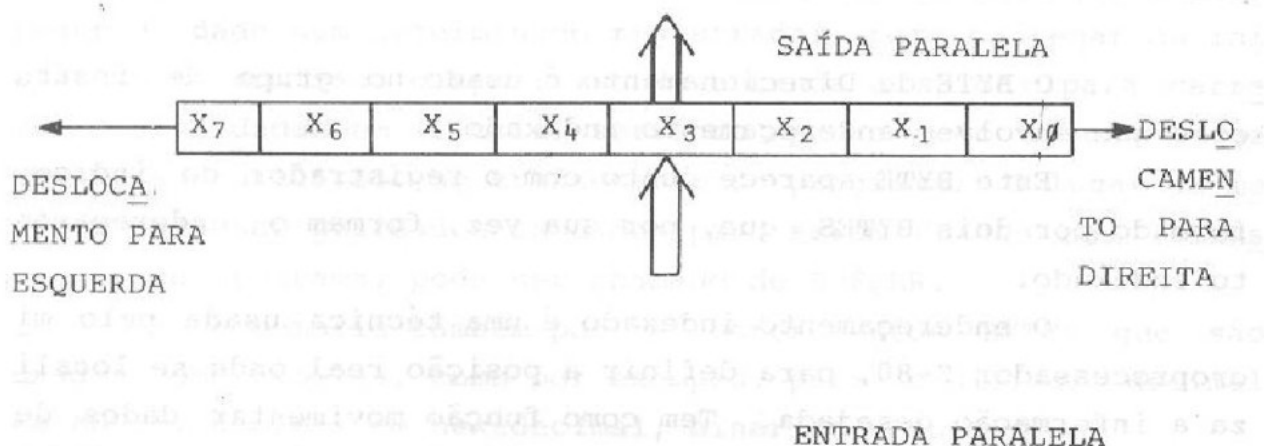


Fig. 2.1 - Registrador Geral.

O registrador de uso geral é formado por 3 pares de registradores, sendo cada um formado por 8 BITS (um BYTE).

GRUPO UM

Registrador B

Registrador C

Registrador D

Registrador E

Registrador H

Registrador L

GRUPO DOIS

Registrador B'

Registrador C'

Registrador D'

Registrador E'

Registrador H'

Registrador L'

Acumulador formado por 8 BITS (A e A')

Indicadores de FLAG formado por 8 BITS (F e F')

Indicadores de STACK POINTER formado por 16 BITS (SP)

Registradores contador de programa formado por 16 BITS (PC)

Registradores de índice formado por 16 BITS ((IX) e (IY))

Registradores de endereçamento de página de interrupção formado por 8 BITS (I)

Registrador de REFRESH de memória formado por 8 BITS (R)

No volume I, os registradores foram explicados com maiores detalhes.

Tipos de Instruções

O 280 possui 158 tipos diferentes de instruções.

Essa gama de instruções é representada por um código binário.

Este código binário é dividido em partes.

As instruções podem ser divididas em:

1. Troca de Informação

2. Lógica

3. Deslocamento

4. Deslocamento

5. Deslocamento

CAPÍTULO 3 - INSTRUÇÃO

Instrução defini-se como sendo "uma ordem dada para executar uma certa tarefa", por exemplo, ligar e desligar uma certa máquina.

Num microprocessador, uma instrução representa ler ou escrever numa posição de memória, fazer operações aritméticas e lógicas, como soma e subtração, e muitas outras operações que serão descritas no transcorrer deste capítulo.

Existem duas maneiras de representar-se uma instrução, utilizando-se um microprocessador. Uma delas é a linguagem de máquina ou linguagem objeto, na qual a instrução é representada pelo código binário ou hexadecimal. Por exemplo, o código (7DH) em hexadecimal ou (01111101)_B em binário, que representa carregar o registrador "A" com o conteúdo do registrador "D". A outra maneira de representar-se uma instrução é utilizar símbolos mnemônicos, onde a instrução é representada por letras como (LD C,H), que representam carregar o registrador "C" com o conteúdo do registrador "H".

Pelo número e tipo de instrução, pode-se determinar quando um microprocessador é superior ou não a outro. Por exemplo, comparando-se o microprocessador Z-80 com o 8080 tem-se que o número de instruções do Z-80 é superior ao do 8080.

3.1 - Tipos de Instruções

O Z-80 possui 158 tipos diferentes de instruções, sendo que cada uma delas é representada por um código mnemônico e pelo seu correspondente código binário ou hexadecimal.

Estas instruções podem ser divididas da seguinte maneira:

Carga e Troca de Informações

Aritméticas e Lógica

Transferência de Blocos

Rotação e Deslocamento

Manipulação de BITS

JMP, CALL, RETORNO

Entrada e Saída

Controle Interno

3.1.1 - Carga e Troca de Informações

Consiste em manusear dados entre a CPU e a memória e vice-versa, e também entre registradores internos, como pode-se ver nas figuras 3.1 e 3.2.



Fig. 3.1 - Transferência de Dados entre CPU e Memória.

CPU DADOS (2 BYTES) ou (1BYTE)

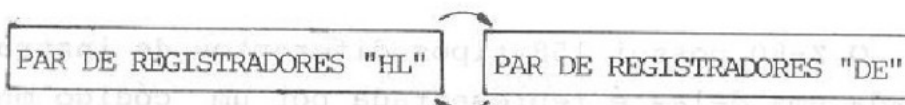


Fig. 3.2 - Transferência de Dados entre os Registradores Internos.

3.1.2 - Operações Aritméticas e Lógicas

Operações aritméticas são utilizadas para realizarem soma e subtração entre registrador e memória. As operações ló

gicas são utilizadas para realizarem funções lógicas, ou podem ser usadas para setar um BIT de um registrador (colocá-lo a nível lógico "um"), ou resetá-lo (colocá-lo a nível lógico "zero"); pela figura 3.3, pode-se acompanhar o procedimento.

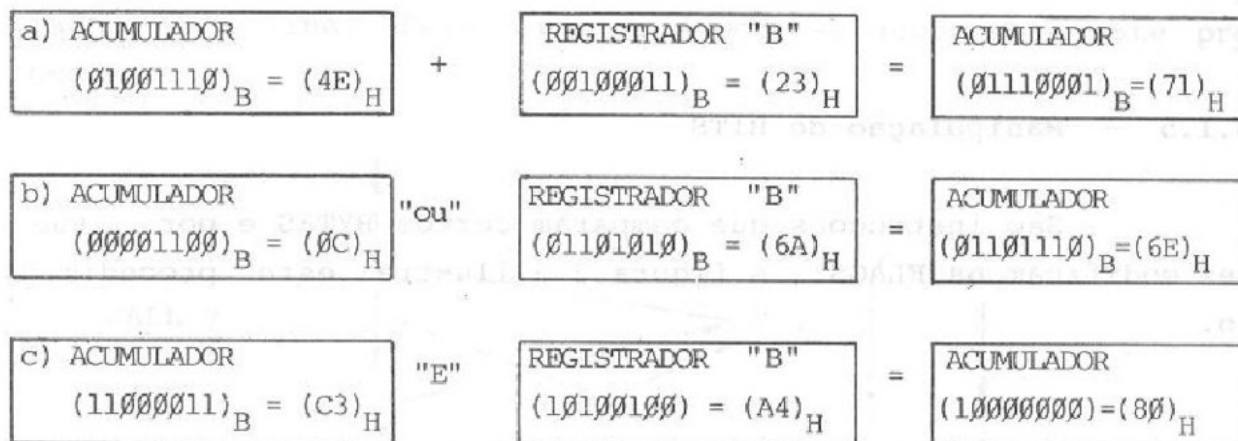


Fig. 3.3 - Exemplo de Operação Aritmética (a) e Lógica (b e c).

3.1.3 - Transferência de Blocos

O microprocessador Z-80 apresenta um grupo de instruções de grande utilidade para transferência de blocos.

Um bloco é um conjunto de BYTES que se localiza numa determinada posição da memória que, por vezes, nos interessa copiar ou simplesmente transferir de uma dada posição para outra. Para tal, lança-se mão das instruções reservadas para este fim. Apenas a título de comparação, os microprocessadores 8080 não possuem instruções de transferência de blocos, sendo necessário o desenvolvimento de uma rotina (um conjunto de instruções) para esse fim.

3.1.4 - Rotação e Deslocamento

As instruções de Rotação e Deslocamento são utilizadas para deslocar uma informação (BITS) para direita ou esquerda, e para outras aplicações, que serão mostradas nos capítulos seguintes; na figura 3.4, pode-se observar uma das aplicações.

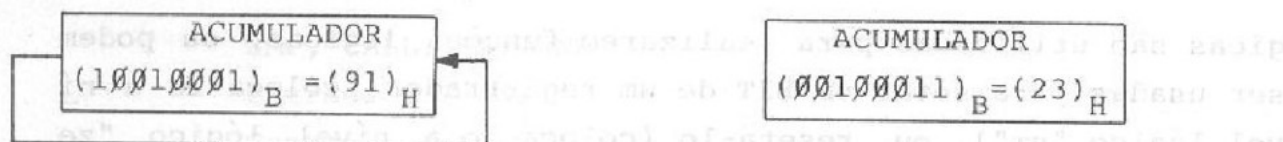


Fig. 3.4 - Rotação de um BIT para a Esquerda.

3.1.5 - Manipulação de BITS

São instruções que comparam certos BYTES e por sua vez modificam os FLAGS. A figura 3.5 ilustra este procedimento.

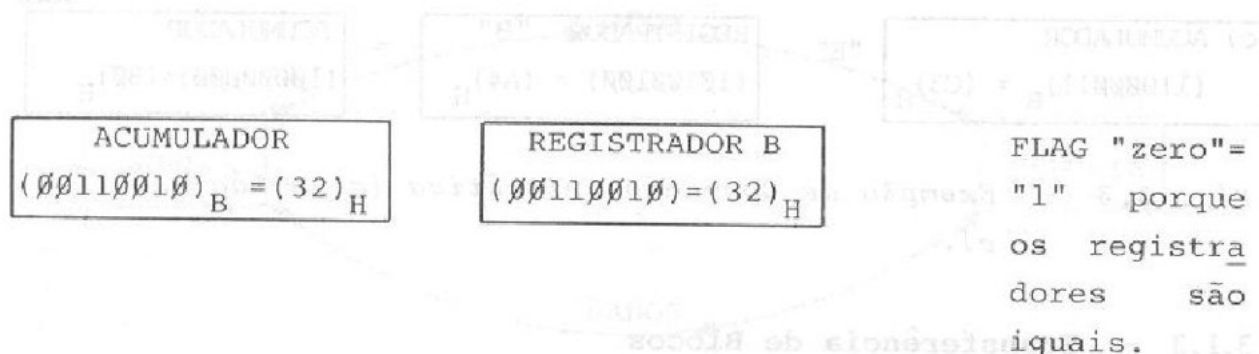


Fig. 3.5 - Manipulação de BITS.

3.1.6 - Salto, CALL e Retornos

As instruções de CALL, Retorno e JMP são usadas para mudar o contador de programa (PC) e seguir num outro endereço pré-determinado.

Os Saltos (JMP) são utilizados para mudar a sequência de um determinado programa.

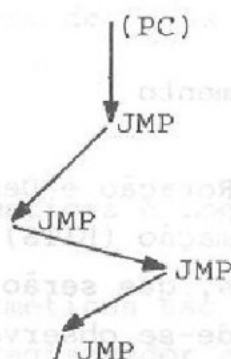


Fig. 3.6 - Saltos.

Utiliza-se CALL quando existe um certo grupo de instruções que se repetem várias vezes no mesmo programa. Este grupo pode ser denominado de uma subrotina, que deverá sempre terminar com uma instrução de retorno. Esta instrução modifica o contador de programa, para voltar na sequência de onde foi chamada a subrotina. Pela figura 3.7, pode-se acompanhar este procedimento.

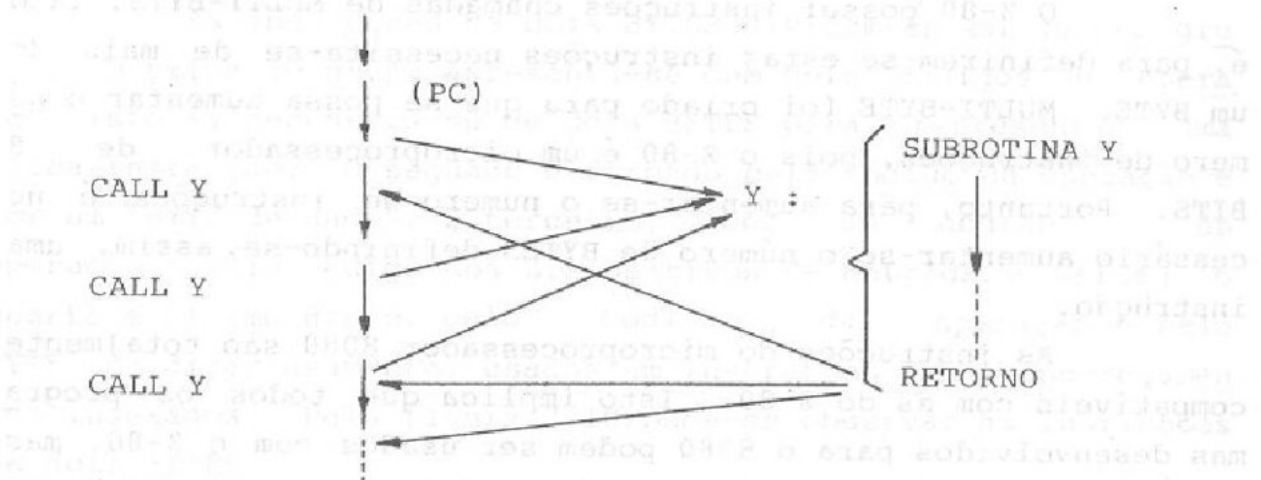


Fig. 3.7 - CALL e Retorno.

3.1.7 - Entrada e Saída

Uma instrução de entrada ou saída possibilita a movimentação de dados para dispositivos externos, como impressora, terminal de vídeo e outros. Na figura 3.8, tem-se um exemplo de transferência de informação para uma impressora.

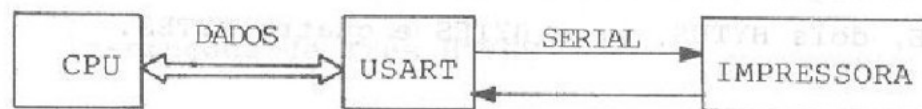


Fig. 3.8 Dispositivo de Entrada e Saída.

3.1.8 - Controle Interno

O microprocessador Z-80 apresenta um grupo de instruções que são utilizadas para controle interno.

Este controle é usado para liberação de interrupção

mascarada; (EI e DI), para designar o modo de interrupção mascarada, (IM0, IM1, IM2) e para parada geral da CPU (HALT).

3.2 - Número de BYTE de uma Instrução

O microprocessador Z-80 possui instruções que utilizam de "um" a "quatro" BYTES.

O Z-80 possui instruções chamadas de MULTI-BYTE, isto é, para definirem-se estas instruções necessita-se de mais de um BYTE. MULTI-BYTE foi criado para que se possa aumentar o número de instruções, pois o Z-80 é um microprocessador de 8 BITS. Portanto, para aumentar-se o número de instruções é necessário aumentar-se o número de BYTES definindo-se, assim, uma instrução.

As instruções do microprocessador 8080 são totalmente compatíveis com as do Z-80. Isto implica que todos os programas desenvolvidos para o 8080 podem ser usados com o Z-80, mas a recíproca não é verdadeira. A única diferença encontrada entre estes microprocessadores está na representação de seus mnemônicos, pois a linguagem de máquina permanece a mesma.

O microprocessador Z-80 possui em seu grupo de instruções até quatro BYTES. Estes tipos de instruções de quatro BYTES não existem no grupo de instruções do microprocessador 8080 constando apenas instruções de um, dois e três BYTES e apenas um código de operação. Já o Z-80, apresenta instruções com até três códigos de operações.

A seguir, será representada a formação das instruções de um BYTE, dois BYTES, três BYTES e quatro BYTES.

3.2.1 - Instruções de um BYTE

As instruções de um BYTE necessitam apenas do código de operação e não há necessidade de nenhuma informação auxiliar. Pela figura 3.9, tem-se um exemplo de instrução de um código de operação.

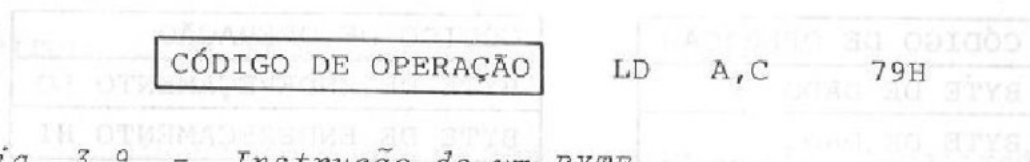


Fig. 3.9 - Instrução de um BYTE.

3.2.2 - Instruções de Dois BYTES

As instruções de dois BYTES dividem-se em quatro grupos. O primeiro grupo apresenta-se com dois códigos de operação, isto é, necessita-se de dois BYTES para representar uma única instrução. O segundo é formado pelo código de operação e por um "BYTE de dado"; o terceiro, por um código de operação e pelo código dos dispositivos de entrada e saída; o quarto e último grupo, pelo código de operação e pelo BYTE de direcionamento, usados em instruções para endereçamentos indexados. Pela figura 3.10, pode-se observar as instruções de dois BYTES.

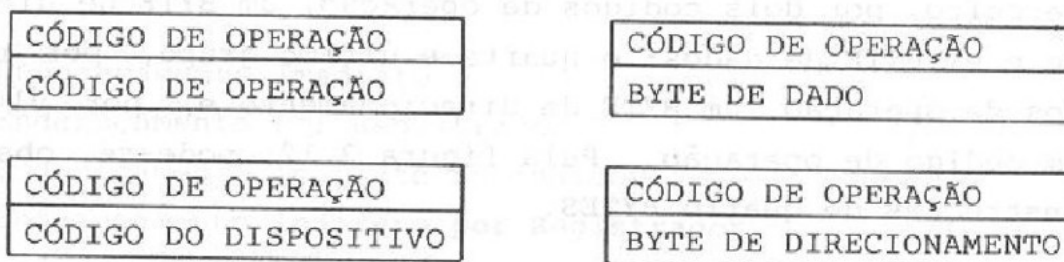


Fig. 3.10 - Instrução de dois BYTES.

3.2.3. - Instruções de Três BYTES

As instruções de três BYTES, dividem-se em três grupos. O primeiro é formado pelo código de operação e por dois BYTES de dados; o segundo, pelo código de operação e por dois BYTES de endereçamento, sendo o primeiro menos significativo (LO) e o segundo (HI); e o terceiro e último é usado em instruções de endereçamento indexado, sendo formado por dois códigos de operação e um BYTE de direcionamento. Pela figura 3.11, pode-se observar as instruções de três BYTES.

CÓDIGO DE OPERAÇÃO
BYTE DE DADO
BYTE DE DADO

CÓDIGO DE OPERAÇÃO
BYTE DE ENDEREÇAMENTO LO
BYTE DE ENDEREÇAMENTO HI

CÓDIGO DE OPERAÇÃO
CÓDIGO DE OPERAÇÃO
BYTE DE DIRECIONAMENTO

Fig. 3.11 - Instruções de Três BYTES.

3.2.4 - Instruções de Quatro BYTES

As instruções de quatro BYTES reúnem-se em quatro grupos. O primeiro é formado por dois códigos de operação e dois BYTES de dados; o segundo, por dois BYTES de endereçamento, sendo o primeiro menos significativo (LO) que o segundo (HI); o terceiro, por dois códigos de operação, um BYTE de direcionamento e um BYTE de dados; o quarto e último grupo, por dois códigos de operação, um BYTE de direcionamento e, por último, por um código de operação. Pela figura 3.12, pode-se observar as instruções de quatro BYTES.

CÓDIGO DE OPERAÇÃO
CÓDIGO DE OPERAÇÃO
BYTE DE DADOS
BYTE DE DADOS

CÓDIGO DE OPERAÇÃO
CÓDIGO DE OPERAÇÃO
BYTE DE ENDEREÇAMENTO LO
BYTE DE ENDEREÇAMENTO HI

CÓDIGO DE OPERAÇÃO
CÓDIGO DE OPERAÇÃO
BYTE DE DIRECIONAMENTO
BYTE DE DADOS

CÓDIGO DE OPERAÇÃO
CÓDIGO DE OPERAÇÃO
BYTE DE DIRECIONAMENTO
CÓDIGO DE OPERAÇÃO

Fig. 3.12 - Instrução de Quatro BYTES.

CAPÍTULO 4 - ENDEREÇAMENTO

O endereço de um dado (informação), localizado numa certa posição da memória ou um dispositivo de entrada e saída, nada mais é do que o lugar físico onde se localiza este dado. Para o microprocessador, endereçamento consiste em colocar-se no ADDRESS BUS (via de endereços) o respectivo endereço de onde se localiza o dado.

O microprocessador Z-80 apresenta dez maneiras diferentes de endereçamento, mostrando grande vantagem sobre o 8080 pela flexibilidade de realizar o endereçamento.

Como se pode notar, existe um limite máximo para realizar o endereçamento. No caso do microprocessador Z-80, este possui 16 BITS de ADDRESS BUS, podendo endereçar até 65536 (64K) posições de memória. Em caso de dispositivo de entrada e saída, este tem a capacidade de endereçar 256 posições.

A seguir estão relacionados os dez modos de endereçamento oferecidos pelo microprocessador Z-80.

Endereçamento Imediato

Endereçamento Por Registrador

Endereçamento Imediato Estendido

Endereçamento Indireto por Registrador

Endereçamento Relativo

Endereçamento Modificado Paginado Zero

Endereçamento Implícito

Endereçamento Indexado

Endereçamento Estendido

Endereçamento De BIT

4.1 - Endereçamento Imediato

O endereçamento imediato é formado pelo código de operação e um segundo BYTE, que contém informação desejada a ser manipulada.

O exemplo que se segue utiliza o endereçamento imediato:

LD E,2FH

Esta instrução representa carregar o registrador "E" com o dado que o segue no caso é "(2FH)". Assim sendo, após a execução desta instrução, o registrador "E" terá um valor igual a "(2FH)".

4.2 - Endereçamento por Registrador

O endereçamento por registrador é formado apenas pelo código de operação, pois o movimento de dados ocorre só internamente. Em outras palavras, ocorrerão trocas de informações apenas entre registradores.

O código de operação é formado da seguinte maneira: dois BITS (01) que representam endereçamento por registradores e dois grupos de 3 BITS "R" e "R'", que representam quais registradores estão sendo envolvidos. Na figura 4.1, a seguir, está representado o código de operação de um endereçamento por registrador.

LD R,R'	01 + R + + R' +	REGISTRADOR R, R'	
			B 000
LD B,C	(01000001B) → (41H)		C 001
(41H) Código de Operação da Instrução LD B,C			D 010
LD H,E	(0110011B) → (63H)		E 011
(63H) Código de Operação da Instrução LD H,E			H 100
			L 101
			A 111

"R" Representa o destino

"R'" Representa o destinatário

Fig. 4.1 - Endereçamento por Registrador.

Por exemplo, se o registrador "B" contiver (5DH) e o registrador "E" (0A7H), quando o microprocessador executar a instrução LD B,E; o registrador "B", após a execução, passará a ter (0A7H) e o registrador "E" ficará inalterado. Pela figura 4.2, pode-se ver a movimentação de dados.

REGISTRADOR "B" 5DH

REGISTRADOR "E" A7H

LD B,E

REGISTRADOR "B" A7H

REGISTRADOR "E" A7H

Fig. 4.2 - Movimentação de Dados.

4.3 - Endereçamento Imediato Estendido

O endereçamento imediato estendido é formado pelo código de operação e por dois BYTES subseqüentes, onde o primeiro BYTE é carregado na parte menos significativa do registrador (LOW-ORDER) e, o segundo, na parte mais significativa do registrador (HIGH-ORDER).

A finalidade deste tipo de endereçamento é carregar pares de registradores ou registrador com 16 BITS, "BC", "DE", "HL", "SP". Pela figura 4.3, pode-se ver o formato da instrução.

BYTE 1	Código de Operação
BYTE 2	BYTE menos significativo (LO)
BYTE 3	BYTE mais significativo (HI)

Fig. 4.3 - Formato de Endereçamento Imediato Estendido.

A figura 4.4 mostra um exemplo de uma instrução de endereçamento imediato estendido.

Linguagem de máquina trabalha diretamente com códigos binários ou hexadecimais. No capítulo VIII será estudada com maior detalhe.

LD DE,7A83H

LINGUAGEM DE MÁQUINA

(00010001B) ou 11H

(10000011B) ou 83H

(01111100B) ou 7AH

Fig. 4.4 - Endereçamento Imediato Estendido.

Após o microprocessador executar esta instrução, o registrador "E" será carregado com o BYTE menos significativo (que no exemplo será (83H)) e o registrador "D" será carregado com o BYTE mais significativo, tendo assim um valor igual a (7AH). Para as demais instruções de endereçamento imediato estendido, segue-se o mesmo raciocínio acima descrito.

4.4 - Endereçamento Indireto por Registrador

O endereçamento indireto por registrador utiliza pares de registradores como "BC", "DE", "HL" e "SP", para indicar o endereço da memória onde reside a informação (dado). Este tipo de endereçamento requer apenas o código de operação. Na figura 4.5, tem-se um exemplo de endereçamento indireto por registrador. O par de registradores estará sempre envolvido por parênteses, representando uma posição da memória.

LD C,(HL)

LD (HL),D

Fig. 4.5 - Endereçamento Indireto por Registrador.

Na montagem dos mnemônicos do microprocessador 2-80 sempre que houver referências a conteúdo de registrador ou algum outro dispositivo, este será envolvido por parênteses. No exemplo acima, a instrução "LD C,(HL)" representa que o registrador "C" será carregado pelo conteúdo endereçado pelo par de registradores "HL" que representam uma posição de memória.

A instrução "LD (HL),D" representa que a posição de memória endereçada pelo par de registradores (HL) será carregada com o valor do conteúdo do registrador "D". Na figura 4.6, pode-se ver o fluxo dos dados.

LD C, (HL)

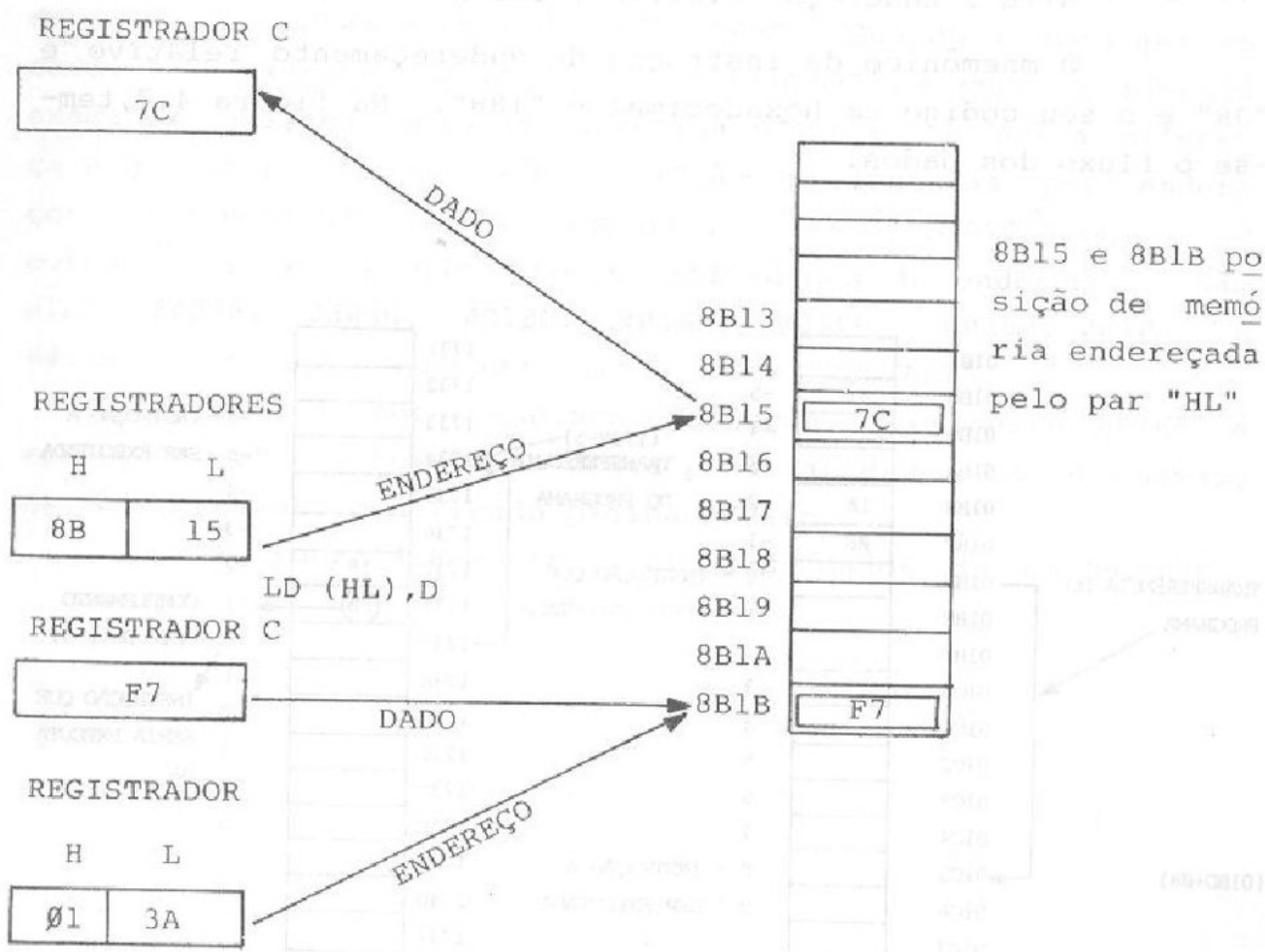


Fig. 4.6 - Fluxo de Dados.

4.5 - Endereçamento Relativo

O endereçamento relativo é formado por dois BYTES. O primeiro BYTE representa o código de operação e, o segundo, o endereço relativo.

Este tipo de endereçamento é aplicado a saltos em de terminados pontos do programa e poderá ocorrer tanto em endereços crescentes como decrescentes.

O salto poderá ocorrer apenas numa faixa de endereço, sendo esta de (+127) a (-128), representado em complemento dois.

A figura 4.7 mostra o formato do endereçamento relativo.

BYTE 1 Código de Operação

BYTE 2 Endereço Relativo (-128 a + 127)

O mnemônico da instrução de endereçamento relativo é "JR" e o seu código em hexadecimal é "18H". Na figura 4.7, tem-se o fluxo dos dados.



Fig. 4.7 - Fluxo de Dados de Endereçamento Relativo.

A grande vantagem de usar-se endereçamento relativo é na relocação de programas. Este tipo de endereçamento não lida com o endereço onde se localiza fisicamente a instrução que se rá executada, mas com a diferença de onde está o programa e pa ra onde irá a instrução a ser executada. Esta diferença será a mesma em qualquer lugar onde se localize o programa. Outra van tagem deste endereçamento é na economia de um BYTE, pois instru ções de saltos ocupam três BYTES, sendo que um BYTE corresponde ao código da instrução e os dois subseqüentes ao endereço de on de deverá prosseguir o programa.

4.6 - Endereçamento Paginado Zero

O endereçamento modificado paginado zero é formado por um único BYTE, que é o código de operação. Este tipo de en dereçamento também é chamado de "RESTART". Quando o microproces sador Z-80 executa uma instrução de "RESTART", é como se tivesse executado uma instrução de subrotina (CALL). A única diferen ça é que as instruções de "RESTART" são apresentadas por endere ços pré-determinados. Existem apenas oito "RESTARTS" ou, em outras palavras, apenas oito possibilidades de endereços. São eles: 0000H, 0008H, 0010H, 0018H, 0020H, 0028H, 0030H e 0038H, sendo uma para cada instrução de "RESTART".

Pode-se notar que, nos endereços, o primeiro "BYTE" é sempre "zero" e é por causa desta peculiaridade que é chamado de endereçamento modificado paginado zero.

A figura 4.8 mostra o fluxo de dados de um "RESTART".

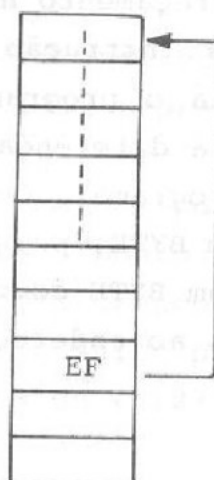
Ex: RST 28H

0028

AB7C

AB7D

AB7E



PROGRAMA COUNTER (PC)

AB7D
0028

PILHA ENDEREÇADA PELO
STACK POINTER (SP)

SP-2

SP-1

7E
AB

Fig. 4.8 - Fluxo de Dado de um "RESTART".

A grande vantagem deste tipo de endereçamento é quando em certos programas existem uma ou mais subrotinas que se repetem inúmeras vezes. É interessante usar as instruções de "RESTART", isto porque instruções para realizarem subrotinas (CALL) utilizam três BYTES, enquanto que as instruções de "RESTART" utilizam apenas um BYTE. Num programa qualquer tem-se uma subrotina que é utilizada dez vezes, utilizando-se "RESTART", tem-se uma economia de vinte BYTES.

4.7 - Endereçamento Implícito

O endereçamento implícito é aquele que ocorre automaticamente na execução de certas instruções.

O microprocessador Z-80 contém este tipo de endereçamento, sendo que as instruções envolvidas neste tipo são de aritmética e lógica.

A instrução "ADD A,H", por exemplo, irá efetuar a soma do registrador "A", com o registrador "H", e o resultado da soma será carregado automaticamente no registrador "A".

4.8 - Endereçamento Indexado

O endereçamento indexado é uma particularidade do microprocessador Z-80. Este tipo de endereçamento é possível, porque o Z-80 possui dois registradores (IX e IY) de 16 BITS cada um, chamados de registradores de índices e sendo utilizados em endereçamento indexado.

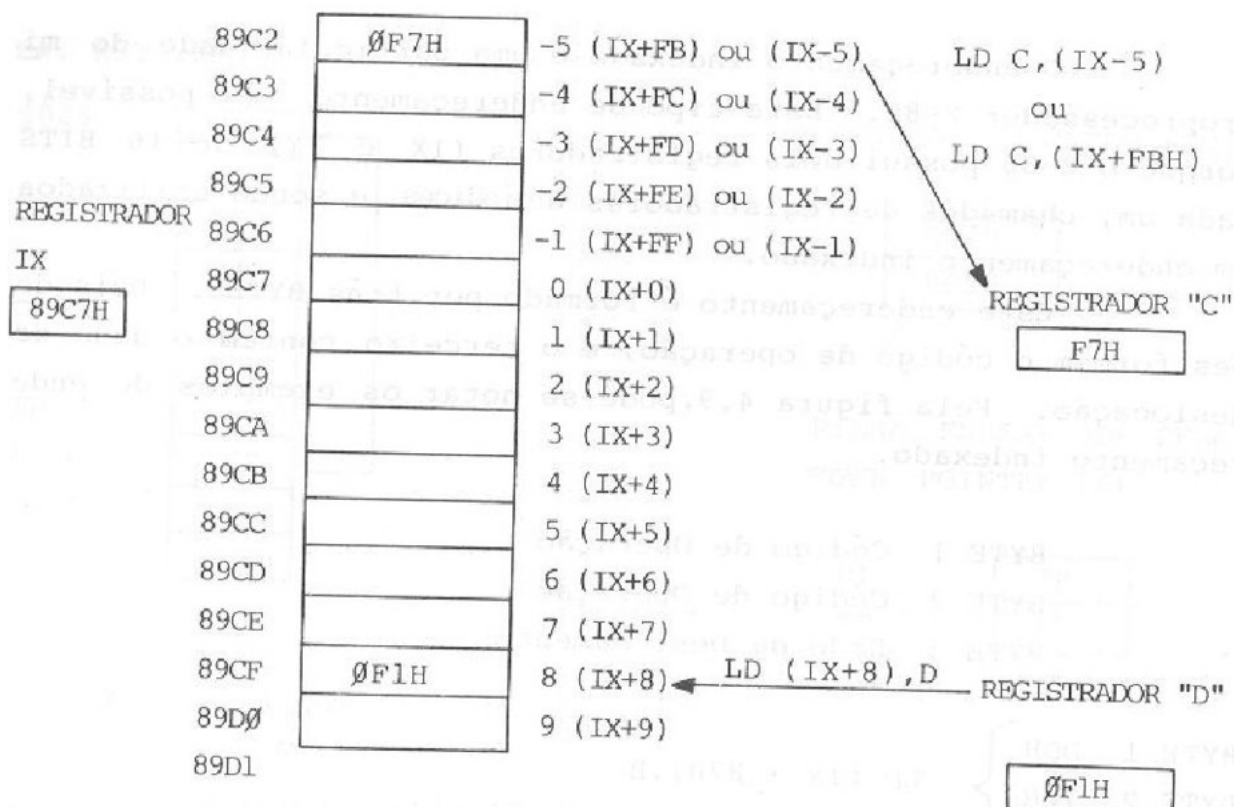
Este endereçamento é formado por três BYTES. Dois deles formam o código de operação, e o terceiro contém o dado de deslocamento. Pela figura 4.9, pode-se notar os exemplos de endereçamento indexado.

0		
1	BYTE 1	Código de Operação
2	BYTE 2	Código de Operação
3	BYTE 3	Dado de Deslocamento

BYTE 1	DDH	} LD (IX + 07H), B
BYTE 2	70H	
BYTE 3	07H	

Fig. 4.9 - Endereçamento Indexado.

Este tipo de endereçamento é muito similar ao endereçamento indireto por registrador. A única diferença que os distingue, é que o endereçamento indexado possui um BYTE, que será somado ou subtraído do registrador de índice, para se determinar o endereço final do BYTE a ser manipulado, enquanto o endereçamento indexado tem a capacidade de endereçar entre (-128 a +127) endereços. Pela figura 4.10, pode-se observar o fluxo de dados.



Obs: para o registrador de índice "IX" manter a mesma filosofia.

Fig. 4.10 - Fluxo de Dados do Endereçamento Indexado.

4.9 - Endereçamento Estendido

O endereçamento estendido é formado pelo código de operação e por dois BYTES subseqüentes, que indicam a posição de onde se localiza o respectivo dado na mesma memória. Este endereçamento é dividido em duas partes: na primeira, o BYTE menos significativo (LO) e, na segunda, o mais significativo (HI).

O que distingue este tipo de endereçamento dos demais é que o endereço está contido na própria instrução. Por exemplo, a instrução LD A, (7DF2H) significa que o dado contido no endereço físico (7DF2H) será carregado no registrador "A".

4.10 - Endereçamento de BIT

O endereçamento de BIT é formado por dois BYTES. O primeiro é o código de operação e o segundo apresenta-se dividi

do em três partes, sendo elas: o oitavo e sétimo BITS fixos (01); o sexto, quinto e o quarto BITS, que definem o BIT a ser manipulado; os BITS restantes que indicam o registrador. Na figura 4.11, pode-se acompanhar o formato da instrução.

BYTE 1 CÓDIGO DE OPERAÇÃO
 BYTE 2 01 + BBB + → RRR →

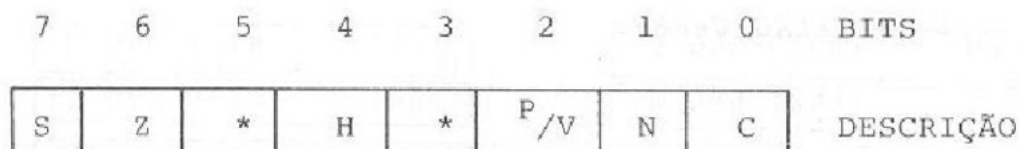
BIT	BBB	REGIS	RRR
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	M	110
7	111	A	111

Fig. 4.11 - Formato da Instrução de Endereçamento de BIT.

Este grupo de instruções têm a finalidade de testar, setar e resetar qualquer BIT de um dos registradores mencionados.

CAPÍTULO 5 - FLAG

Os registradores F e F' indicam, principalmente em operações lógicas e aritméticas, as condições que envolvem o resultado, como sinal, CARRY e outras condições que podem ser vistas na figura 5.1.

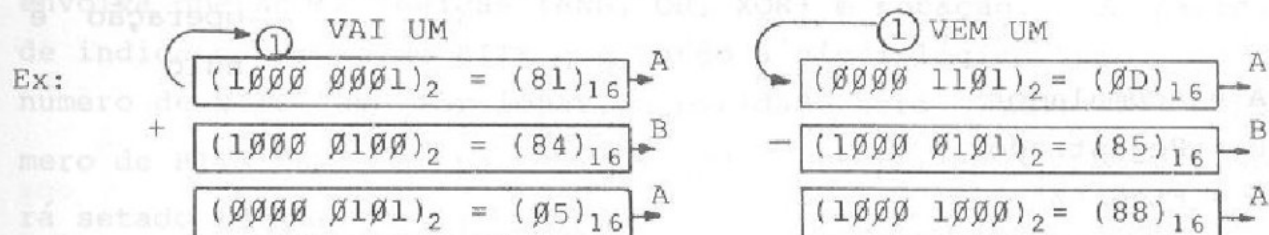


* Não é definido

Fig. 5.1 - Registrador de FLAG.

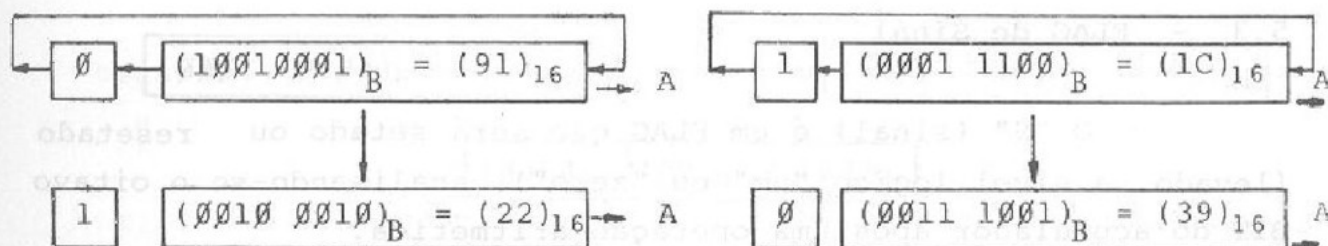
5.1 - FLAG de CARRY

O "C" (CARRY) é um FLAG que será setado (elevado a nível lógico "um") quando, em operações de adição, houver "vai um" no BIT "7", em operações de subtração, quando o BIT "7" necessitar de "vem um"; e, por último, quando envolver instruções de deslocamento. Na figura 5.2, pode-se acompanhar o exemplo de como o CARRY BIT pode ser afetado.



O FLAG C = 1 porque houve "vai um"

O FLAG C = 1 porque houve "vem um"



A = Acumulador
B = Registrador B
C = CARRY

Obs: { O índice B = 2 = Binário
O índice H = 16 = Hexadecimal

Fig. 5.2 - FLAG CARRY.

5.2 - FLAG de Zero

O "Z" (zero) é um FLAG que será setado (levado a nível lógico "um") quando, em operações lógicas aritméticas e de deslocamento, ocorrer no acumulador um resultado igual a zero. Na figura 5.3, pode-se acompanhar a operação.

+	① "vai um"		
	(1000 0000) _B = (80) _H	A	(1001 1001) _B = (99) _H
	(1000 0000) _B = (80) _H	B	(1001 1001) _B = (99) _H
	(0000 0000) _B = (00) _H	A	(0000 0000) _B = (00) _H

O FLAG Z=1 porque o resultado da operação é nulo

O FLAG Z=1 porque o resultado da operação é nulo

A = Acumulador

B = Registrador B

Z = FLAG "Z"

Fig. 5.3 - FLAG Zero.

5.3 - FLAG de Sinal

O "S" (sinal) é um FLAG que será setado ou resetado (levado a nível lógico "um" ou "zero"), analisando-se o oitavo BIT do acumulador após uma operação aritmética.

Se o oitavo BIT for "um", o FLAG "S" será setado e,

se for "zero", o FLAG "S" será resetado. Em outras palavras, o FLAG "S" é a cópia do oitavo BIT do acumulador.

Este FLAG é usado para indicar se um número é positivo ou negativo. Nesta técnica, podem-se representar números de (+127) a (-128). Na figura 5.4, pode-se acompanhar a operação.

Ex:	SINAL		"vem um"
		1	SINAL
	$(00001010) = (0A)_H$	A	$(00001000) = (04)_H$
+	$(00000110) = (06)_H$	B	$(00001000) = (08)_H$
	$(00010000) = (10)_H$	A	$(11111000) = (FC)_H$
S = "0" porque oitavo BIT é "zero"		S = "1" porque o oitavo BIT é "um"	
A = Acumulador		B = Registrador B	
		S = FLAG "S"	

Fig. 5.4 - FLAG Sinal.

5.4 - FLAG de Paridade e OVERFLOW

O BIT P/V (Paridade e OVERFLOW) é um FLAG que tem dupla função. A primeira indica a paridade do resultado quando envolve operações lógicas (AND, OR, XOR) e rotação. A paridade indica o número de BITS que estão a nível lógico "um". Se o número de BITS "um" for ímpar, a paridade será ímpar; se o número de BITS "um" for par, a paridade será par. O FLAG P/V será setado (levado a nível lógico "um"), quando a paridade for par. Caso contrário, o FLAG será resetado (levado a nível lógico "zero"). Pela figura 5.5, pode-se ver quando o FLAG Paridade será afetado.

$$(00010100)_B = (14)_H$$

AND

$$(11001110)_B = (CE)_H$$

$$(00001000)_B = (04)_H$$

FLAG (P/V) = "0" porque o número de BITS "um" é ímpar.

$$(0110 \ 1001)_B = (69)_H$$

OR

$$(1100 \ 0011)_B (C3)_H$$

$$(1110 \ 1011)_B = (EB)_H$$

FLAG (P/V) = "1" porque o número de BITS "um" é par.

Fig. 5.5 - FLAG de Paridade.

A segunda função é utilizada nas operações aritméticas que trabalham com complemento "dois", porque nesta condição são representados números entre (+127 e -128). Isto é, os 7 primeiros BITS indicam o número, sendo que o BIT mais significativo indica o sinal, como visto no FLAG "S". O FLAG P/V será setado quando exceder a máxima capacidade permitida em complemento dois e será resetado quando a operação estiver dentro dos conformes requeridos em complemento dois. Na figura 5.6, pode-se ver quando o FLAG de OVERFLOW será afetado.

Ex:

$$\begin{array}{rcl}
 & 1 \text{ "vai um"} & \\
 & 1 \quad 111 \quad 1001 \text{ número } (-7) \text{ em complemento dois} & \\
 + & 1 \quad 111 \quad 0001 \text{ número } (-15) \text{ em complemento dois} & \\
 \hline
 & 1 \quad 110 \quad 1010 \text{ número } (-22) \text{ em complemento dois} &
 \end{array}$$

SINAL

O FLAG P/V = "0", porque o número em complemento dois está correto.

O FLAG C = "1", porque houve "vai um".

Ex:

$$\begin{array}{rcl}
 & 1 \text{ "vai um"} & \\
 & 1 \quad 001 \quad 1100 \text{ número } (-100) \text{ em complemento dois} & \\
 + & 1 \quad 100 \quad 1110 \text{ número } (-50) \text{ em complemento dois} & \\
 \hline
 & 0 \quad 110 \quad 1010 \text{ incorreto} &
 \end{array}$$

SINAL

O FLAG P/V = "1", porque o número (- 150) em complemento Dois ocupa 8 BITS e o permitido é apenas 7 BITS.

O FLAG C = "1", porque houve "vai um".

OBS: O FLAG CARRY não tem nenhuma relação com o FLAG sinal.

Fig. 5.6 - FLAG de OVERFLOW.

5.5 - FLAGS H e N

Os BITS "H" e "N" serão utilizados em operações binárias codificadas em "BCD". A instrução "DAA" analisará estes FLAGS e tomará as decisões para o ajuste. Este ajuste tem o mesmo princípio dos Algoritmos estudados no primeiro volume.

O FLAG H (HALF-CARRY e HALF-BORROW) será setado, quando em operação aritmética houver "vai um" (HALF-CARRY) e "vem um" (HALF-BORROW) no meio do BYTE, isto é, do quarto para o quinto BIT e vice-versa.

O FLAG N é resetado em operação de adição e setado em operação de subtração.

1 "vai um" 1 "vai um"				1 "vem um"			
1111	1000	F8 _H	A	0110	0100	64 _H	A
+ 0000	1001	09 _H	B	- 0100	1000	48 _H	B
<hr/>				<hr/>			
0000	0001	01 _H	A	0001	1100	1C _H	A

H = "1" porque houve "vai um"
no meio do BYTE.

N = "0" porque é adição

H = "1" porque teve "vem um"
no meio do BYTE.

N = "1" porque é subtração

+				-			
0100	0011	43 _H	A	0101	0111	57 _H	A
+ 0011	0101	35 _H	B	- 0100	0110	46 _H	B
<hr/>				<hr/>			
0111	1000	78 _H	A	0001	0001	11 _H	A

H = "Ø" porque não houve "vai um"
no meio do BYTE

H = "ø" porque não teve
"vai um" no meio do
BYTE

N = "Ø" porque é adição

N = "1" porque é subtração

A = Acumulador

B = Registro B

H = HALF-CARRY

N = HALF-BORROW

Fig. 5.7 - FLAG HALF-CARRY e HALF-BORROW.

6.1 - Introdução

Interrupção é o nome dado quando, em um determinado instante, a Unidade Central de Processamento (CPU) pára a execução de um certo programa e começa processar um outro. Ao término deste programa, automaticamente a (CPU) volta a executá-lo a partir de onde houve a interrupção.

A maioria das interrupções são feitas por dispositivos periféricos, como (CTC), (PIO) e outros circuitos.

O Z-80 possui dois FLIP-FLOPS internos, "IFF1" e "IFF2". O "IFF1" determina se pode haver um pedido de interrupção; o "IFF2" é uma cópia do "IFF1", sendo usado para guardar o STATUS do "IFF1". As instruções que ativam e desativam estes FLIP-FLOPS são, respectivamente, (DI e EI).

O microprocessador Z-80 possui dois tipos básicos de interrupção: a mascarada e a não mascarada.

Na interrupção mascarada existe primeiramente uma verificação das condições dos FLIP-FLOPS antes da aceitação pela "CPU". Se estes estiverem ativos, a interrupção será aceita, caso contrário será recusada.

Já na interrupção não mascarada, não é necessário verificar se "IFF1" está ativo ou não, pois serão sempre processadas. Esta é utilizada em sistemas onde existem rotinas de grande prioridade.

Uma aplicação desta interrupção é em sistemas de grande porte, onde, por exemplo, em caso de falta de energia elétrica, entra automaticamente uma energia gerada por baterias.

Como esta energia tem certas limitações, a "CPU" executará apenas as rotinas de maior prioridade, como refrescamento das memórias dinâmicas.

6.2 - Interrupção Não Mascarada

Para que ocorra esta interrupção, é necessário que a entrada \overline{NMI} seja ativada, sendo efetuada através do nível lógico

co "zero".

Logo após o pedido de interrupção ser aceito pela CPU, esta carregará na posição de memória endereçada pelo STACK POINTER "SP", o endereço da instrução que a CPU iria executar se não houvesse este pedido.

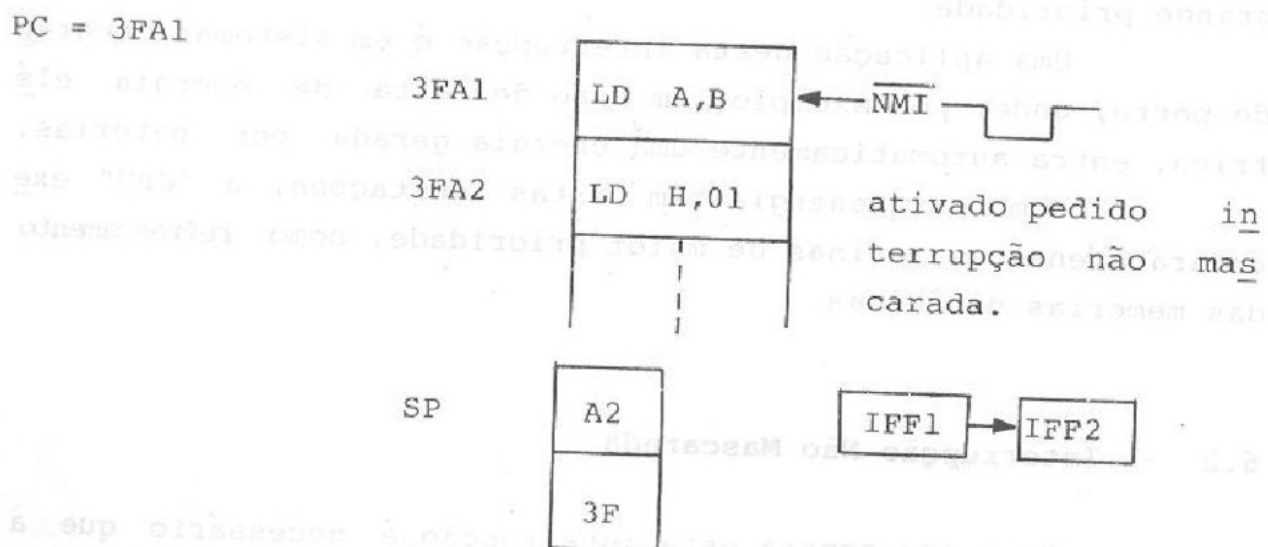
O "IFF1" será resetado, isto ocorre para que não haja um pedido de interrupção mascarada, enquanto estiver sendo executado.

Neste tipo de interrupção, o programa COUNTER (PC) será sempre carregado com o endereço (0066H). Geralmente neste endereço existe um "JUMP" para uma determinada rotina.

Um detalhe muito importante é com relação aos STATUS dos registradores. Uma vez que a interrupção é aceita, não se pode modificar o conteúdo dos registradores e então restauram-se os STATUS dos registradores. Esta pode ser realizada de duas maneiras: a primeira seria utilizar as instruções "EXX e EX AF,AF'" e a segunda seria com "PUSH AF, PUSH BC,...".

Para o término desta rotina, restauram-se de volta os STATUS dos registradores e as interrupções não mascaradas deverão terminar com a instrução "RETN". Esta carregará o conteúdo do "IFF2" em "IFF1" e o Programa COUNTER (PC) com o conteúdo da memória, endereçada pelo STACK POINTER (SP).

Na figura 6.1, tem-se o fluxo completo para interrupção não mascarada.



PC = 0066

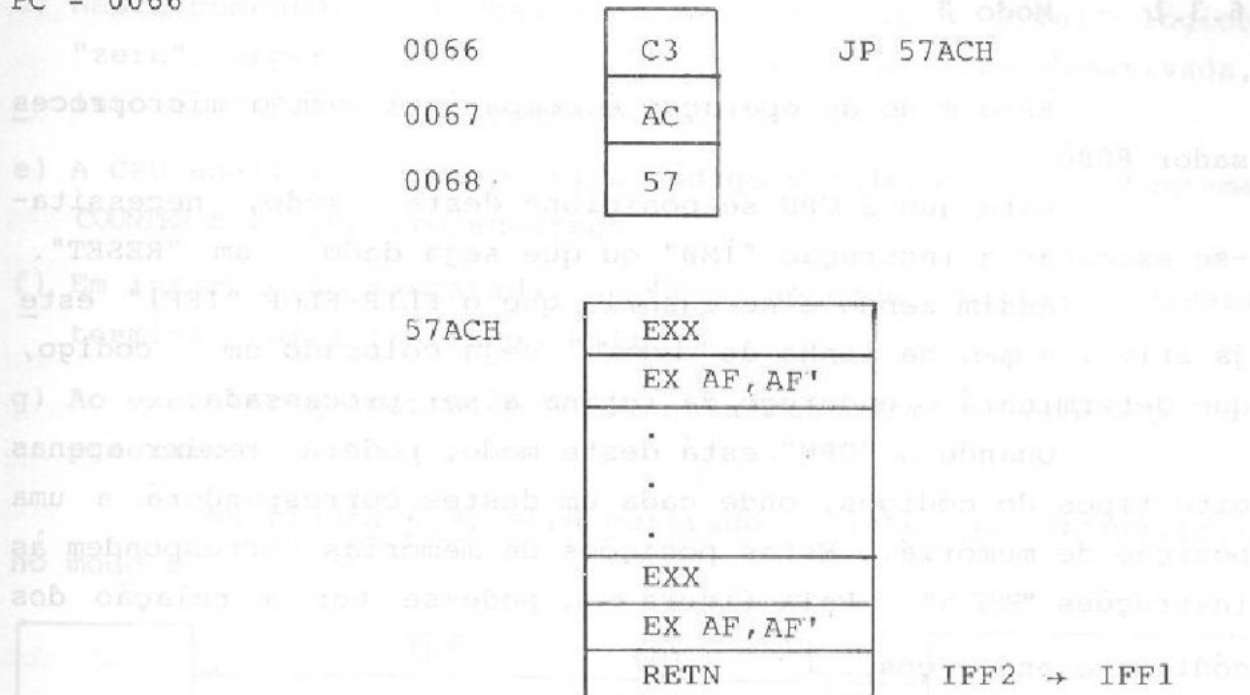


Fig. 6.1 - Fluxo da Interrupção não Mascarada.

6.3 - Interrupção Mascarada

A interrupção mascarada necessita de um controle pré-estabelecido por SOFTWARE. As instruções que determinam este controle são: "EI, DI", que habilitam e desabilitam a interrupção e "IM0, IM1 e IM2", que determinam o modo de interrupção.

Para que ocorra uma interrupção mascarada, necessita-se que a entrada "INT" seja ativada, isto é, elevada ao nível lógico "zero" e que o FLIP-FLOP "IFF1" esteja em condição para que se possa aceitar. Caso contrário, a "CPU", ignorará este pedido de interrupção.

O microprocessador Z-80 possui três maneiras para executar uma interrupção mascarada, designadas por: modo 0, modo 1 e modo 2.

O modo 0 é automaticamente selecionado pela "CPU" quando é dado um "RESET", isto é, quando a entrada \overline{RST} é colocada a nível lógico "zero".

6.3.1 - Modo 0

Este modo de operação é compatível com o microprocessador 8080.

Para que a CPU se posicione neste modo, necessita-se executar a instrução "IM0" ou que seja dado um "RESET".

Assim sendo é necessário que o FLIP-FLOP "IFF1" esteja ativo, e que, na linha de "dados", seja colocado um código, que determinará o endereço da rotina a ser processada.

Quando a "CPU" está deste modo, poderá receber apenas oito tipos de códigos, onde cada um destes corresponderá a uma posição de memória. Estas posições de memórias correspondem às instruções "RST n". Pela figura 6.2, pode-se ter a relação dos códigos e endereços.

CÓDIGOS	ENDEREÇOS	INSTRUÇÕES
C7H	0000H	RST 0
CFH	0008H	RST 8
D7H	0010H	RST 16
DFH	0018H	RST 24
E7H	0020H	RST 32
EEH	0028H	RST 40
F7H	0030H	RST 48
FFH	0038H	RST 56

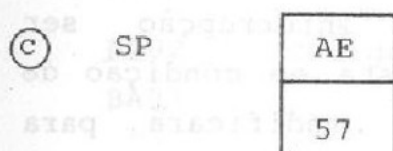
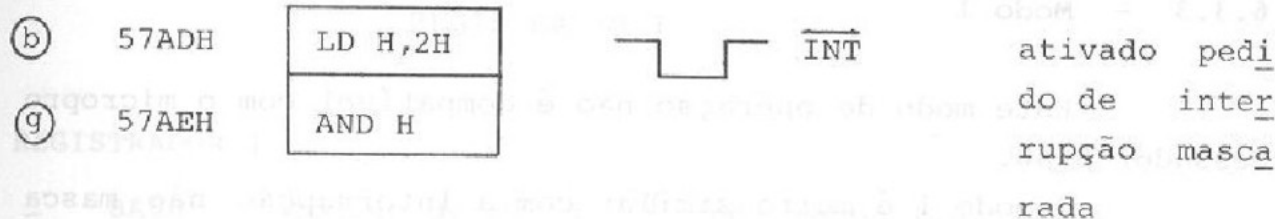
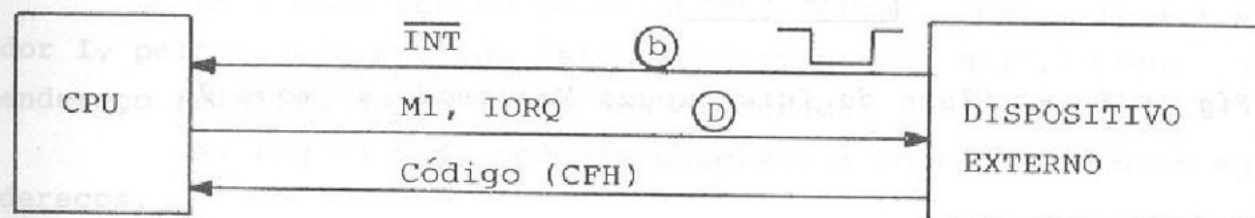
Fig. 6.2 - Relação dos Códigos e Endereços no modo "zero".

6.3.2 - Roteiro do Modo 0

- A CPU deverá estar no modo 0 e o FLIP-FLOP "IFF1" deverá estar em condições de receber uma interrupção.
- A linha $\overline{\text{INT}}$ deverá ser acionada.
- Na posição de memória, endereçada pelo STACK-POINTER (SP), deverá ser carregado o endereço da instrução, que a "CPU" iria executar se não tivesse ocorrido uma interrupção.

- d) Neste momento, as linhas $\overline{M1}$ e \overline{IORQ} irão para nível lógico "zero", esperando o código e a interrupção será desativada, isto é, "IFF1" será resetado.
- e) A CPU analisará o respectivo código e colocará no programa COUNTER o respectivo endereço.
- f) Em interrupção mascarada, no final de cada rotina, deverá terminar com a instrução RETI.
- g) Ao executar a instrução "RETI", esta restaura o endereço do programa.

Na figura 6.3, será mostrado o fluxo de interrupção no modo 0.



(e) PC = 0008

0008	C3	JP 9A07 H
0009	07	
000A	9A	

Fig. 6.3

PUSH AF
PUSH BC
PUSH DE
PUSH HL
POP HL
POP DE
POP BC
POP AF
EI
RET I

Fig. 6.3 - Fluxo da Interrupção Mascarada no modo 1.

6.3.3 - Modo 1

Este modo de operação não é compatível com o microprocessador 8080.

O modo 1 é muito similar com a interrupção não mascarada, exceto que a entrada, que deve ser aplicada ao sinal avisando uma interrupção, é $\overline{\text{INT}}$. Antes de uma interrupção ser aceita, é verificado se o FLIP-FLOP "IFF1" está em condição de aceitá-la, o programa COUNTER (PC) sempre se modificará para "0038H".

No modo 1, não há necessidade do dispositivo enviar um código de interrupção, porque o Programa Counter(PC) sempre se modificará para o mesmo endereço.

6.3.4 - Modo 2

Neste modo, o microprocessador Z-80 pode endereçar diretamente qualquer rotina por interrupção. Pode também ter até 128 dispositivos de entrada e saída, que necessitam trabalhar

com interrupção. Os dispositivos com "CTC" e "PIO", estudados no volume I, estão preparados para operar no modo 2.

O registrador I é utilizado neste modo para indicar a página da interrupção, em outras palavras, este indicará os oito BITS mais significativos de uma posição de memória.

Quando uma interrupção for aceita e a "CPU" estiver operando no modo 2, o dispositivo que requereu a interrupção deverá mandar, pela via de dados, um código que corresponderá à parte menos significativa de uma posição de memória. O BIT menos significativo do código enviado pelo dispositivo deverá estar em nível lógico "zero".

Este endereço de memória "M", formado pelo registrador I, pelo código enviado pelo dispositivo e "M+1", conterá o endereço relativo da rotina a ser executada.

Na figura 6.4, pode-se observar a montagem destes endereços.

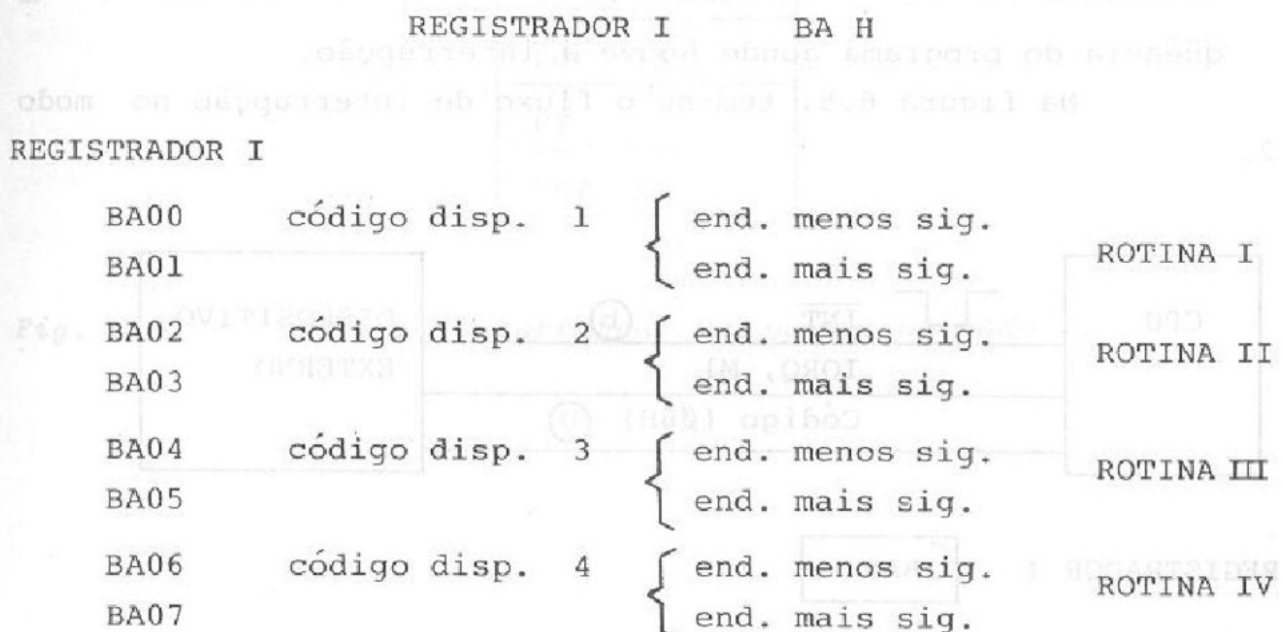
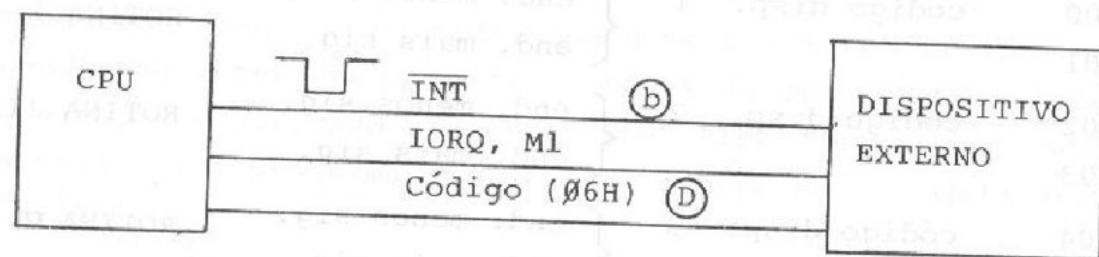


Fig. 6.4 - Montagem no Endereçamento no modo 2.

6.3.5 - Roteiro do Modo 2

- a) A CPU deverá estar no modo 2, o FLIP-FLOP "IFF1" deverá estar em condições de receber uma interrupção e o registrador I deverá conter a respectiva página de endereçamento.
- b) A linha INT deverá ser acionada.
- c) Na posição de memória, endereçada pelo STACK-POINTER "SP", deverá ser carregado o endereço da instrução que a "CPU" iria executar, se não tivesse ocorrido uma interrupção, e o FLIP-FLOP "IFF1" será resetado.
- d) Neste momento o dispositivo enviará pela via de dados o seu código.
- e) O registrador I, junto com o código do dispositivo, formará a posição de memória "M" e "M+1",
- f) O programa COUNTER "PC" será carregado com o conteúdo das memórias "M+1" e "M", respectivamente.
- g) No final desta rotina deverá conter a instrução "RETI", que carregará no "PC" o endereço, para voltar a executar a sequência do programa aonde houve a interrupção.

Na figura 6.5, tem-se o fluxo de interrupção no modo 2.



REGISTRADOR I

BAH

(b) 7730

LD A,D

(h) 7731

LD A,01H

INT ativa do pedido de interrupção mascarada.

(c) SP

31

77

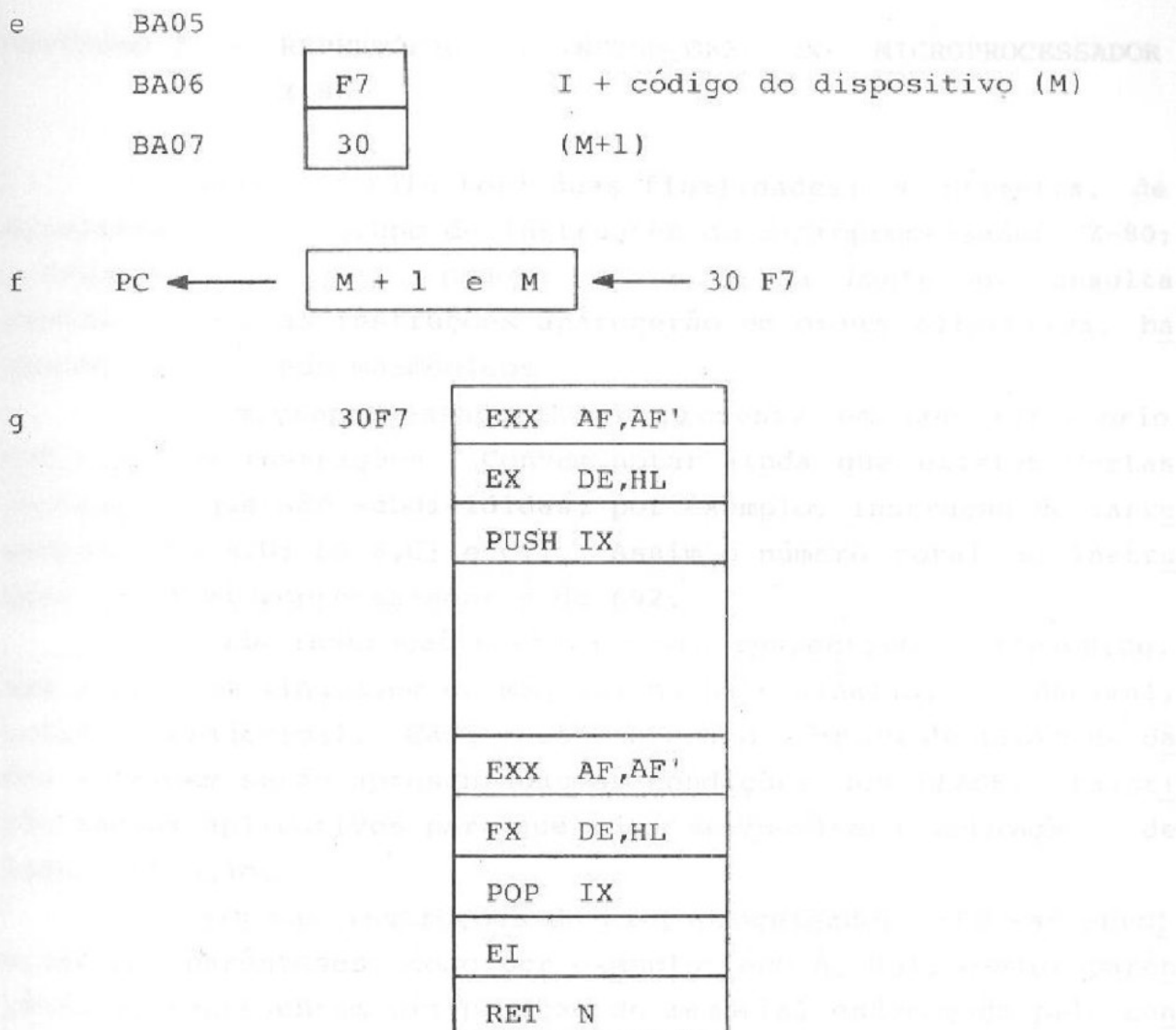


Fig. 6.5 - Fluxo da Interrupção Mascarada no modo 2.

CAPÍTULO 7 - REPERTÓRIO DAS INSTRUÇÕES DO MICROPROCESSADOR Z-80

Este capítulo terá duas finalidades: a primeira, de apresentar todo o grupo de instruções do microprocessador Z-80; a segunda, é que poderá ser uma fonte de consulta rápida, porque as instruções aparecerão em ordem alfabética, ba seando-se nos seus mnemônicos.

O microprocessador Z-80 apresenta em seu repertório 158 tipos de instruções. Convém notar ainda que existem certas instruções que são subdivididas, por exemplo, instrução de carre gamento (LD A,D; LD B,C; etc.). Assim o número total de instruções deste microprocessador é de 692.

Cada instrução conterà o seu respectivo mnemônico, seu código em linguagem de máquina na base binária, decimal, octal e hexadecimal. Cada caso conterà o sentido de fluxo de da dos e também serão apresentadas as condições dos FLAGS. Existi rão também aplicativos para que melhor se visualize a aplicação de cada instrução.

Algumas instruções do microprocessador Z-80 são envolvidas por parênteses, como por exemplo, ADC A,(HL), e estes parênteses representam uma posição de memória, endereçada pelo conteúdo do registrador envolvido pelos mesmos.

Certas instruções aparecerão abreviadas, a seguir será apresentado um sumário destes termos abreviados.

a) r e r' → Representa os registros B, C, D, E, H, L, A.

REGISTRADOR	CÓDIGO
B	000
C	001
D	010
E	011
H	100
L	101
A	111

b) SS → representa os registradores BC, DE, HL, SP.

REGISTRADOR	CÓDIGO
BC	00
DE	01
HL	10
SP	11

c) b → Representa o BIT a ser testado

BIT	CÓDIGO
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

d) T → Representa os registradores de índice IX e IY

REGISTRADORES	CÓDIGO
IX	0
IY	1

e) DD → Representa os registradores BC, DE, HL, AF

REGISTRADORES	CÓDIGO
BC	00
DE	01
HL	10
SP	11

1) ADC A,n

Operação Simbólica

AC ← AC + n + CY

Descrição:

Esta instrução é formada por dois BYTES, pelo código de operação e por um BYTE de dados. Esta instrução soma o conteúdo do acumulador com o BYTE de dados mais o CARRY BIT, sendo

o resultado carregado no acumulador:

Código de Operação

Hexadecimal	CE n
Octal	316 n
Decimal	206 n
Binário	11001110 n

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC A,n	C, Z, P/V, S,	2	2	7
Registrador	N, H.			

Condição dos FLAGS

- C SET se houve vai "um" do BIT 7
- Z SET se o resultado da operação for zero
- P/V SET se teve OVERFLOW
- S SET se o BIT 7 for "um"
- N RESET em qualquer condição
- H SET se houve vai "um" do BIT "3"

Exemplo: ADC A,25 H

Acumulador	=	1A H		0001	1010 B
Dado	=	25 H		0010	0101 B
CARRY BIT	=	01 H	+	0000	0001 B
Acumulador	=	40 H		0100	0000 B

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 1

2) ADC A,A

Operação Simbólica

A ← A + A + CY

Descrição;

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução soma o conteúdo do acumulador

mulador consigo mesmo, mais o CARRY BIT, sendo o resultado carregado no próprio acumulador.

Código de Operação

Hexadecimal	8F
Octal	217
Decimal	143
Binário	100001111

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC A,A	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver ADC A,n

Exemplo: ADC A,A

Acumulador	13	H	0001	0011
Acumulador	13	H	0001	0011
CARRY BIT	+	00	H	0000
Acumulador	26	H	0010	0110

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 0

3) ADC A,r

Operação Simbólica

A + A + r + CY

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução soma o conteúdo do acumulador com o conteúdo de um dos registradores (B, C, D, E, H, L), mais o CARRY BIT, sendo o resultado carregado no acumulador.

Código de Operação

Binário 10001 ← r →

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC A,r	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver ADC A,n

Exemplo: ADC A,B

Acumulador	4A	H	0100	1010
Registrador "B"	50	H	0101	0000
CARRY BIT	+ 01	H	+ 0000	0001
Acumulador	9B	H	1001	1011

C = 0 Z = 0 P/V = 0 S = 1 N = 0 H = 0

Exemplo: ADC A,L

Acumulador	5C	H	0101	1100
Registrador "L"	1D	H	0001	1101
CARRY BIT	+ 00	H	0000	0000
Acumulador	79	H	0111	1001

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 1

4) ADC A,(HL)

Operação Simbólica

AC ← AC + (HL) + CY

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução soma o conteúdo do acumulador com o conteúdo de memória, endereçada pelo par de registradores "HL", mais o CARRY BIT, e o resultado é carregado no acumulador.

Código de Operação

Hexadecimal	8E
Octal	216
Decimal	142
Binário	10001110

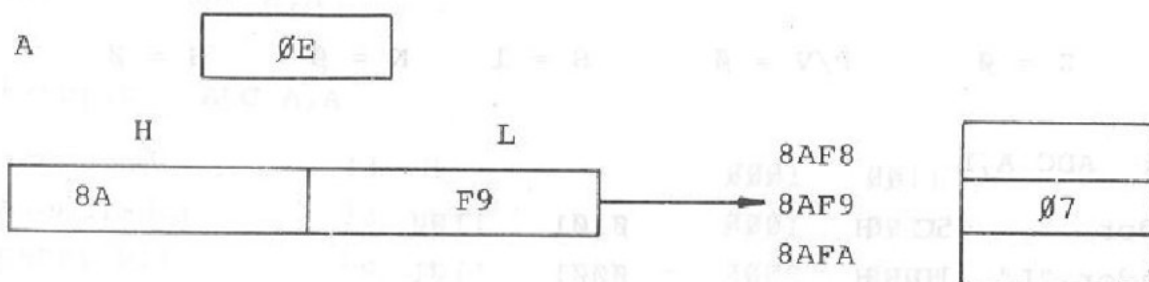
Parâmetros

Mnemônico	FLAGS Afetados	nº de BYTES	Ciclos de Máq.	Estados
ADC A, (HL)	C, Z, P/V, S, N, H	1	2	7

Condição dos FLAGS

Ver ADC A, n

Exemplo: ADC A, (HL)



Acumulador	0E	H	0000	1110		
Memória(8AF9H)	07	H	0000	0111		
CARRY BIT	+	<u>01</u>	H	+	<u>0000</u>	<u>0001</u>
Acumulador	16	H	0001	0110		

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 1

5) ADC A, (IX+d)

Operação Simbólica

AC ← AC + (IX+d) + CY

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação, e o terceiro será somado com o registrador de índice (IX). Ao efetuar esta instrução, o

conteúdo do acumulador será somado à posição de memória, endereçada pelo registrador de índice mais o CARRY BIT, e o resultado será carregado no acumulador.

Acumulador

Código de Operação

Hexadecimal	DD	8E	d
Octal	335	216	d
Decimal	221	142	d
Binário	11011101	10001110	d

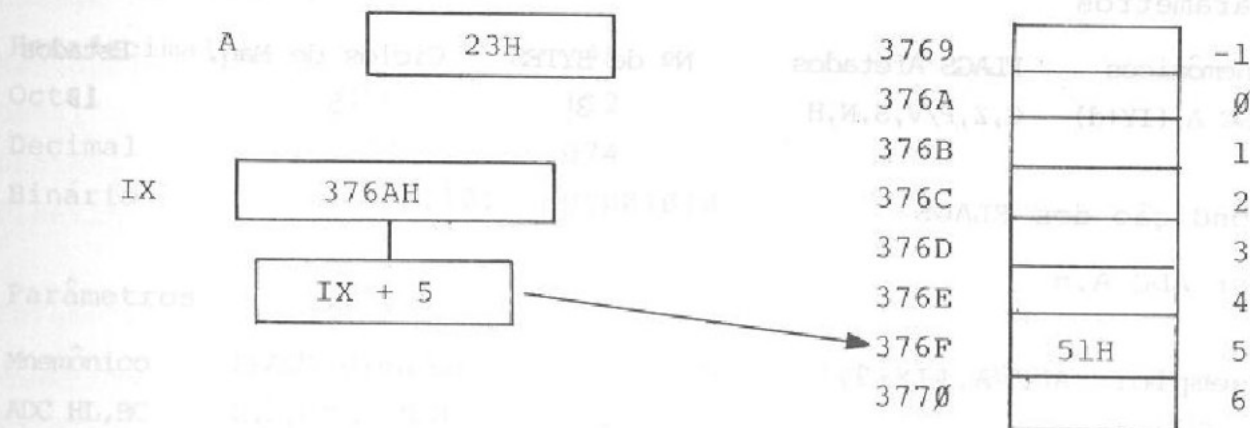
Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC A, (IX+d)	C,Z,P/V,S,N,H.	3	5	19

Condição dos FLAGS

Ver ADC A,n

Exemplo: ADC A, (IX+5)



Acumulador	23 H	0010	0011
Memória(IX+5)	51 H	0101	0001
CARRY BIT	+ 01 H	0000	0001
Acumulador	75 H	0111	0101

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 0

6) ADC A, (IY+ d)

Operação Simbólica

$AC \leftarrow AC + (IY + d) + CY$

Descrição:

Esta instrução é formada por três BYTES, sendo que dois BYTES correspondem ao código de operação, e o terceiro é somado com o registrador de índice (IY). Ao efetuar esta instrução, o conteúdo do acumulador será somado com a posição de memória, endereçada pelo registrador de índice mais o CARRY BIT, e o resultado será carregado no acumulador.

Código de Operação

Hexadecimal	FD	8E	d
Octal	375	216	d
Decimal	253	142	d
Binário	11111101	10001110	d

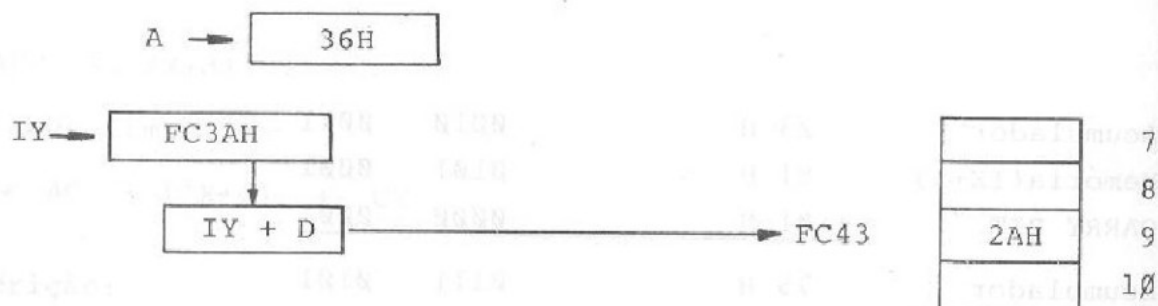
Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC A, (IY+d)	C, Z, P/V, S, N, H	3	5	19

Condição dos FLAGS

Ver ADC A, n

Exemplo: ADC A, (IY+9)



Acumulador	36 H	0011	0110
Memória(IY+d)	2A H	0010	1010
CARRY BIT	+ 00 H	+ 0000	0000
Acumulador	60 H	0110	0000

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 1

7) ADC HL,BC

Operação Simbólica

HL ← HL + BC + CY

Descrição:

Esta instrução é formada por dois BYTES, representando ambos o código de operação. O intuito desta instrução é o de somar o conteúdo do par de registradores "HL" com o conteúdo do par de registradores "BC", mais o CARRY BIT, e o resultado deverá ser carregado no par de registradores "HL".

Código de Operação

Hexadecimal	ED	4A
Octal	355	112
Decimal	237	074
Binário	11101101	01001010

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC HL,BC	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

C	SET se houve vai "um" do BIT 15
Z	SET se o resultado for "zero"
P/V	SET se teve OVERFLOW
S	SET se o BIT 15 for "um"
N	RESET em qualquer situação
H	SET se houve vai "um" do BIT 11.

Exemplo: ADC HL,BC

Registrador	HL 6D1C	H	0110	1101	0001	1100
Registrador	BC 127A	H	0001	0010	0111	1010
CARRY BIT	0001	H	+ 0000	0000	0000	0001
	7F97	H	0111	1111	1001	0111

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 0

8) ADC HL,DE

Operação Simbólica

HL ← HL + DE + CY

Descrição:

Esta instrução é formada por dois BYTES, representam do ambos o código de operação. Ao se efetuar esta instrução, o conteúdo do par de registradores "HL" é somado com o conteúdo do par de registradores "DE", mais o CARRY BIT, e o resultado deverá ser carregado no par de registradores "HL".

Código de Operação

Hexadecimal	ED	5A
Octal	355	132
Decimal	237	090
Binário	11101101	01011010

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC HL,DE	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver ADC HL,BC.

Exemplo:

Registrador	HL	6BA3	H	0110	1011	1010	0011
Registrador	BC	047A	H	0000	0100	0111	1010
CARRY BIT		+ 0000	H	+ 0000	0000	0000	0000
Decimal		7010	H	0111	0000	0001	1101

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 1

9) ADC HL,HL

Operação Simbólica

HL ← HL + HL + CY

Descrição:

Esta instrução é formada por dois BYTES, representam do ambos o código de operação. Ao se efetuar esta instrução, o conteúdo do par de registradores "HL" é somado com si próprio, mais o CARRY BIT, e o resultado deverá ser carregado no par de registradores "HL".

Código de Operação

Hexadecimal	ED	6A
Octal	355	152
Decimal	237	106
Binário	11101101	01101010

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC HL,HL	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver ADC HL,BC

Exemplo:

Registrador	HL	017A	H	0000	0001	0111	1010
Registrador	HL	017A	H	0000	0001	0111	1010
CARRY BIT		+ 0000	H	0000	0000	0000	0000
		02F4	H	0000	0010	1111	0100

10) ADC HL,SP

Operação Simbólica

$HL \leftarrow HL + SP + CY$

Descrição:

Esta instrução é formada por dois BYTES, representando ambos o código de operação. Ao se efetuar esta instrução, o conteúdo do par de registradores "HL" é somado com o conteúdo do registrador "SP" (STACK POINTER), mais o CARRY BIT, e o resultado deverá ser guardado no par de registradores "HL".

Código de Operação

Hexadecimal	ED	7A
Octal	355	172
Decimal	237	122
Binário	11101101	01111010

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADC HL,HL	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver ADC HL,BC

11) ADD A,(HL)

Operação Simbólica

$A \leftarrow A + (HL)$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução irá efetuar a operação de soma entre o acumulador e a posição de memória, endereçada pelo par de registradores "HL", e o resultado ficará retido no acumulador.

Código de Operação

Hexadecimal	86
Octal	206
Decimal	134
Binário	10000110

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD A,(HL)	C,Z,P/V,S,N,H.	1	2	7

Condição dos FLAGS

Ver ADC A,n

Exemplo:

ADD A,(HL)

Par."HL" → (76DAH) → A2H

Acumulador	36 H	0011	0110
Memória (76DAH)	+ A2 H	1010	0010
	08 H	1101	1000

C = 0 Z = 0 P/V = 0 S = 1 N = 0 H = 0

12) ADD A,(IX+d)

Operação Simbólica

A ← A + (IX+d)

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação; o terceiro indica o BYTE que deverá ser somado ou subtraído do registrador (IX).

Esta instrução soma o acumulador com a posição de memória endereçada pelo registrador de índice (IX), e o resultado é carregado no acumulador.

Código de Operação

Hexadecimal	DD	86	d
Octal	335	206	d
Decimal	221	134	d
Binário	11011101	10000110	d

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD A, (IX+d)	C, Z, P/V, S, N, H.	3	5	19

Condição dos FLAGS

Ver ADC A, n

Exemplo:

ADD A, (IX+0AH)

(IX) = AB10

Acumulador	1A H	0001	1010
Memória (IX+0AH)	63 H	0110	0011
	7D H	0111	1101

C = 0 Z = 0 P/V = 0 S = 1 N = 0 H = 0

13) ADD A, (IY + d)

Operação Simbólica

$A \leftarrow A + (IY + d)$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação; o terceiro indica o BYTE que deverá ser somado ou subtraído do registrador (IY). Esta instrução soma o acumulador com a posição de memória indicada pelo registrador de índice (IY), e o resultado é carregado no acumulador.

Código de Operação

Hexadecimal	FD	86	d
-------------	----	----	---

Octal 375 206 d
 Decimal 253 134 d
 Binário 11111101 10000110 d

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de Ciclos	Ciclos de Máq.	Estados
ADD A, (IX+d)	C, Z, P/V, S, N, H	3	5	19

Condição dos FLAGS

Ver ADC A, n

14) ADD A, n

Operação Simbólica

$A \leftarrow A + n$

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação, e o segundo, um BYTE de dados. Ao se efetuar esta instrução, o conteúdo do acumulador é somado com o BYTE de dados, e o resultado é guardado no acumulador.

Código de Operação

Hexadecimal C6 n
 Octal 306 n
 Decimal 198 n
 Binário 11000110 n

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD A, n	C, Z, P/V, S, N, H	2	2	7

Condição dos FLAGS

Ver ADC A, n

15) ADD A, r

Operação Simbólica

AC ← AC + r

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução soma o conteúdo do acumulador com o conteúdo de um dos registradores (B, C, D, E, H, L), sendo o resultado carregado no acumulador.

Código de Operação

Binário 100000 + r +

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD A,r	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver ADC A,n

Exemplo: ADD A,D

Acumulador	4A H	0100	1010
Registro "D"	+ 50 H	+ 0101	0000
Acumulador	9A H	1001	1010

C = 0 Z = 0 P/V = 0 S = 1 N = 0 H = 0

ADD A,H

Acumulador	5C H	0101	1100
Registro "H"	+ 1D H	+ 0001	1101
Acumulador	79 H	0111	1001

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 1

16) ADD HL,BC

Operação Simbólica

HL ← HL + BC

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução soma o conteúdo do par de registradores "HL" com o conteúdo do par de registradores "BC", e o resultado é guardado no par de registradores "HL".

Código de Operação

Hexadecimal	09
Octal	11
Decimal	9
Binário	00001001

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD HL,BC	C,N,H	1	3	11

Condição dos FLAGS

C SET se houve vai um do BIT 15

Z Não afetado

P/V Não afetado

N RESET em qualquer condição

H SET se houve vai um do BIT 11

Exemplo;

Registrador	HL	7890H	0111	1000	1001	0000
Registrador	BC	01FAH	0000	0001	1111	1010
Registrador	HL	7A8AH	0111	1010	1000	1010

C = 0

N = 0

H = 0

17) ADD HL,DE

Operação Simbólica

HL ← HL + DE

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução soma o conteúdo do par de registradores "HL" com o conteúdo do par de registradores "DE", e o resultado é guardado no par de registradores "HL".

Código de Operação

Hexadecimal	19
Octal	31
Decimal	25
Binário	00011001

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD HL,DE	C,N,H	1	3	11

Condição dos FLAGS

Ver ADD HL,BC

Exemplo: ADD HL, DE

1 vai um

Registrador	"HL"	FF00H	1111	1111	0000	0000
Registrador	"DE"	+ A17BH	+ 1010	0001	0111	1011
Registrador	"HL"	A07BH	1010	0000	0111	1011

C = 1

N = 0

H = 1

18) ADD HL,HL

Operação Simbólica

HL ← HL + HL

Descrição:

Esta instrução é formada por um único BYTE, sendo este código de operação. Esta instrução soma o conteúdo do par de registradores "HL" com si próprio, e o resultado é guardado no par de registradores "HL".

Código de Operação

Hexadecimal	29
Octal	51
Decimal	41
Binário	00101001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD HL,HL	C,N,H	1	3	11

Condição dos FLAGS

Ver ADD HL,BC

19) ADD HL,SP

Operação Simbólica

HL ← HL + SP

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução soma o conteúdo do par de registradores "HL" com o conteúdo do registrador "SP" (STACK POINTER), e o resultado é guardado no par de registradores "HL".

Código de Operação

Hexadecimal	39
Octal	71
Decimal	57
Binário	00111001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD HL,SP	C,N,H	1	3	11

Condição dos FLAGS

Ver ADD HL,BC

Exemplo:

Utilizando-se esta instrução convenientemente, pode-se examinar o conteúdo do registrador "SP". Para isto, zera-se o conteúdo do par de registradores "HL" e soma-se com o registrador "SP". Após esta soma, no par de registradores "HL" tem-se o valor do registrador "SP".

Registrador	HL	0000H	0000	0000	0000	0000
Registrador	SP	+ 7BAFH	+ 0111	1011	1010	1111
		7BAFH	0111	1011	1010	1111

C = 0

N = 0

H = 0

20) ADD IX, WW

SS representa os registradores BC, DE, IX, SP

Registrador	SS	Código
BC		00
DE		01
IX		10
SP		11

Operação Simbólica

IX ← IX + WW

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução soma o conteúdo do registrador de índice "IX" com o conteúdo do registrador "SS", e o resultado é guardado no registrador "IX".

Código de Operação

Binário 11011101 00 + WW + 1001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD IX,WW	C,N,H	2	4	15

Condição dos FLAGS

Ver ADD HL,BC

Exemplo: ADD IX,DE

Registrador	IX	7F1A	H	0111	1111	0001	1010
Registrador	DE	+ 10F0	H	+ 0001	0000	1111	0000
		900A	H	1001	0000	0000	1010

C = 0

N = 0

H = 1

21) ADD IY,WW

SS representa os registradores BC, DE, IX, SP

Registradores WW	Códigos
BC	00
DE	01
IY	10
SP	11

Operação Simbólica

IY ← IY + SS

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução soma o conteúdo do registrador de índice IY com o conteúdo do registrador "WW", e o resultado é guardado no registrador "IY".

Código de Operação

Binário 11111101 00 + WW + 1001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
ADD IY,WW	C,N,H	2	4	15

Condição dos FLAGS

Ver ADD HL,BC

Exemplo: ADD IY,IY

1 vai um

Registrador IY	B17A	H	1011	0001	0111	1010
Registrador IY	+ B17A	H	+ 1011	0001	0111	1010
Registrador IY	62F4	H	0110	0010	1111	0100

C = 1

N = 0

H = 0

22) AND r

r representa os registradores A,B,C,D,E,H,L

Registrador r	Código
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Operação Simbólica

$A \leftarrow A \wedge r$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução efetua a lógica "AND", entre o conteúdo do acumulador e o conteúdo de um dos registradores descritos, sendo o resultado carregado no acumulador.

Código de Operação

Binário 10100 + _ r _ +

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
AND r	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

- C RESET em qualquer situação
- Z SET se o resultado for "zero"
- P/V SET se paridade for par (se no número de "um" for par)
- S SET se o BIT "7" for igual a "um"
- N RESET em qualquer situação
- H SET em qualquer situação

Exemplo: AND B

Acumulador 8F H 1000 1111

Registrador B B0 H 1011 0000

Acumulador 80 H 1000 0000

S = 1 Z = 0 H = 1 P/V = 0 N = 0 C = 0

Acumulador C8 H 1100 1000

Registrador H 17 H 0001 0111

Acumulador 00 H 0000 0000

S = 0 Z = 1 H = 1 P/V = 1 N = 0 C = 0

23) AND n

Operação Simbólica

$A \leftarrow A \wedge n$

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação e o segundo um BYTE de dados. Esta instrução efetua a lógica AND, entre o acumulador e o BYTE de dados, sendo o resultado carregado no acumulador.

Código de Operação

Hexadecimal	E6	n
Octal	346	n
Decimal	230	n
Binário	11100110	n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
AND n	C,Z,P/V,S,N,H	2	2	7

Condição dos FLAGS

Ver AND r

24) AND (HL)

Operação Simbólica

$A \leftarrow A \wedge (HL)$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução realiza a lógica AND, entre o acumulador e a posição de memória indicada pelo par de registradores "HL", sendo o resultado carregado no acumulador.

Código de Operação

Hexadecimal	A6
Octal	246
Decimal	166
Binário	10100110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
AND (HL)	C,Z,P/V,S,N,H	1	2	7

Condição dos FLAGS

Ver AND r

Exemplo: AND (HL)

Acumulador	73H	0111	0011		
(3704H)=HL	<u>3EH</u>	<u>0011</u>	<u>1110</u>		
Acumulador	32H	0011	0010		
S = 0	Z = 0	H = 1	P/V = 0	N = 0	C = 0

25) AND (IX+d)

Operação Simbólica

$A \leftarrow A \wedge (IX+d)$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação e o terceiro um BYTE de dados, que deverá ser somado ou subtraído do registrador de índice (IX), que apontará a posição do dado a ser manuseado. Esta instrução executa a lógica "AND" entre o acumulador e a posição indicada pelo registrador de índice (IX), com o seu respectivo BYTE de dados, sendo o resultado carregado no acumulador.

Código de Operação

Hexadecimal	DD	A6	d
Octal	335	246	d
Decimal	221	166	d
Binário	11011101	10100110	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
AND (IX+d)	(C,Z,P/V,S,N,H)	3	5	19

Condição dos FLAGS

Ver AND R

Exemplo: AND (IX+10H)

Posição de memória IX = (7AC7H)

Posição de memória IX + 10H = (7AD7H)

Acumulador	7A H	0111 1010
Posição de memória (IX+10)	F1 H	1111 0001
Acumulador	70 H	0111 0000

S = 0 Z = 0 H = 1 P/V = 0 N = 0 C = 0

26) AND (IY+d)

Operação Simbólica

$A \leftarrow A \wedge (IY+d)$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação e o terceiro um BYTE de dados, que deverá ser somado ao registrador de índice (IY), que apontará a posição do dado a ser manuseado. Esta instrução executa a lógica AND, entre o acumulador e a posição indicada pelo registrador de índice (IY), com o seu respectivo BYTE de dados, sendo o resultado carregado na memória.

Código de Operação

Hexadecimal	FD	A6	d
Octal	375	246	d
Decimal	253	166	d
Binário	11111101	10100110	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
AND(IY+d)	C,Z,P/V,S,N,H	3	5	19

Condição dos FLAGS

Ver AND r

27) BIT b,r

Operação Simbólica

"FLAG Z" $\leftarrow \overline{r_b}$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução testa um único BIT de um registrador. O FLAG "Z" é responsável por este teste, porque o BIT testado for "um", o FLAG "Z" será resetado; caso contrário, será setado, em outras palavras, complementa-se o valor do BIT

testado e coloca-se no FLAG "Z".

Código de Operação

11001011 01 $\leftarrow b \rightarrow$ $\leftarrow r \rightarrow$

Parâmetros

Mnemônico	FLAG Afetados	Nº de BYTES	Ciclos de Máq.	Estados
BIT b,r	Z,P/V,S,N,H	3	2	8

Condição dos FLAGS

C Não afetado
 Z SET se o BIT testado for "zero"
 P/V Não determinado
 S Não determinado
 N Reseta em qualquer condição
 H Seta em qualquer condição

Exemplo: BIT 5,H

Registrador H = 7EH 01111110

Z = 0

BIT 3,0

Registrador D = B2H 10110010

Z = 1

28) BIT b, (HL)

Operação Simbólica

"FLAG Z" $\leftarrow \overline{M_b}$

Descrição:

Esta instrução é formada por dois BYTES, sendo os dois o código de operação. Esta instrução testa um BIT de memória endereçada pelo par de registradores (HL), afetando o "FLAG Z".

Código de Operação

11001011

01 + _ b _ + 110

Parâmetros

Mnemônicos	FLAG Afetados	Nº de BYTES	Ciclos de Máq.	Estados
BIT b,(HL)	Z,P/V,S,N,H	2	3	12

Condição dos FLAGS

Ver BIT b,r

Exemplo: BIT 0,(HL)

7BADH

+ HL

7BAD

6A

M (HL) = 6AH 01101010

Z = 1

29) BIT b,(IX+d)

Operação Simbólica

"FLAG Z" + M_b

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto, o código de operação. O terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução testa um BIT de uma determinada posição de memória, endereçada pelo registrador de índice (IX), com o seu respectivo BYTE de dados.

Código de Operação

Binário 11011101 11001011 d 01 + b + 110

Parâmetros

Mnemônico	FLAG Afetados	Nº de BYTES	Ciclos de Máq.	Estados
BIT b, (HL)	Z, P/V, S, N, H	4	5	20

Condição dos FLAGS

Ver BIT b, r

30) BIT b, (IY+d)

Operação Simbólica

"FLAG Z" + M_b

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto, o código de operação. O terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução testa um BIT de uma determinada posição de memória, endereçada pelo registrador de índice (IY), com o seu respectivo BYTE de dados.

Código de Operação

11111101 11001011 d 01 + b + 110

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
BIT b, (IY+d)	Z, P/V, S, N, H	4	5	20

Condição dos FLAGS

Ver BIT b, r

31) CALL cc, nn

cc Condição

NZ	Se não zero
Z	Se zero
NC	Se não CARRY

Código

000
001
010

FLAGS

Z
Z
C

C	Se CARRY	011	C
PO	Se paridade ímpar	100	P/V
PE	Se paridade par	101	P/V
P	Se sinal positivo	110	S
M	Se sinal negativo	111	S

Operação Simbólica

(SP - 1) + PCH

(SP - 2) + PCL

PC + nn

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação. Os dois últimos representam o endereço que irá modificar o programa COUNTER (PC). Esta instrução é utilizada para chamada de subrotinas, só que antes de chamar estas respectivas subrotinas, verificam-se as condições dos FLAGS. Este tipo de instrução é denominada de instrução condicional, pois depende de alguma condição para ser executada. Se as condições dos FLAGS forem favoráveis, esta instrução colocará o conteúdo do "PC" (Programa COUNTER) na posição de memória endereçada pelo STACK POINTER. Esta operação indica aonde voltar o programa COUNTER (PC), depois da execução desta respectiva subrotina. Em seguida, colocar-se-á no PC o endereço da subrotina, se as condições dos FLAGS forem favoráveis, caso contrário, a instrução será ignorada.

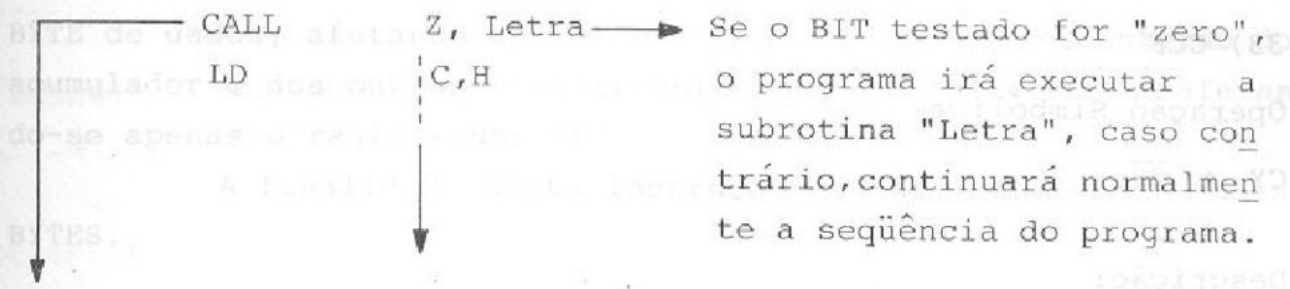
Parâmetros

Mnemônico	FLAG Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CALL cc,nn	nenhum	3	3 5	10 falsa 17 verdadeira

Código de Operação

Binário 11 + cc + 100 nn

Exemplo: BIT 1,A



Letra: LD A,B
INC. (HL)
RET

32) CALL nn

Operação Simbólica

(SP - 1) ← PCH

(SP - 2) ← PCL

PC ← nn

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação, e os dois últimos o endereço que irá modificar o programa COUNTER (PC). Este tipo de instrução é chamado de instrução incondicional, pois não depende de nenhuma condição para ser executada.

Ao executar-se esta instrução, colocar-se-á no "PC", o par de dados (nn), e no STACK POINTER (SP), o endereço de retorno ao qual o programa voltará após a execução de uma dada rotina.

Código de Operação

Hexadecimal	CD	nn
Octal	315	nn
Decimal	205	nn
Binário	11001101	nn

Parâmetros

Mnemônico	FLAG Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CALL nn	nenhum	3	5	17

33) CCF

Operação Simbólica

$CY \leftarrow \overline{CY}$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução complementa o CARRY BIT.

Código de Operação

Hexadecimal	3F
Octal	77
Decimal	63
Binário	00111111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CCF	H,N,C	1	1	4

Condição dos FLAGS

C SET se o CY era "zero" resseta se era "um"

Z Não modificado

P/V Não modificado

S Não modificado

N Resetado em qualquer condição

H Copia o CY anterior antes da execução desta instrução

34) CP n

Operação Simbólica

A - n

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação e o segundo o BYTE de dados. Esta instrução efetua a subtração entre o conteúdo do acumulador e o

BYTE de dados, afetando apenas os FLAGS, sendo que o conteúdo do acumulador e dos outros registradores não são alterados, afetando-se apenas o registrador "F".

A finalidade desta instrução é a de comparar dois BYTES.

Código de Operação

Hexadecimal	FE	n
Octal	376	n
Decimal	254	n
Binário	11111110	n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados.
CP n	C,Z,P/V,S,N,H	2	2	7

Condição dos FLAGS

C	SET se teve vem "um" para o BIT 7
Z	SET se o resultado for zero
P/V	SET se teve OVERFLOW
S	SET se o BIT 7 for "um"
N	SET em qualquer situação
H	SET se teve vem "um" do BIT 4

35) CP r

Operação Simbólica

A - r

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução efetua a subtração entre o acumulador e o registrador "r", afetando apenas os FLAGS.

Código de Operação

Binário 10111 + - r - +

Parâmetros				
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CP r	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver CP n

Exemplo: CP B

Acumulador	6C	H	0110	1100
Registro B	- 61	H	- 0110	0001
	0B	H	0000	1011
				Resultado ignorado

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 0

36) CP (HL)

Operação Simbólica

A - (HL)

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução efetua a subtração entre o acumulador e a posição de memória, endereçada pelo par de registradores "HL", afetando apenas os FLAGS.

Código de Operação

Hexadecimal	BE
Octal	276
Decimal	190
Binário	10111110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CP (HL)	C,Z,P/V,S,N,H	1	1	7

Condição dos FLAGS

Ver CP n

37) CP (IX + d)

Operação Simbólica

A - (IX + d)

Descrição:

Esta instrução é formada por 3 BYTES, sendo os dois primeiros o código de operação, e o terceiro um BYTE de dados. Esta instrução efetua a subtração entre o acumulador e a posição de memória, endereçada pelo registrador de índice "IX" em conjunto com o BYTE "d", afetando apenas os FLAGS.

Código de Operação

Hexadecimal	DD	BE	d
Octal	335	276	d
Decimal	221	190	d
Binário	11011101	10111110	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CP (IX+d)	C,Z,P/V,S,N,H	3	5	19

Condição dos FLAGS

Ver CP r

38) CP (IY + d)

Operação Simbólica

A - (IY + d)

Descrição:

Esta instrução é formada por 3 BYTES, sendo os dois primeiros o código de operação e o terceiro um BYTE de dados. Esta instrução efetua a subtração entre o acumulador e a posição de memória, endereçada pelo registrador de índice "IY" em conjunto com o BYTE "d", afetando apenas os FLAGS.

Código de Operação

Hexadecimal	FD	BE	d
Octal	375	276	d
Decimal	253	190	d
Binário	11111101	10111110	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CP (IY+d)	C,Z,P/V,S,N,H	3	5	19

Condição dos FLAGS

Ver CP n

39) CPD

Operação Simbólica

A	-	(HL)
HL	+	HL - 1
BC	+	BC - 1

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução subtrai o conteúdo do acumulador da posição de memória, endereçada pelo par de registradores (HL), afetando apenas os FLAGS. Após esta operação, o par de registradores "HL" é decrementado, o que indicará a próxima posição de memória a ser comparada; também o par de registradores "BC" é decrementado, funcionando como um contador, indicando a quantidade de BYTES a ser comparada.

Código de Operação

Hexadecimal	ED	A9
Octal	355	251
Decimal	237	169
Binário	11101101	10101001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CPD	Z,P/V,S,N,H	2	4	16

Condição dos FLAGS

C	Não modificado
Z	SET se A = (HL)
P/V	SET se BC - 1 \neq 0
S	SET se o BIT 7 é igual a "um"
N	SET em qualquer condição
H	SET se vem vai um do BIT 4

Exemplo:

Acumulador "B7"

HL	ADFE
BC	0004

ADFB
ADFC
ADFD
ADFE

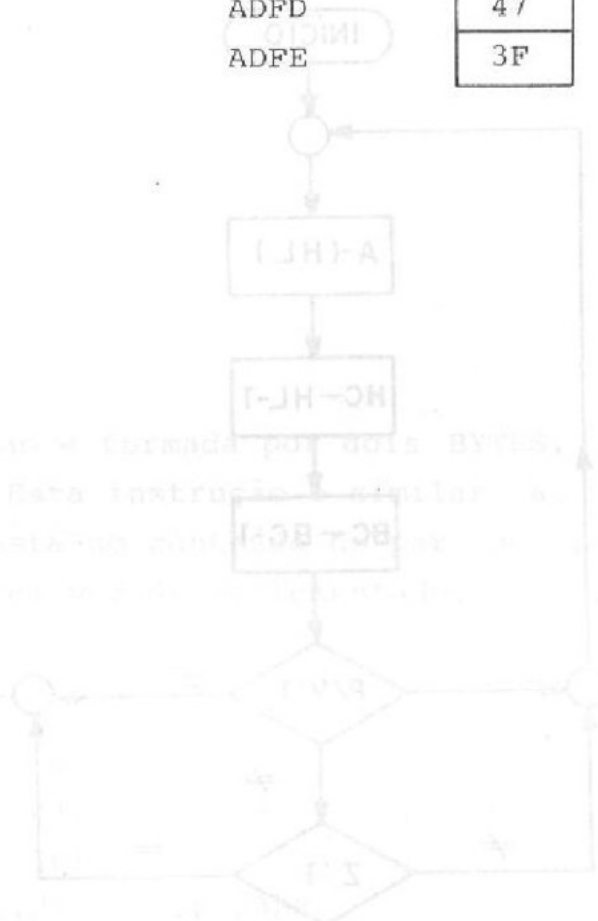
7A
37
47
3F

CPD	
P/V = 0	Z = 0
HL	ADFD
BC	0003

CPD	
P/V = 0	Z = 0
HL	ADFC
BC	0002

CPD	
P/V = 0	Z = 1
HL	ADFC
BC	0001

CPD	
P/V = 1	Z = 0
HL	ADFB
BC	0000



40) CPDR

Operação Simbólica

A ← (HL)

HL ← HL + 1

BC ← BC - 1

PC ← PC - 2 se B ≠ 0 ou A = (HL)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. A única diferença entre CPD CPDR é que esta instrução, uma vez executada, entra em repetição, saindo desta apenas quando o conteúdo do acumulador for igual à posição de memória, endereçada pelo par de registradores "HL", ou quando o conteúdo do par de registradores "BC", for igual a zero. Pela figura 7.1, pode-se ver a estrutura da instrução.

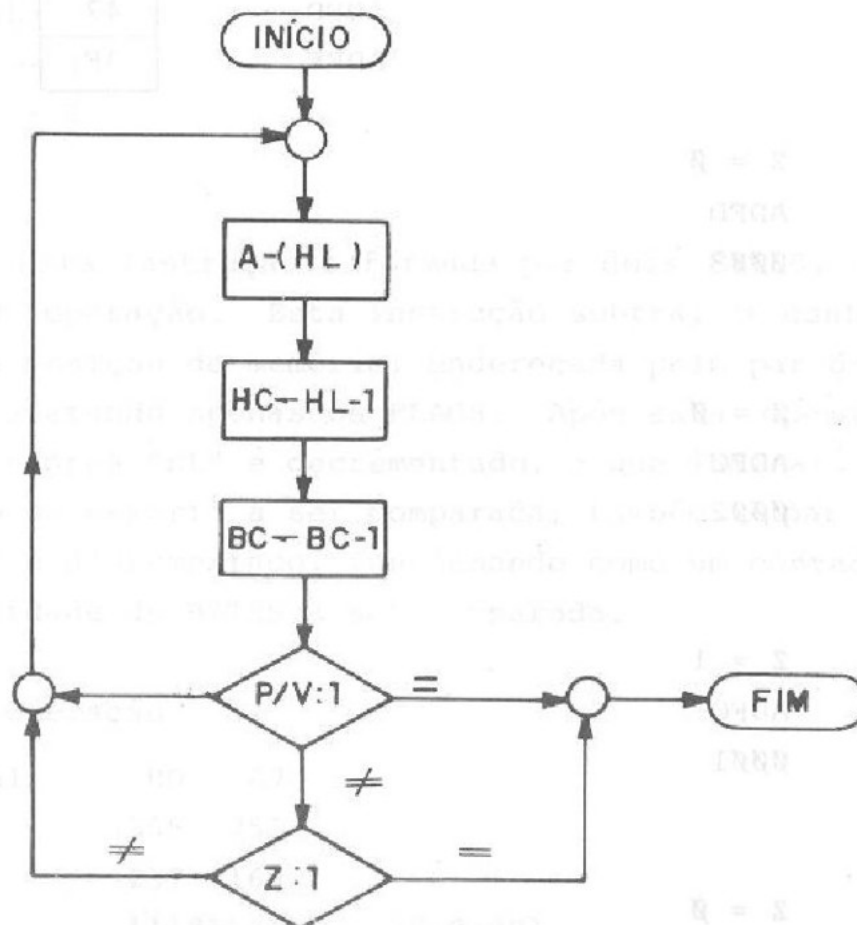


Fig. 7.1 - Estrutura da instrução CPDR.

Código de Operação

Hexadecimal	ED	B9
Octal	355	271
Decimal	237	185
Binário	11101101	10111001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CPDR	Z,P/V,S,N,H	2	5 4	21 16

BC = 0 e A ≠ (HL)

BC = 0 ou A = (HL)

Condição dos FLAGS

Ver CPD

41) CPI

Operação Simbólica

A - (HL)

HL ← HL + 1

BC ← BC - 1

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar ao CPD, sendo que a única diferença está no conteúdo do par de registradores HL, que é incrementado em vez de decrementado.

Código de Operação

Hexadecimal	ED	A1
Octal	355	241
Decimal	237	161
Binário	11101101	10100001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CPI	Z,P/V,S,N,H	2	4	16

Condição dos FLAGS

Ver CPD

42) CPIR

Operação Simbólica

A - (HL)

HL + HL + 1

BC + BC - 2

PC + PC - 2 se B ≠ 0 ou A = (HL)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à CPDR, sendo que a única diferença está no conteúdo do par de registradores "HL", que é incrementado em vez de decrementado.

Código de Operação

Hexadecimal	ED	B1
Octal	355	261
Decimal	237	177
Binário	11101101	10110001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CPIR	Z,P/V,S,N,H	2	5	21

BC ≠ 0 e A ≠ (HL)

BC = 0 ou A = (HL)

Condição dos FLAGS

Ver CPR

43) CPL

Operação Simbólica

$$A \leftarrow \bar{A}$$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução complementa o conteúdo do acumulador ou, em outras palavras, o inverte "BIT a BIT".

Código de Operação

Hexadecimal	2F
Octal	57
Decimal	47
Binário	00101111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
CPL	N, H	1	1	4

Condição dos FLAGS

C	Não modificado
Z	Não modificado
P/V	Não modificado
S	Não modificado
N	SET em qualquer condição
H	SET em qualquer condição

Exemplo:

Acumulador	F3H	11110011
CPL		
Acumulador	0CH	00001100

44) DAA

Descrição:

Esta instrução é formada por um único BYTE, sendo es

te o código de operação. É utilizada quando se opera em aritmética decimal (BCD). Efetuando-se esta instrução depois de uma operação de soma ou subtração, o conteúdo do acumulador será ajustado automaticamente. Para melhor compreensão, acompanhar a tabela 7.2.

Instruções usadas anteriormente	CY antes do ajuste	H antes do ajuste	N antes do ajuste	Acumulador B7 B4	Acumulador B3 B0	Ajuste	CY depois do ajuste
	0	0	0	0-9	0-9	00	0
	0	0	0	0-8	A-F	06	0
ADD	0	1	0	0-9	0-3	06	0
ADC	0	0	0	A-F	0-9	00	1
INC	0	0	0	9-F	A-F	06	1
	0	1	0	A-F	0-3	06	1
	1	0	0	0-2	0-9	00	1
	1	0	0	0-2	A-F	FA	1
	1	1	0	0-3	0-3	A0	1
	0	0	1	0-9	0-9	00	0
	0	1	1	6-F	0-8	FA	0
	1	0	1	0-9	7-F	A0	1
	1	1	1	F-F	6-F	9A	1

Fig. 7.2 - Ajuste para BCD.

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DAA	C,Z,P/V,S,H	1	1	4

Código de Operação

Hexadecimal	27
Octal	47
Decimal	39
Binário	00100111

Condição dos FLAGS

C Ver tabela

Z SET se o acumulador for zero depois do "DAA"

P/V SET se paridade for par depois do "DAA"

S SET se o BIT 7 for um depois de "DAA"
 N Não é modificado
 H Ver tabela

45) DEC r

Operação Simbólica

$r \leftarrow r - 1$

Descrição:

Esta instrução é formada por um BYTE, sendo este o código de operação. Esta instrução decrementa de uma unidade o registrador r.

Código de Operação

Binário $00 _ r _ \rightarrow 101$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DEC r	Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

C Não modificado
 Z Seta se o ressetado for zero
 P/V Seta se o conteúdo do registrador for (80) Hex
 S Seta se o BIT 7 for "um"
 N Seta em qualquer condição
 H Seta se houve vai "um" do BIT 3

46) DEC (IX+d)

Operação Simbólica

$(IX+d) \leftarrow (IX+d) - 1$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação, e o terceiro um BYTE de dados. Esta instrução decrementa a posição de memória, endereça

da pelo registrador de índice "IX", em conjunto com o BYTE de da dos "d".

Código de Operação

Hexadecimal	DD	35	d
Octal	335	65	d
Decimal	221	53	d
Binário	11011101	00110101	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DEC(IX+d)	Z,P/V,S,N,H	3	6	23

Condição dos BITS

Ver DEC r

47) DEC (IY+d)

Operação Simbólica

$(IY+d) \leftarrow (IY+d) - 1$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação, e o terceiro um BYTE de da dos. Esta instrução decrementa a posição de memória, endereça da pelo registrador de índice "IY", em conjunto com o BYTE de da dos.

Código de Operação

Hexadecimal	FD	35	d
Octal	375	65	d
Decimal	253	53	d
Binário	11111101	00110101	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DEC(IY+d)	Z,P/V,S,N,H	3	6	23

Condição dos FLAGS

Ver DEC r

48) DEC (HL);

Operação Simbólica

$(HL) + (HL) - 1$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução decrementa de uma unidade a posição de memória endereçada pelo par de registradores (HL).

Código de Operação

Hexadecimal	35
Octal	65
Decimal	53
Binário	00110101

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DEC(HL)	Z,P/V,S,N,H	1	1	11

Condição dos FLAGS

Ver DEC r

49) DEC SS

Operação Simbólica

$SS + SS - 1$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução decrementa de uma unidade o conteúdo do par de registradores "SS". Um detalhe importante é que esta instrução não afeta nenhum dos FLAGS.

Código de Operação

Binário 00 + SS + 1011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DEC SS	nenhum	1	1	6

50) DEC IX

Operação Simbólica

IX ← (IX - 1)

Descrição:

Esta instrução é formada por dois BYTES, sendo eles o código de operação. Esta instrução decrementa o conteúdo do registrador de índice IX.

Código de Operação

Hexadecimal	DD	2B
Octal	335	53
Decimal	221	43
Binário	11011101	00101011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DEC IX	nenhum	2	2	10

51) DEC IY

Operação Simbólica

IY ← (IY - 1)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução decrementa o conteúdo do registrador de índice IY.

Código de Operação

Hexadecimal	FD	2B
Octal	375	53
Decimal	253	43
Binário	11111101	00101011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DEC IY	nenhum	2	2	10

52) DJNZ e

Operação Simbólica

$B \leftarrow (B - 1)$

$B = 0 \quad PC \leftarrow PC$

$B \neq 0 \quad PC \leftarrow (PC + e)$

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação, e o segundo, um BIT de dados. Ao se executar esta instrução, o registrador "B" será decrementado e o programa COUNTER modificado ($PC - e$), se o conteúdo do registrador "B" for diferente de zero; caso contrário, continua normalmente ou, em outras palavras, o Programa COUNTER "PC" não será modificado. Conforme se nota o registrador "B" funciona como contador, podendo-se, portanto, efetuar uma operação qualquer (n) vezes. Na figura 7.3, pode-se verificar o fluxo de dados.

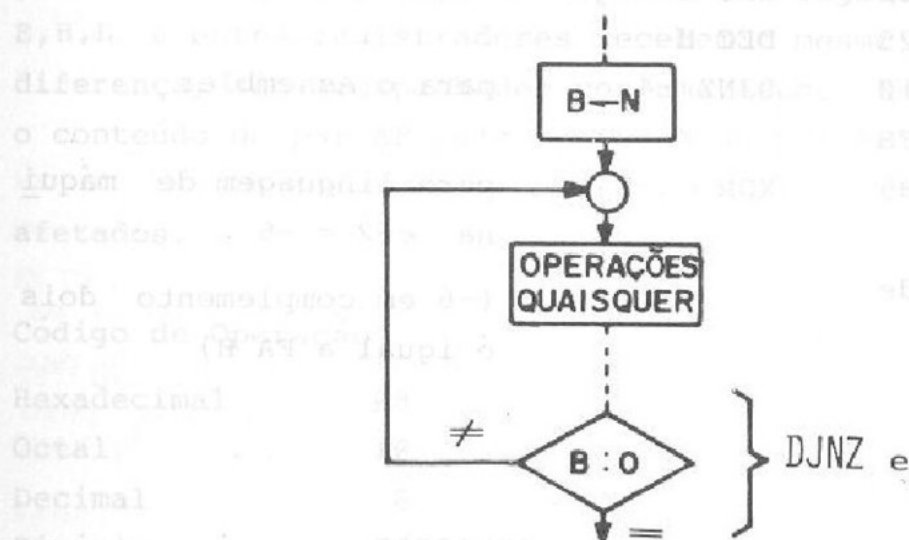


Fig. 7.3 - Estrutura da instrução DJNZ e.

Código de Operação

Hexadecimal	10	e-2
Octal	20	e-2
Decimal	16	e-2
Binário	00010000	e-2

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
DJNZ e	nenhum	2	2	8
			3	13

Se $B = 0$

Se $B \neq 0$

Obs: é utilizado (e-2) para se ter o valor em linguagem de máquina. O Assembler utiliza como referência a própria instrução, que no caso é "DJNZ"; já a linguagem de máquina utiliza como referência a próxima instrução, que no caso é "XOR L".

Exemplo:

Assembler

-4	-6	06	LD B, 87 H	
-3	-5	87		
-2	-4	10	INC E	
-1	-3	25	DEC H	
0	-2	10	DJNZ	-4 para o Assembler
1	-1	FB		
2	0	85	XOR L	para linguagem de máquina na $e-2 = -6$ (-6 em complemento dois é igual a FA H)

53) EI

Operação Simbólica

IFF1 + 1

IFF2 ← 1

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução habilitará a interrupção mascarada, setando os FLIP FLOPS interno, IFF1 e IFF2.

Código de Operação

Hexadecimal	FB
Octal	373
Decimal	251
Binário	11111011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
EI	nenhum	1	1	4

54) EX AF,AF'

Operação Simbólica

AF ↔ AF'

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Como foi visto anteriormente, o micro processador Z-80 possui um espelho dos registradores A,F,B,C,D,E,H,L e estes registradores recebem o mesmo nome, mas com uma diferença, são acompanhados por uma aspa. Esta instrução troca o conteúdo do par AF pelo conteúdo do par AF'.

Ao executar-se esta instrução, todos os FLAGS serão afetados.

Código de Operação

Hexadecimal	08
Octal	10
Decimal	8
Binário	00001000

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
EX AF,AF'	C,Z,P/V,S,N,H	1	1	4

55) EX DE,HL

Operação Simbólica

DE ↔ HL

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução troca o conteúdo do par de registradores "DE" com o conteúdo do par de registradores "HL".

Código de Operação

Hexadecimal	EB
Octal	353
Decimal	235
Binário	11101011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
EX DE,HL	nenhum	1	1	4

56) EX (SP),HL

Operação Simbólica

L + (SP)

H + (SP + 1)

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução troca o conteúdo do par de registradores "HL" com a posição de memória endereçada pelo STACK POINTER (SP).

Operação Simbólica

Código de Operação

Hexadecimal	E3	
Octal	343	
Decimal	227	
Binário	11100011	

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
EX (SP),HL	nenhum	1	5	19

57) EX (SP),IX

Operação Simbólica

IXL \leftarrow (SP)

IXH \leftarrow (SP + 1)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução troca o conteúdo do registrador de índice IX pelo posição de memória endereçada pelo STACK POINTER. A parte menos significativa de IX (IXL) é trocada com (SP) e a parte mais significativa de IX (IXH) é trocada com (SP + 1).

Código de Operação

Hexadecimal	DD	E3
Octal	335	343
Decimal	343	227
Binário	11011101	11100011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
EX DE,HL	nenhum	1	1	4

58) EX (SP),IY

Operação Simbólica

IYL \leftarrow (SP)

IYH \leftarrow (SP + 1)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução troca o conteúdo do registrador de índice IY pela posição de memória endereçada pelo STACK POINTER(SP). A parte menos significativa de IY (IYL) é trocada com (SP) e a parte mais significativa de IY (IYH) é trocada com (SP + 1).

Código de Operação

Hexadecimal	FD	E3
Octal	375	343
Decimal	253	227
Binário	11111101	11100011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
EX(SP), IY	nenhum	2	6	23

59) EXX

Operação Simbólica

BC ↔ BC'

DE ↔ DE'

HL ↔ HL'

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução troca o conteúdo dos pares de registradores BC, DE, HL pelo conteúdo dos pares de registradores BC', DE', HL'. As instruções EX AF, AF' E EXX são usadas para resguardar o conteúdo dos registradores dentro do próprio microprocessador.

Código de Operação

Hexadecimal	D9
Octal	331
Decimal	217
Binário	11011001

Parâmetros				
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
EXX	nenhum	1	1	4

60) HALT

Operação Simbólica

CPU em Retenção

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Ao se executar esta instrução a CPU entra num estado de retenção. Este estado fará com que a CPU pare de executar qualquer outra instrução após esta. A CPU pode sair deste estado de duas maneiras: a primeira seria por interrupção e a segunda por um RESET. Os registradores e os STATUS não são afetados e, nesta condição, o refrescamento das memórias dinâmicas continuará operando normalmente.

Código de Operação

Hexadecimal	76
Octal	166
Decimal	118
Binário	01110110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
HALT	nenhum	1	1	4

61) IM 0

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução coloca a CPU no modo "Zero", isto é, quando a linha \overline{INT} é colocada em nível lógico "zero", provocará um salto no programa para uma posição, dependendo do código colocado no DATA BUS. Pela figura 7.4, pode-se observar os códigos e as posições de saltos.

CÓDIGO NO BUS DE DADOS	ENDEREÇOS DE SALTOS
C7H	0000H
CFH	0008H
D7H	0010H
DFH	0018H
E7H	0020H
EFH	0028H
F7H	0030H
FFH	0038H

Fig. 7.4 - Código das posições dos saltos.

Código de Operação

Hexadecimal	ED	46
Octal	335	106
Decimal	237	070
Binário	11101101	01000110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IM ϕ	nenhum	2	2	8

62) IM 1

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução coloca a CPU no modo "UM", isto é, quando a linha $\overline{\text{INT}}$ é colocada em nível lógico "zero", provoca um salto no programa sempre para o endereço (0038 H), não sendo preciso colocar nenhuma informação no DATA BUS.

Código de Operação

Hexadecimal	ED	56
Octal	335	126
Decimal	237	86
Binário	11101101	01010110

Parâmetros				
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IM 1	nenhum	2	2	8

63) IM 2

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução coloca a CPU no modo "Dois", isto é, quando a linha INT é colocada em nível lógico "zero", provoca um salto no programa. Este salto é feito para posições sucessivas de memória, sendo a parte menos significativa em (M) e a parte mais significativa em (M+1). Esta posição de memória será endereçada pelo registrador I, e por um código colocado no BUS de dados, sendo necessário apenas 7 BITS, sendo que o BIT "zero" deverá ser sempre "zero".

REGISTRADOR I 7 BITS Periférico 0

Código de Operação

Hexadecimal	ED	5E
Octal	355	136
Decimal	237	94
Binário	111011101	01011110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IM 2	nenhum	2	2	8

64) IN A, (n)

Operação Simbólica

A ← (n)

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de máquina e o segundo um BYTE de Dados, que indicará o endereço do dispositivo de entrada e saída. Esta ins

trução colocará no conteúdo do acumulador o conteúdo do dispositivo de entrada e saída, endereçado pelo segundo BYTE da instrução.

Código de Operação

Hexadecimal	DB	n
Octal	333	n
Decimal	219	n
Binário	11011011	n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IN (A,n)	nenhum	2	3	10

65) IN r,(C)

Operação Simbólica

$r \leftarrow (C)$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução coloca no conteúdo do registrador (r) no conteúdo do dispositivo de entrada e saída, endereçado pelo registrador "C".

Código de Operação

Binário 11101101 01 $\leftarrow r \leftarrow$ 000

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IN r,(C)	Z,P/V,S,N,H	2	3	11

Condição dos FLAGS

C	Não modificado
Z	Seta se for "zero"
P/V	Seta se for par
S	Seta se o BIT 7 for "um"

N Reseta em qualquer condição
H Reseta em qualquer condição

66) INC (HL)

Operação Simbólica

(HL) + (HL) + 1

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução incrementa de uma unidade a posição de memória endereçada pelo par de registradores "HL".

Código de Operação

Hexadecimal 34
Octal 64
Decimal 52
Binário 00110100

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INC r	Z,P/V,S,N,H	1	3	11

Condição dos FLAGS

Ver INC r

67) INC r

Operação Simbólica

r + r + 1

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução incrementa de uma unidade o conteúdo dos registradores (A,B,C,D,E,H,L,).

Código de Operação

Binário $00 \leftarrow r \rightarrow 100$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de máq.	Estados
INC r	Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver DEC r

68) INC (IX+d)

Operação Simbólica

$(IX+d) \leftarrow (IX+d) + 1$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação e o terceiro um BYTE de dados. Esta instrução decrementa a posição de memória endereçada pelo registrador de índice "IX", em conjunto com o BYTE de dados.

Código de Operação

Hexadecimal	DD	34	d
Octal	335	64	d
Decimal	221	052	d
Binário	11011101	00110100	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INC(IX+d)	Z,P/V,S,N,H	3	6	23

Condição dos FLAGS

Ver DEC r

69) INC (IY+d)

Operação Simbólica

$(IY + d) \leftarrow (IY + d) + 1$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação e o terceiro um BYTE de dados. Esta instrução decrementa a posição da memória endereçada pelo registrador de índice "IX", em conjunto com o BYTE de dados.

Código de Operação

Hexadecimal	FD	34	d
Octal	375	64	d
Decimal	253	52	d
Binário	11111101	00110100	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INC(IX d)	Z,P/V,S,N,H	3	6	23

Condição dos FLAGS

Ver DEC r

70) INC SS

Operação Simbólica

SS ← SS + 1

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução incrementa de uma unidade o conteúdo do par de registradores "SS" e não afeta nenhum FLAG.

Código de Operação

Binário 00 ← SS → 0011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INC SS	nenhum	1	1	6

71) INC IX

Operação Simbólica

$IX \leftarrow (IX + 1)$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução incrementa de uma unidade o conteúdo do registrador de índice IX.

Código de Operação

Hexadecimal	DD	23
Octal	335	43
Decimal	221	35
Binário	11011101	001000011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INC IX	nenhum	2	2	10

72) INC IY

Operação Simbólica

$IY \leftarrow (IY + 1)$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução incrementa o conteúdo do registrador de índice "IY".

Código de Operação

Hexadecimal	FD	23
Octal	375	43
Decimal	253	35
Binário	11111101	001000011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INC IY	nenhum	2	2	10

73) IND

Operação Simbólica

(HL) ← (C)

HL ← (HL - 1)

B ← (B - 1)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega o conteúdo de um dispositivo de entrada e saída, endereçado pelo registrador "C", em uma posição de memória endereçada pelo par de registradores "HL". O registrador B é utilizado como contador, que é decrementado a cada execução desta instrução; já o par de registradores HL é decrementado para carregamento de outros dados.

Código de Operação

Hexadecimal	ED	AA
Octal	355	252
Decimal	237	170
Binário	11101101	10101010

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IND	Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

C Não modificado

Z Seta se B = 0

P/V Não determinado

S Não determinado

N Setado em qualquer condição

H Não determinado

Ex: Existe um dispositivo de entrada e saída cujo endereço é 5BH. A entrada deste dispositivo varia com o tempo. Desejando-se ler quatro vezes este canal, na execução de um programa qualquer, carrega-se o contador com 04H, o que no caso é o registrador "B", o par de registradores HL com o endereço de memória, a ser carregado os dados 72DAH, e o registrador "C" com o endereço do dispositivo 5BH.

IND
: FLAG Z = 0

IND
: FLAG Z = 0

IND
: FLAG Z = 0

IND
FLAG Z = 1

72D7
72D8
72D9
72DA

27
14
BA
75

Dispositivo
de entrada
e saída

75 H
BA H
14 H
27 H

endereço (5BH)

74) INDR

Operação Simbólica

(HL) ← (C)

HL ← (HL - 1)

B ← (B - 1)

PC ← PC - 2 se B ≠ 0

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução segue os mesmos parâmetros da IND, entra em LOOP, fazendo as mesmas operações, e sairá deste quando o contador chegar ao fim ou, em outras palavras, se o registrador "B" for igual a "zero".

Código de Operação

Hexadecimal ED BA

Octal	355	272
Decimal	237	186
Binário	11101101	10111010

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INDR	Z,P/V,S,N,H	2	4	16
			5	21

se $B = \emptyset$

se $B \neq \emptyset$

Condição dos FLAGS

C	Não afetado
Z	SET em qualquer condição
H	Não determinado
P/V	Não determinado
N	SET em qualquer condição
H	Não determinado

75) INI

Operação Simbólica

$(HL) \leftarrow (C)$

$HL \leftarrow (HL + 1)$

$B \leftarrow (B - 1)$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à instrução IND, exceto que o conteúdo do par de registrador "HL" é incrementado em vez de decrementado.

Código de Operação

Hexadecimal	ED	A2
Octal	355	242
Decimal	237	162
Binário	11101101	10100010

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INI	Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver INDR

76) INIR

Operação Simbólica

(HL) + (C)

HL + (HL + 1)

B + (B - 1)

PC + PC - 2 se B = 0

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à instrução INDR, exceto que o conteúdo do par de registradores "HL" é incrementado em vez de decrementado.

Código de Operação

Hexadecimal	ED	B2
Octal	355	262
Decimal	237	178
Binário	11101101	10110010

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INIR	Z,P/V,S,N,H	2	4	16
			5	21

se B = 0

se B ≠ 0

Condição dos FLAGS

Ver INDR

77) JP CC,nn

CC Representam os FLAGS

CC	Condição	Código	FLAGS
NZ	Se não zero	000	Z
Z	Se zero	001	Z
NC	Se não CARRY	010	C
C	Se CARRY	011	C
PO	Se paridade ímpar	100	P/V
PE	Se paridade par	101	P/V
P	Se sinal positivo	110	S
M	Se sinal negativo	111	S

Operação Simbólica

PC ← nn se CC for verdadeiro

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação, e os dois últimos o endereço que irá modificar o Programa COUNTER (PC). Esta instrução é utilizada para dar saltos condicionais, no transcorrer de um programa, isto é, antes de realizar estes saltos, verificam-se as condições dos FLAGS; se favoráveis, o salto é realizado, caso contrário, o salto é ignorado.

Código de Operação

Binário 11 ← cc → 010 nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
JP CC,nn	nenhum	3	3	10

Exemplo:

ADD	A,C	Se após a operação de soma do
JC	MICRO	acumulador com o registrador C,
		houver CARRY, o programa reali
MICRO: LD	A,C	zará o salto, caso contrário,
		não haverá salto.

78) JP (HL)

Operação Simbólica

PC ← (HL)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução realiza um salto incondicional, especificado pelo conteúdo do par de registradores "HL".

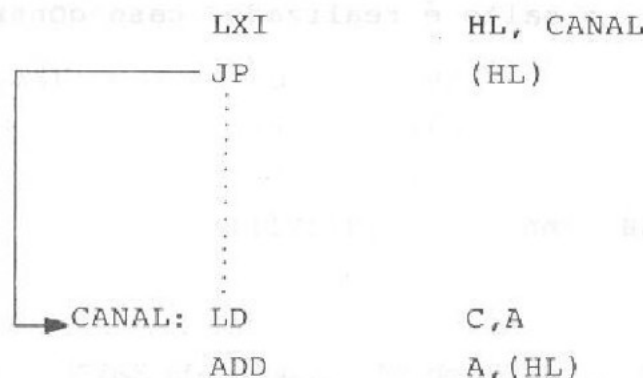
Código de Operação

Hexadecimal	E9
Octal	351
Decimal	233
Binário	11101001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
JP (HL)	nenhum	1	1	4

Exemplo:



79) JP (IX)

Operação Simbólica

PC ← (IX)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução realiza um salto incondi

cional, especificado pelo conteúdo do registrador de índice "IX".

Código de Operação:

Hexadecimal	DD	E9
Octal	335	351
Decimal	221	233
Binário	11011101	11101001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
JP (IX)	nenhum	2	2	8

80) JP (IY)

Operação Simbólica

PC ← (IY)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução realiza um salto incondicional, especificado pelo conteúdo do registrador de índice "IY".

Código de Operação

Hexadecimal	FD	E9
Octal	375	351
Décimal	253	233
Binário	11111101	11101001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
JP (IY)	nenhum	2	2	8

81) JP nn

Operação Simbólica

PC ← (nn)

Descrição:

Esta instrução é formada por três BYTES, sendo o "pri^{meiro} o código de operação e os dois últimos o endereço que irá modificar o Programa COUNTER (PC). Este tipo de instrução é chamado de incondicional, pois o salto independe de qualquer con^{dição}.

Código de Operação

Hexadecimal	C3	nn
Octal	303	nn
Decimal	195	nn
Binário	11000011	nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
JP nn	nenhum	3	3	10

82) JR e

Operação Simbólica

PC ← PC + e

Descrição:

Esta instrução é formada por dois BYTES, sendo o pri^{meiro} o código de operação e o segundo um BYTE de dados. Esta instrução efetua um salto relativo, isto é, somando-se ou sub^{traindo-se do Programa COUNTER (PC), o salto ocorre incondicio^{nalmente}, não depende das condições dos FLAGS.}

Condição de Operação

Hexadecimal	18	e-2
Octal	30	e-2
Decimal	24	e-2
Binário	00011000	e-2

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
JR e	nenhum	2	3	12

Exemplo:

Referência

			Assembler	Linguagem de máquina
	CB 84	RES	0,H	
Assem.	18 07	JP	09	0
L.M.	41	LD	B,C	0
	70	LD	(HL),B	
	CD AB 77	CALL	NC,77ABH	
	CB DC	SET	3,H	
	CB 27	SLA	A	7

A grande vantagem desta instrução em relação à instrução "JMP" é que economiza um BYTE; mas existe uma restrição: o salto relativo pode ser realizado numa faixa de (-128 a 127).

83) JR cc,e

CC Representa condição dos FLAGS

cc	Condição	Código	FLAGS
NC	Se não zero	00	Z
Z	Se zero	01	Z
NC	Se não CARRY	10	C
C	Se CARRY	11	C

Operação Simbólica

$PC \leftarrow (PC + e)$ Se cc for verdadeiro

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação e o segundo um BYTE de dados, que efetuará o salto relativo. Esta instrução efetuará um salto relativo condicional, isto é, antes de efetuar o salto, verifica as condições dos FLAGS; se favoráveis, o salto é realizado, caso

contrário, não será efetuado.

Código de Operação

Binário 001 + cc + 000 e-2

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
JR cc,e	nenhum	2	2	7
			3	12
cc falso				
cc verdadeiro				

Exemplo:

Referência

Assembler Linguagem de Máquina

DD7717	LD (IX+17H),A
4F	LD (C,A)
6D	LD (L,L)
E5	PUSH HL
A0	AND B
Assemb. 28	FA JR Z,-4
L.M. 60	LD H,B

Como o salto é negativo, necessita achar o complemento dois do número.

Linguagem máquina	(-6)	00000110
Complemento dois	(FAH)	11111010

84) LD A,(TT)

"TT" representa os registradores BC e DE

Registradores (TT) Códigos

BC 0

DE 1

Operação Simbólica

A + (SS)

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução carrega no acumulador, o conteúdo da memória, endereçada pelo par de registradores "BC" ou "DE".

Código de Operação

Binário 000 ← TT → 1010

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD A, (TT)	nenhum	1	2	7

Exemplo:

Acumulador	ACH	1416
BC	1417H	1417
	LD A, (BC)	1418
Acumulador	F2H	

85) LD A, (nn)

Operação Simbólica

A ← (nn)

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação, e os outros dois BYTES o endereço da posição de memória. Esta instrução carrega no acumulador o conteúdo da posição de memória, endereçada por (nn).

Código de Operação

Hexadecimal	3A	nn
Octal	72	nn
Decimal	58	nn
Binário	00111010	nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD A,(nn)	nenhum	3	4	13

86) LD A,I

Operação Simbólica

A ← I

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega no acumulador o conteúdo do registrador de interrupção. Um detalhe importante é que, ao efetuar esta instrução, os FLAGS são afetados.

Código de Operação

Hexadecimal ED 57

Octal 355 127

Decimal 237 87

Binário 11101101 01010111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD A,I	Z,P/V,S,N,H	2	2	9

Condição dos FLAGS

C Não modificado

Z SET se for zero

P/V SET se FLAG de interrupção IFF2 estiver setado

S SET se o BIT 7 for "um"

N RESET em qualquer condição

H RESET em qualquer condição

87) LD A,R

Operação Simbólica

Descrição:

Esta instrução é formada por dois BYTES, sendo estes

o código de operação. Esta instrução carrega, no acumulador, o conteúdo do registrador de REFRESH. Esta instrução é usada quando se lida com memórias dinâmicas, que necessitam de REFRESH. Um detalhe importante é que, ao efetuar esta instrução, os FLAGS são afetados.

Código de Operação

Hexadecimal	ED	5F
Octal	355	137
Decimal	237	95
Binário	11101011	01011111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD A,R	Z,P/V,S,N,H	2	2	9

Condição dos FLAGS

Ver LD A,I

88) LD (TT),A

SS representa os registradores BC e DE

Registradores (TT)	Códigos
BC	0
DE	1

Operação Simbólica

(TT) + A

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução carrega o conteúdo do acumulador no conteúdo de memória, endereçada pelo par de registradores "BC ou DE".

Código de Operação

Binário 000 + TT + 0010

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (TT),A	nenhum	1	2	7

89) LD SS,nn

Operação Simbólica

SS ← nn

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação e os dois últimos o valor que será carregado nos pares de registradores. Esta instrução carrega, no conteúdo dos pares de registradores "SS", o valor dos BYTES "nn".

Código de Operação

Binário 00 + SS + 0001 nn

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclo de Máq.	Estados
LD SS,nn	nenhum	3	3	10

Exemplo:

Registrador BC + 7DF9H
LD BC,6A04H

Registrador BC + 6A04H

90) LD SS,(nn)

SS representa os registradores BC, DE, SP

Registrador SS Código

BC	00
DE	10
SP	11

Operação Simbólica

SSH ← (nn + 1)

SSL ← (nn)

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros o código de operação, e os dois últimos indicam uma posição de memória. Esta instrução carrega, no conteúdo do par de registradores mais significativo, o conteúdo da memória endereçada por (nn + 1), e carrega, no conteúdo do par de registradores menos significativo, o conteúdo da memória endereçada por (nn).

Código de Operação

Binário 11101101 01 ← SS → 1011 nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD SS, (nn)	nenhum	4	6	20

Registrador DE → 7AF9H

LD DE, (ACDEH)

ACDE

14

ACDF

2A

Registrador DE → 2A14H

91) LD (HL), n

Operação Simbólica

(HL) ← n

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação e o segundo um BYTE de dados. Esta instrução carrega o dado (n) no conteúdo da memória, endereçada pelo par de registradores (HL).

Código de Operação

Hexadecimal 36 n

Octal	66 n
Decimal	54 n
Binário	00110110 n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (HL),n	nenhum	2	3	10

92) LD HL,(nn)

Operação Simbólica

HHL + (nn + 1)

LHL + (nn)

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação; os dois últimos indicam uma posição de memória. Esta instrução carrega no registrador H, o conteúdo de memória endereçada por (nn + 1), e carrega no registrador L, o conteúdo de memória endereçada por (nn).

Código de Operação

Hexadecimal	2A nn
Octal	52 nn
Decimal	42 nn
Binário	00101010 nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD HL,(nn)	nenhum	3	5	16

92) LD (HL),r

Operação Simbólica

(HL) ← r

Descrição:

Esta instrução é formada por dois BYTES, sendo estes

o código de operação. Esta instrução carrega o conteúdo do registrador "r", na posição de memória endereçada pelo par de registradores HL.

Código de Operação

Binário 01110 $\leftarrow r \rightarrow$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (HL), r	nenhum	1	2	7

93) LD I, A

Operação Simbólica

I \leftarrow A

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega no registrador de interrupção o conteúdo do acumulador.

Código de Operação

Hexadecimal	ED	47
Octal	355	107
Decimal	237	71
Binário	11101011	01000111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD I, A	nenhum	2	2	9

94) LD T, nn

T representa os registradores de índice IX e IY

Registro (T)	Código
--------------	--------

IX	0
IY	1

Operação Simbólica

IX ← nn

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros o código de operação e, os dois últimos, BYTES de dados. Esta instrução carrega no registrador IX ou IY, o dado "nn".

Código de Operação

Binário 11 + T + 11101 00100001 nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD T, nn	nenhum	4	4	14

95) LD T, (nn)

T representa os registradores de índice IX e IY

Registrador (T) Código

IX

0

IY

1

Operação Simbólica

TH ← (nn + 1)

TL ← (nn)

Descrição:

Esta instrução é formada por dois BYTES, sendo os dois primeiros o código de operação; os dois últimos indicam uma posição de memória. Esta instrução carrega, no registrador mais significativo, o conteúdo da memória endereçada por (nn + 1), e carrega, no registrador menos significativo, o conteúdo da memória endereçada por (nn).

Código de Operação

Binário 11 + T + 11101 00101010 nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD T,(nn)	nenhum	4	6	20

96) LD (T + d),n

T representa os registradores de índice IX e IY

Registrador T	Código
IX	0
IY	1

Operação Simbólica

(T + d) ← n

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros o código de operação; o terceiro indica uma posição de memória junto com o registrador de índice "IX" ou "IY"; o quarto e último BYTE indica o valor que será carregado na memória. Esta instrução carrega um dado "n", numa posição de memória endereçada pelo registrador de índice IX ou IY, mais o dado "d", que será somado ou subtraído do registrador de índice.

Código de Operação

Binário 11 + T + 11101 00110110 d n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (T+d),n	nenhum	4	5	19

Exemplo:

(IX -4) = (IX+FC)
 (IX -3) = (IX+FD)
 (IX -2) = (IX+FE)
 (IX -1) = (IX+FF)
 (IX+0)
 (IX+1)
 (IX+2)
 (IX+3)
 (IX+4)

4A
CD
7E
B9
14
39
FA
01
0F

LD (IX + 3), 0D7H

Posição de (IX + 3) após efetuar esta instrução ficará igual a "D7".

(IX + 2)	FA
(IX + 3)	D7
(IX + 4)	0F

97) LD (T + d), r

Operação Simbólica

(T + d) ← r

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação; o terceiro BYTE indica uma posição de memória junto com o registrador de índice "IX" ou "IY". Esta instrução carrega o conteúdo do registrador "r" numa posição de memória, endereçada pelo registrador de índice "IX" ou "IY", mais o dado "d", que será somado ou subtraído do registrador de índice.

Código de Operação

Binário 11 + T + 11101 01110 ← r → d

Parâmetro

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (T+d), r	nenhum	3	5	19

Exemplo:

(IY - 5) =	(IY + FB)	3A
(IY - 4) =	(IY + FC)	14
(IY - 3) =	(IY + FD)	F9
(IY - 2) =	(IY + FE)	04
(IY - 1) =	(IY + FF)	20
	(IY + 0)	IF
	(IY + 1)	CA
	(IY + 2)	ED

(IY + 3)

(IY + 4)

(IY + 5)

F0
BA
1A

LD (IY-3),C Registrador C → 05H

Posição de (IX-3) após efetuar esta instrução ficará igual a "05".

(IY - 4)

(IY - 3)

(IY - 2)

14
05
04

98) LD (nn),A

Operação Simbólica

(nn) ← A

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação; os dois últimos indicam uma posição de memória. Esta instrução carrega o conteúdo do acumulador em uma posição de memória, endereçada pelos BYTES (nn).

Código de Operação

Hexadecimal	32	nn
Octal	62	nn
Decimal	50	nn
Binário	00110010	nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (n,n)	nenhum	3	4	13

Exemplo:

Conteúdo do acumulador	36H
Conteúdo de memória	(79A1H)=4AH

LD (79A1H),A

Conteúdo do acumulador 36H
Conteúdo de memória (79A1H)=36H

99) LD (nn),SS

Operação Simbólica

(nn) ← SSL Registro menos significativo

(nn+1) ← SSH Registro mais significativo

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros os códigos de operação; os dois últimos indicam uma posição de memória. Esta instrução carrega o conteúdo do registrador menos significativo, na posição de memória (nn) e a parte mais significativa, na posição de memória (nn+1).

Código de Operação

Binário 11101101 01 ← SS → 0011 nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (nn),SS	nenhum	4	6	20

Exemplo:

Conteúdo do par de registradores BC = F0A3H

Conteúdo de memória (140CH) = 14H

Conteúdo de memória (140DH) = 5AH

LD (140CH),BC

Conteúdo do par de registradores BC = F0A3H

Conteúdo de memória (140CH) = A3

Conteúdo de memória (140DH) = F0

100) LD (nn),HL

Operação Simbólica

(nn) ← L

(nn+1) ← H

Descrição:

Esta instrução é formada por três BYTES, sendo o primeiro o código de operação; os dois últimos indicam uma posição de memória. Esta instrução carrega o conteúdo do registrador "L" na posição de memória (nn), e o conteúdo do registrador "H", na posição de memória endereçada por (nn+1).

Exemplo:

Código de Operação

Hexadecimal	22	nn
Octal	42	nn
Decimal	33	nn
Binário	001000010	nn

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (nn),HL	nenhum	4	5	16

101) LD (nn),T

T representa os registradores de índice IX e IY

Registrador	I	Código
IX		0
IY		1

Operação Simbólica

(nn) ← TL parte menos significativa

(nn+1) ← TH parte mais significativa

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros o código de operação; os dois últimos indicam uma posição de memória. Esta instrução carrega o conteúdo do registrador menos significativo, na posição de memória (nn), e a parte mais significativa, na posição de memória (nn + 1).

Código de Operação

Binário	11 + T + 11101	001000010	nn
---------	----------------	-----------	----

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD (nn),T	nenhum	4	6	20

102) LD R,A

Operação Simbólica

R ← A

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega o conteúdo do acumulador no registrador "R". Este registrador (R) é chamado de REFRESH, e é usado apenas quando existirem no circuito memórias dinâmicas (RAM's).

Código de Operação

Hexadecimal	ED	4F
Octal	355	117
Decimal	237	79
Binário	11101101	01001111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD R,A	nenhum	2	2	9

103) LD r,(HL)

Operação Simbólica

r ← (HL)

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução carrega o conteúdo de posição, endereçada pelo par de registradores "HL", no registrador "r".

Código de Operação

Binário $01 \leftarrow \underline{r} \rightarrow 110$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD r, (HL)	nenhum	1	2	7

Exemplo:

Conteúdo do par de registradores HL = 7BAFH
 Conteúdo do registrador C = DCH

	7BAF	BC
	7BAF	2A
	7BBC	7A

LD C, (HL)

Conteúdo do par de registradores HL = 7BAFH
 Conteúdo do registrador C = 2A

104) LD r, (T + d)

Operação Simbólica

$r \leftarrow (T + d)$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação; o terceiro indica uma posição de memória junto com o registrador de índice "IX" ou "IY". Esta instrução carrega o conteúdo da posição de memória, endereçada pelo registrador de índice "IX" ou "IY", mais o dado "d", que será somado ou subtraído do registrador de índice, no registro "r".

Código de Operação

Binário $11 \leftarrow \underline{T} \rightarrow 11101 \quad 01 \leftarrow \underline{r} \rightarrow 110 \quad d$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de máq.	Estados
LD r,(T+d)	nenhum	3	5	19

Exemplo:

(IX - 5) = (IX + FA)

(IX - 4) = (IX + FC)

(IX - 3) = (IX + FD)

(IX - 2) = (IX + FE)

(IX - 1) = (IX + FF)

(IX + 0)

(IX + 1)

(IX + 2)

(IX + 3)

(IX + 4)

(IX + 5)

36
44
52

A1
BA
0F
DE
F1
CA
14
34
44
AF
CD

Conteúdo do registrador B = 36

LD B,(IX + 4)

Conteúdo do registrador B = AF

105) LD r,n

Operação Simbólica

r ← n

Descrição:

Esta operação é formada por dois BYTES, sendo o primeiro o código de operação e o segundo um BYTE de dados. Esta instrução carrega no conteúdo do registrador "r" o valor do BYTE de dados (n).

Código de Operação

Binário 00 ← r → 110 n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD r,n	nenhum	2	2	7

Exemplo:

Conteúdo dos registrador H = C0H

LD H,1CH

Conteúdo do registrador H = 1CH

106) LD r,r'

Operação Simbólica

$r \leftarrow r'$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução carrega o conteúdo do registrador r', no registrador r.

Código de Operação

01 $\leftarrow r \rightarrow \leftarrow r' \rightarrow$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD r,r'	nenhum	1	1	4

Exemplo:

Conteúdo do registrador B = 27H

Conteúdo do registrador D = A5H

LD B,D

Conteúdo do registrador B = A5H

Conteúdo do registrador D = A5H

107) LD SP,HL

Operação Simbólica

$SP \leftarrow HL$

Descrição:

Esta instrução é formada por um único BYTE, sendo es

te o código de operação. Esta instrução carrega o conteúdo do par de registradores "HL" no registrador STACK POINTER "SP".

Código de Operação

Hexadecimal	F9
Octal	371
Decimal	249
Binário	11111001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD SP,HL	nenhum	1	1	6

Exemplo:

Conteúdo do par de registradores HL = 16D9H

Conteúdo do registrador SP = 75FAH

LD SP,HL

Conteúdo do par de registradores HL = 16D9H

Conteúdo do registrador SP = 16D9H

108) LD SP,T

Operação Simbólica

SP, + T

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega o conteúdo do par de registradores "HL" no registrador de índice.

Código de Operação

Binário	11, + T + 11101	11111001
---------	-----------------	----------

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LD SP,T	nenhum	2	2	10

109) LDD

Operação Simbólica

(DE) ← (HL)

BC ← BC - 1

DE ← DE - 1

HL ← HL - 1

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega o conteúdo de memória endereçada pelo par de registradores "HL", na posição de memória endereçada pelo par de registradores "DE", e os pares de registradores BC, DE, HL, são decrementados de uma posição.

Código de Operação

Hexadecimal	ED	A8
Octal	355	250
Decimal	237	168
Binário	11101101	10101000

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LDD	P/V,N,H	2	4	16

Condição dos FLAGS

C	Não modificado
Z	Não modificado
P/V	Seta se BC for diferente de "zero"
S	Não modificado
N	Reseta em qualquer condição
H	Reseta em qualquer condição

Exemplo:

0764

0765

0766

7E

F1

03

Conteúdo do par de registradores BC 0765 H
 Conteúdo do par de registradores DE 100D H
 Conteúdo do par de registradores HL ACF1 H

ACF1 **D5**

LDD

100D **L4**

Conteúdo do par de registradores BC 0764
 Conteúdo do par de registradores DE 100C
 Conteúdo do par de registradores HL ACFD

ACF1 **D5**

H = 0 P/V = 1 N = 0

100D **D5**

110) LDDR

Operação Simbólica

(DE) ← (HL)
 BC ← BC - 1
 DE ← DE - 1
 HL ← HL - 1
 PC ← PC - 2 se BC ≠ 0

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega o conteúdo da posição de memória endereçada pelo par de registradores "HL", na posição de memória endereçada pelo par de registradores "DE", e os pares de registradores BC, DE, HL, são decrementados de uma posição. O Programa COUNTER (PC) é decrementado de duas posições, se o conteúdo do par "BC" for diferente de zero.

Código de Operação

Hexadecimal	ED	B8
Octal	355	270
Decimal	237	184
Binário	11101101	10111000

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LDDR	P/V,N,H	2	5	21
			4	16

se BC \neq 0

se BC = 0

Condição dos FLAGS

C	Não modificado
Z	Não modificado
P/V	RESET em qualquer condição
S	Não modificado
N	RESET em qualquer condição
H	RESET em qualquer condição

Exemplo:

Conteúdo do par de registradores BC 0003H
 Conteúdo do par de registradores DE A173H
 Conteúdo do par de registradores HL 7315H

7313	10
7314	C9
7315	48

A171	FD
A172	37
A173	2A

LDDR

Conteúdo do par de registradores BC 0000H
 Conteúdo do par de registradores DE A170H
 Conteúdo do par de registradores HL 7312H

7313	10
7314	C9
7315	48

A171	10
A172	C9
A173	48

111) LDI

Operação Simbólica

(DE) \leftarrow (HL)

BC \leftarrow BC - 1

DE \leftarrow DE + 1

HL \leftarrow HL + 1

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "LDD", apenas os pares de registradores "DE" e "HL", são incrementados em vez de decrementados.

Código de Operação

Hexadecimal	ED	AØ
Octal	355	24Ø
Decimal	277	16Ø
Binário	111Ø11Ø1	1Ø1ØØØØØ

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LDI	P/V,N,H	2	4	16

Condição dos FLAGS

Ver LDD

112) LDIR

Operação Simbólica

(DE) \leftarrow (HL)

BC \leftarrow BC - 1

DE \leftarrow DE + 1

HL \leftarrow HL + 1

PC \leftarrow PC - 2 se BC \neq Ø

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "LDDR", apenas os pares de registradores "DE" e "HL" são incrementados em vez de decrementados.

Código de Operação

Hexadecimal	ED	B0
Octal	355	260
Decimal	237	176
Binário	11101101	10110000

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
LDIR	P/V,N,H	2	5	21
			4	16

se BC \neq 0

se BC = 0

Condição dos FLAGS

Ver LDDR

113) NEG

Operação Simbólica

$A \leftarrow 0 - A$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução executa a subtração do conteúdo do acumulador por zero ou, em outras palavras, é efetuado o complemento para dois do acumulador.

Código de Operação

Hexadecimal	ED	44
Octal	355	104
Decimal	237	68

Binário 11101101 01000100

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
NEG	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

C SET se o acumulador não era "zero" antes da operação

Z SET se o resultado for "zero"

P/V SET se o acumulador era "80H" antes da operação

S SET se o BIT 7 for igual a "1"

N SET em qualquer condição

H SET se "vem um" do BIT 4 para BIT 3

Exemplo:

Acumulador = 3A

NEG

		1		
0000		0000		(00H)
-	0011	1010		(3AH)
	1100	0110		(C6H)

Acumulador = C6 H

C = 1 Z = 0 P/V = 0 S = 1 N = 1 H = 1

114) NOP

Operação Simbólica

Nenhuma Operação

Descrição:

Esta instrução é formada por um BYTE, sendo este o código de operação. Esta instrução não altera os FLAGS registradores e posições de memórias, em outras palavras, é como se nenhuma instrução fosse executada.

Código de Operação

Hexadecimal 00

Octal 00

Decimal 00
Binário 00000000

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
NOP	nenhum	1	1	4

115) OR r

Operação Simbólica

A ← A ∨ r

Descrição:

Esta instrução é formada por um BYTE, sendo este o código de operação. Esta instrução executa a operação lógica "OR", entre o conteúdo do acumulador e o conteúdo do registrador "r", e o resultado é armazenado no acumulador.

Código de Operação

Binário 10110 ← _ r _ →

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OR r	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver AND r

Exemplo:

Conteúdo do acumulador = 7AH 0111 1010

Conteúdo do registrador B = C2H 0110 0010

1111 1010

OR B

Conteúdo acumulador = FAH

Conteúdo registrador B = C2H

C = 0 Z = 0 P/V = 1 S = 1 N = 0 H = 1

116) OR (HL)

Operação Simbólica

A ← A ∨ (HL)

Descrição:

Esta instrução é formada por um BYTE, sendo este o código de operação. Esta instrução executa a operação lógica "OR", entre o conteúdo do acumulador e o conteúdo de memória, endereçada pelo par de registradores "HL", e o resultado é armazenado no acumulador.

Código de Operação

Hexadecimal	B6
Octal	266
Decimal	182
Binário	10110110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OR (HL)	C,Z,P/V,S,N,H	1	2	7

Condição dos FLAGS

Ver AND r

117) OR (T + d)

Operação Simbólica

A ← A ∨ (T + d)

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação e o terceiro um BYTE de dados, que será somado ou subtraído do registrador de índice "IX" ou "IY". Esta instrução executa a lógica "OR", entre o acumulador e a posição indicada pelo registrador de índice "IX" ou

"IY", com o seu respectivo BYTE de dados.

Código de Operação

Binário 11 + T + 11101 10110110 a

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OR (T+d)	C,Z,P/V,S,N,H	3	5	19

Condição dos FLAGS

Ver AND r

118) OUT (C),r

Operação Simbólica

(C) + B

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução coloca no registrador (r), o conteúdo do dispositivo de entrada e saída, endereçada pelo registrador "C".

Código de Operação

Binário 11101101 01 + r + 001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OUT (C),r	nenhum	2	3	12

119) OUTD

Operação Simbólica

(C) + (HL)

HL + HL - 1

BC + BC - 1

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução carrega o conteúdo de uma posição de memória, endereçada pelo par de registradores "HL", num dispositivo de entrada e saída endereçado pelo registrador "C". O registrador "B" é utilizado como contador, e é decrementado a cada execução desta instrução; já o par de registradores "HL" é decrementado para carregamento de outros dados.

Código de Operação

Hexadecimal	ED	AB
Octal	355	253
Decimal	237	171
Binário	11101101	10101011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OUTD	Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Z	SET se o registrador B = 0
C	Não modificado
P/V	Não determinado
S	Não determinado
N	SET em qualquer condição
H	Não determinado

120) OUTI

Operação Simbólica

(C) ← (HL)
BC ← BC - 1
HL ← HL + 1

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à instrução

OUTD, exceto que o conteúdo do par de registradores "HL" é incrementado em vez de decrementado.

Código de Operação

Hexadecimal	ED	A3
Octal	355	243
Decimal	237	163
Binário	11101101	10100011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OUTI	Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver OUTD

121) OTDR

Operação Simbólica

(C) ← (HL)

B. ← B - 1

HL ← HL + 1

PC ← PC - 2 se B ≠ 0

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução segue os mesmos parâmetros da instrução "OUTD". Entrará em LOOP, fazendo as mesmas operações da instrução OUTD, e sairá deste LOOP, quando o contador chegar ao fim ou, em outras palavras, se o registrador "B" for igual a zero.

Código de Operação

Hexadecimal	ED	BB
Octal	355	272
Decimal	237	186
Binário	11101101	10111011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OTDR	Z,P/V,S,N,H	2	4	16
			5	21

se B \neq 0

se B = 0

Condição dos FLAGS

C Não modificado

Z SET em qualquer condição

H Não determinado

P/V Não determinado

N Em qualquer condição

H Não determinado

122) OTIR

Operação Simbólica

(C) \leftarrow (HL)

HL \leftarrow HL + 1

B \leftarrow B - 2

PC \leftarrow PC - 2 se B \neq 0

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à instrução "OUTI". Entrará em LOOP, fazendo as mesmas operações da instrução OUTD, e sairá deste LOOP quando o contador chegar ao fim.

Código de Operação

Hexadecimal	ED	B3
Octal	355	263
Decimal	237	179
Binário	11101101	10110011

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
OTIR	Z,P/V,S,N,H	2	4	16
			5	21

se $B = \emptyset$

se $B \neq \emptyset$

Condição dos FLAGS

Ver OTDR

123) POP DD

Operação Simbólica

DDL \leftarrow (SP)

DDH \leftarrow (SP + 1)

SP \leftarrow (SP + 2)

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução carrega, na parte menos significativa de DD (DDL), o conteúdo de memória endereçada pelo STACK POINTER (SP); e carrega na parte mais significativa de DD (DDH), o conteúdo de memória endereçada pelo STACK POINTER mais "um" (SP + 1); por último, o conteúdo do registrador "SP" incrementado de duas posições. Esta instrução é usada para resguardo de registradores.

Obs.: A instrução "POP AF" afeta os FLAGS.

Código de Operação

Binário $11 \leftarrow \underline{DD} \rightarrow 0001$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
POP SS	nenhum	1	3	10

Exemplo:

Conteúdo do par de registradores HL = 017FH

Conteúdo do registrador SP ABL8H

POP HL

Conteúdo do par de registradores HL = 4913H

Conteúdo do registrador SP = ABLAH

ABL8	13
ABL9	49
ABLA	1A

124) POP T

Operação Simbólica

TL ← (SP)

TH ← (SP + 1)

SP ← SP + 2

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "POP SS", sendo utilizados os registradores de índice IX e IY.

Código de Operação

Binário $11 + T \rightarrow 11101$ 11100001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
POP T	nenhum	2	4	14

125) PUSH DD

Operação Simbólica

(SP - 1) ← H DD

(SP - 2) ← L DD

SP ← SP - 2

Descrição:

Esta instrução é formada por um único BYTE, sendo es

te o código de operação. Esta instrução carrega numa posição de memória, endereçada pelo registrador STACK POINTER menos "um" ($SP - 1$), o conteúdo da parte mais significativa de SS(DDH), carrega numa posição de memória endereçada pelo registrador STACK POINTER menos "dois" ($SP - 2$) e o conteúdo da parte menos significativa de DD (DDL); por último, o conteúdo do registrador "SP" decrementado de duas posições. Esta instrução é usada para recuperação de pares de registradores.

Código de Operação

Binário $11 \leftarrow \underline{DD} \rightarrow 0101$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
PUSH DD	nenhum	1	3	11

Exemplo:

Conteúdo do par de registradores BC = 07F4H

Conteúdo do registrador SP = FABCH

PUSH HL

Conteúdo do par de registradores BC = 07F4H

Conteúdo do registrador SP = FABAH

"SP"	FABA	F4
	FABB	07
	FABC	7A
	FABD	27

126) PUSH T

Operação Simbólica

$(SP - 1) \leftarrow TH$

$(SP - 2) \leftarrow TL$

$SP \leftarrow SP - 2$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "PUSH SS", sendo utilizados os registradores de índice "IX" e "IY".

Código de Operação

Binário 11 + T + 11101 11100101

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
PUSH T	nenhum	1	4	14

127) RES b,r

Operação Simbólica

$r_b \leftarrow 0$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução reseta, isto é, coloca a nível lógico "zero", um único BIT "b" de um registrador "r".

Código de Operação

Binário 11001011 10 + b + +r +

Parâmetros

mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RES b,r	nenhum	2	2	8

Exemplo:

Conteúdo do registrador B = 7AH 0111 1010

RES 6,B

Conteúdo do registrador B = 3AH 0011 1010

128) RES b,(HL)

Operação Simbólica

$(HL)_b \leftarrow 0$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução reseta o BIT "b" de uma posição de memória, endereçada pelo par de registradores "HL".

Código de Operação

Binário 11001011 $10 \leftarrow b \rightarrow 110$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RES b, (HL)	nenhum	2	4	15

129) RES b, (T + d)

Operação Simbólica

$(T + d)_b \leftarrow 0$

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto BYTE o código de operação; o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução reseta um BIT "b" de uma determinada posição de memória, endereçada pelo registrador de índice IX ou IY, com o seu respectivo BYTE de dados.

Código de Operação

Binário $11 \leftarrow T \rightarrow 11101$ 11001011 d $10 \leftarrow b \rightarrow 110$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RES b, (T+d)	nenhum	4	6	23

130) RET

Operação Simbólica

$PCL \leftarrow (SP)$
 $PCH \leftarrow (SP - 1)$
 $SP \leftarrow (SP - 2)$

Descrição

Esta instrução é formada por um único código de máquina, sendo esta o código de operação. Esta instrução colocará incondicionalmente, isto é, sem verificar as condições dos FLAGS, o conteúdo de memória endereçada pelo STACK POINTER (SP), na parte menos significativa do Programa COUNTER (PC), e carregará a posição de memória endereçada pelo registrador (SP + 1), na parte mais significativa do PC. Esta instrução é usada para indicar o fim de uma subrotina.

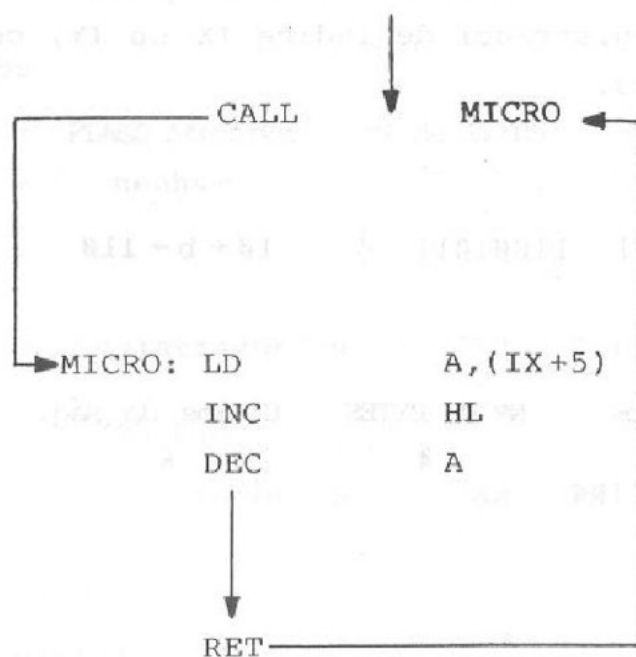
Código de Operação

Hexadecimal	C9
Octal	311
Decimal	201
Binário	11001001

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RET	nenhum	1	3	10

Exemplo:



131) RET CC

CC condição dos FLAGS

	Condição CC	Código	FLAGS
NZ	Se não zero	000	Z
Z	Se zero	001	Z
NC	Se não CARRY	010	C
C	Se CARRY	011	C
PO	Se paridade ímpar	100	P/V
PE	Se paridade par	101	P/V
P	Se sinal positivo	110	S
M	Se sinal negativo	111	S

Operação Simbólica

LPC ← (SP)

HPC ← (SP + 1)

SP ← (SP - 2)

SE ← CC for verdadeiro

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução é similar à RET, apenas, antes de ser efetuada, verificam-se as condições dos FLAGS; se favoráveis, esta será efetuada, caso contrário, será ignorada.

Código de Operação

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RET CC	nenhum	1	1	5 se for falso
			3	11 se for verdadeiro

132) RETI

Operação Simbólica

L PC ← (SP)

H PC \leftarrow (SP + 1)

SP \leftarrow (SP - 2)

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução executa, os mesmos parâmetros da instrução RET, sendo que esta instrução manda um sinal para o dispositivo que causou a interrupção, avisando que a interrupção foi executada.

Código de Operação

Hexadecimal	ED	4D
Octal	355	115
Decimal	237	77
Binário	11101101	01001101

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RETI	nenhum	2	4	14

133) RETN

Operação Simbólica

L PC \leftarrow (SP)

H PC \leftarrow (SP + 1)

SP \leftarrow (SP - 2)

IFF1 \leftarrow IFF2

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução executa os mesmos parâmetros da instrução "RET", sendo que é usada em interrupção não mascarada. O FLIP-FLOP "IFF2" contém a cópia de "IFF1", e após executar esta instrução, a informação de "IFF2" é colocada em "IFF1".

Código de Operação

Hexadecimal	ED	45
-------------	----	----

Octal	355	105
Decimal	237	69
Binário	11101101	010000101

Parâmetros

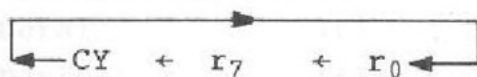
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RETN	nenhum	2	4	14

134) RL r

r representa os registradores B,C,D,E,H,L

Registrador	Código
B	000
C	001
D	010
E	011
H	100
L	101

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca de uma posição para a esquerda o conteúdo do registrador "r", sendo o BIT D7 carregado no "CARRY", e o conteúdo do CARRY, carregado no BIT D0.

Código de Operação

Binário	11001011	00010 ± r →
---------	----------	-------------

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RL r	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

C Carregamento com o valor do BIT "D7"
 Z SET se o resultado for igual a zero
 P/V SET se for par
 S SET se o BIT "D7" for igual a "zero"
 N RESET em qualquer condição
 H RESET em qualquer condição

Exemplo:

Conteúdo do registrador H DEH = 1011 1110
 CY = 0

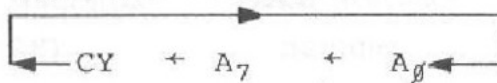
RL H

Conteúdo do registrador H 7CH = 0111 1100

CY = 1 Z = 0 P/V = 0 S = 1 N = 0 H = 0

135) RLA

Operação Simbólica



Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução é similar à "RL r", exceto que o acumulador é que é envolvido.

Código de Operação

Hexadecimal 17
 Octal 27
 Decimal 23
 Binário 00010111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RLA	C, N, H	1	1	4

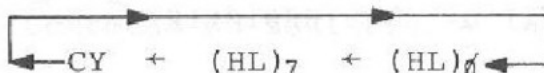
Condição dos FLAGS

C Carregamento com o valor do BIT "D7"

Z Não modificado
P/V Não modificado
S Não modificado
N RESET em qualquer condição
H RESET em qualquer condição

136) RL (HL)

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "RL r", excepto que o conteúdo de memória, endereçada pelo par "HL", é que sofrerá a operação.

Código de Operação

Hexadecimal	CB	16
Octal	313	26
Decimal	203	22
Binário	11001011	00010110

Parâmetros

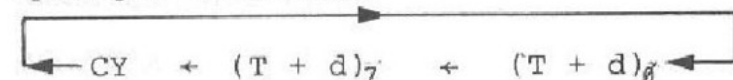
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RL (HL)	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver RL r

137) RL (T + d)

Operação Simbólica



Descrição:

Esta instrução é formada por quatro BYTES, sendo que os dois primeiros e o quarto representam o código de operação, e o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução é similar à "RL r", exceto que o conteúdo da memória endereçada pelo registrador de índice "IX ou IY", com o seu respectivo BYTE de dados, é quem sofre esta operação.

Código de Operação

Binário 11 + T → 1101 11001011 d 00010110

Parâmetros

Mnemônico	FLAGS Afetados	nº de BYTES	Ciclos de Máq.	Estados
RL (T+d)	Z,C,P/V,S,N,H	4	6	23

Condição dos FLAGS

Ver "RL r"

138) RLC r

Operação Simbólica

CY ← r₇ ← r₀

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca o conteúdo do registrador "r" de uma posição para a esquerda, sendo o BIT "D7" carregado no "CARRY" e o BIT "D0" e os demais, deslocados de uma posição. O conteúdo do CARRY é afetado após a execução desta instrução.

Código de Operação

Binário 11001011 00010 + _ r _ →

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RLC r	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

Ver RL r

Exemplo:

Conteúdo do registrador B \rightarrow 73H \rightarrow 01110011

CY = 1

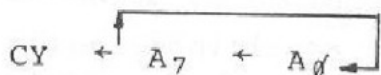
RLC B

Conteúdo do registrador B E6H \rightarrow 1110 0110

CY = 0 Z = 0 P/V = 0 S = 1 N = 0 H = 0

139) RLCA

Operação Simbólica



Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução é similar à "RLC r", exceto que é o acumulador quem será envolvido.

Código de Operação

Hexadecimal	07
Octal	07
Decimal	07
Binário	00000111

Parâmetros

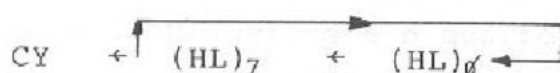
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RLCA	C, N, H	1	1	4

Condição dos FLAGS

Ver RLA

140) RLC (HL)

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "RLC r", exceto que o conteúdo de memória endereçada pelo par "HL" é quem sofre a operação.

Código de Operação

Hexadecimal	CB	06
Octal	313	6
Decimal	203	6
Binário	11001011	00000110

Parâmetros

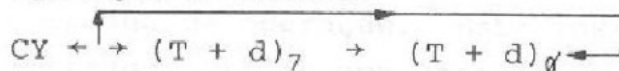
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RLC (HL)	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver RL r

141) RLC (T + d)

Operação Simbólica



Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto o código de operação; o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução é similar à "RLC r", exceto que o conteúdo de memória endereçada pelo registrador de índice IX ou IY, com seu respectivo BYTE de dados, é quem sofre esta operação.

Código de Operação

Binário	11 + <u>T</u> + 1101	11001011 d	00000110
---------	----------------------	------------	----------

Parâmetros

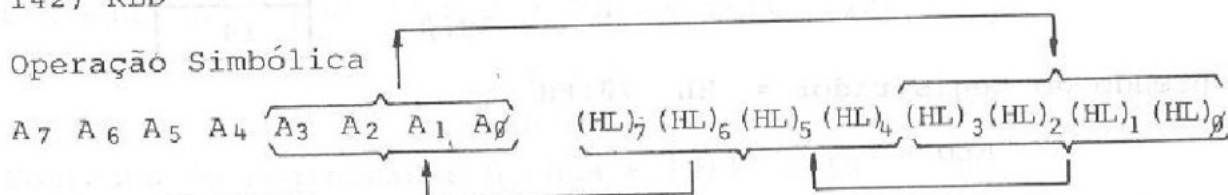
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RL (T + d)	Z,C,P/V,S,N,H	4	6	23

Código de Operação

Ver RL r

142) RLD

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca os quatro BYTES menos significativos da memória, endereçada pelo par "HL", para a parte mais significativa da mesma posição de memória. A parte mais significativa desta posição de memória é deslocada para a parte menos significativa do acumulador; por último, a parte menos significativa do acumulador é carregada na parte menos significativa da posição de memória, endereçada pelo par "HL".

Código de Operação

Hexadecimal	ED	6F
Octal	355	157
Decimal	237	111
Binário	11101101	01101111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RLD	Z,P/V,S,N,H	2	5	18

Condição dos FLAGS

C	Não afetado
Z	SET se o acumulador for "zero"

P/V SET se a paridade do acumulador for par
 S SET se o BIT 7 do acumulador for "um"
 N RESET em qualquer condição
 H RESET em qualquer condição

Exemplo:

Conteúdo do acumulador = 5EH 701E 6F
 701F A3
 7020 14

Conteúdo do Registrador = HL 701FH

RLD

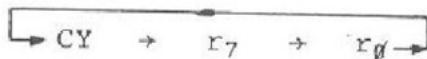
Conteúdo do acumulador 5AH 701E 6F
 701F 3E
 7020 14

Conteúdo do par de registradores HL 701FH

Z = 0 P/V = 1 S = 0 N = 0 H = 0

143) RR r

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca de uma posição para a direita, o conteúdo do registrador "r", sendo o "CARRY" carregado no BIT "D7", e o BIT "D0" carregado no "CARRY".

Código de Operação

Binário 11001011 00011 ← r →

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RR r	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

C	Carregamento com o valor do BIT "D0"
Z	SET se o resultado for igual a zero
P/V	SET se for par
S	SET se o BIT "D7" for igual a "zero"
N	RESET em qualquer condição
H	RESET em qualquer condição

Conteúdo do registrador D → 7CH = 0111 1100

CY = 1

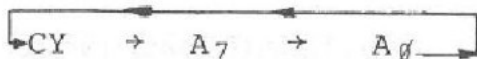
RR D

Conteúdo do registrador D → BEH = 1011 1110

CY = 0 Z = 0 P/V = 1 S = 1 N = 0 H = 0

144) RRA

Operação Simbólica



Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução é similar à "RR r", exceto que o acumulador será envolvido.

Código de Operação

Hexadecimal	1F
Octal	37
Decimal	31
Binário	00011111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRA	C, N, H	1	2	8

Condição dos FLAGS

C	Carregamento com o valor do BIT "D0"
---	--------------------------------------

Z Não modificado
P/V Não modificado
S Não modificado
N RESET em qualquer condição
H RESET em qualquer condição

145) RR (HL)

Operação Simbólica

$\text{CY} \rightarrow (\text{HL})_7 \rightarrow (\text{HL})_8 \rightarrow$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "RR r", exceto que o conteúdo de memória endereçada pelo par "HL" é quem sofrerá a operação.

Código de Operação

Hexadecimal	CB	1E
Octal	313	36
Decimal	203	30
Binário	11001011	00011110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RR (HL)	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver RR r

146) RR (T + d)

Operação Simbólica

$\text{CY} \rightarrow (\text{T} + \text{d})_7 \rightarrow (\text{T} + \text{d})_8 \rightarrow$

Descrição:

Esta instrução é formada por quatro BYTES, sendo os

dois primeiros e o quarto o código de operação; o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução é similar à "RR r", exceto que o conteúdo de memória endereçada pelo registrador de índice IX ou IY, com seu respectivo BYTE de dados, é quem sofre esta operação.

Código de Operação

Binário 11 \leftarrow T \rightarrow 11101 11001011 d 00011100

Parâmetros

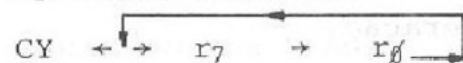
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RR (T + d)	Z,C,P/V,S,N,H	4	6	23

Condição dos FLAGS

Ver RR r

147) RRC r

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca de uma posição para a direita, o conteúdo do registrador "r", sendo o BIT "D0" carregado no CARRY e no BIT "D7", e os demais BITS, deslocados de uma posição. O conteúdo do CARRY BITS é perdido após a execução desta instrução.

Código de Operação

Binário 11001011 00001 \leftarrow _ r _ \rightarrow

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRC r	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

Ver RRR

Exemplo:

Conteúdo do registrador C \rightarrow A8H = 10101000

CY = 0

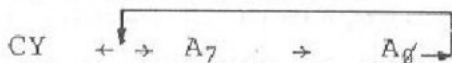
RLC C

Conteúdo do registrador C \rightarrow 54H = 0101 0100

CY = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 0

148) RRCA

Operação Simbólica



Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução é similar à "RLC r", exceto que o acumulador é quem sofre a operação.

Código de Operação

Hexadecimal 0F

Octal 17

Decimal 15

Binário 00001111

Parâmetros

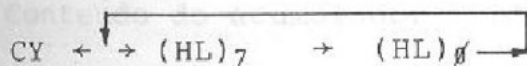
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRCA	C, N, H	1	1	4

Condição dos FLAGS

Ver RRA

149) RRC (HL)

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "RRC r", exce_{to} que o conteúdo de memória, endereçada pelo par "HL", é quem sofre a operação.

Código de Operação

Hexadecimal	CB	0E
Octal	313	16
Decimal	203	14
Binário	11001011	00001110

Parâmetros

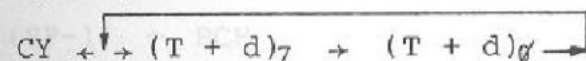
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRC (HL)	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver RR r

150) RRC (T + d)

Operação Simbólica



Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto o código de operação; o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução é similar à "RLC r", exceto que o conteúdo de memória, endereçada pelo registrador de índice IX ou IY, com seu respectivo BYTE de dados, é quem se envolve nesta operação.

Código de Operação

Binário	11 + T + 1101	11001011	d	00001110
---------	---------------	----------	---	----------

Parâmetros

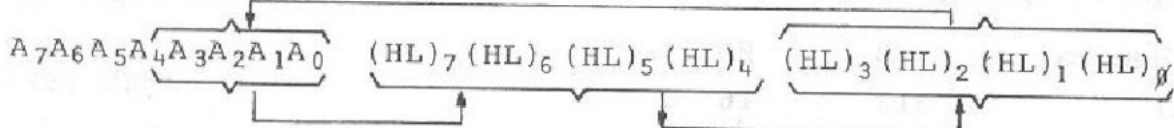
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRC (T+d)	Z,C,P/V,S,N,H	4	6	23

Código de Operação

Ver RR r

151) RRD

Operação Simbólica



Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca os quatro BITS menos significativos do acumulador, para a parte mais significativa de uma posição de memória, indicada pelo par de registradores "HL". A parte mais significativa desta posição é deslocada para a parte menos significativa desta mesma posição e, por último, a parte menos significativa desta posição de memória, indicada pelo par "HL", é deslocada para a parte menos significativa do acumulador.

Código de Operação

Hexadecimal	ED	67
Octal	355	147
Decimal	237	103
Binário	11101101	01100111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRD	Z,P/V,S,N,H	2	5	18

Condição dos FLAGS

Ver RLD

Exemplo:

Conteúdo do acumulador DAH

A31B

3E

Conteúdo do Registrador HL = A31BH

Parâmetros RLD

Conteúdo do acumulador = DEH

Conteúdo do par de registradores HL = A31BH

A31B

A3

152) RST P

P representa os endereços 0, 8, 10H, 18H, 20H, 28H, 30H, 38H.

Endereços	P	Código
0		000
8		001
10H		010
18H		011
20H		100
28H		101
30H		110
38H		111

Operação Simbólica

(SP-1) ← PCH

(SP-2) ← PCL

PCH ← 0

PCL ← P

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução apresenta o mesmo princípio da instrução "CALL", com apenas uma diferença: o contador de programa "PC" assume um endereço pré-determinado, tendo uma variedade de 8 endereços. Esta instrução pode ser utilizada quando, em um programa, uma subrotina é utilizada várias vezes. Em vez de chamar-se esta subrotina por CALL, que utiliza

três BYTES, aplica-se um "RSP", economizando-se dois BYTES.

Código de Operação

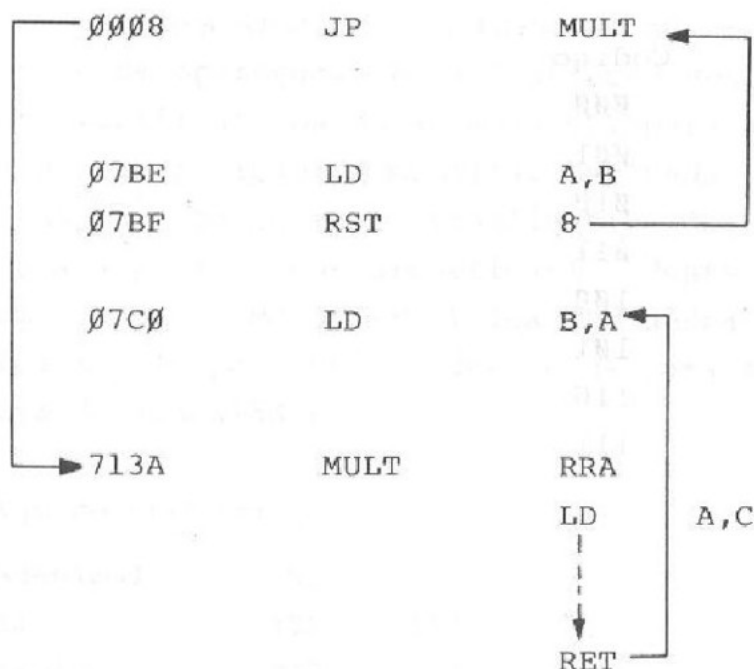
Binário 11 ← P → 111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RST P	nenhum	1	3	11

Exemplo:

Aplicação de RST P



153) SBC A,r

Operação Simbólica

$A \leftarrow A - r - CY$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução efetua a subtração entre o conteúdo do acumulador, o conteúdo do registrador "r" e o "CARRY", sendo o resultado carregado no próprio acumulador.

Código de Operação

10011 ← r →

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SBC A,r	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver CP n

Exemplo:

Conteúdo do acumulador A	→ 71H	=	0111	0001
Conteúdo do registrador B	→ 3AH	=	0011	1010
Conteúdo do CARRY	0 → 01H	= -	0000	0001
Resultado	36H	=	0011	0110

SBC A,B

Conteúdo do acumulador A → 36H
 Conteúdo do registrador B → 3AH

C = 0 Z = 0 P/V = 0 S = 0 N = 1 H = 1

154) SBC A,n

Operação Simbólica

A ← A - n - CY

Descrição:

Esta instrução é formada por dois BYTES, o primeiro é o código de operação, e o segundo um BYTE de dados. Esta instrução subtrai o conteúdo do acumulador com o BYTE de dados, menos o CARRY BIT, sendo o resultado carregado no acumulador.

Código de Operação

Hexadecimal	DE n
Octal	336 n

Decimal 222 n
Binário 11011110 n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SBC A,n	C,Z,P/V,S,N,H	2	2	7

Condição dos FLAGS

Ver CP n

155) SBC A,(HL)

Operação Simbólica

$A \leftarrow A - (HL) - CY$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução subtrai o conteúdo do acumulador com o conteúdo de memória, endereçada pelo par de registradores "HL", menos o CARRY BIT, sendo o resultado carregado no acumulador.

Código de Operação

Hexadecimal 9E
Octal 236
Decimal 158
Binário 10011110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SBC	Z,C,P/V,S,N,H	1	2	7

Condição dos FLAGS

Ver CP n

156) SBC A,(T + d)

Operação Simbólica

$$A \leftarrow A - (T + d) - CY$$

Descrição:

Esta instrução é formada por três BYTES, sendo dois BYTES o código de operação; o terceiro é somado ou subtraído junto com o registrador de índice (IX ou IY). Ao efetuar-se esta instrução, o conteúdo do acumulador é subtraído da posição de memória, endereçada pelo registrador de índice, menos o CARRY BIT, e o resultado é carregado no acumulador.

Código de Operação

Binário $11 \leftarrow T \leftarrow 11101$ 10011110 d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SBC A, (T+d)	Z, C, P/V, S, N, H	3	5	19

Condição dos FLAGS

Ver CP n

157) SBC HL, SS

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução subtrai o conteúdo do par de registradores "HL", do conteúdo do registrador "SS", menos o CARRY BIT, e o resultado é carregado no par de registradores "HL".

Código de Operação

Binário 11101101 $01 \leftarrow \underline{SS} \rightarrow 0010$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SBC HL, SS	C, Z, P/V, S, N, H	2	4	15

Condição dos FLAGS

C	SET se vem "um" para o BIT 15
Z	SET se o resultado for "zero"
P/V	SET se teve OVERFLOW
S	SET se o BIT "15" for "um"
N	SET em qualquer situação
H	SET se houve "vem um" do BIT 12

158) SCF

Operação Simbólica

CY + 1

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução coloca o conteúdo do CARRY BIT em nível lógico "um", ou em outras palavras, o CARRY BIT é setado.

Código de Operação

Hexadecimal	37
Octal	67
Decimal	55
Binário	00110111

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SCF	C, N, H	1	1	4

Condição dos FLAGS

C	SET em qualquer condição
Z	Não modificado
P/V	Não modificado
S	Não modificado
N	RESET em qualquer condição
H	RESET em qualquer condição

159) SET b,r

Operação Simbólica

$r_b \leftarrow 1$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução coloca em nível lógico "um", um único bit "B" de um registrador "R".

Código de Operação

Binário 11001011 $11 \leftarrow b \rightarrow r$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SET b,r	nenhum	2	2	8

Exemplo:

Conteúdo do registrador B = 7CH \rightarrow 0111 1100

SET 1,B

Conteúdo do registrador B = 7EH \rightarrow 0111 1110

160) SET b,(HL)

Operação Simbólica

$(HL)_b \leftarrow 1$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução coloca em nível lógico "um" um BIT "b", de uma posição de memória endereçada pelo par de registradores "HL".

Código de Operação

Binário 11001011 $11 \leftarrow b \rightarrow 110$

Parâmetros

Mnemônico	FLAGS Afetados	nº de BYTES	Ciclos de Máq.	Estados
SET b,HL	nenhum	2	4	15

161) SET b (T + d)

Operação Simbólica

$(T + d)_b \leftarrow 1$

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto o código de operação; o terceiro é somado ou subtraído do registrador de índice. Esta instrução seta um BIT "b" de uma determinada memória, endereçada pelo registrador de índice IX ou IY, com o seu respectivo BYTE de dados.

Código de Operação

Binário $11 \leftarrow T \rightarrow 11101$ 11001011 d $11 \leftarrow b \rightarrow 110$

162) SLA r

Operação Simbólica

$CY \leftarrow r7 \leftarrow r0 \leftarrow 0$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca de uma posição para a esquerda, o conteúdo do acumulador, sendo o conteúdo do BIT 7 carregado no CARRY BIT, e no conteúdo do BIT 0 carrega-se "zero".

Código de Operação

Binário 11001011 $00100 \leftarrow r \rightarrow$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SLA r	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

Ver RL r

Exemplo:

Conteúdo do registrador H = 37 H = 0011 0111

SLA H

Conteúdo do registrador H = 6E H 0110 1110

C = 0 Z = 0 P/V = 0 S = 0 N = 0 H = 0

163) SLA (HL)

Operação Simbólica

CY ← (HL)₇ ← (HL)₀ ← 0

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "SLA r", apenas que o conteúdo da memória, endereçada pelo par de registradores "HL", é envolvido nesta operação.

Código de Operação

Hexadecimal	CB	26
Octal	316	46
Decimal	203	38
Binário	11001011	00100110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SLA (HL)	C,Z,P/V,S,N,H	2	2	15

Condição dos FLAGS

Ver RL R

164) SLA (T + d)

Operação Simbólica

$CY \leftarrow (T + d)_7 \leftarrow (T + d)_8 \leftarrow 0$

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto o código de operação; o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução é similar à "SLA r", exceto que o conteúdo de memória, endereça da pelo registrador de índice IX ou IY, com seu respectivo BYTE de dados, é quem se envolve nesta operação.

Código de Operação

Binário $11 \leftarrow T \rightarrow 11101 \quad 11001011 \quad d \quad 00100110$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SLA (T+d)	C,Z,P/V,S,N,H	4	4	23

Condição dos FLAGS

Ver RL r

165) SRA r

Operação Simbólica

$r_7 \rightarrow r_0 \rightarrow CY$

Descrição:

Esta instrução é formada por dois BYTES, sendo eles o código de operação. Esta instrução desloca o conteúdo do registrador "r" de uma posição para a direita, e o conteúdo do BIT 0 será carregado no CARRY BIT, e o conteúdo do BIT 7 será carregado com seu próprio valor.

Código de Operação

Binário $11011011 \quad 00101 \leftarrow \underline{r} \rightarrow$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SRA r	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

Ver RR r

Exemplo:

Conteúdo do registrador E \rightarrow 5CH 0101 1100

SRA E

Conteúdo do registrador E \rightarrow 2EH 0010 1110

C = 0 Z = 0 P/V = 1 S = 0 N = 0 H = 0

166) SRA (HL)

Operação Simbólica

(HL)7 \rightarrow (HL)0 \rightarrow CY

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "SRA r", apenas o conteúdo da memória, endereçada pelo par de registradores "HL", é envolvido nesta operação.

Código de Operação

Hexadecimal	CB	2E
Octal	313	56
Decimal	203	46
Binário	11001011	00101110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SRA (HL)	C,Z,P/V,S,N,H	2	2	15

Condição dos FLAGS

Ver RR (HL)

167) SRA (T + d)

Operação Simbólica

$$(T + d)_7 \rightarrow (T + d)_8 \rightarrow CY$$

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto o código de operação; o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução é similar à "SRA r", exceto que o conteúdo de memória, endereçada pelo registrador de índice IX ou IY, com seu respectivo BYTE de dados, é quem se envolve nesta operação.

Código de Operação

Binário $11 + \underline{T} \rightarrow 11101$ 11001011 d 00101110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SLA (T+d)	C,Z,P/V,S,N,H	6	4	23

Condição dos FLAGS

Ver RR r

168) SRL r

Operação Simbólica

$$0 \rightarrow r_7 \rightarrow r_0 \rightarrow CY$$

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução desloca de uma posição para esquerda, o conteúdo do acumulador, sendo o conteúdo do BIT 7 carregado com "zero", e o conteúdo do BIT 0 carregado no CARRY BIT.

Código de Operação

Binário 11001011 $00111 + \underline{r} \rightarrow$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SAL r	C,Z,P/V,S,N,H	2	2	8

Condição dos FLAGS

Ver RR r

Exemplo:

Conteúdo do registrador L = 01H = 00000001

SRL L

Conteúdo do registrador L = 00H = 00000000

C = 1 Z = 1 P/V = 1 S = 0 N = 0 H = 0

169) SRL (HL)

Operação Simbólica

0 → (HL)₇ → (HL)₀ → CY

Descrição:

Esta instrução é formada por dois BYTES, sendo estes o código de operação. Esta instrução é similar à "SRL r", apenas que o conteúdo da memória, endereçada pelo par de registradores "HL", é envolvido nesta operação.

Código de Operação

Hexadecimal	CB	3E
Octal	313	76
Decimal	203	62
Binário	11001011	00111110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SRL (HL)	C,Z,P/V,S,N,H	2	2	15

Condição dos FLAGS

Ver RR (HL)

170) SRL (T + d)

Operação Simbólica

$\emptyset \rightarrow (T + d)_7 \rightarrow (T + d)_8 \rightarrow CY$

Descrição:

Esta instrução é formada por quatro BYTES, sendo os dois primeiros e o quarto o código de operação; o terceiro BYTE é somado ou subtraído do registrador de índice. Esta instrução é similar à "SRL r", exceto que o conteúdo de memória, endereçada pelo registrador de índice "IX ou IY", com o seu respectivo BYTE de dados, é quem se envolve nesta operação.

Código de Operação

Binário $11 \leftarrow T \rightarrow 111\emptyset1$ $11\emptyset\emptyset1\emptyset11$ d $\emptyset\emptyset11111\emptyset$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SRA (T + d)	C,Z,P/V,S,N,H	4	6	23

Condição dos FLAGS

Ver "RR r"

171) SUB r

Operação Simbólica

$AC \leftarrow AC - r$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução subtrai o conteúdo do acumulador com o registro (r) sendo o resultado guardado no acumulador.

Código de Operação

Binário $1\emptyset\emptyset1\emptyset \leftarrow r \rightarrow$

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SUB r	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver CP n

Exemplo:

Conteúdo do acumulador = 73H → 0111 0011

Conteúdo do registrador C = 3AH → $\begin{array}{r} 0011 \\ - 0011 \\ \hline 0011 \end{array}$ 1010

SUB B

Conteúdo do acumulador = 39H

Conteúdo registrador C = 3AH

C = 0 Z = 0 P/V = 0 S = 0 N = 1 H = 1

172) SUB n

Operação Simbólica

A ← A - n

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação, e o segundo um BYTE de dados. Esta instrução subtrai o conteúdo do acumulador com o BYTE de dados, e o resultado é guardado no acumulador.

Código de Operação

Hexadecimal	D6	n
Octal	326	n
Decimal	214	n
Binário	11010110	n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SUB n	C,Z,P/V,S,N,H	2	2	7

Condição dos FLAGS

Ver CP n

173) SUB (HL)

Operação Simbólica

$A \leftarrow A - (HL)$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução irá efetuar a operação de subtração entre o acumulador e a posição de memória, endereçada pelo par de registradores "HL", e o resultado ficará no acumulador.

Código de Operação

Hexadecimal	96
Octal	226
Decimal	150
Binário	10010110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SUB (HL)	C,Z,P/V,S,N,H	1	2	7

Condição dos FLAGS

Ver CP n

174) SUB (T + d)

Operação Simbólica

$A \leftarrow A - (T + d)$

Descrição:

Operação

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação; o terceiro indica o BYTE que deverá ser somado ou subtraído do registrador de índice. Esta instrução subtrai o conteúdo do acumulador com a posição de memória, endereçada pelo registrador de índice IX e IY.

Código de Operação

Binário 11 + T + 11101 10010110 d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
SUB (T + d)	C,Z,P/V,S,N,H	3	5	19

Condição dos FLAGS

Ver CP n

175) XOR r

Operação Simbólica

$A \leftarrow A \oplus r$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução executa a lógica "OU exclusivo", entre o conteúdo do acumulador e o conteúdo do registrador "r", sendo o resultado carregado no acumulador.

Código de Operação

Binário 10101 + r +

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
XOR	C,Z,P/V,S,N,H	1	1	4

Condição dos FLAGS

Ver AND r

Exemplo:

Conteúdo do acumulador	3 A	0011	1010
Conteúdo do registrador E	8 7	+ 1000	0111
		1011	1101

XOR E

Conteúdo do acumulador → = BDH

Conteúdo do registrador E = 87H

C = 0 Z = 0 P/V = 1 S = 1 N = 0 H = 1

176) XOR n

Operação Simbólica

A ← A ⊕ n

Descrição:

Esta instrução é formada por dois BYTES, sendo o primeiro o código de operação e o segundo um BYTE de dados. Esta instrução executa a lógica "OU exclusivo", entre o conteúdo do acumulador e o BYTE de dados, sendo o resultado guardado no acumulador.

Código de Operação

Hexadecimal	EE	n
Octal	356	n
Decimal	238	n
Binário	11101110	n

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
XOR n	C,Z,P/V,S,N,H	2	2	7

Condição dos FLAGS

Ver AND r

177) XOR (HL)

Operação Simbólica

$A \leftarrow A \oplus (HL)$

Descrição:

Esta instrução é formada por um único BYTE, sendo este o código de operação. Esta instrução executa a operação lógica "OU exclusivo", entre o conteúdo do acumulador e a posição de memória, endereçada pelo par de registradores "HL", ficando o resultado no acumulador.

Código de Operação

Hexadecimal	AE
Octal	256
Decimal	174
Binário	10101110

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
XOR (HL)	C,Z,P/V,S,N,H	1	2	7

Condição dos FLAGS

Ver AND r

178) XOR (T + d)

Operação Simbólica

$A \leftarrow A \oplus (T + d)$

Descrição:

Esta instrução é formada por três BYTES, sendo os dois primeiros o código de operação; o terceiro indica o BYTE que deverá ser somado ou subtraído do registrador de índice. Esta instrução subtrai o conteúdo do acumulador com a posição de memória, endereçada pelo registrador de índice.

Código de Operação

Binário $11 \leftarrow T \rightarrow 11101$ 10101110 d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
XOR (T + d)	C,Z,P/V,S,N,H	3	5	19

Condição dos FLAGS

Ver AND n

CAPÍTULO 8 - LINGUAGEM

Desde o princípio da era do computador, a principal preocupação foi de como poderia ser feita a comunicação entre o homem e a máquina.

Como se pode notar, tanto o computador como a máquina utilizam-se apenas de níveis de tensão, chamados níveis lógicos.

A grande preocupação do homem é utilizar o computador em todas as áreas e, para isso, necessita de uma comunicação de fácil acesso com o meio externo. Isto, em outras palavras, significa transformar níveis de tensão em informações, com o tipo de linguagem empregada pela maioria dos homens.

Uma das primeiras comunicações com o computador foi a linguagem de máquina ou objeto. Esta linguagem é muito cansativa por lidar diretamente com códigos binários, isto é, níveis lógicos "zero" e "um".

Em função da dificuldade apresentada por esta linguagem, surgiu a linguagem ASSEMBLY, que transforma códigos binários em mnemônicos, isto é, nomeia cada código para que este possa ser utilizado mais facilmente.

Isto pode ser verificado, por exemplo, nos microcomputadores Z-80 e 8080, onde seus mnemônicos são totalmente incompatíveis, mas a maioria de seus códigos binários são compatíveis.

Com o surgimento desta linguagem, houve a necessidade de transformar os mnemônicos em códigos binários. Esta transformação é chamada de Compilador Assembler. Pela figura 8.1, pode-se notar o fluxo de dados.

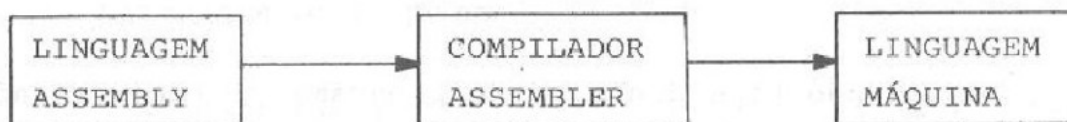


Fig. 8.1 - Programa Assembler.

Mesmo com o surgimento da linguagem ASSEMBLY, o operador continuava encontrando muitas dificuldades para manusear este grupo de instruções. Mediante estas dificuldades, outras linguagens foram-se desenvolvendo.

Com o aparecimento de outras linguagens, houve uma divisão entre as linguagens de baixo nível e as linguagens de alto nível.

As linguagens de alto nível surgiram especificamente para cada área, como "FORTRAN" para área científica; "COBOL" para área comercial e outras linguagens para cada finalidade como BASIC, PASCAL, PL/1, etc.

As linguagens de alto nível deram margem ao surgimento dos programas tradutores. Como o próprio nome está dizendo, o computador necessita de um programa que traduza tais linguagens para que tenha condições de executar as instruções a ele designadas.

Atualmente existem dois tipos de programa tradutor. O primeiro é chamado de compilador, e transforma um determinado programa com linguagem de alto nível, em linguagem de máquina, para posteriormente o executar. Na figura 8.2, pode-se ver o fluxo do programa compilador.

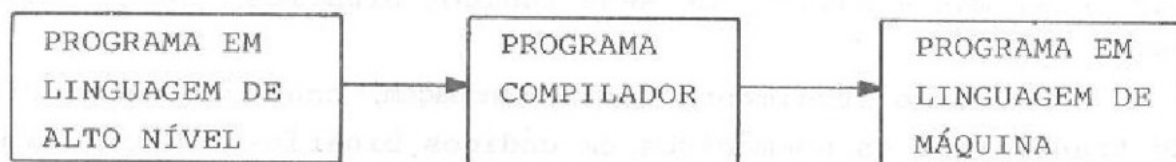


Fig. 8.2 - Programa Compilador.

O segundo tipo denomina-se programa interpretador, tendo o seguinte princípio: cada instrução é transformada em linguagem de máquina, executada e assim sucessivamente. Na figura 8.3, pode-se ver o fluxo de um programa interpretador.

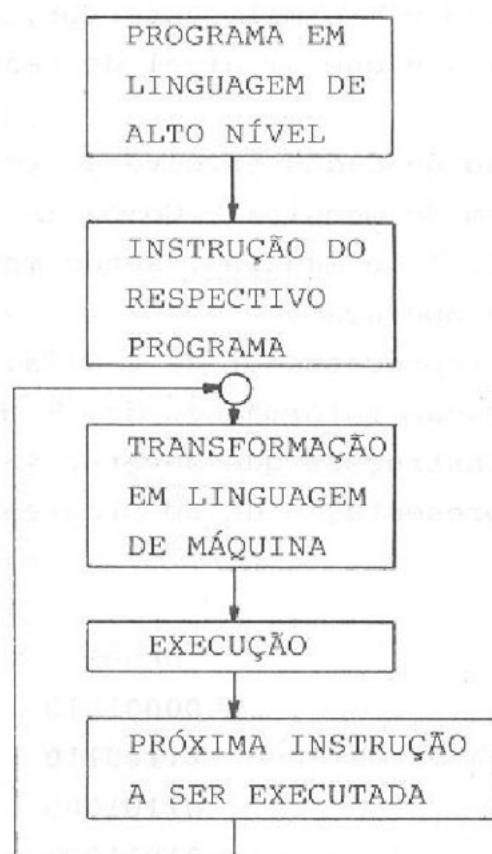


Fig. 8.3 - Programa Interpretador.

A execução de um programa compilado é mais rápida que a de um programa interpretado. Porém, o programa interpretado pode sofrer qualquer modificação e, em seguida, ser executado. Já, se o programa compilado sofrer qualquer modificação, precisará, antes, ser modificado na fonte e depois compilado novamente, isto é, transformar-se todo o programa fonte em linguagem de máquina, para então executá-lo.

8.1 - Linguagem de Máquina

No microprocessador Z-80, como em qualquer outro existente na praça, para que se possa estabelecer uma comunicação com o meio exterior, memórias e outros dispositivos auxiliares, é necessário que exista uma linguagem.

O Z-80, como é um sistema eletrônico digital, é ca

paz de realizar manipulação com o meio exterior, apenas em códigos binários, que nada mais é que um nível de tensão, (5 Volts) nível lógico "zero".

Esta manipulação de dados envolvendo codificação binária é chamada de linguagem de máquina. Como o próprio nome diz, é uma linguagem a nível de máquina, sendo muito trabalhosa e cansativa, em termos de operação.

O Z-80 é um microprocessador de 8 BITS. Portanto, na memória deverão ser guardadas informações de 8 BITS, que por ventura poderão ser as instruções que deverão ser processadas. A figura 8.4 mostra a representação de um programa em linguagem de máquina.

ENDEREÇO	DADOS
00000001 01100000	00001010
00000001 01100001	01100110
00000001 01100010	01101000
00000001 01100011	01011001
00000001 01100100	00000000
00000001 01100111	01101001
00000001 01100110	01000000
00000001 01100101	00111111
00000001 01101000	01010110
00000001 01101011	00000001
00000001 01101010	00001110
00000001 01101001	11110000
00000001 01101100	11000011

Fig. 8.4 - Linguagem de Máquina em Binário.

Como se pode notar, é muito difícil distinguir que tipo de instrução está sendo processada, por isso este tipo de codificação binária pouco usado. Existe um artifício que é muito utilizado em microprocessador: a codificação em hexadecimal. Como foi visto no volume I, para transformar codificação binária em hexadecimal, divide-se em grupos de quatro elementose, com a ajuda da tabela, efetua-se a transformação. Pela figura 8.5, pode-se notar a diferença.

ENDEREÇO	DADO
Ø16Ø	ØA
Ø161	66
Ø162	68
Ø163	59
Ø164	ØØ
Ø165	69
Ø166	4Ø
Ø167	3F
Ø168	56
Ø169	Ø1
Ø16A	ØE
Ø16B	FØ
Ø16C	A3

Fig. 8.5 - Linguagem de Máquina em Hexadecimal.

A linguagem de máquina é também chamada de programa objeto, não necessita de interpretador, e uma vez carregado na memória, está pronto para ser processado.

8.2 - Linguagem ASSEMBLY

Introdução

Como foi visto nos capítulos anteriores, a Linguagem ASSEMBLY foi a primeira linguagem que surgiu nesta área de computação.

O microprocessador Z-80 possui uma variedade muito grande de instruções, e cada uma possui o seu correspondente conjunto de mnemônicos.

Esta linguagem tem como finalidade facilitar o manuseio das instruções, para que possam desenvolver programas de todos os tipos, desde uma simples rotina para efetuar uma multiplicação, até um sofisticado programa para controle de dispositivos para indústria, para o lar ou qualquer outro tipo.

Estrutura da Linguagem Assembly

A linguagem ASSEMBLY é dividida em grupos. Na tabela 8.6, pode-se acompanhar esta divisão.

LABEL	Código de Operação	Operando ou Endereço	Comentário
Começo:	LD	SP,FIM	; Inicialização STACK POINTER
	IM	2	; Modo de interrupção dois
LOOP 1:	IN	A,(Portal)	; Lê o porto Al
	AND	0FH	; Máscara a parte mais significativa do porto.

Fig. 8.6 - Estrutura da Linguagem Assembly.

Obs: "Espaço", "Vírgula", "Dois Pontos", "Ponto e Vírgula" são chamados de delimitadores.

8.2.1 - LABEL

A principal característica da linguagem ASSEMBLY é o uso de LABEL. O LABEL torna a programação em ASSEMBLY mais rápida e segura, porque ao invés de lidar com um endereço absoluto, por exemplo, JP 1FA3H, lida com um conjunto de caracteres (LABEL), por exemplo JP LOOP1. Este artifício é útil para seguir o fluxo de dados, e também quando se acrescenta ou se retira alguma instrução, pois o novo endereço do LABEL é automatizado.

camente corrigido pelo programa Compilador Assembler.

A única restrição é que o LABEL deve conter no máximo seis caracteres sendo que o primeiro carácter deve ser sempre uma letra, isto é, o LABEL não pode começar com um número.

8.2.2 - Código de Operação

O código de operação é simplesmente a instrução pura como "LD", "SUB", "AND" e outras. No código de operação aparecerá apenas o conjunto de caracteres, que representará o tipo de instrução como: soma; carregamento; teste e outras.

8.2.3 - Operando ou Endereço

O terceiro grupo é dividido em duas partes. A primeira parte da informação é chamada de Operando, sendo a designação dada para mencionar o fluxo de dados. A segunda é chamada de Endereço, e é utilizada pelas instruções de "CALL" e "JP".

8.2.4. - Comentário

O quarto e último grupo se destina ao comentário que, em linguagem ASSEMBLY, é o que há de mais importante num programa.

Os comentários designam uma fácil visualização do fluxo do programa. O comentário deve, na maioria das vezes, dizer para que se destina a instrução e não simplesmente descrevê-la.

8.2.5 - Delimitadores

Os delimitadores são usados para separação dos grupos.

Um espaço ou um conjunto de espaços em branco é usado

para separar cada grupo, por exemplo, entre o código de operação e o operando ou endereço.

Dois pontos delimitam o fim de um LABEL. Em Pseudo-Instrução não há necessidade de dois pontos depois de um LABEL.

A vírgula separa operando de endereço, por exemplo: (A,B; C,D; A,7AH; B,00100110B; HL,7FA3H).

O ponto e vírgula indica o início de comentário.

8.3 - Pseudo-Instrução

Pseudo-Instrução é um termo usado em Linguagem ASSEMBLY, também chamado de "Instrução Especial".

Pseudo-Instrução é usada na definição de LABEL, definição de nome de programa, definição de STRING, definição do endereço inicial do programa, reserva de área de memória e definição de BYTE.

Sempre que se define um dado hexadecimal em um programa que começa com (A, B, C, D, F,) este deverá ser precedido pelo algarismo zero, por exemplo: (0BA12H), (0A3H).

Todo número representado em hexadecimal deverá ser acompanhado pela letra "H", todo número em binário deverá ser acompanhado pela letra "B"; todo número em octal deverá ser acompanhado pela letra "O" ou "Q" e todo número em decimal deverá ser acompanhado pela letra "D" ou, simplesmente, o número sozinho.

Exemplo:

Hexadecimal	07BAH, 0ABH, 0BFC0H
Binário	100100B, 011101B, 10010011B
Octal	371O, 670O, 14O ou 371Q, 670Q, 14Q
Decimal	77D, 66790D ou 77, 66790

TITTLE	define	Título ao programa
DEFB	define	BYTE
DEFL	define	LABEL
DEFM	define	STRING

DEFS	define	Área de memória
DEFW	define	Palavra
ORG	define	A origem do programa
EQU	define	EQUATE
END	define	Fim de programa
\$	define	Representa o endereço aonde se localiza o programa COUNTER.

1 - TITTLE

Operação simbólica

TITTLE - Descrição qualquer

Descrição:

Esta pseudo-instrução é usada para dar título ao programa a que está se referindo.

Exemplo:

Código de Operação	Operando
TITTLE	Multiplicação entre dois números.

2 - DEFB

Operação simbólica

LABEL	Código de Operação	Operando
nome	DEFB	N

Descrição:

Esta pseudo-instrução é usada quando se deseja inserir um BYTE na execução de um programa, quando se quer definir uma tabela de dados ou, simplesmente, inserir um dado no meio de um programa.

Exemplo:

1)	0000	7F	DEFB	7FH
	0001	0F	DEFB	15
	0002	41	DEFB	41H
	0003	3A	DEFB	00111100B
	0004	04	DEFB	04H
	0005	1A	DEFB	1AH

	0006	33	DEFB	33H
	0007	0A	DEFB	10
2)	0000	CB0F	RRC	A
	0002	60	LD	H,B
	0003	FA	DEFB	0FAH
	0004	0E20	LD	C,14H
	0006	CB85	RES	0,L
	0008	6C	LD	L,H
	0008	DD	DEFB	0DDH
	000A	2119A0	LD	HL,0A019H

3 - DEFL

Operação Simbólica

LABEL	Código de Operação	Operando
Nome	DEFL	NN

Descrição:

Esta instrução é usada para especificar um certo endereço a um determinado LABEL.

Usando desta pseudo-instrução, o LABEL poderá ser definido novamente no transcorrer do programa.

Exemplo:

0000	3F0E	LD	A,14
=LABO=	MULT	DEFL	1AB0H
0002	11B01A	LD	DE,MULT
0005	80	ADD	A,B
0006	12	LD	(DE),A
=7B7E=	MULT	DEFL	7B7EH
0007	217E7B	LD	HL,MULT
000A	7E	LD	A,(HL)

4 - DEFM

Operação Simbólica

LABEL	Código de Operação	Operando
Nome	DEFM	"STRING"

Descrição:

Esta pseudo-instrução é usada quando se deseja de

signar um conjunto de STRINGS, isto é, um conjunto de caracteres. Uma nota importante é que o conjunto de STRINGS deverá estar contido entre aspas.

O programa Assembler transformará todo o conjunto de STRINGS, contido entre aspas, em hexadecimal, representando a sua equivalência em ASCII.

```
0000      4D 49 43 52      DEFM      "Microprocessador Z-80"
0004      4F 50 52 4F
0008      43 45 53 53
000C      41 44 4F 52
0010      20 5A 38 30
0011      3E 71      LD      A,71H
```

5 - DEFS

Operação Simbólica

	Código de Operação	Operando
DEFS		NN

Descrição:

Esta pseudo-instrução reserva uma determinada área de memória e o dado "NN" é quem define o tamanho desta área.

Exemplo:

```
0000      78      LD      A,B
0001      21A07E    LD      HL,7EA0H
0004      DEFS     7
000B      41      LD      B,C
```

6 - DEFW

Operação Simbólica

LABEL	Código de Operação	Operando
Nome	DEFW	NN

Descrição

Esta pseudo-instrução carrega numa posição de memória, onde o Programa Counter (PC) está posicionado, o par de dados "NN", sendo que no (PC) será carregado (NN+1) e no (PC+1) será carregado NN.

Exemplo:

0000	B37A	DEFW	7AB3H
0002	0614	LD	B,14H
0004	80	ADD	A,B
0005	1407	DEFW	0714H
0007	111417	LD	DE,1714H

7 - ORG

Operação Simbólica

Código de Operação	Operando
ORG	NN

Descrição

Esta pseudo-instrução especifica qual o valor que o (PROGRAMA COUNTER) deverá assumir, onde o dado NN é quem vai de terminar o endereço deste programa.

Exemplo:

		ORG	2000H
2000	DD214800	LD	IX,0048H
2004	CBBF	RES	7, A
2006	CBC0	SET	0,B

8 - EQU

LABEL	Código de Operação	Operando
nome	EQU	NN

Descrição:

Esta pseudo-instrução especifica ao LABEL um certo valor especificado pelo dado "NN". Esta, é similar a DEFL, ex ceto que não pode ser danificada no transcorrer do programa.

Exemplo:

		ORG	5000H
=0FFF=	FIM	EQU	0FFFH
=7A=	PORT1	EQU	7AH
=8A=	PORT2	EQU	8AH
5000 31FF0F		LD	SP,FIM
5003 DB7A		IN	A,(PORT1)
5005 D38A		OUT	(PORT2),A

9 - END

Operação Simbólica

Código de Operação

END

Descrição:

Esta pseudo-instrução representa o fim de um programa.

10 - \$

Descrição:

Este símbolo representa o endereço onde se localiza o Programa Counter. Este artifício é usado em "JUMPS" relativos em outros casos, que serão vistos no capítulo seguinte.

Exemplo:

		ORG	7B00H	
7B00	79	LD	A,B	
7B01	1801	JR	Z,LOOP1-\$	{ Representa LOOP1-PC
7B03	44	LD	B,H	
7B04	C620 LOOP1:	ADD	A,20H	

CAPÍTULO 9 - PRÁTICA DE PROGRAMAÇÃO

9.1 - Introdução

Neste capítulo serão desenvolvidos programas em linguagem Assembly. Estes programas abrangerão a maioria das instruções desenvolvidas nos capítulos anteriores e serão apresentados exemplos de programações da "PIO" e "CTC".

Outro ponto importante deste capítulo é que serão implementados alguns algoritmos desenvolvidos no volume de HARDWARE.

Em todos os programas desenvolvidos serão apresentadas explicações minuciosas e fluxogramas.

Os programas elaborados neste capítulo poderão ser executados em qualquer KIT de microprocessador, que utilize o Z-80 e que trabalhe em Linguagem de Máquina, ou qualquer sistema de médio e grande porte, que apresente em seu sistema operacional o Assembler do microprocessador Z-80.

9.2 - Método de Programação

Em qualquer linguagem que se deseje elaborar um programa, necessita-se de um roteiro. Este roteiro pode variar de acordo com o método de cada programador.

Todos os programas devem, de uma maneira geral, apresentar uma breve descrição, onde conste a filosofia utilizada, limitações e utilidades.

Após esta descrição deverão apresentar um diagrama de fluxo, mostrando as bifurcações e a estrutura do mesmo.

Por último, deverão os programas serem elaborados na sua respectiva linguagem. No caso da linguagem Assembly, os programas deverão possuir comentários explicando o fluxo, pois torna-se difícil acompanhar o mesmo, seguindo-se somente as instruções.

9.3 - Aplicação Prática de algumas Instruções

9.3.1. Na figura 9.1, pode-se verificar se o conteúdo do acumu

lador é zero, bastando apenas executar a função "AND A" ou "OR A". Estas instruções não alteram o conteúdo do acumulador, afetando apenas os FLAGS.

```

'0000 A7          AND    A
'0001 C20800'     JP     NZ,INICIO

'0004 B7          OR     A
'0005 CA0800'     JP     Z,FIM

'0008 4F          INICIO: LD     C,A
'0009 87          ADD    A,A
'000A C9          RET

'000B 3EFF        FIM:   LD     A,OFFH
'000D 21800D      LD     HL,03456
'0010 C9          RET

```

Fig. 9.1 - Teste do Acumulador.

A instrução "CP 00H" também não altera o conteúdo do acumulador, modificando apenas os FLAGS. Esta instrução contém apenas dois BYTES e, usando-se as instruções "AND" ou "OR", economiza-se um BYTE. Pela figura 9.2, pode-se acompanhar a estrutura desta instrução.

```

'0011 FE00        CP     00H
'0013 CA1900'     JP     Z,LOOP1
'0016 3EFF        LD     A,OFFH
'0018 C9          RET
'0019 3E03        LOOP1: LD     A,03H
'001B C9          RET

```

Fig. 9.2 - Teste do Acumulador.

9.3.2. Para multiplicar o conteúdo do acumulador por 2, basta apenas executar a instrução ADD A,A, que soma o acumulador consigo próprio; ou executar a instrução SLA A, que desloca o conteúdo do acumulador de uma posição para a esquerda. Na figura 9.2.A, pode-se observar esta aplicação.

'001C	87	ADD	A,A	;ACUMULADOR VEZES DOIS
'001D	87	ADD	A,A	;ACUMULADOR VEZES QUATRO
'001E	87	ADD	A,A	;ACUMULADOR VEZES OITO
'001F	CB27	SLA	A	;ACUMULADOR VEZES DOIS
'0021	CB27	SLA	A	;ACUMULADOR VEZES QUATRO
'0023	CB27	SLA	A	;ACUMULADOR VEZES OITO

Fig. 9.2.A - Multiplica por dois o conteúdo do acumulador.

Exemplo:

A = 00000010

$ \begin{array}{r} 0000 \ 0010 \leftarrow A \\ + \ 0000 \ 0010 \leftarrow A \\ \hline 0000 \ 0100 \leftarrow A \times 2 \end{array} $	<div style="text-align: center;"> <p>A</p> <div style="border: 1px solid black; display: inline-block; padding: 5px;"> 0000 0010 </div> <p>← 0</p> <div style="border: 1px solid black; display: inline-block; padding: 5px;"> 0000 0100 </div> <p>← A x 2</p> </div>
ADD A,A	SLA A

Fig. 9.2.B

9.3.3. Para divisão do conteúdo do acumulador por 2, basta apenas executar a instrução "SRL A", que desloca o conteúdo do acumulador para direita, o que pode ser observado na figura 9.3.

'0025	CB3F	SRL	A	;ACUMULADOR DIVIDIDO POR DOIS
'0027	CB3F	SRL	A	;ACUMULADOR DIVIDIDO POR QUATRO
'0029	CB3F	SRL	A	;ACUMULADOR DIVIDIDO POR OITO

Fig. 9.3 - Divide por dois o conteúdo do acumulador.

9.3.4. Quando se deseja zerar o conteúdo do acumulador, basta executar a instrução "SUB A" ou "XOR A"; o único detalhe é que estas instruções afetam os FLAGS.

Esta operação poderia ser realizada através da instrução "LD A,00H", mas esta ocupa dois BYTES.

O segundo método é usado quando não se deseja modificar os FLAGS. A figura 9.4 ilustra o exposto.

		ALTERAM OS FLAGS	
'002B	97	SUB	A ;ACUMULADOR IQUAL A ZERO
'002C	AF	XOR	A ;ACUMULADOR IQUAL A ZERO
		NAO ALTERA OS FLAGS	
'002D	3E00	LD	A,00H ;ACUMULADOR IQUAL A ZERO

Fig. 9.4 - Zerar o conteúdo do acumulador.

9.3.5. A figura 9.5 apresenta uma técnica para se inverter um determinado BIT do conteúdo do acumulador. Para tal, basta executar a instrução "XOR n", onde o "n" é o valor do BIT a ser invertido.

A = (0110 1110) B Deseja-se inverter "BIT 0" e o
"BIT 6"

'002F EE41 XOR 01000001B

A = (0010 1111) B

Fig. 9.5 - Inverter um BIT do conteúdo do acumulador.

9.3.6. Quando se deseja testar se um determinado registrador (B, C, D, E, H, L) é zero, pode-se proceder como no exemplo da figura 9.6.

'0031	05	DEC	B	
'0032	04	INC	B	
'0033	CA3900'	JP	Z,TERMIN	;CONT. DE "B" NAO FOI AFETADO ;SE FOR ZERO VAI PARA TERMIN ;CASO CONTRARIO CONTINUE
'0034	0EFF	LD	C,0FFH	;SETA O REGISTRO "C"
'0038	C9	RET		
'0039	0E00	TERMIN: LD	C,000H	;RESETA O REGISTRO "C"
'003B	C9	RET		

Fig. 9.6 - Verificação do conteúdo de um registrador.

9.3.7. Pelo grupo de instruções do Z-80, pode-se notar que existem poucas instruções de troca de par de registradores, como "EX HL,DE". Para que se possa trocar o conteúdo de um par de registradores como "BC" com "HL", "DE com IX", basta executar a rotina da figura 9.7.

```

*****
*                                     *
*               TROCA DE PARES DE REGISTRO               *
*                                     *
*****

'003C  C5          PUSH  BC      ;GUARDA NA POS. DE MEMORIA END.
'003D  D5          PUSH  DE      ;PELO "SP" O CONTEUDO DOS
                                ;DOIS REGISTRADORES.
'003E  C1          POP   BC      ;CARREGA O CONT. DOS REG. COM
'003F  D1          POP   DE      ;POSICAO DE MEMORIA ENDER. PELO
                                ;"SP" NA MESMA ORDEM QUE FOI
                                ;EXECUTADA OS PUSH

```

Fig. 9.7 - Programa de troca de registradores.

Exemplo:

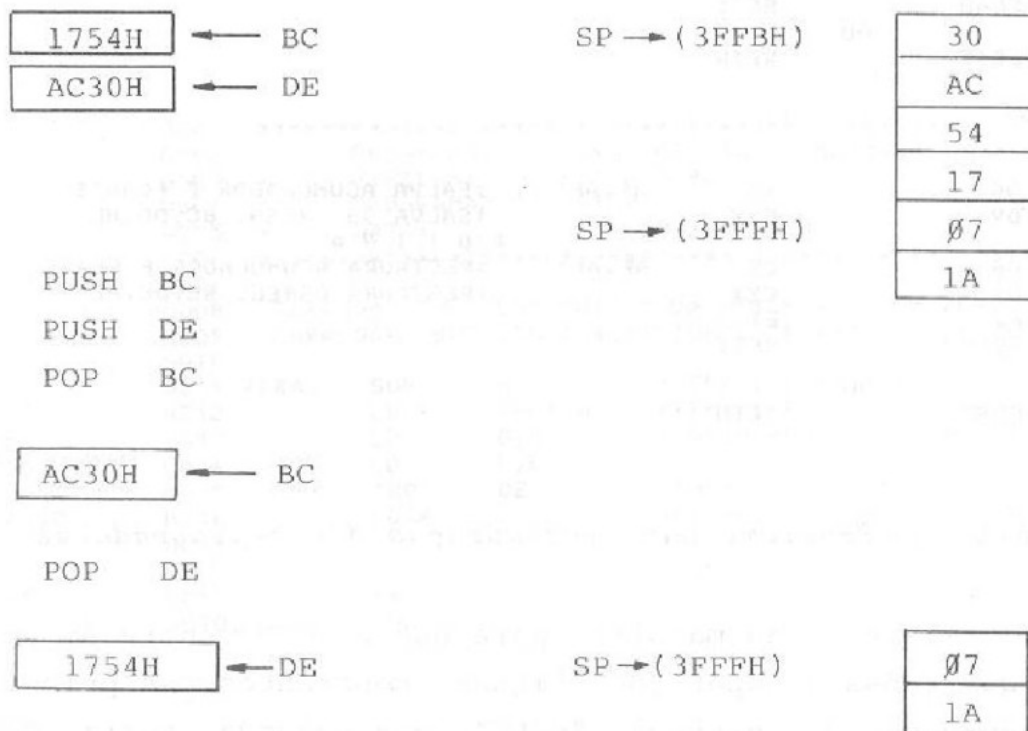


Fig. 9.7.A - Fluxo de dados de troca de registradores.

9.3.8. As rotinas de interrupção não podem, de maneira alguma, modificar os registradores e FLAGS; para isto, devem-se salvar todos os registradores que porventura possam ser destruídos.

Existem duas maneiras para salvar os registradores. A primeira é através das instruções de "PUSH" e "POP"; a segunda é quando apenas o acumulador e os registradores de uso geral forem afetados. A figura 9.8, ilustra o modo pelo qual os registradores podem ser restaurados.

```

'0040 F5      INTER:  PUSH    AF      ;SALVAR OS REGISTRADORES
'0041 C5              PUSH    BC
'0042 D5              PUSH    DE
'0043 E5              PUSH    HL
'0044 DDE5          PUSH    IX
'0046 FDE5          PUSH    IY

                                R O T I N A

'0048 FDE1          POP     IY      ;RESTAURA OS REGISTRADORES
'004A DDE1          POP     IX
'004C E1            POP     HL
'004D D1            POP     DE
'004E C1            POP     BC
'004F F1            POP     AF
'0050 F8            EI
'0051 ED4D          RETI
'0053 ED45          OU      RETN

'0055 08            EX      AF,AF'  ;SALVA ACUMULADOR E FLAG'S
'0056 D9            EXX          ;SALVA OS RESG. BC,DE,HL
                                R O T I N A
'0057 08            EX      AF,AF'  ;RESTAURA ACUMULADOR E FLAGS
'0058 D9            EXX          ;RESTAURA OSREG. BC,DE,HL
'0059 FB            EI
'005A ED4D          RETI
'005C ED45          OU      RETN

```

Fig. 9.8 - Programa para restauração dos registradores.

9.3.9. Existem duas maneiras para que a "CPU" entre em um estado em que todas as operações fiquem inoperantes. A primeira seria a execução da instrução "HALT", e a segunda seria dar um "JUMP" para o próprio endereço. Ambas sairão desta situação se houver uma interrupção ou um RESET.

Existem certas estruturas de programas onde a CPU

trabalha apenas com interrupção. As rotinas são executadas apenas quando existir uma interrupção. Na figura 9.9, pode-se verificar o exposto.

```
A)  '005F  76          HALT

B)  '0060  C36000'    WAIT:  JP      WAIT
```

Fig. 9.9 - WAIT por SOFTWARE.

9.4 - Programa Simples

9.4.1 - Zera Conteúdo de Memória

Esta rotina zera o conteúdo de memória, e para isto deve-se carregar, no par de registradores "HL", o início da memória a ser zerada, e no par de registradores "BC", a quantidade de BYTES a ser efetuados nesta operação, como mostra a figura 9.10.

```
0001 *****
0002 *      PROGRAMA:  ZERA POSICAO DE MEMORIA      *
0003 *      SISTEMA:   MICRO Z-80                    *
0004 *      PROGRAMADOR: CARLA                        *
0005 *      DATA:     8/FEV/84                      *
0006 *****
0007
0008      CARREGAR "HL" COM INICIO DA MEMORIA A SER ZERADA
0009      CARREGAR "BC" COM A QUANTIDADE DE BYTE A SER ZERADO
0010
'0000  97      0011 ZERA:  SUB    A          ;ZERA O CONT. DO ACUMUL.
'0001  77      0012      LD    (HL),A      ;ZERA A PRIM. POS. MEM.
'0002  54      0013      LD    D,H        ;COPIA REG. "HL" EM "DE"
'0003  50      0014      LD    E,L
'0004  13      0015      INC    DE          ;INCREMENTA "DE"
'0005  ED80    0016      LDIR           ;CARREGA O CONT. DE MEM. ENDER.
                                ;PELO "HL" NA POS. DE MEM. END.
                                ;PELO "DE"
                                0017
                                0018
'0007  C9      0019      RET
                                0020      END
```

Fig. 9.10 - Zera o conteúdo de memória.

9.4.2 - Verificação da Paridade de um Número

A paridade de um número é definida como sendo a quantidade de BITS, em um determinado BYTE, em um certo nível lógico.

Para saber se um certo número é par ou ímpar, basta carregar este número no acumulador e executar a instrução "AND A" ou "OR A", pois esta afetará apenas os "FLAGS".

Depois da execução de "AND A" ou "OR A", a instrução "JP PO" efetuará o "JUMP" se for ímpar, e "JP PE" se for par. Estas instruções decidirão se o número é ímpar ou par.

O programa terá controle sobre uma posição de memória designada por "INDICA"; se for ímpar esta será setada e se for par será resetado, como mostra a figura 9.11.

```
0001 *****
0002 *      PROGRAMA:      CALCULO DA PARIDADE DE UM NUMERO  *
0003 *      SISTEMA:      MICRO-Z80                          *
0004 *      PROGRAMADOR:   ERICA                               *
0005 *      DATA:         9-FER-84                            *
0006 *****
0007
0008      O NUMERO DEVERA ESTAR NO ACUMULADOR
0009
>12CA 0010 INDICA EQU      12CAH      ;FLAG INDICA SE E PAR OU IMPAR
0011
'0000 A7      0012 PARIDA: AND      A      ;CALCULO DA PARIDADE
'0001 E20700' 0013 JP      PO, IMPAR ;SE FOR PAR CONTINUA
'0004 97      0014 SUB      A      ;RESETA FLAG
'0005 1802    0015 JR      FIM-$
'0007 3EFF    0016 IMPAR: LD      A, OFFH ;SETA FLAG
'0009 32CA12  0017 FIM:  LD      (INDICA), A
'000C C9      0018 RET
0019      END
```

Fig. 9.11 - Programa que verifica se o conteúdo do acumulador é par ou ímpar.

9.5 - LOOP

LOOP é uma técnica que em qualquer programa de médio e grande porte é empregado.

Esta técnica consiste em repetir um certo grupo de

instruções. O controle feito sobre o "LOOP" é de quantas vezes este será repetido, e geralmente é feito através de um registrador usado como contador.

Antes que o programa entre em "LOOP", é carregado em um registrador quantas vezes se deseja que o grupo de instruções se repita. No fim deste "LOOP", este registrador é decrementado de uma unidade e é verificado se chegou ao fim, se positivo, o LOOP terminará, caso contrário, continuará.

No conjunto de instruções do microprocessador Z-80 existe uma instrução que é usada em "LOOP"; esta decrementa o registrador "B" e verifica o seu conteúdo. Se for diferente de zero, o Programa COUNTER (PC) assumirá o endereço do início do "LOOP"; caso contrário, o programa seguirá a seqüência normal. A desvantagem desta instrução é que o registrador "B" possui apenas um BYTE, e poderá ser carregado com um valor entre $(0 \text{ a } 255)_{10}$, ou $(0 \text{ a } FF)_{16}$; portanto o "LOOP" poderá repetir apenas nesta faixa. Na figura 9.12, pode-se ter uma visão desta técnica.

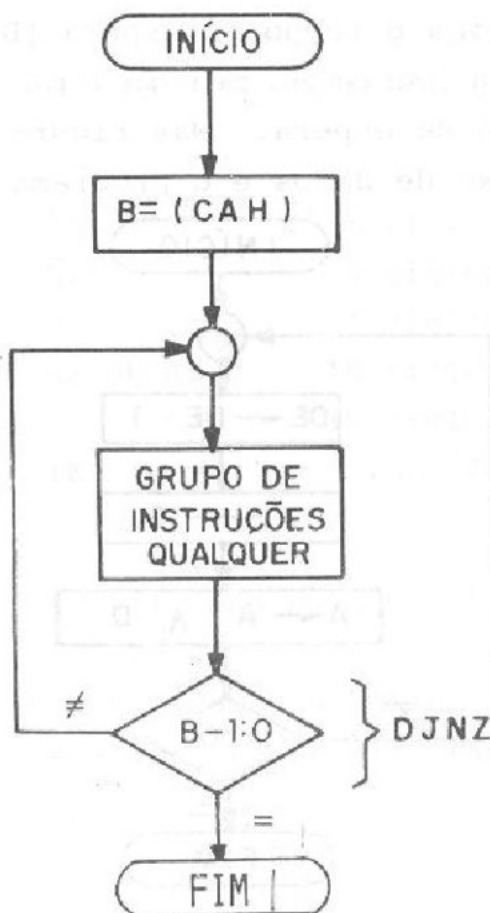


Fig. 9.12 - LOOP.

9.5.1 - DELAY

Muitas vezes na execução de um programa utiliza-se uma rotina chamada "DELAY", que é usada quando necessita-se, por exemplo, que uma entrada de algum dispositivo se estabilize antes da leitura deste sinal. Na figura 9.13, pode-se observar esta aplicação.

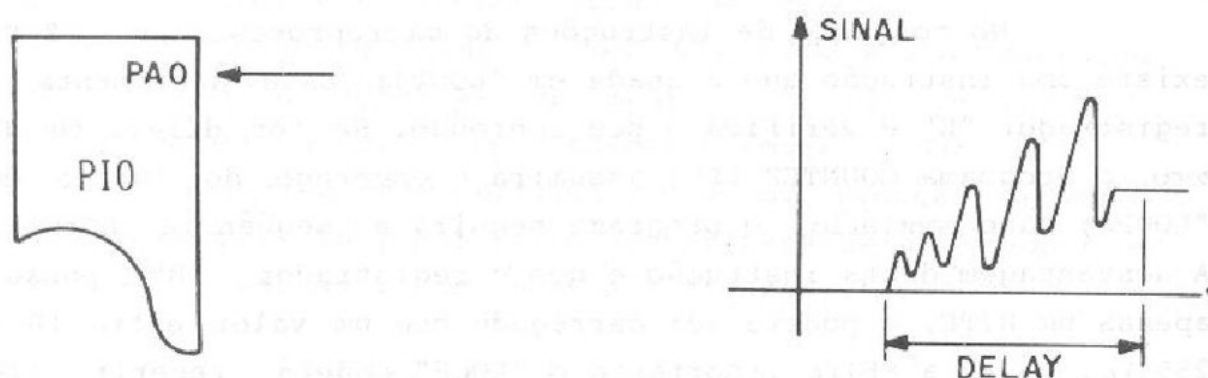


Fig. 9.13 - Aplicação da rotina "DELAY".

Nesta rotina o tempo de espera (DELAY) existirá uma variável de controle; para isso deverá carregar o par de registradores "DE" com o valor do tempo de espera. Nas figuras 9.14 e 9.15, pode-se acompanhar o fluxo de dados e o programa.

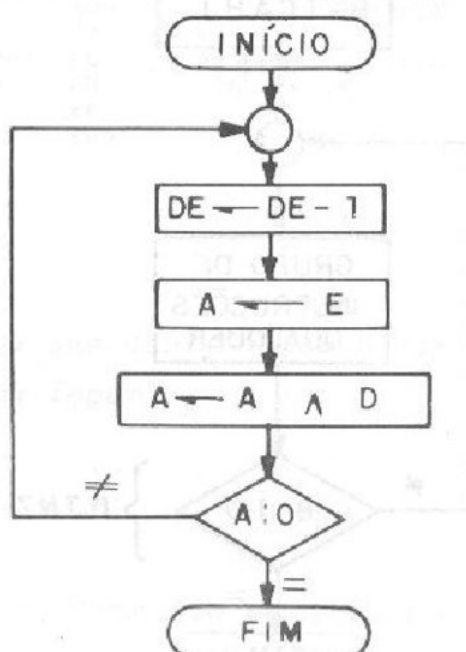


Fig. 9.14 - Fluxo da rotina "DELAY".

```

0001 *****
0002 *      PROGRAMA:      DELAY POR SOFTWARE      *
0003 *      SISTEMA:      MICRO Z-80              *
0004 *      PROGRAMADOR:    SILVIA                  *
0005 *      DATA:          10-FER-84              *
0006 *****
0007
0008      CARREGAR "DE" COM VALOR DO TEMPO DO DELAY
0009
'0000 18      0010 DELAY: DEC DE      ;DECREMENTA O CONTADOR
'0001 78      0011      LD A,E
'0002 B2      0012      OR D      ;VER. SE "D" E "E" SAO ZEROS
'0003 20FB     0013      JR NZ,DELAY-1 ;SE "D" E "E" NAO FOREM ZERO
0014      ;CONTINUA DECREMENTANDO
'0005 C9      0015      RET
0016      END

```

Fig. 9.15 - Programa de DELAY.

Para cálculo do valor a ser carregado no par de registradores "DE", necessita-se, em primeiro lugar, ter conhecimento do valor do CLOCK injetado no microprocessador.

Em seguida, necessita-se saber qual é o tempo de execução de cada instrução. Na figura 9.16, pode-se acompanhar esta estrutura.

DELAY:	DEC	DE	6 ciclos
	LD	A,E	4 ciclos
	OR	D	4 ciclos
	JP	NZ,DELAY	10 ciclos
	RET		10 ciclos
		TE =	T.(nn.24 + 10)

T = Período do CLOCK utilizado pela "CPU"

TE = Tempo total do DELAY

nn = Valor do par de registradores "DE" que será decrementado até zero, e esse número é que vai completar o tempo necessário para o DELAY.

Fig. 9.16 - Programa DELAY.

Exemplo:

$$\text{CLOCK} = 2\text{MHZ} = 5 \times 10^{-7} \text{ seg.}$$

$$\text{Tempo de execução} = 0,5 \text{ seg.}$$

$$0,5 = 5 \times 10^{-7} \quad (\text{nn} \cdot 24 + 10)$$

$$\text{nn} = (41666)_{10} = (\text{A2B2})_{16}$$

$$\text{Portanto "DE"} = (\text{A2B2H})$$

OBS:.. O máximo número que "nn" pode assumir é $(65535)_{10}$ ou $(\text{FFFF})_{16}$; se necessitar que "nn" assuma um número maior que este, basta apenas acrescentar "NOP" no programa, ou outra instrução que não perturbe a estrutura do mesmo.

9.5.2 - Paridade do Código ASCII

O código ASCII é formado por "7" BITS (BIT6 a BIT0), sendo o "BIT 7" reservado para a paridade. Este código é utilizado quando se deseja transmitir ou receber dados de impressora, vídeo ou troca de informações entre computadores.

Quando há necessidade de troca de informações com o meio externo, principalmente entre computadores de grande porte, existe a preocupação de um protocolo de comunicação chamado também de "HANDSHAKE", que consiste em analisar a paridade de dados recebidos. Se a paridade estiver de acordo, o dado será aceito, caso contrário, o computador que está recebendo pedirá que seja retransmitido o dado novamente.

Na figura 9.17, será desenvolvido um programa para cálculo da paridade do código ASCII, e para isso deverá ser carregado no par "HL" o início do bloco, e no registrador "B" a quantidade de BYTES.

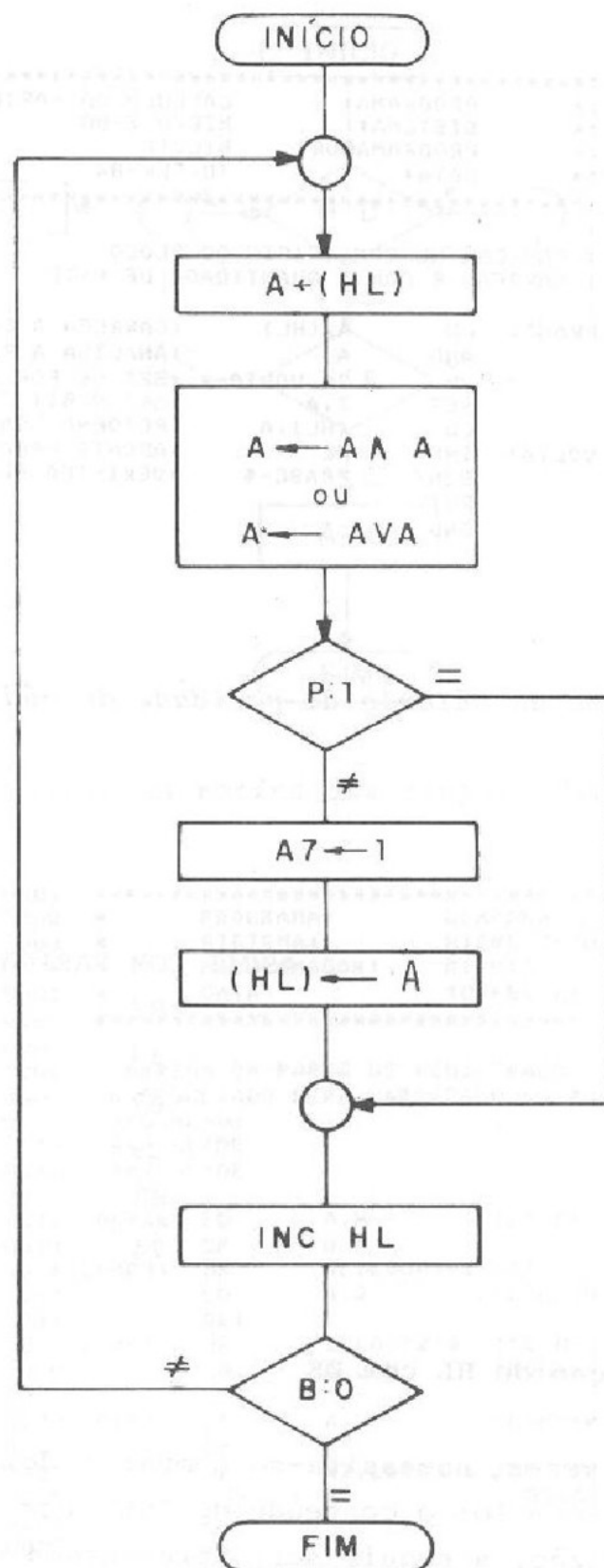


Fig. 9.17 - Fluxo do Cálculo da Paridade do Código ASCII.

```

0001 *****
0002 ;*      PROGRAMA:      CALCULO DA PARIDADE DO ASCII
0003 ;*      SISTEMA:      MICRO Z-80
0004 ;*      PROGRAMADOR:   SILVIA
0005 ;*      DATA:         10-FER-84
0006 *****
0007 ;
0008 ; CARREGA HL COM INICIO DO BLOCO
0009 ; CARREGA B COM A QUANTIDADE DE BYTE
0010 ;
0011 PRASC: LD      A,(HL)      ;CARREGA A COM O DADO
0012        AND      A          ;ANALISA A PARIDADE
0013        JP       PE,VOLTA-$ ;SET SE FOR IMPAR
0014        SET      7,A        ;SET O BIT 7
0015        LD      (HL),A      ;RETORNA COM O DADO
0016 VOLTA: INC      HL        ;APONTA PARA O PROXIMO DADO
0017        DJNZ     PRASC-$    ;VERIFICA SE JA ACABOU
0018        RET
0019        END

```

9.17.A - Programa do Cálculo da paridade do código ASCII.

Exemplo:

ASCII	ASCII COM PARIDADE
32	B2
33	33
37	B7
49	C9
58	B8
74	74

9.5.3 - Comparação de HL com DE

Muitas vezes, necessita-se comparar dois pares de registradores; por exemplo: o conteúdo de "HL" com o conteúdo de "DE". Por esta razão, a seguir será apresentada uma rotina que compara o par de registradores "HL" com o par de registradores "DE", e o acumulador será carregado com "zero" se "HL" = "DE", com "UM" se "HL" > "DE" e com "DOIS" se "HL" < "DE". Na figura 9.18, será apresentado o fluxo desta rotina.

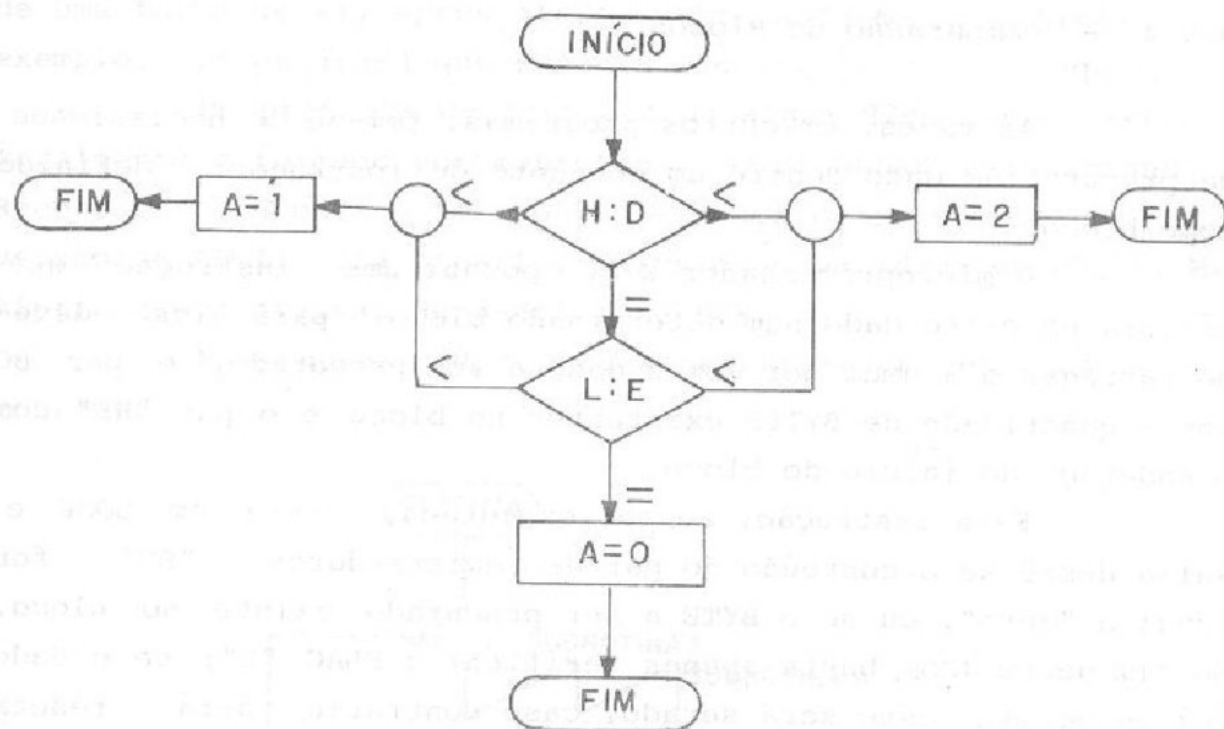


Fig. 9.18 - Fluxo da rotina que compara "HL" com "DE".

```

0001 *****
0002 *   PROGRAMA:   COMPARA "HL" POR "DE"   *
0003 *   SISTEMA:   MICRO Z-80              *
0004 *   PROGRAMADOR: SILVIA                 *
0005 *   DATA:     10-FEV-84                *
0006 *****
0007
0008 CARREGA OS PARES DE REGISTRADORES "HL" E "DE"
0009 O RESULTADO SERA CARREGADO NO ACUMULADOR
0010 A=0 HL=DE
0011 A=1 HL<DE
0012 A=2 HL>DE
0013
'0000 7C      0014 COMPAR: LD      A,H      ;COMPARA "H" COM "D"
'0001 8A      0015 CP      D              ;
'0002 3003    0016 LOOP3: JR      NC,LOOP1-$ ;SE HOUVE CARRY HL<DE
'0004 3E02    0017 LD      A,2          ;CODIGO PARA HL<DE
'0006 C9      0018 RET                      ;
'0007 2803    0019 LOOP1: JR      Z,LOOP2-$ ;SE NAO FOR ZERO HL>DE
'0009 3E01    0020 LD      A,01H        ;CODIGO PARA HL>DE
'000B C9      0021 RET                      ;
'000C 7D      0022 LOOP2: LD      A,L      ;COMPARA "L" COM "E"
'000D BB      0023 CP      E              ;
'000E 20F2    0024 JR      NZ,LOOP3-$ ;SE FOR ZERO HL=DE
'0010 97      0025 SUB      A            ;CODIGO PARA HL=DE
'0011 C9      0026 RET                      ;
0027 END

```

Fig. 9.18.A - Programa que compara "HL" com "DE".

9.5.4 - Comparação de Blocos

Às vezes, em certos programas, tem-se a necessidade de procurar um dado dentre um conjunto de instruções definido como bloco.

O microprocessador Z-80 possui uma instrução que procura um certo dado num determinado bloco; para isso deve-se carregar o acumulador com o dado a ser procurado, o par BC com a quantidade de BYTES existentes no bloco e o par "HL" com o endereço do início do bloco.

Esta instrução, ao ser executada, entra em LOOP e sairá deste se o conteúdo do par de registradores "BC" for igual a "zero", ou se o BYTE a ser procurado existe no bloco. No fim deste LOOP, basta apenas verificar o FLAG "Z"; se o dado for encontrado, este será setado, caso contrário, será reseta do.

Na figura 9.19 pode-se acompanhar o programa.

```
0001 *****
0002 *      PROGRAMA:      COMPARACAO DE BLOCO      *
0003 *      SISTEMA:      MICRO Z-80                *
0004 *      PROGRAMADOR:   ANDREZA                   *
0005 *      DATA:        10-FER-84                 *
0006 *****
0007
0008 CARREGAR O ACUMULADOR COM O DADO A SER PROCURADO
0009 CARREGAR "HL" COM O INICIO DO BLOCO
0010 CARREGAR "BC" COM A QUANTIDADE DE BYTE A SER PROCURADA
0011 O ACUMULADOR RETORNARA COM:
0012 A=0 SE ENCONTROU
0013 A=FF SE NAO ENCONTROU
'0000 EDB1 0014 BLOCO: CPIR          ;PROCURAR DADO
'0002 2002 0015 JR NZ,SET-$;SET O ACU. SE NAO ENCONT. O DADO
'0004 97   0016 SUB A          ;RESET O ACU. O DADO NAO ENCONTR.
'0005 C9   0017 RET
'0006 3EFF 0018 SET: LD A,0FFH ;SET O ACUM. DADO NAO ENCONTRADO
'0008 C9   0019 RET
0020 ;
```

Fig. 9.19 - Programa para comparação de blocos.

9.6 - Rotinas

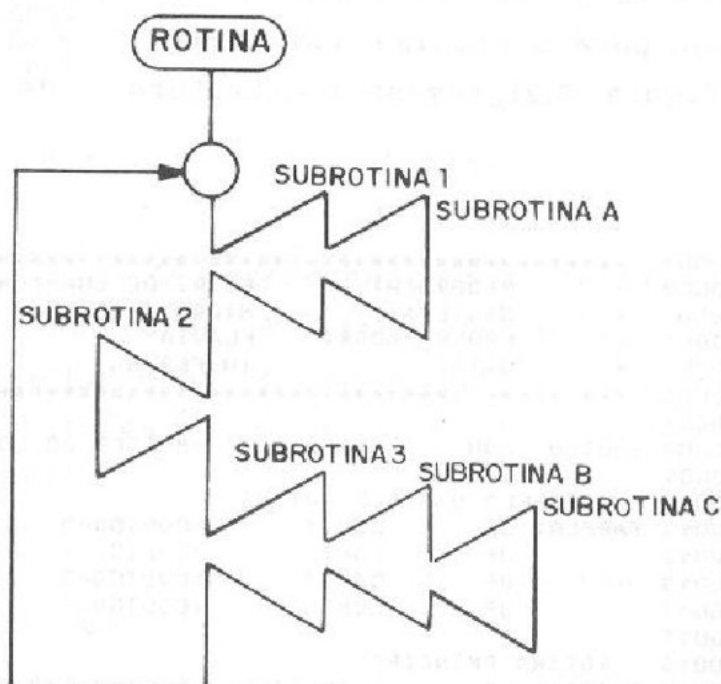
9.6.1 - Introdução

A maioria dos programas existentes no mercado são,

de uma forma geral, apresentados sobre a forma de rotinas. Por exemplo: Um programa que executa o controle de um maquinário.

Um programa de controle de algum dispositivo externo geralmente é formado por subrotinas como: DELAY, Transmissão e Recepção de dados (se for controlado por outro computador), Conversão de ASCII para binário, Cálculo de paridade do ASCII, Multiplicação e divisão de dados e outros.

Na figura 9.20, tem-se a estrutura de chamadas de subrotinas.



Exemplo:

Subrotina 1	Leitura de dados (BCD) de uma "PIO"
Subrotina A	Transforma "BCD" em "ASCII"
Subrotina 2	Transmissão de dados para um computador central
Subrotina 3	Análise dos dados recebida
Subrotina B	Executa uma determinada função
Subrotina C	Executa um DELAY

Fig. 9.20 - Estrutura de uma rotina.

9.6.2 - Estrutura para Chamada de Subrotinas

Quando se tem um programa formado por subrotinas, que podem ser chamadas dependendo de alguma condição que ocorra no transcorrer do programa, elas podem ser executadas, como se detalha a seguir.

Em primeiro lugar, colocam-se numa tabela todas as rotinas sendo chamadas por "JUMP", o par de registradores "HL" de verá ser carregado com o início da tabela e o acumulador terá o código da subrotina a ser executada, e esta por sua vez atualizará este código para a próxima subrotina.

Na figura 9.21, tem-se a estrutura de subrotina.

```

0001 *****
0002 *      PROGRAMA:      ESTR. DE CHAMADA DE SUB-ROTINAS *
0003 *      SISTEMA:      MICRO Z-80 *
0004 *      PROGRAMADOR:   FLAVIA *
0005 *      DATA:         10-FER-84 *
0006 *****
0007
>5520 0008 CODIGO EQU 5520H ;BUFFER DO CODIGO DA SUB-ROT.
0009
0010 TABELA DAS SUB ROTINAS
'0000 C31F00' 0011 TABELA: JP CONT1 ;CODIGO-0
'0003 C32000' 0012 JP CONT2 ;CODIGO-1
'0006 C32100' 0013 JP CONT3 ;CODIGO-2
'0009 C32200' 0014 JP CONT4 ;CODIGO-3
0015
0016 ROTINA PRINCIPAL
'000C 3A2055 0017 INICIO: LD A,(CODIGO) ;CODIGO DAS SUB-ROTINAS
'000F 210000' 0018 LD HL,TABELA ;INICIO DA TABELA
'0012 5F 0019 LD E,A
'0013 87 0020 ADD A,A ;MULTIPLICA POR DOIS
'0014 83 0021 ADD A,E ;MULTIPLICA POR TRES PORQUE A
0022 ;TABELA COMREENDE TRES BYTES
0023 ;PARA CADA SUB-ROTINA
'0015 5F 0024 LD E,A ;CARREGA O CODIGO REAL
'0016 1600 0025 LD D,00
'0018 19 0026 ADD HL,DE ;APONTA PARA A SUB-ROTINA
'0019 CD1E00' 0027 CALL EXECUT
'001C 18EE 0028 JR INICIO-4
0029
'001E E9 0030 EXECUT: JP (HL) ;EXECUTA A SUB-ROTINA CUJO
0031 ;QUE ESTA EM "HL"
0032
0033 SUB-ROTINA UM
'>001F 0034 CONT1:
0035
'001F C9 0036 RET
0037
0038
0039 SUB-ROTINA DOIS

```

```

'>0020      0040 CONT2:
            0041
            0042
'0020 C9      0043      RET
            0044
            0045 SUB-ROTINA TRES
'>0021      0046 CONT3:
            0047
            0048
'0021 C9      0049      RET
            0050
            0051 SUB-ROTINA QUATRO
'>0022      0052 CONT4:
            0053
            0054
'0022 C9      0055      RET
            0056      END

```

Fig. 9.21 - Estrutura de Subrotinas.

Este tipo de procedimento é muito usado na estrutura de programa de controle. A rotina apresentada tem a incumbência apenas de selecionar as subrotinas; estas, por sua vez, executarão as suas tarefas e atualizarão o código para a execução de outras rotinas.

Outra vantagem desta estrutura é que ela pode sofrer modificações ou acréscimos de outras subrotinas, sem alterar o fluxo das outras rotinas.

9.7 - Algoritmo

9.7.1 - Soma e Subtração

Neste item serão desenvolvidas a soma e subtração de MULTI-BYTES, isto é, de vários BYTES.

A soma e a subtração serão efetuadas utilizando-se posições de memórias, isto é, entre as posições do operando "B", e o resultado será carregado no operador "A".

Este programa será dividido em quatro partes, sendo soma e subtração em binário e soma e subtração em "BCD", e o número de BYTES utilizado será de quinze. Na figura 9.21, pode-se ver a estrutura e o fluxograma das operações.

OPERANDO A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
OPERANDO B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RESULTADO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Fig. 9.21.A - Estrutura da operação de soma e subtração.

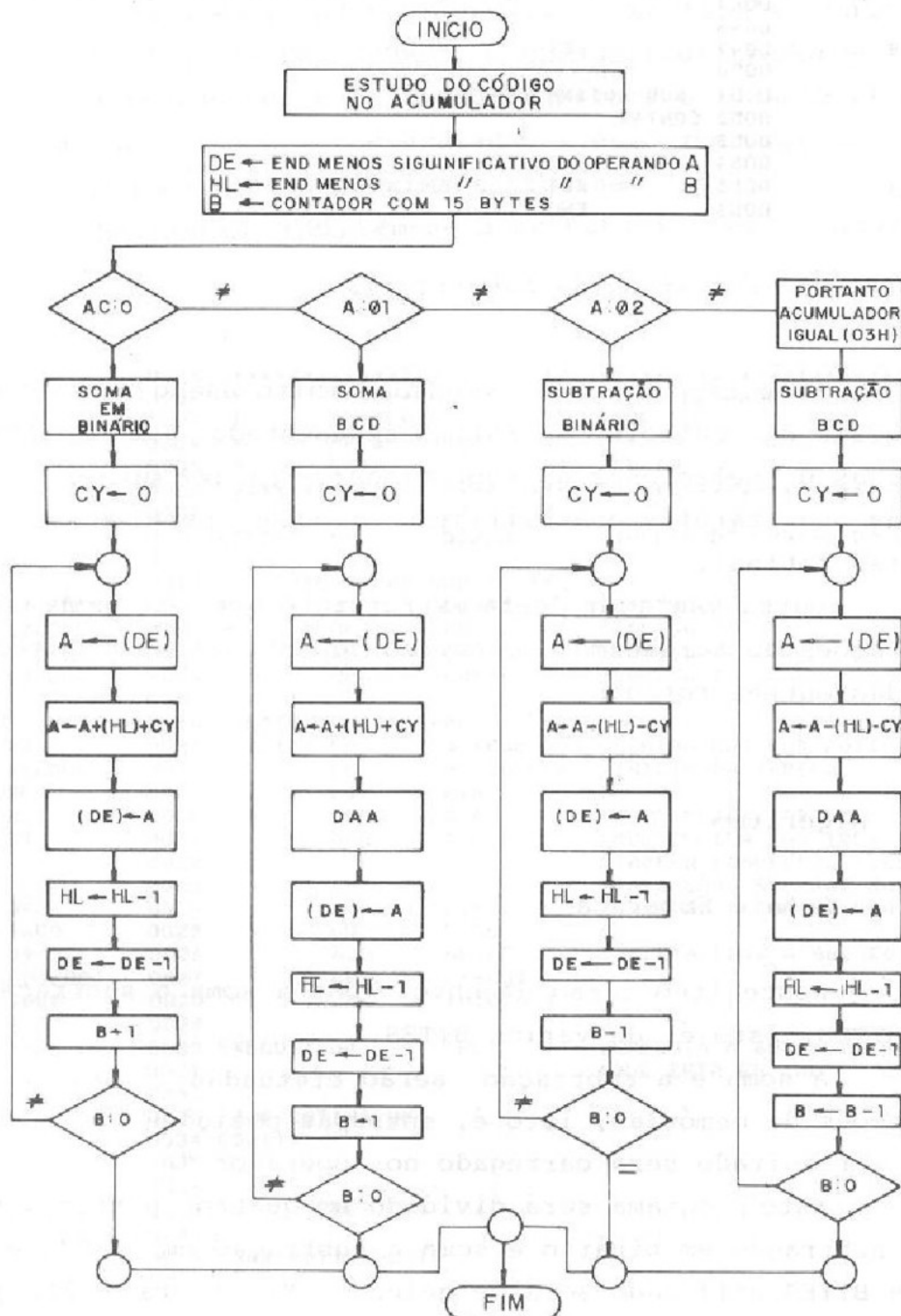


Fig. 9.22 - Fluxo da operação soma ou subtração.


```

0001 *****
0002 *      PROGRAMA:      SOMA E SUBTRCAO DE MULTI BYTE *
0003 *      SISTEMA:      MICRO Z-80 *
0004 *      PROGRAMADOR:   CRISTINE *
0005 *      DATA:         10-FER-84 *
0006 *****
0007
0008 OS DADOS DEVEM SER CARREGADO EM "OPERA" E "OPERB"
0009 O RESULTADO SERA CARREGADO EM "OPERA"
0010 NO ACUMULADOR DEVERA SER CARREGADO COM:
0011 A=0 SE FOR SOMA EM BINARIO
0012 A=1 SE FOR SOMA EM BCD
0013 A=2 SE FOR SUBTRACAO EM BINARIO
0014 A=3 SE FOR SUBTRACAO EM BCD
0015
>1000 0016 OPERA EQU 1000H ;ENDERECO DO OPERANDO "A"
>100F 0017 OPERB EQU OPERA+15 ;ENDERECO DO OPERANDO "B"
0018 ;
0019 ;
'0000 110E10 0020 INICIO: LD DE,OPERA+14 ;END. PRIM. BYTE DO OPER. A
'0003 211D10 0021 LD HL,OPERB+14 ;END. SEG. BYTE DO OPERANDO B
'0006 060F 0022 LD B,15 ;CONT. QUE INDICA O NUMERO
'0008 FE01 0023 CP 01 ;VERIFICA O CODIGO
'000A 3008 0024 JR NC,SOMBCD-$ ;SE CY=1 SOMA EM BINARIO
'000C A7 0025 AND A ;RESET CARRY
'000D 1A 0026 LOOP1: LD A,(DE) ;CARREGA O OPERANDO A
'000E 8E 0027 ADC A,(HL) ;SOMA COM OPERANDO "B"
'000F 12 0028 LD (DE),A ;GUARDA O RESULTADO
'0010 28 0029 DEC HL ;DECREMENTA OS PONTEIROS
'0011 18 0030 DEC DE
'0012 10F9 0031 DJNZ LOOP1-$
0032 ;
'0014 200A 0033 SOMBCD: JR NZ,SUBBIN-$ ;SE FOR ZERO E SOMA EM BCD
'0016 A7 0034 AND A ;RESET CARRY
'0017 1A 0035 LOOP2: LD A,(DE) ;CARREGA OPERANDO A
'0018 8E 0036 ADC A,(HL) ;SOMA COM OPERANDO B
'0019 27 0037 DAA ;AJUSTA PARA BCD
'001A 12 0038 LD (DE),A ;GUARDA O RESULTADO
'001B 28 0039 DEC HL ;DECREMENTA OS PONTEIROS
'001C 18 0040 DEC DE
'001D 10FB 0041 DJNZ LOOP2-$ ;VER. SE JA ACABOU A OPERACAO
'001F C9 0042 RET
0043 ;
'0020 FE02 0044 SUBBIN: CP 02H ;VER. SE E SUBTR. EM BINARIO
'0022 2009 0045 JR NZ,SUBBCD-$ ;SE DIF. ZERO SUBTRACAO BCD
'0024 A7 0046 AND A ;RESET CARRY
'0025 1A 0047 LOOP3: LD A,(DE) ;CARREGA OPERANDO A
'0026 9E 0048 SBC A,(HL) ;SUBTRAI DO OPERANDO B
'0027 12 0049 LD (DE),A ;GUARDA O RESULTADO
'0028 28 0050 DEC HL ;DECREMENTA OS PONTEIROS
'0029 28 0051 DEC HL
'002A 10F9 0052 DJNZ LOOP3-$ ;VERIF. JA ACABOU A OPERACAO
'002C C9 0053 RET
0054 ;
'002D A7 0055 SUBBCD: AND A ;RESET O CARRY
'002E 1A 0056 LOOP4: LD A,(DE) ;CARREGA OPERANDO A
'002F 9E 0057 SBC A,(HL) ;SUBTRAI O OPERANDO B
'0030 27 0058 DAA ;AJUSTA PARA BCD

```


'0031	12	0059	LD	(DE),A	GUARDA O RESULTADO
'0032	28	0060	DEC	HL	DECREMENTA OS PONTEIROS
'0033	28	0061	DEC	HL	
'0034	10F8	0062	DJNZ	LOOP4-4	VER. SE JA ACABOU A OPERACAO
'0036	C9	0063	RET		
		0064	END		

Fig. 9.22.A - Programa da operação de soma e subtração.

9.7.2 - Multiplicação

Em microprocessadores como 8080 e Z-80, não existe a operação de multiplicação. Para que um microprocessador realize a operação de multiplicação, é preciso elaborar uma rotina específica para se realizar tal operação.

A princípio existem dois métodos para realizar-se a multiplicação, utilizando-se microprocessadores. O primeiro é o método de somas sucessivas, e o segundo o de deslocamentos e somas. O primeiro método apresenta uma grande desvantagem, que é o tempo de execução. Por exemplo: deseja-se multiplicar três mil(3000) por dois mil(2000); para tal precisa-se somar consecutivamente duas mil(2000) vezes o valor três mil(3000). Como o tempo gasto em média, para a execução de uma instrução num microprocessador, é da ordem de microsegundos, para realizar-se uma simples contabilidade de uma firma, onde existem milhares de multiplicações, seriam gastas muitas horas, por esta razão o primeiro método não é utilizado. Na realização das operações de multiplicação é aplicado o método de soma e deslocamento, que consiste em analisar o BIT menos significativo do multiplicador. Se for "zero", apenas desloca-se o resultado; se for "um", soma-se o resultado com o multiplicando e desloca-se, repetindo-se este processo até que todos os BITS do multiplicador sejam examinados.

O programa dado a seguir executará a multiplicação do conteúdo do registrador "D" com o conteúdo do registrador "E", e o resultado será apresentado no par (HL). Na figura 9.23, tem-se o fluxo e o programa de multiplicação.

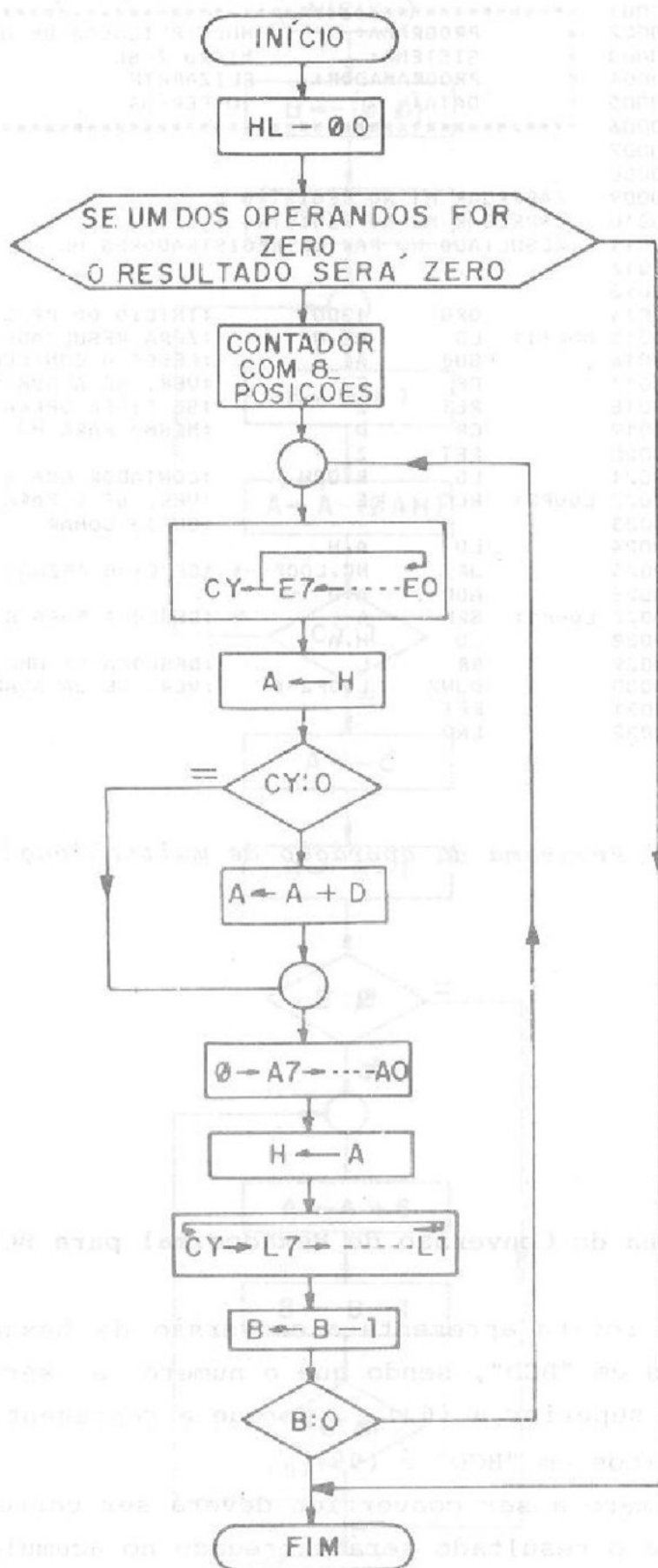


Fig. 9.23 - Fluxo da operação de multiplicação.

```

0001 *****
0002 *      PROGRAMA:      MULTIPLICACAO DE UM BYTE      *
0003 *      SISTEMA:      MICRO Z-80                    *
0004 *      PROGRAMADOR:   ELIZABETE                    *
0005 *      DATA:         10-FER-84                    *
0006 *****
0007
0008
0009      CARREGAR M1 NO REGISTRO D
0010      CARREGAR M2 NO REGISTRO E
0011      RESULTADO NO PAR DE REGISTRADORES HL
0012
0013
0014      ORG      1300      ; INICIO DO PROGRAMA
0015 MULT1: LD      HL,00      ; ZERA RESULTADO
0016      SUB      A          ; RESET O CONTEUDO DO ACULM.
0017      CP      E          ; VER. SE ALGUM OPERANDO E ZERO
0018      RET      Z          ; SE TIVER OPERACAO TERMINADA
0019      CP      D          ; MESMO PARA M2
0020      RET      Z
0021      LD      B,08H      ; CONTADOR COM 8 BITS
0022 LOOP2: RLC      E      ; VER. SE E PARA SOMAR E DESL.
0023      ; OU SO SOMAR
0024      LD      A,H
0025      JR      NC,LOOP1-$ ; SE CY=0 APENAS DESLOCA
0026      ADD      A,D
0027 LOOP1: SRL      A      ; DESLOCA PARA DIREITA
0028      LD      H,A
0029      RR      L          ; DESLOCA DE UMA POSICAO "L"
0030      DJNZ     LOOP2-$ ; VER. SE JA ACABOU A OPERACAO
0031      RET
0032      END

```

Fig. 9.23.A - Programa da operação de multiplicação.

9.7.3 - Rotina de Conversão de Hexadecimal para BCD

Esta rotina apresenta a conversão de hexadecimal para dois dígitos em "BCD", sendo que o número a ser convertido não poderá ser superior a $(63)_{16}$, porque a representação máxima de dois dígitos em "BCD" é $(99)_{10}$.

O número a ser convertido deverá ser carregado no registrador "C" e o resultado será carregado no acumulador.

Na figura 9.24, pode-se acompanhar o fluxo e o programa de conversão.

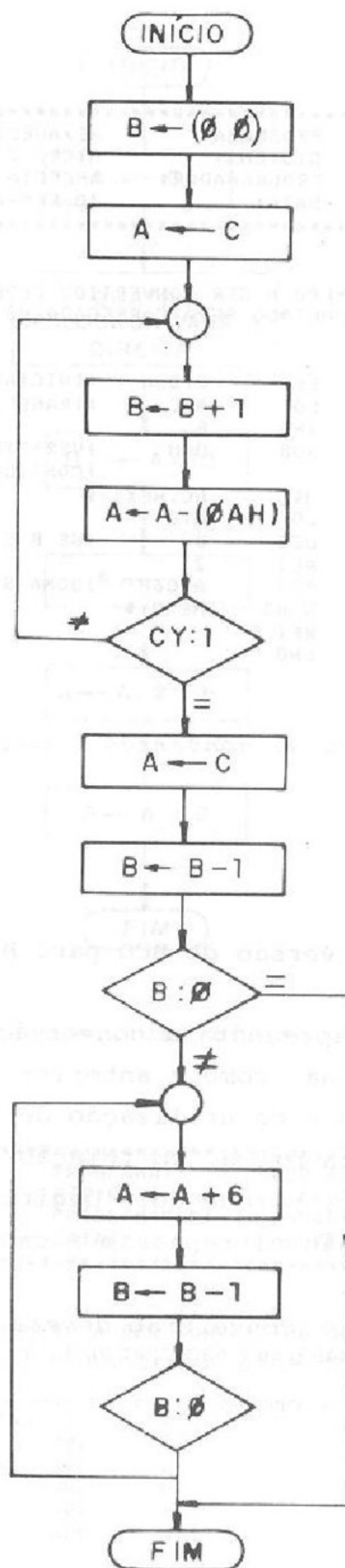


Fig. 9.24 - Fluxo da conversão de Hexadecimal para BCD.

```

0001 ; *****
0002 ;*      PROGRAMA:      HEXADECIMAL PARA BCD      *
0003 ;*      SISTEMA:      MICRO Z-80                *
0004 ;*      PROGRAMADOR:   ANDREIA                  *
0005 ;*      DATA:         10-FER-84                *
0006 ; *****
0007 ;
0008 ;
0009 ; O NUMERO A SER CONVERTIDO DEVERA ESTAR NO REGISTRO "C"
0010 ; O RESULTADO SERA CARREGADO NO ACUMULADOR
0011 ;
0012 ;
0000 0600 0013 HEXBCD: LD      B,00H      ;INICIALIZA "B" COM ZERO
0002 79    0014          LD      A,C      ;TRANSF. O RESULTADO PARA O ACUM.
0003 04    0015 HEX1:  INC      B
0004 D60A 0016          SUB      0AH      ;VERIFICA QUANTAS VEZES DEZ ESTA
0006 30FB 0017          ;CONTIDO NUMERO
0008 79    0018          JR      NC,HEX1-$
0009 05    0019          LD      A,C
000A C8    0020          DEC      B      ;SE B E ZERO A CONVERSAO TERMINOU
000B C606 0021          RET      Z
000D 10FC 0022 HEX2:  ADD      A,06H      ;SOMA SEIS NO ACUMULADOR
000F C9    0023          DJNZ    HEX2-$
0024          RET
0025          END

```

Fig. 9.24.A - Programa da conversão Hexadecimal para BCD.

9.7.4 - Rotina de Conversão de BCD para Hexadecimal

Esta rotina apresenta a conversão de BCD para Hexadecimal. Tanto esta rotina como a anterior são usadas em sistemas onde há necessidade de utilização de DISPLAY de sete segmentos ou similar. Esta rotina, em relação à anterior, não apresenta nenhuma limitação, porque dois dígitos em BCD $(99)_{10}$, e em Hexadecimal $(63)_{16}$, não ultrapassam a capacidade de um registrador que é de $(FF)_{16}$.

O número a ser convertido deverá ser carregado no acumulador e o resultado será armazenado também no acumulador. Na figura 9.25, pode-se acompanhar o fluxo e o programa.

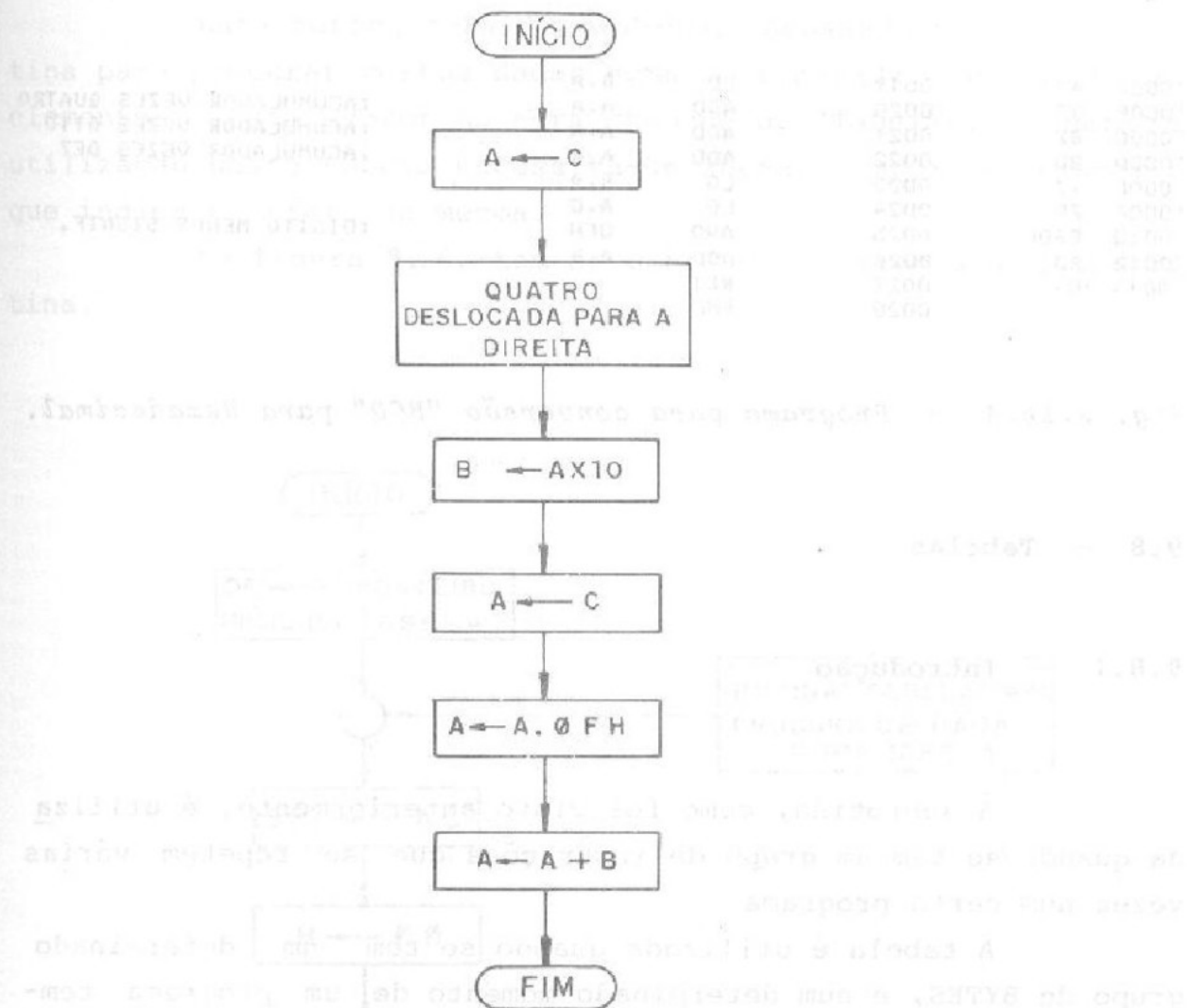


Fig. 9.25 - Fluxo da conversão BCD para Hexadecimal.

```

0001 ; *****
0002 ;*   PROGRAMA:   BCD PARA - HEXADECIMAL   *
0003 ;*   SISTEMA:   MICRO Z-80               *
0004 ;*   PROGRAMADOR: LAURA                 *
0005 ;*   DATA:     10-FER-B4                 *
0006 ; *****
0007 ;
0008 ;
0009 ; O NUMERO A SER CONVERTIDO DEVERA SER CARREGADO NO REG.
0010 ; "C" O RESULTADO SER CARREGADO NO ACUMULADOR
0011 ;
0012 ;
'0000 79      0013 BCDHEX: LD      A,C      ;NUMERO A SER CONVERTIDO
'0001 C83F    0014          SRL     A       ;RESET O DIG. MENOS SIGN.
'0003 C83F    0015          SRL     A,A
'0005 C83F    0016          SRL     A,A
'0007 C83F    0017          SRL     A,A
'0009 87      0018          ADD     A,A      ;ACUMULADOR VEZES DOIS

```

*000A	47	0019	LD	B,A	
*000B	87	0020	ADD	A,A	;ACUMULADOR VEZES QUATRO
*000C	87	0021	ADD	A,A	;ACUMULADOR VEZES OITO
*000D	80	0022	ADD	A,B	;ACUMULADOR VEZES DEZ
*000E	47	0023	LD	B,A	
*000F	79	0024	LD	A,C	
*0010	E60F	0025	AND	0FH	;DÍGITO MENOS SIGNIF.
*0012	80	0026	ADD	A,B	
*0013	C9	0027	RET		
		0028	END		

Fig. 9.25.A - Programa para conversão "BCD" para Hexadecimal.

9.8 - Tabelas

9.8.1 - Introdução

A subrotina, como foi visto anteriormente, é utilizada quando se tem um grupo de instruções que se repetem várias vezes num certo programa.

A tabela é utilizada quando se tem um determinado grupo de BYTES, e num determinado momento de um programa tem-se a necessidade de obter um desses BYTES.

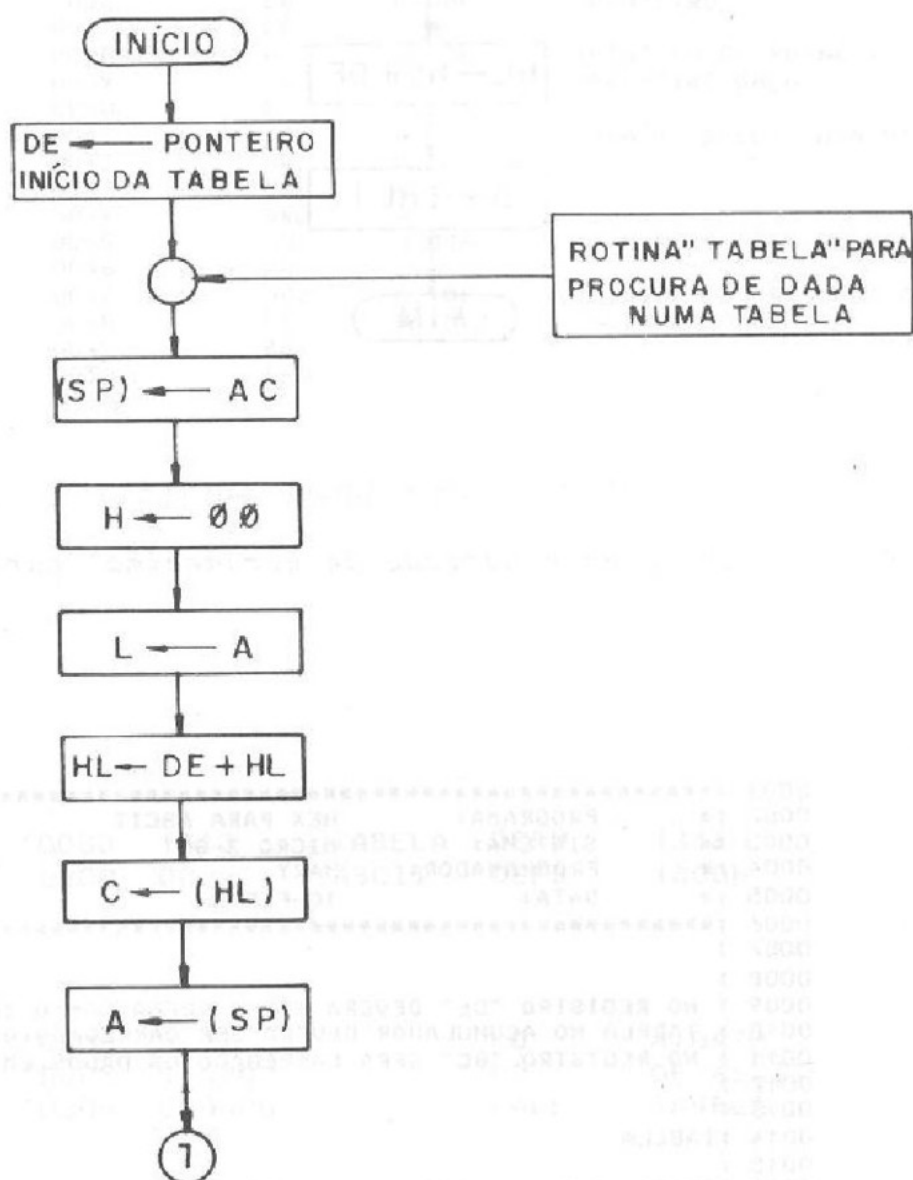
9.8.2 - Transformação de Hexadecimal para ASCII

Esta rotina é muito utilizada quando se deseja imprimir dados em hexadecimal, em uma determinada impressora que trabalha com o código ASCII.

Esta rotina irá utilizar a filosofia de tabela, isto é, os dados a serem transformados são colocados numa tabela com um ponteiro indicando o início da tabela. O mecanismo desta rotina é o seguinte: o BYTE a ser transformado deverá ser carregado no acumulador. Como o acumulador é formado por "8 BITS", a parte mais significativa "DADO A" (BIT 7 a 4), e a menos significativa "DADO B" (BIT 3 a 0), cada um destes dados corresponderá a um código em ASCII, que será carregado no par de registradores "BC".

Esta rotina será desenvolvida, pensando-se numa subro-
tina para procurar certos dados numa determinada tabela de 16
elementos. Esta subrotina será chamada de "Tabela", e para a
utilização desta tabela necessita-se apenas mudar o ponteiro
que indica o início da mesma.

Na figura 9.26, tem-se o fluxo e o programa desta ro-
tina.



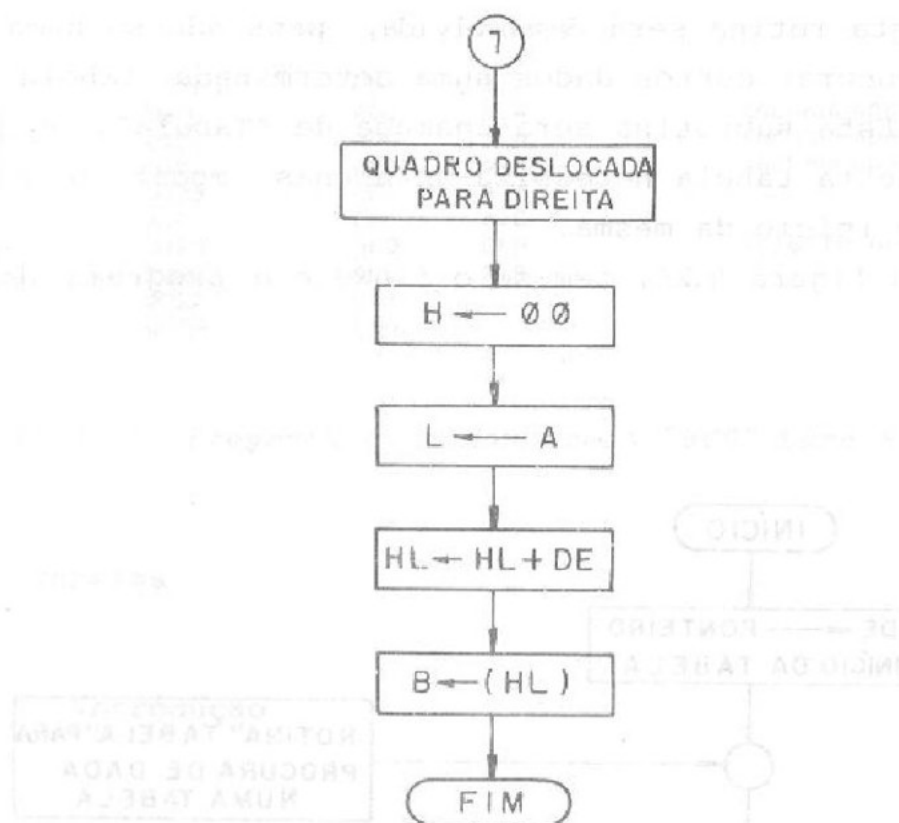


Fig. 9.26 - Fluxo para conversão de Hexadecimal para ASCII.

```

0001 ;*****
0002 ;*      PROGRAMA:      HEX PARA ASCII      *
0003 ;*      SISTEMA:      MICRO Z-80          *
0004 ;*      PROGRAMADOR:   MARY                *
0005 ;*      DATA:         10-FER-84           *
0006 ;*****
0007 ;
0008 ;
0009 ; NO REGISTRO "DE" DEVERA SER CARREGA COM O INICIO DA
0010 ; TABELA NO ACUMULADOR DEVERA SER CARREGADO O DADO
0011 ; NO REGISTRO "BC" SERA CARREGADO OS DADOS EM ASCII
0012 ;
0013 ;
0014 ;TABELA
0015 ;
0016 ASCII:  DEFB      30H      ;"1"
0017         DEFB      31H      ;"2"
0018         DEFB      32H      ;"3"
  
```

'0000 30
'0001 31
'0002 32

'0003	33	0019	DEFB	33H	;"4"
'0004	34	0020	DEFB	34H	;"5"
'0005	35	0021	DEFB	35H	;"6"
'0006	36	0022	DEFB	36H	;"7"
'0007	37	0023	DEFB	37H	;"8"
'0008	38	0024	DEFB	38H	;"9"
'0009	39	0025	DEFB	39H	;"A"
'000A	41	0026	DEFB	41H	;"B"
'000B	42	0027	DEFB	42H	;"C"
'000C	43	0028	DEFB	43H	;"D"
'000D	44	0029	DEFB	44H	;"E"
'000E	45	0030	DEFB	45H	;"F"
		0031			
		0032	PROGRAMA		
		0033			
'000F	F5	0034	TABELA: PUSH	AF	;"GUARDA O DADO A SER TRANSF."
'0010	E60F	0035	AND	0FH	;"MASCARA DO DADO MENOS SIGNIF."
'0012	2600	0036	LD	H,00H	;"PONTEIRO"
'0014	6F	0037	LD	L,A	
'0015	19	0038	ADD	HL,DE	;"POSICAO DO PRIMEIRO DADO"
'0016	4E	0039	LD	C,(HL)	;"PRIMEIRO DADO"
'0017	F1	0040	POP	AF	
'0018	CB3F	0041	SRL	A	;"QUATRO DESL. PARA DIREITA"
'001A	CB3F	0042	SRL	A	
'001C	CB3F	0043	SRL	A	
'001E	CB3F	0044	SRL	A	
'0020	2600	0045	LD	H,00H	;"PONTEIRO PARA SEGUNDO DADO"
'0022	6F	0046	LD	L,A	
'0023	19	0047	ADD	HL,DE	;"POSICAO DO SEGUNDO DADO"
'0024	46	0048	LD	B,(HL)	;"SEGUNDO DADO"
'0025	C9	0049	RET		
		0050	END		

Fig. 9.26.A - Programa Hexadecimal para ASCII.

Exemplo:

'0000	1613	TABELA	DEFW	1316H
'0002	00	ASCII	DEFB	1300H
'0003	3EA7		LD	A,0A7H
'0005	110200		LD	DE,ASCII
'0008	CD0000		CALL	TABELA

RESULTADO

REGISTRADOR B 41H
REGISTRADOR C 37H

9.8.3 - Transformação de ASCII para Hexadecimal

Esta rotina tem a finalidade de transformar um dado ASCII em Hexadecimal. Este dado deverá estar no acumulador, e o resultado será carregado no mesmo.

Pela tabela 9.27.A pode-se observar que aos números de (30 a 39) basta subtrair (30H), e aos números de (41 a 42) basta subtrair (37H). Fazendo-se esta operação o número será transformado para hexadecimal.

Na figura 9.27, pode-se acompanhar o fluxo desta rotina.

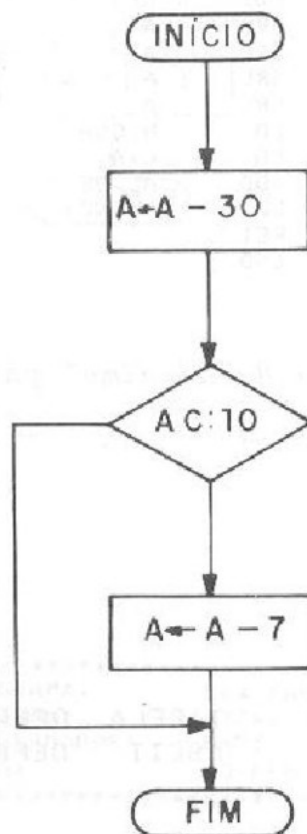


Fig. 9.27 - Fluxo da rotina.

B)	HEX	ASCII
	30	0
	31	1
	32	2
	33	3
	34	4
AC - 30	35	5
	36	6
	37	7
	38	8
	39	9
	41	A
	42	B
AC - 30 - 7	43	C
	44	D
	45	E
	46	F

Fig. 9.27.A - Códigos do ASCII.

```

0001 ;*****
0002 ;*      PROGRAMA:      ASCII PARA HEXADECIMAL      *
0003 ;*      SISTEMA:      MICRO Z-80                  *
0004 ;*      PROGRAMADOR:   TANIA                      *
0005 ;*      DATA:         10-FER-B4                  *
0006 ;*****
0007 ;
0008 ;
0009 ; NO ACUMULADOR DEVERA SER CARREGADO A DADO
0010 ; A CONVERSAO SERA CARREGADO NO ACUMULADOR
0011 ;
0012 ;
'0000 D630 0013 ASCHEX: SUB      30H      ;SUBTRAI "30"
'0002 FE0A 0014 CP          10      ;VERIFICA SE ESTA ENTRE 0 E 9
'0004 D8    0015 RET          C      ;RETORNA SE HOUVE CARRY
'0005 D6D7 0016 SUB      07H      ;SUBTRAI SETE
'0007 C9    0017 RET          ;RETORNA
0018 END

```

Fig. 9.27.B - Programa de conversão de ASCII para hexadecimal.

9.8.4 - Conversão de Hexadecimal para Sete Segmentos

Usa-se esta rotina quando se utiliza DISPLAY.

O intuito deste programa é o de transformar o conteúdo do acumulador composto por dois dados, o mais significativo (dado A), e o menos significativo (dado B), num código para poder acionar o DISPLAY de sete segmentos.

O dado a ser transformado deverá ser carregado no acumulador, e o resultado será carregado no par de registradores "BC".

Este programa utilizará uma tabela, e, para a formação desta tabela, pode-se acompanhar a figura 9.28.

A)	DIGITO	CÓDIGO
	0	3FH
	1	06H
	2	5BH
	3	4FH
	4	66H
	5	6DH
	6	7DH
	7	07H
	8	7FH
	9	6FH
	A	77H
	B	7CH
	C	39H
	D	5EH
	E	79H
	F	71H

Fig. 9.28 - Código de sete segmentos

B)

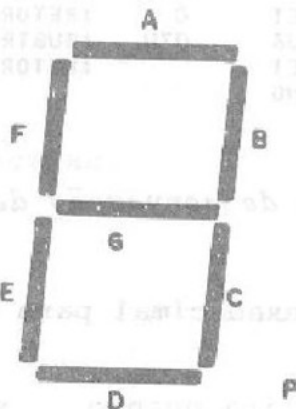


Fig. 9.28.A - DISPLAY de sete segmentos.

BIT	7	6	5	4	3	2	1	1
	*	Sg	Sf	Se	Sd	Sc	Sb	Sa

* O BIT 7 pode corresponder ao ponto dependendo da aplicação.

Fig. 9.29 - Tabela para DISPLAY de sete segmentos.



Fig. 9.29.A - Fluxo para transformação de Hexadecimal em sete segmentos.

Exemplo:

		0001					
		0002					
		0003					
		0004					
		0005					
		0006	:		COOIG0	-	DIGITO
'0000	3F	0007	SETSEG	DEFB	03FH	:	"0"
'0001	06	0008		DEFB	006H	:	"1"
'0002	58	0009		DEFB	058H	:	"2"
'0003	4F	0010		DEFB	04FH	:	"3"
'0004	66	0011		DEFB	066H	:	"4"
'0005	60	0012		DEFB	060H	:	"5"
'0006	70	0013		DEFB	070H	:	"6"
'0007	07	0014		DEFB	007H	:	"7"

'0008	7F	0015	DEFB	07FH	;"8"
'0009	6F	0016	DEFB	06FH	;"9"
'000A	77	0017	DEFB	077H	;"A"
'000B	7C	0018	DEFB	07CH	;"B"
'000C	39	0019	DEFB	039H	;"C"
'000D	5E	0020	DEFB	05EH	;"D"
'000E	79	0021	DEFB	079H	;"E"
'000F	71	0022	DEFB	071H	;"F"
		0023			
		0024			
		0025			
		0026			
		0027			
		0028			
'0010	1613	0029	TABELA	DEFW	1316H
		0030			
		0031			
		0032			
		0033			
'0012	3E87	0034	LD	A,087H	;"DADO A SER CONVERTIDO"
'0014	110000'	0035	LD	DE,SETSEG	;"INICIO DA TABELA"
'0017	CD1000'	0036	CALL	TABELA	;"PROCURA OS DADOS NA TABELA"
		0037	END		

Fig. 9.29.B - Programa Hexadecimal para BCD.

Conteúdo do Registrador B 7F H
 Conteúdo do Registrador C 71 H

9.9 - Dispositivo I/O

9.9.1 - Introdução

Neste item serão apresentadas rotinas que utilizam dispositivos de entrada e saída com CTC e PIO. Em algumas rotinas será apresentada a estrutura de chamada de programa por interrupção.

9.9.2 - DELAY por CTC

Nesta rotina será apresentada uma rotina onde, através do "CTC", pode-se executar um DELAY.

Quando se lida com "CTC", o primeiro passo é, saber que tipo de contagem será utilizada, neste caso deseja-se um DELAY múltiplo de (10ns ou 100 Hz). O segundo passo é determi

nar se o CLOCK será externo ou o próprio CLOCK do sistema, e neste caso, por conveniência, será usado o CLOCK do sistema que é de 2,4765 MHz, que é automaticamente injetado na pastilha.

Com a descrição dada, chega-se à conclusão que o modo de operação será o modo TIMER. Neste modo, contam-se os pulsos de CLOCK interno e divide-se a frequência de entrada por 16 ou 255, e pelo seu registrador interno, que pode variar entre (255 a 0).

Para cálculo do valor do registrador interno do CTC, basta apenas aplicar-se a fórmula dada a seguir:

$$R = \frac{Fc}{D \cdot Fp} \quad \text{onde,}$$

R = Valor do registrador interno do CTC

Fc = Frequência do CLOCK injetado no CTC

D = Fator (Prescale) 255 ou 16

Fp = Frequência pedida

Exemplo:

$$Fc = 2,4756 \text{ MHz}$$

$$D = 256$$

$$FP = 100 \text{ HZ}$$

$$R = (97)_{10} = (61)_{16}$$

Obs.: Se o fator "D" for igual a 16, o valor do registrador "R" será superior a 255.

Para se determinar as palavras de controle, ver volume I, capítulo "CTC".

Na figura 9.30, temos o programa, o qual está sob a forma de subrotina, mas poderá ser juntado com qualquer outro programa onde o FLAG "TIMOUT" poderá ser verificado no transcorrer de qualquer outro programa.

Neste programa foi utilizado apenas um contador do

CTC, os outros poderiam ser utilizados para contagens de outros tempos.

```

0001 *****
0002 *      PROGRAMA:      DELAY POR CTC      *
0003 *      SISTEMA:      MICRO Z-80        *
0004 *      PROGRAMADOR:   MIRIAN            *
0005 *      DATA:         10-FER-84         *
0006 *****
0007
0008      CARREGAR NA POSICAO DE MEMORIA DESEGNADA POR TIMOUT O
0009      VALOR DO DELAY
0010
0011      ENDERECAMENTO DOS DISPOSITIVO DE ENTRADA E SAIDA
0012
>0081      0013 CTCA0 EQU 81H ;ENDERECO DO "CTC0"
>0082      0014 CTCA1 EQU 82H ;ENDERECO DO "CTC1"
>0083      0015 CTCA2 EQU 83H ;ENDERECO DO "CTC2"
>0084      0016 CTCA3 EQU 84H ;ENDERECO DO "CTC3"
0017
>2000      0018 TIMOUT EQU 2000H ;POSICAO DO DELAY
0019      MAPEAMENTO DAS INTERRUPCOES
0020
'0000 2900' 0021 INT1: DEFW RCTCO ;INTERRUPCAO DO "CTCO"
0022
0023      PROGRAMA PRINCIPAL
0024
'0002 3E00 0025 DELCTC: LD A,00H ;VETOR DE INTERRUPCAO
'0004 ED47 0026 LD I,A
'0006 ED5E 0027 IM 2 ;MODO DOIS DE INTERRUPCAO
'0008 DE81 0028 LD C,CTCA0 ;ENDERECO DO "CTCO"
'000A 3E03 0029 LD A,03H ;CODIGO PARA RESET DE "CTC"
'000C 0604 0030 LD B,04H ;CONTADOR PARA QUATRO CTC
'000E ED79 0031 VOLT1: OUT (C),A ;RESET "CTC"
'0010 0C 0032 INC C ;PROXIMO "CTC"
'0011 10FB 0033 DJNZ VOLT1-$
'0013 FB 0034 EI ;HABILITA A INTERRUPCAO
'0014 210000' 0035 LD HL,INT1 ;CODIGO DE INTER. PARA "CTC"
'0017 7D 0036 LD A,L
'0018 D381 0037 OUT (CTCA0),A ;VETOR PARA "CTC"
'001A 3EA5 0038 LD A,0A5H ;PALAVRA DE TRABALHO DO CTC
'001C D381 0039 OUT (CTCA0),A
'001E 3E61 0040 LD A,61H ;CONTADOR COM 100Hz
'0020 D381 0041 OUT (CTCA0),A
'0022 3A0020 0042 VOLT2: LD A,(TIMOUT) ;VER. SE O DELAY TERMINOU
'0025 A7 0043 AND A ;ESTA EM ZERO?
'0026 20FA 0044 JR NZ,VOLT2-$ ;SE FOR ZERO RETORNE
'0028 C9 0045 RET
0046
0047      ROTINA DE INTER.
0048
'0029 08 0049 RCTCO: EX AF,AF' ;SALVA ACUMULADOR
'002A 3A0020 0050 LD A,(TIMOUT) ;DECREMENTA "TIMOUT"
'002D A7 0051 AND A ;VER. SE TIMOUT E ZERO
'002E 2804 0052 JR Z,FIH-$

```

```

'0030 3D      0053 DEC    A
'0031 320020  0054 LD     (TIMOUT),A ;GUARDA O VALOR DECREMENTADO
'0034 08      0055 FIM:  EX    AF,AF' ;RESTAURA O ACUMULADOR
'0035 FB      0056 EI     ;HABILITA A INTER.
'0036 ED4D    0057 RETI
          0058 END

```

Fig. 9.30 - Programa do contador.

Exemplo:

Deseja-se um DELAY de 100 ms:

$$\text{TIMOUT} = \frac{T}{T_{\text{CTC}}}$$

TIMOUT = BUFFER que é decrementado a cada interrupção

T = Tempo de DELAY desejado

T_{CTC} = Tempo de interrupção do CTC, neste caso é de 10 ms

$$\text{TIMOUT} = 100 \text{ ms} = (10)_{10} = (0A)_{16}$$

```
LD A,0AH
```

```
LD (TIMOUT),A 100 ms
```

```
CALL DELCTC
```

9.9.3 - PIO

A PIO é o dispositivo que é usado como interface paralela controlada pela CPU. No volume I, no capítulo "PIO", foi estudada toda a sua estrutura interna; neste capítulo, serão mostradas algumas aplicações.

Este dispositivo oferece quatro modos de utilização. Na figura 9.31, tem-se as aplicações destes modos.

```

0001
0002
0003
0004
>008F 0005 PI01AC EQU 8FH ;PORTO "A"
>0091 0006 PI01BC EQU 91H ;PORTO "B"
0007

```

		A)	MODO "ZERO"	
'0000	3E0F	LD	A,00001111B ;CODIGO DE SAIDA	
'0002	D38F	OUT	(PI01AC),A	
'0004	D391	OUT	(PI01BC),A	
		B)	MODO "UM"	
'0006	3E4F	LD	A,01001111B ;CODIGO DE ENTRADA	
'0008	328F00	LD	(PI01AC),A	
'000B	329100	LD	(PI01BC),A	
		C)	MODO "DOIS"	
'000E	3E8F	LD	A,10001111B ;CODIGO DE BIDEREACIONAL	
'0010	328F00	LD	(PI01AC),A	
'0013	328F00	LD	(PI01AC),A	
		D)	MODO "TRES"	
'0016	3ECF	LD	A,11001111B ;CONTROLE DO MODO TRES	
'0018	D38F	OUT	(PI01AC),A	
'001A	D391	OUT	(PI01BC),A	
'001C	3E81	LD	A,10000001B ;BIT "7" E "0" COMO ENTRADA E ;E OS RESTANTES COMO SAIDA	
'001E	D38F	OUT	(PI01AC),A	
'0020	3EF0	LD	A,11110000B ;BIT "7,6,5,4" COMO ENTRADA E ;OS RESTATE COMO SAIDA	
'0022	D391	OUT	(PI01BC),A	

OBS: A nomenclatura PI01AD, PI01AC, PI01BD, PI01BC significa.

PIO	Porto
A,B	Porto A ou Porto B
1,2,3	Número do porto
D,C	Dado ou controle

Fig. 9.31 - Programação da PIO.

9.9.4 - PIO como Contador

Neste programa será apresentada uma estrutura para contagem de peças numa esteira.

Na figura 9.32, pode-se acompanhar a estrutura do programa.

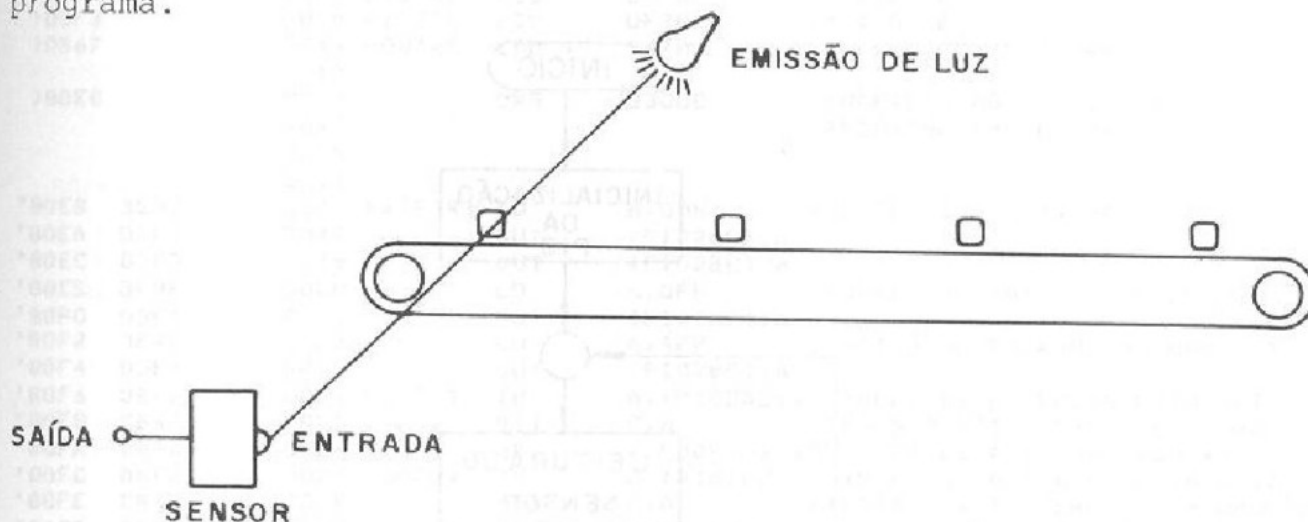


Fig. 9.32 - Estrutura para contagem de unidade em uma esteira.

A saída do sensor será acoplada a um "BIT" de um porto que, no caso, será o BIT "0" do porto B, e no porto A, teremos dois DISPLAYS, que contarão de 0 a 99 unidades.

Na figura 9.33, tem-se o esquema de ligação entre a CPU, PIO e o SENSOR.

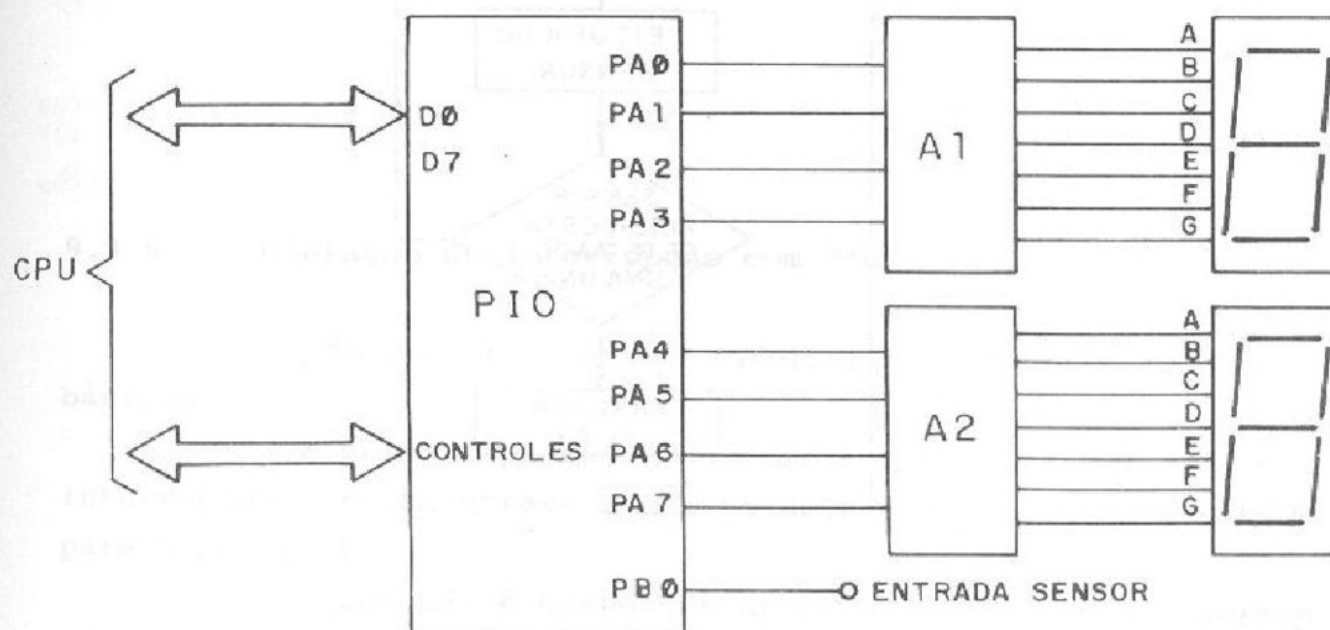


Fig. 9.33 - Esquema de ligação entre CPU, PIO e SENSOR.

O porto "A" será programado como saída para acionamento dos DISPLAYS; já o porto "B", como entrada.

Na figura 9.34, tem-se o fluxo e o programa de contagem de peças numa esteira.



Fig. 9.34 - Fluxo de contagem de unidade.

```

0001 *****
0002 *      PROGRAMA:      PIO COM CONTADOR      *
0003 *      SISTEMA:      MICRO Z-80      *
0004 *      PROGRAMADOR:    FERNANDA      *
0005 *      DATA:          10-FER-84      *
0006 *****
0007
0008
0009 PIO2AC EQU 0F1H ;PORTO "A"
0010 PIO2BC EQU 0F3H ;PORTO "B"
0011 CONTAD EQU 567H ;BUFFER DE CONTADOR
0012
0013 ORG 33000 ;ENDEREÇO DO INÍCIO DO
0014 ;PROGRAMA EM DECIMAL
0015
0016
0017 EXTEIR: LD A,03H ;DESABILITA A INTERRUPÇÃO
0018 OUT (PIO2AC),A
0019 OUT (PIO2BC),A
0020 LD A,0FH ;CÓDIGO DE SAÍDA NO MODO "0"
0021 OUT (PIO2AC),A
0022 LD A,4FH ;CÓDIGO DE ENTRADA NO MODO "1"
0023 OUT (PIO2BC),A
0024 LOOP1: IN A,(PIO2BC-1) ;VER. SE O SENSOR ESTÁ ACIO.
0025 BIT 0,A ;TESTA O BIT "ZERO" DO ACUMUL.
0026 JR Z,LOOP1-$ ;SE ESTIVER ACIONADO CONT+1
0027 LOOP2: IN A,(PIO2BC-1) ;VER. SE O SENSOR ESTÁ ACIO.
0028 BIT 0,A ;TESTA O BIT "ZERO" DO ACUMULADOR
0029 JR NZ,LOOP2-$ ;FICA EM LOOP ATÉ QUE O
0030 ;SENSOR SEJA DESATIVADO
0031 LD A,(CONTAD) ;BUFFER DO CONTADOR
0032 ADD A,01H ;INCREMENTA CONTADOR
0033 DAA ;AJUSTA PARA DECIMAL
0034 LD (CONTAD),A ;GUARDA CONTADOR CORRIGIDO
0035 OUT (PIO2AC-1),A ;ATUALIZAR O PORTO
0036 JR LOOP1-$
0037 END

```

Fig. 9.35 - Programa de contagem de unidade.

9.9.5 - Operação de Interrupção com PIO

A PIO pode operar com interrupção de duas maneiras básicas:

- a primeira é quando se opera nos modos 0/1 e 2, e a interrupção é feita através do pino ASTB para Porto "A", e BSTB para o porto "B".

- a segunda é quando se opera no modo 3, e a interrupção pode ser feita se todos os BITS estão ativos ou se apenas um está ativo.

Na figura 9.36, tem-se a aplicação de interrupção.

A) Interrupção pelo STB

	0001		
	0002		
	0003		
	0004		
	0005		
	0006		
'0000	3E4F	LD	A,01001111B ;DEFINE COMO MODO DE SAIDA
'0002	03A4	OUT	(0A4H),A
'0004	3E04	LD	A,04H ;VETOR DE INTERRUPTAO
'0006	03A4	OUT	(0A4H),A
'0008	3E87	LD	A,10000111B ;HABILITA A INTERRUPTAO
'000A	03A4	OUT	(0A4H),A
	0013		
	0014		
	0015		

B) Interrupção pelo próprio BIT

	0016		
	0017		
	0018		
	0019		
'000C	3ECF	LD	A,11001111B ;PALAVRA DE CONTR. MODO TRES
'000E	0378	OUT	(78H),A
'0010	3E80	LD	A,10000000B ;APENAS O BIT 7 COMO ENTRADA
'0012	034E	OUT	(78),A
'0014	3EF4	LD	A,0F4H ;VETOR DE INTERRUPTAO
'0016	034E	OUT	(78),A
'0018	3E87	LD	A,10110111B ;BIT 7 HABILITA A INTERRUPTAO
	0027		;BIT 6 INTER. EM UM UNICO BIT
	0028		;BIT 5 LOGICA POSITIVA
	0029		;BIT 4 MASCARA DA INTERRUPTAO
	0030		;BIT 3,2,1,0 CONTROLE
'001A	034E	OUT	(78),A
	0032		
	0033		
	0034		

Fig. 9.36 - Modos de interrupção da PIO.

9.9.6 - Contador Crescente e Decrescente

Este programa através de interrupção poderá incrementar e decrementar um contador de dois dígitos.

O controle será feito por interrupção através dos pínos "B7", para incrementar, e "B6", para decrementar do porto "B". No porto "A", serão colocados dois DISPLAYS.

Este programa poderá ser aplicado em controle de maquinário como prensa, injetora, etc..

Na figura 9.37, tem-se o esquema de um contador crescente e decrescente.

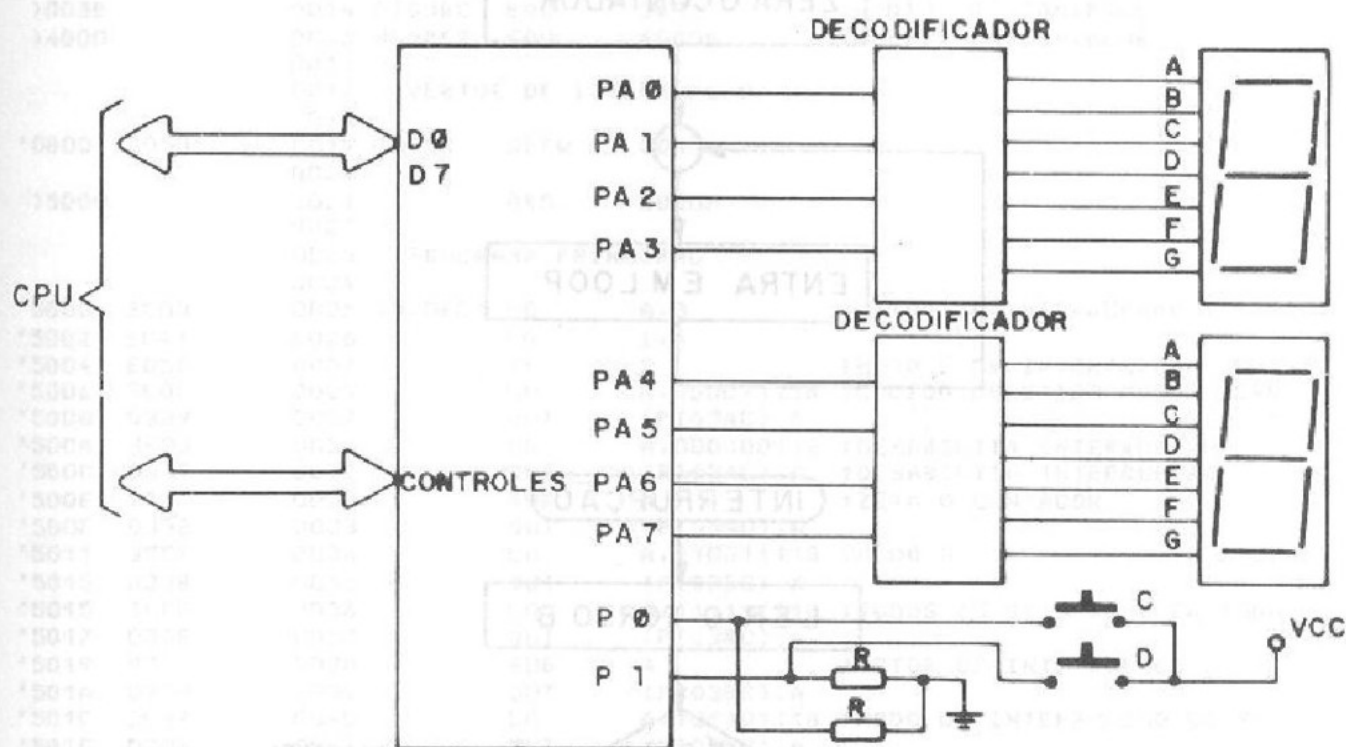


Fig. 9.37 - Esquema de um contador.

Nas figuras 9.38, pode-se acompanhar o fluxo e o programa do contador.



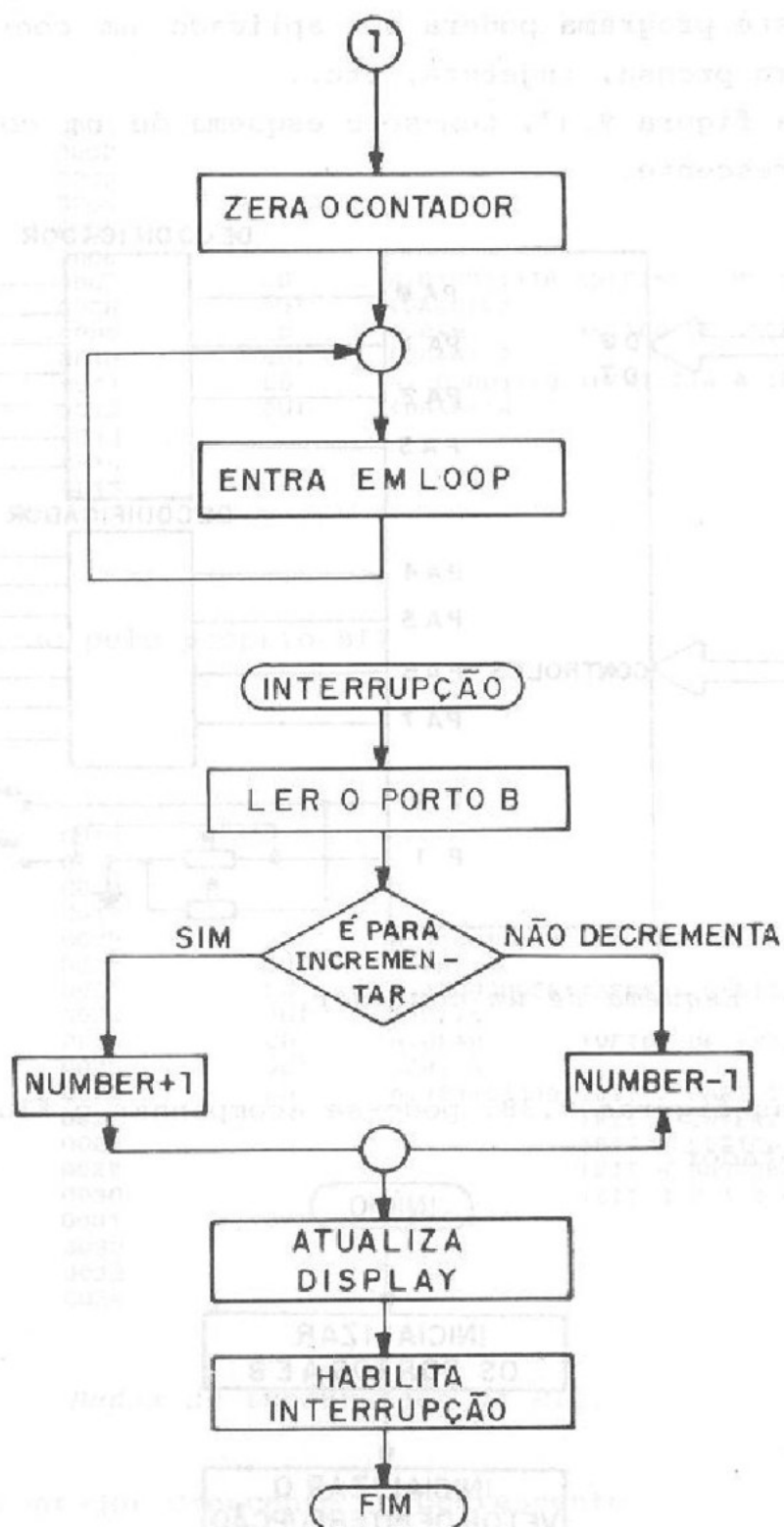


Fig. 9.38 - Fluxo do contador.


```

0001 *****
0002 *      PROGRAMA:      CONTADOR      *
0003 *      SISTEMA:      MICRO Z-80      *
0004 *      PROGRAMADOR:    CAROLINA      *
0005 *      DATA:          10-FER-84      *
0006 *****
0007
0008
0009      ENDERECAAMENTO DO PORTO
0010
>0038 0011 PIO3AD EQU 38H ;PORTO "A" DADO
>0039 0012 PIO3AC EQU 39H ;PORTO "A" CONTROLE
>003A 0013 PIO3BD EQU 3AH ;PORTO "B" DADO
>003B 0014 PIO3BC EQU 3BH ;PORTO "B" CONTROLE
>4000 0015 NUMBER EQU 4000H ;BUFFER DO CONTADOR
0016
0017      VETOR DE INTERRUPCAO
0018
'0000 2850' 0019 INTER DEFW CONTA
0020
>5000 0021 ORG 5000H
0022
0023      PROGRAMA PRINCIPAL
0024
'5000 3E00 0025 INCDEC: LD A,0 ;VETOR DE INTERRUPCAO
'5002 ED47 0026 LD I,A
'5004 ED5E 0027 IM 2 ;MODO 2 DE INTERRUPCAO
'5006 3E0F 0028 LD A,00001111B ;CODIGO DE SAIDA MODO "ZERO"
'5008 D339 0029 OUT (PIO3AC),A
'500A 3E03 0030 LD A,00000011B ;DESABILITA INTERRUPCAO
'500C D339 0031 OUT (PIO3AC),A ;DESABILITA INTERRUPCAO
'500E 97 0032 SUB A ;ZERA O CONTADOR
'500F D338 0033 OUT (PIO3AD),A
'5011 3ECF 0034 LD A,11001111B ;MODO 3
'5013 D33B 0035 OUT (PIO3BC),A
'5015 3EFF 0036 LD A,11111111B ;TODOS OS BITS COMO ENTRADA
'5017 D33B 0037 OUT (PIO3BC),A
'5019 97 0038 SUB A ;VETOR DE INTERRUPCAO
'501A D33B 0039 OUT (PIO3BC),A
'501C 3EB7 0040 LD A,10110111B ;MODO DE INTERRUPCAO DO PIO
'501E D33B 0041 OUT (PIO3BC),A
'5020 3E3F 0042 LD A,00111111B ;HAB. A INTER. BITS 7 E 6
'5022 D33B 0043 OUT (PIO3BC),A
'5024 F8 0044 EI ;HABILITA A INTERRUPCAO
'5025 C32550' 0045 WAIT: JP WAIT ;ENTRA EM LOOP ESPERANDO INT.
0046
0047
0048      PROGRAMA DE INTERRUPCAO
0049
'5028 F5 0050 CONTA: PUSH AF ;SALVA ACUMULADOR
'5029 DB3A 0051 IN A,(PIO3BD) ;VERIF. SE E PARA INCREM. OU
0052 ;DECREMENTAR
'5028 CB7F 0053 BIT 7,A
'502D 3A0040 0054 LD A,(NUMBER)
'5030 2004 0055 JR NZ,CRES-$ ;SE O BIT 7 FOR "UM" E CRESC.
'5032 D601 0056 SUB 01H ;DECREMENTA CONTADOR
'5034 1802 0057 JR FIM-$
'5036 C601 0058 CRES: ADD A,01H ;INCREMENTA CONTADOR

```

'5038	27	0059	FIM:	DAA		:AJUSTA PARA DECIMAL
'5039	320040	0060		LD	(NUMBER),A	
'503C	D338	0061		OUT	(PI03AD),A	:ATUALIZA DISPLY
'503E	F1	0062		POP	AF	:RESTAURA ACUMULADOR E FLAGS
'503F	FB	0063		EI		:HABILITA INTERUPCAO
'5040	ED4D	0064		RETI		
		0065		END		

Fig. 9.38.A - Programa contador decrescente.



CAPÍTULO 10 - CONCLUSÃO

Com este volume e o primeiro, onde foi desenvolvida a parte de HARDWARE, pode-se acompanhar toda a estrutura dos vários componentes, arquitetura, ciclos de tempo e exemplos de cada circuito para uma determinada área.

O II volume foi praticamente dedicado à programação, nele se desenvolve o estudo das instruções, aplicações e programas.

Com o desenvolvimento da linguagem Assembly, foram elaboradas várias rotinas, que poderão servir de exemplos e de métodos na criação de outros programas.

A parte mais importante da programação é o estudo da utilização e a aplicação de cada instrução, para que se possa realizar um programa ou uma simples rotina, com maior eficiência.

Com a automatização que está sendo desenvolvida nos dias de hoje, o HARDWARE vai-se tornar a etapa elementar de um projeto, sendo o SOFTWARE a mais complexa.

Na maioria dos projetos, o desenvolvimento do SOFTWARE se torna, muitas vezes, mais prolongado que o desenvolvimento do HARDWARE.

Deve-se ter em conta que a parte de programação não é tão maleável como se imagina e, nesta área, existe um limite que é determinado pelo HARDWARE. Um exemplo típico é quando se projeta o HARDWARE e não se prevê uma área de memória suficiente para a execução do SOFTWARE.

Assim sendo, de posse de todas as informações contidas nestes dois volumes, restará ao leitor apenas colocá-las em prática. Esperamos, desta forma, ter contribuído para o alcance deste objetivo.

APÊNDICE A

TABELA DE CONVERSÃO DOS SISTEMAS DE NUMERAÇÃO

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL
000	0000 0000	000	00
001	0000 0001	001	01
002	0000 0010	002	02
003	0000 0011	003	03
004	0000 0100	004	04
005	0000 0101	005	05
006	0000 0110	006	06
007	0000 0111	007	07
008	0000 1000	010	08
009	0000 1001	011	09
010	0000 1010	012	0A
011	0000 1011	013	0B
012	0000 1100	014	0C
013	0000 1101	015	0D
014	0000 1110	016	0E
015	0000 1111	017	0F
016	0001 0000	020	10
017	0001 0001	021	11
018	0001 0010	022	12
019	0001 0011	023	13
020	0001 0100	024	14
021	0001 0101	025	15
022	0001 0110	026	16
023	0001 0111	027	17
024	0001 1000	030	18
025	0001 1001	031	19
026	0001 1010	032	1A
027	0001 1011	033	1B
028	0001 1100	034	1C
029	0001 1101	035	1D
030	0001 1110	036	1E
031	0001 1111	037	1F
032	0010 0000	040	20

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL	APÊNDICE A
033	0010 0001	041	21	
034	0010 0010	042	22	
035	0010 0011	043	23	
036	0010 0100	044	24	
037	0010 0101	045	25	
038	0010 0110	046	26	
039	0010 0111	047	27	
040	0010 1000	050	28	
041	0010 1001	051	29	
042	0010 1010	052	2A	
043	0010 1011	053	2B	
044	0010 1100	054	2C	
045	0010 1101	055	2D	
046	0010 1110	056	2E	
047	0010 1111	057	2F	
048	0011 0000	060	30	
049	0011 0001	061	31	
050	0011 0010	062	32	
051	0011 0011	063	33	
052	0011 0100	064	34	
053	0011 0101	065	35	
054	0011 0110	066	36	
055	0011 0111	067	37	
056	0011 1000	070	38	
057	0011 1001	071	39	
058	0011 1010	072	3A	
059	0011 1011	073	3B	
060	0011 1100	074	3C	
061	0011 1101	075	3D	
062	0011 1110	076	3E	
063	0011 1111	077	3F	
064	0100 0000	100	40	
065	0100 0001	101	41	
066	0100 0010	102	42	
067	0100 0011	103	43	
068	0100 0100	104	44	
069	0100 0101	105	45	

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL	DECIMAL
070	0100 0110	106	46	107
071	0100 0111	107	47	108
072	0100 1000	110	48	109
073	0100 1001	111	49	110
074	0100 1010	112	4A	111
075	0100 1011	113	4B	112
076	0100 1100	114	4C	113
077	0100 1101	115	4D	114
078	0100 1110	116	4E	115
079	0100 1111	117	4F	116
080	0101 0000	120	50	117
081	0101 0001	121	51	118
082	0101 0010	122	52	119
083	0101 0011	123	53	120
084	0101 0100	124	54	121
085	0101 0101	125	55	122
086	0101 0110	126	56	123
087	0101 0111	127	57	124
088	0101 1000	130	58	125
089	0101 1001	131	59	126
090	0101 1010	132	5A	127
091	0101 1011	133	5B	128
092	0101 1100	134	5C	129
093	0101 1101	135	5D	130
094	0101 1110	136	5E	131
095	0101 1111	137	5F	132
096	0110 0000	140	60	133
097	0110 0001	141	61	134
098	0110 0010	142	62	135
099	0110 0011	143	63	136
100	0110 0100	144	64	137
101	0110 0101	145	65	138
102	0110 0110	146	66	139
103	0110 0111	147	67	140
104	0110 1000	150	68	141
105	0110 1001	151	69	142
106	0110 1010	152	6A	143

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL	DECIMAL
107	0110 1011	153	6B	070
108	0110 1100	154	6C	071
109	0110 1101	155	6D	072
110	0110 1110	156	6E	073
111	0110 1111	157	6F	074
112	0111 0000	160	70	075
113	0111 0001	161	71	076
114	0111 0010	162	72	077
115	0111 0011	163	73	078
116	0111 0100	164	74	079
117	0111 0101	165	75	080
118	0111 0110	166	76	081
119	0111 0111	167	77	082
120	0111 1000	170	78	083
121	0111 1001	171	79	084
122	0111 1010	172	7A	085
123	0111 1011	173	7B	086
124	0111 1100	174	7C	087
125	0111 1101	175	7D	088
126	0111 1110	176	7E	089
127	0111 1111	177	7F	090
128	1000 0000	200	80	091
129	1000 0001	201	81	092
130	1000 0010	202	82	093
131	1000 0011	203	83	094
132	1000 0100	204	84	095
133	1000 0101	205	85	096
134	1000 0110	206	86	097
135	1000 0111	207	87	098
136	1000 1000	210	88	099
137	1000 1001	211	89	100
138	1000 1010	212	8A	101
139	1000 1011	213	8B	102
140	1000 1100	214	8C	103
141	1000 1101	215	8D	104
142	1000 1110	216	8E	105
143	1000 1111	217	8F	106

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL
144	1001 0000	220	90
145	1001 0001	221	91
146	1001 0010	222	92
147	1001 0011	223	93
148	1001 0100	224	94
149	1001 0101	225	95
150	1001 0110	226	96
151	1001 0111	227	97
152	1001 1000	230	98
153	1001 1001	231	99
154	1001 1010	232	9A
155	1001 1011	233	9B
156	1001 1100	234	9C
157	1001 1101	235	9D
158	1001 1110	236	9E
159	1001 1111	237	9F
160	1010 0000	240	A0
161	1010 0001	241	A1
162	1010 0010	242	A2
163	1010 0011	243	A3
164	1010 0100	244	A4
165	1010 0101	245	A5
166	1010 0110	246	A6
167	1010 0111	247	A7
168	1010 1000	250	A8
169	1010 1001	251	A9
170	1010 1010	252	AA
171	1010 1011	253	AB
172	1010 1100	254	AC
173	1010 1101	255	AD
174	1010 1110	256	AE
175	1010 1111	257	AF
176	1011 0000	260	B0
177	1011 0001	261	B1
178	1011 0010	262	B2
179	1011 0011	263	B3
180	1011 0100	264	B4

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL	DECIMAL
181	1011 0101	265	B5	181
182	1011 0110	266	B6	182
183	1011 0111	267	B7	183
184	1011 1000	270	B8	184
185	1011 1001	271	B9	185
186	1011 1010	272	BA	186
187	1011 1011	273	BB	187
188	1011 1100	274	BC	188
189	1011 1101	275	BD	189
190	1011 1110	276	BE	190
191	1011 1111	277	BF	191
192	1100 0000	300	C0	192
193	1100 0001	301	C1	193
194	1100 0010	302	C2	194
195	1100 0011	303	C3	195
196	1100 0100	304	C4	196
197	1100 0101	305	C5	197
198	1100 0110	306	C6	198
199	1100 0111	307	C7	199
200	1100 1000	310	C8	200
201	1100 1001	311	C9	201
202	1100 1010	312	CA	202
203	1100 1011	313	CB	203
204	1100 1100	314	CC	204
205	1100 1101	315	CD	205
206	1100 1110	316	CE	206
207	1100 1111	317	CF	207
208	1101 0000	320	D0	208
209	1101 0001	321	D1	209
210	1101 0010	322	D2	210
211	1101 0011	323	D3	211
212	1101 0100	324	D4	212
213	1101 0101	325	D5	213
214	1101 0110	326	D6	214
215	1101 0111	327	D7	215
216	1101 1000	330	D8	216
217	1101 1001	331	D9	217

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL
218	1101 1010	332	DA
219	1101 1011	333	DB
220	1101 1100	334	DC
221	1101 1101	335	DD
222	1101 1110	336	DE
223	1101 1111	337	DF
224	1110 0000	340	E0
225	1110 0001	341	E1
226	1110 0010	342	E2
227	1110 0011	343	E3
228	1110 0100	344	E4
229	1110 0101	345	E5
230	1110 0110	346	E6
231	1110 0111	347	E7
232	1110 1000	350	E8
233	1110 1001	351	E9
234	1110 1010	352	EA
235	1110 1011	353	EB
236	1110 1100	354	EC
237	1110 1101	355	ED
238	1110 1110	356	EE
239	1110 1111	357	EF
240	1111 0000	360	F0
241	1111 0001	361	F1
242	1111 0010	362	F2
243	1111 0011	363	F3
244	1111 0100	364	F4
245	1111 0101	365	F5
246	1111 0110	366	F6
247	1111 0111	367	F7
248	1111 1000	370	F8
249	1111 1001	371	F9
250	1111 1010	372	FA
251	1111 1011	373	FB
252	1111 1100	374	FC
253	1111 1101	375	FD
254	1111 1110	376	FE
255	1111 1111	377	FF

DECIMAL	HEXADECIMAL	DECIMAL	HEXADECIMAL
218	1000 0010	218	1000 0010
219	1000 0011	219	1000 0011
220	1000 0010	220	1000 0010
221	1000 0011	221	1000 0011
222	1000 0010	222	1000 0010
223	1000 0011	223	1000 0011
224	1000 0010	224	1000 0010
225	1000 0011	225	1000 0011
226	1000 0010	226	1000 0010
227	1000 0011	227	1000 0011
228	1000 0010	228	1000 0010
229	1000 0011	229	1000 0011
230	1000 0010	230	1000 0010
231	1000 0011	231	1000 0011
232	1000 0010	232	1000 0010
233	1000 0011	233	1000 0011
234	1000 0010	234	1000 0010
235	1000 0011	235	1000 0011
236	1000 0010	236	1000 0010
237	1000 0011	237	1000 0011
238	1000 0010	238	1000 0010
239	1000 0011	239	1000 0011
240	1000 0010	240	1000 0010
241	1000 0011	241	1000 0011
242	1000 0010	242	1000 0010
243	1000 0011	243	1000 0011
244	1000 0010	244	1000 0010
245	1000 0011	245	1000 0011
246	1000 0010	246	1000 0010
247	1000 0011	247	1000 0011
248	1000 0010	248	1000 0010
249	1000 0011	249	1000 0011
250	1000 0010	250	1000 0010
251	1000 0011	251	1000 0011
252	1000 0010	252	1000 0010
253	1000 0011	253	1000 0011
254	1000 0010	254	1000 0010
255	1000 0011	255	1000 0011

APÊNDICE B

TABELA DE EXPONENCIAIS

2^n	n
1	0
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24
33 554 432	25
67 108 864	26
134 217 728	27
268 435 456	28
536 870 912	29
1 073 741 824	30

	16^n	n
	1	0
	16	1
	256	2
	4 096	3
	65 536	4
	1 048 576	5
	16 777 216	6
	268 435 456	7
	4 294 967 296	8
	68 719 476 736	9
	1 099 511 627 776	10

APÉNDICE B
TABELA DE EXPONENCIAIS

n	
1	1
2	2
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1 024
11	2 048
12	4 096
13	8 192
14	16 384
15	32 768
16	65 536
17	131 072
18	262 144
19	524 288
20	1 048 576
21	2 097 152
22	4 194 304
23	8 388 608
24	16 777 216
25	33 554 432
26	67 108 864
27	134 217 728
28	268 435 456
29	536 870 912
30	1 073 741 824

APÊNDICE C

TABELA DE EQUIVALÊNCIA HEXADECIMAL E ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTER CHANGE)

HEXA	ASCII	DESCRIÇÃO
00	NUL	NULO (Nenhuma Operação)
01	SOH	INÍCIO DO CABEÇÁRIO
02	STX	INÍCIO DO TEXTO
03	ETX	FIM DE TEXTO
04	EOT	FIM DE TRANSMISSÃO
05	ENQ	PERGUNTAR, CONSULTAR
06	ACK	RECONHECIMENTO
07	BEL	CAMPAINHA
08	BS	RETROCESSO
09	HT	TABULAÇÃO HORIZONTAL
0A	LF	AVANÇO DE LINHA
0B	VT	TABULAÇÃO VERTICAL
0C	FF	AVANÇO DE FOLHA
0D	CR	RETORNO DO CARRO
0E	SO	SAÍDA ALTA (Letra Maiúscula)
0F	SI	SAÍDA BAIXA (Letra Minúscula)
10	DLE	LINHA DE SAÍDA DE DADOS
11	DC1	CONTROLE 1
12	DC2	CONTROLE 2
13	DC3	CONTROLE 3
14	DC4	CONTROLE 4
15	NAK	RECONHECIMENTO ERRADO
16	SYN	SINCRONIZAÇÃO VAZIA
17	ETB	FIM DO BLOCO DE TRANSMISSÃO
18	CAN	CANCELAMENTO
19	EM	FIM DO MEIO
1A	SUB	SUBSTITUIÇÃO
1B	ESC	ESCAPE
1C	FS	SEPARAÇÃO DE ARQUIVO
1D	GS	SEPARAÇÃO DE GRUPO
1E	RS	SEPARAÇÃO DE GRAVAÇÃO

1F US SEPARAÇÃO DE UNIDADE
20 SP ESPAÇO

APÊNDICE C

TABELA DE EQUIVALÊNCIA HEXADECIMAL E ASCII (AMERICAN STANDARD)

CODE FOR INFORMATION INTER CHANGE

HEXA	ASCII	DESCRIÇÃO
00	NUL	NULO (termina operação)
01	SOH	INÍCIO DO CABEÇALHO
02	STX	INÍCIO DO TEXTO
03	ETX	FIM DE TEXTO
04	EOT	FIM DE TRANSMISSÃO
05	ENO	PERGUNTA, CONSULTA
06	AF	RECONHECIMENTO
07	BSL	CAMPAINHA
08	BS	RETRASO
09	HT	TABUAÇÃO HORIZONTAL
0A	LF	AVANÇO DE LINHA
0B	VT	TABUAÇÃO VERTICAL
0C	FF	AVANÇO DE FOLHA
0D	CR	RETORNO DO CARRO
0E	EU	SAÍDA ALTA (para maiúsculas)
0F	LU	SAÍDA BAIXA (para minúsculas)
10	DEL	LINHA DE SAÍDA DE DADOS
11	DC1	CONTROLE 1
12	DC2	CONTROLE 2
13	DC3	CONTROLE 3
14	DC4	CONTROLE 4
15	NAK	RECONHECIMENTO ERRADO
16	SYN	SINCRONIZAÇÃO Vazia
17	ETB	FIM DO BLOCO DE TRANSMISSÃO
18	CAN	CANCELAMENTO
19	EM	FIM DO MEIO
1A	SUB	SUBSTITUIÇÃO
1B	ESC	ESCAPE
1C	FS	SEPARAÇÃO DE ARQUIVO
1D	GS	SEPARAÇÃO DE GRUPO
1E	RS	SEPARAÇÃO DE SEÇÃO

HEXA	ASCII	HEXA	ASCII	HEXA	ASCII
21	!	46	F	6B	k
22	"	47	G	6C	l
23	#	48	H	6D	m
24	\$	49	I	6E	n
25	%	4A	J	6F	o
26	&	4B	K	70	p
27	'	4C	L	71	q
28	(4D	M	72	r
29)	4E	N	73	s
2A	*	4F	O	74	t
2B	+	50	P	75	u
2C	,	51	Q	76	v
2D	-	52	R	77	w
2E	.	53	S	78	x
2F	/	54	T	79	y
30	0	55	U	7A	z
31	1	56	V	7B	{
32	2	57	W	7C	
33	3	58	X	7D	}
34	4	59	Y	7E	~
35	5	5A	Z	7F	DEL
36	6	5B	[
37	7	5C	\		
38	8	5D]		
39	9	5E	^		
3A	:	5F	_		
3B	;	60	`		
3C	<	61	a		
3D	=	62	b		
3E	>	63	c		
3F	?	64	d		
40	@	65	e		
41	A	66	f		
42	B	67	g		
43	C	68	h		
44	D	69	i		
45	E	6A	j		

HEXA	DECIT	HEXA	DECIT	HEXA	DECIT
21	1	46	9	46	9
22	"	47	"	47	"
23	8	48	8	48	8
24	2	49	2	49	2
25	5	4A	5	4A	5
26	4	4B	4	4B	4
27	"	4C	"	4C	"
28	1	4D	1	4D	1
29	"	4E	"	4E	"
2A	"	4F	"	4F	"
2B	"	50	"	50	"
2C	7	51	7	51	7
2D	-	52	-	52	-
2E	"	53	"	53	"
2F	4	54	4	54	4
30	0	55	0	55	0
31	1	56	1	56	1
32	2	57	2	57	2
33	3	58	3	58	3
34	4	59	4	59	4
35	5	5A	5	5A	5
36	6	5B	6	5B	6
37	7	5C	7	5C	7
38	8	5D	8	5D	8
39	0	5E	0	5E	0
3A	:	5F	:	5F	:
3B	:	60	:	60	:
3C	<	61	<	61	<
3D	=	62	=	62	=
3E	>	63	>	63	>
3F	3	64	3	64	3
40	0	65	0	65	0
41	2	66	2	66	2
42	8	67	8	67	8
43	C	68	C	68	C
44	D	69	D	69	D
45	B	6A	B	6A	B

APÊNDICE D

ESTRUTURA DOS REGISTRADORES DO Z-80

D.1) Organização Interna dos Registradores do Z-80

ACUMULADOR "A"	B	D	H
FLAGS "F"	C	E	L

ACUMULADOR "A' "	B'	D'	H'
FLAGS "F' "	C'	E'	L'

REGISTRADOR "I"
REGISTRADOR DE INDEX "IX"
STACK POINTER "SP"

REGISTRADOR "R"
REGISTRADOR DE INDEX "IY"
PROGRAMA COUNTER "PC"

D.2) Organização dos FLAGS

S	Z	*	H	*	P/V	N	C
---	---	---	---	---	-----	---	---

onde

S FLAG de Sinal

Z FLAG de zero

H FLAG de HALF CARRY

P/V FLAG de paridade ou OVERFLOW

N FLAG de adição/subtração

C FLAG de CARRY

* Não usado

ESTRUTURA DOS REGISTRADORES DO S-80

D.1) Organização interna dos Registradores do S-80

ACUMULADOR "A"	E	D	C	B	A
FLAG "F"	E	D	C	B	A

REGISTRADOR "R"	REGISTRADOR DE INDEX "IX"	STACK POINTER "SP"
REGISTRADOR "R"	REGISTRADOR DE INDEX "IX"	PROGRAMA COUNTER "PC"

D.2) Organização dos FLAG

0	1	2	3	4	5	6	7	8	9	A	B	C
---	---	---	---	---	---	---	---	---	---	---	---	---

onde	
8	FLAG de SINAL
7	FLAG de ZERO
6	FLAG de HALF CARRY
5,4	FLAG de paridade ou OVERFLOW
3	FLAG de adição/subtração
2	FLAG de CARRY
1	não usado

APÊNDICE E

INTERRUPÇÕES DO Z-80

E.1) NÃO MASCARADA ($\overline{\text{NMI}}$)

ENDEREÇO (0066)_H

E.2) MASCARADA (INT)

MODO 0

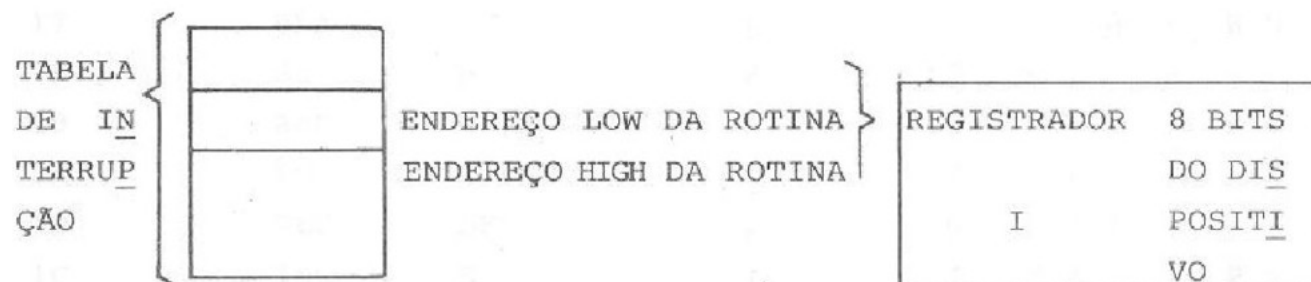
Código no DATA BUS, podendo até endereçar oito posições de memória.

(0000)_H, (0008)_H, (0010)_H, (0018)_H, (0020)_H, (0028)_H, (0030)_H, (0038)_H.

E.3) MODO 1

ENDEREÇO (0038)_H

E.4) MODO 2



E.5) HABILITAÇÃO E DESABILITAÇÃO DO FLIP-FLOP "IFF1" e "IFF2"

CONDIÇÃO	IFF1	IFF2	COMENTÁRIO
CPU RESET	Ø	Ø	
DI	Ø	Ø	
EI	1	1	
LD A,I	-	-	IFF2 → FLAG P/V
LD A,R	-	-	IFF2 → FLAG P/V
NMI	Ø	-	
RETN	IFF2	-	IFF1 ← IFF2
INT	Ø	Ø	
RETI	-	-	

APÊNDICE F

CÓDIGO DE OPERAÇÃO POR ORDEM NUMÉRICA

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS
					S, Z, H, P/V, N, C
00	NOP		1	4	- - - - -
01 YYXX	LD	BC, XXYY	3	10	- - - - -
02	LD	(BC), A	1	7	- - - - -
03	INC	BC	1	6	- - - - -
04	INC	B	1	4	S Z H V 0 -
05	DEC	B	1	4	S Z H V 1 -
06 XX	LD	B, XX	2	7	- - - - -
07	RLCA		1	4	- - 0 - 0 C
08	EX	AF, AF'	1	4	S Z H P/V N C
09	ADD	HL, BC	1	11	- - X - 0 C
0A	LD	A, (BC)	1	7	- - - - -
0B	DEC	BC	1	6	- - - - -
0C	INC	C	1	4	S Z H V 0 -
0D	DEC	C	1	4	S Z H V 1 -
0E XX	LD	C, XX	2	7	- - - - -
0F	RRCA		1	4	- - 0 - 0 C
10 e-2	DJNZ	e	2	8/13	- - - - -
11 YYXX	LD	DE, XXYY	3	10	- - - - -
12	LD	(DE), A	1	7	- - - - -
13	INC	DE	1	6	- - - - -
14	INC	D	1	4	S Z H V 0 -
15	DEC	D	1	4	S Z H V 1 -
16 XX	LD	D, XX	1	4	- - - - -
17	RLA		1	4	- - 0 - 0 C
18 e-2	JR	e	2	12	- - - - -
19	ADD	HL, DE	1	11	- - X - 0 C
1A	LD	A, (DE)	1	7	- - - - -
1B	DEC	DE	1	6	- - - - -
1C	INC	E	1	4	S Z H V 0 -
1D	DEC	E	1	4	S Z H V 1 -
1E XX	LD	E, XX	2	4	- - - - -
1F	RRA		1	4	- - 0 - 0 C

HEXA	COP	OPER.	Nº BYTES	Nº EST.	S, Z, H, P/V, N, C					
20 e-2	JR	NZ,e	2	7/12	-	-	-	-	-	-
21 YYXX	LD	HL,XXYY	3	10	-	-	-	-	-	-
22 ppqq	LD	(qqpp),HL	3	16	-	-	-	-	-	-
23	INC	HL	1	6	-	-	-	-	-	-
24	INC	H	1	4	S	Z	H	V	0	-
25	DEC	H	1	4	S	Z	H	V	1	-
26 XX	LD	H,XX	2	4	-	-	-	-	-	-
27	DAA		1	4	S	Z	H	P	-	C
28 e-2	JR	Z,e	2	7/12	-	-	-	-	-	-
29	ADD	HL,HL	1	11	-	-	X	-	0	C
2A ppqq	LD	HL,(qqpp)	3	16	-	-	-	-	-	-
2B	DEC	HL	1	6	-	-	-	-	-	-
2C	INC	L	1	4	S	Z	H	V	0	-
2D	DEC	L	1	4	S	Z	H	V	1	-
2E XX	LD	L,XX	2	4	-	-	-	-	-	-
2F	CPL		1	4	-	-	1	-	1	-
30 e-2	JR	NC,e	2	7/12	-	-	-	-	-	-
31 YYXX	LD	SP,XXYY	3	10	-	-	-	-	-	-
32 ppqq	LD	(qqpp),A	3	13	-	-	-	-	-	-
33	INC	SP	1	6	-	-	-	-	-	-
34	INC	(HL)	1	11	S	Z	H	V	0	-
35	DEC	(HL)	1	11	S	Z	H	V	1	-
36 XX	LD	(HL),XX	2	10	-	-	-	-	-	-
37	SCF		1	4	-	-	0	-	0	1
38 e-2	JR	C,e	2	7/12	-	-	-	-	-	-
39	ADD	HL,SP	1	11	-	-	X	-	0	C
3A ppqq	LD	A,(qqpp)	3	13	-	-	-	-	-	-
3B	DEC	SP	1	6	-	-	-	-	-	-
3C	INC	A	1	4	S	Z	H	V	0	-
3D	DEC	A	1	4	S	Z	H	V	1	-
3E XX	LD	A,XX	2	7	-	-	-	-	-	-
3F	CCF		1	4	-	-	H	-	0	C
40	LD	B,B	1	4	-	-	-	-	-	-
41	LD	B,C	1	4	-	-	-	-	-	-
42	LD	B,D	1	4	-	-	-	-	-	-
43	LD	B,E	1	4	-	-	-	-	-	-

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS
					S Z H P/V N C
44	LD	B,H	1	4	- - - - -
45	LD	B,L	1	4	- - - - -
46	LD	B,(HL)	1	7	- - - - -
47	LD	B,A	1	4	- - - - -
48	LD	C,B	1	4	- - - - -
49	LD	C,C	1	4	- - - - -
4A	LD	C,D	1	4	- - - - -
4B	LD	C,E	1	4	- - - - -
4C	LD	C,H	1	4	- - - - -
4D	LD	C,L	1	4	- - - - -
4E	LD	C,(HL)	1	7	- - - - -
4F	LD	C,A	1	4	- - - - -
50	LD	D,B	1	4	- - - - -
51	LD	D,C	1	4	- - - - -
52	LD	D,D	1	4	- - - - -
53	LD	D,E	1	4	- - - - -
54	LD	D,H	1	4	- - - - -
55	LD	D,L	1	4	- - - - -
56	LD	D,(HL)	1	7	- - - - -
57	LD	D,A	1	4	- - - - -
58	LD	E,B	1	4	- - - - -
59	LD	E,C	1	4	- - - - -
5A	LD	E,D	1	4	- - - - -
5B	LD	E,E	1	4	- - - - -
5C	LD	E,H	1	4	- - - - -
5D	LD	E,L	1	4	- - - - -
5E	LD	E,(HL)	1	7	- - - - -
5F	LD	E,A	1	4	- - - - -
60	LD	H,B	1	4	- - - - -
61	LD	H,C	1	4	- - - - -
62	LD	H,D	1	4	- - - - -
63	LD	H,E	1	4	- - - - -
64	LD	H,H	1	4	- - - - -
65	LD	H,L	1	4	- - - - -
66	LD	H,(HL)	1	7	- - - - -
67	LD	H,A	1	4	- - - - -

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS					
					S	Z	H	P/V	N	C
68		LD	L,B	1	4	-	-	-	-	-
69		LD	L,C	1	4	-	-	-	-	-
6A		LD	L,D	1	4	-	-	-	-	-
6B		LD	L,E	1	4	-	-	-	-	-
6C		LD	L,H	1	4	-	-	-	-	-
6D		LD	L,L	1	4	-	-	-	-	-
6E		LD	L,(HL)	1	7	-	-	-	-	-
6F		LD	L,A	1	4	-	-	-	-	-
70		LD	(HL),B	1	7	-	-	-	-	-
71		LD	(HL),C	1	7	-	-	-	-	-
72		LD	(HL),D	1	7	-	-	-	-	-
73		LD	(HL),E	1	7	-	-	-	-	-
74		LD	(HL),H	1	7	-	-	-	-	-
75		LD	(HL),L	1	7	-	-	-	-	-
76		HALT		1	4	-	-	-	-	-
77		LD	(HL),A	1	7	-	-	-	-	-
78		LD	A,B	1	4	-	-	-	-	-
79		LD	A,C	1	4	-	-	-	-	-
7A		LD	A,D	1	4	-	-	-	-	-
7B		LD	A,E	1	4	-	-	-	-	-
7C		LD	A,H	1	4	-	-	-	-	-
7D		LD	A,L	1	4	-	-	-	-	-
7E		LD	A,(HL)	1	7	-	-	-	-	-
7F		LD	A,A	1	4	-	-	-	-	-
80		ADD	A,B	1	4	S	Z	H	V	Ø C
81		ADD	A,C	1	4	S	Z	H	V	Ø C
82		ADD	A,D	1	4	S	Z	H	V	Ø C
83		ADD	A,E	1	4	S	Z	H	V	Ø C
84		ADD	A,H	1	4	S	Z	H	V	Ø C
85		ADD	A,L	1	4	S	Z	H	V	Ø C
86		ADD	A,(HL)	1	7	S	Z	H	V	Ø C
87		ADD	A,A	1	4	S	Z	H	V	Ø C
88		ADC	A,B	1	4	S	Z	H	V	Ø C
89		ADC	A,C	1	4	S	Z	H	V	Ø C
8A		ADC	A,D	1	4	S	Z	H	V	Ø C
8B		ADC	A,E	1	4	S	Z	H	V	Ø C

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS
					S Z H P/V N C
8C	ADC	A,H	1	4	S Z H V Ø C
8D	ADC	A,L	1	4	S Z H V Ø C
8E	ADC	A,(HL)	1	7	S Z H V Ø C
8F	ADC	A,A	1	4	S Z H V 1 C
90	SUB	B	1	4	S Z H V 1 C
91	SUB	C	1	4	S Z H V 1 C
92	SUB	D	1	4	S Z H V 1 C
93	SUB	E	1	4	S Z H V 1 C
94	SUB	H	1	4	S Z H V 1 C
95	SUB	L	1	4	S Z H V 1 C
96	SUB	(HL)	1	7	S Z H V 1 C
97	SUB	A	1	4	S Z H V 1 C
98	SBC	B	1	4	S Z H V 1 C
99	SBC	C	1	4	S Z H V 1 C
9A	SBC	D	1	4	S Z H V 1 C
9B	SBC	E	1	4	S Z H V 1 C
9C	SBC	H	1	4	S Z H V 1 C
9D	SBC	L	1	4	S Z H V 1 C
9E	SBC	(HL)	1	7	S Z H V 1 C
9F	SBC	A	1	4	S Z H V 1 C
A0	AND	B	1	4	S Z 1 P Ø Ø
A1	AND	C	1	4	S Z 1 P Ø Ø
A2	AND	D	1	4	S Z 1 P Ø Ø
A3	AND	E	1	4	S Z 1 P Ø Ø
A4	AND	H	1	4	S Z 1 P Ø Ø
A5	AND	L	1	4	S Z 1 P Ø Ø
A6	AND	(HL)	1	7	S Z 1 P Ø Ø
A7	AND	A	1	4	S Z 1 P Ø Ø
A8	XOR	B	1	4	S Z 1 P Ø Ø
A9	XOR	C	1	4	S Z 1 P Ø Ø
AA	XOR	D	1	4	S Z 1 P Ø Ø
AB	XOR	E	1	4	S Z 1 P Ø Ø
AC	XOR	H	1	4	S Z 1 P Ø Ø
AD	XOR	L	1	4	S Z 1 P Ø Ø
AE	XOR	(HL)	1	7	S Z 1 P Ø Ø
AF	XOR	A	1	4	S Z 1 P Ø Ø

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS
					S Z H P/V N C	
B0	OR	B	1	4	S Z 1	P Ø Ø
B1	OR	C	1	4	S Z 1	P Ø Ø
B2	OR	D	1	4	S Z 1	P Ø Ø
B3	OR	E	1	4	S Z 1	P Ø Ø
B4	OR	H	1	4	S Z 1	P Ø Ø
B5	OR	L	1	4	S Z 1	P Ø Ø
B6	OR	(HL)	1	7	S Z 1	P Ø Ø
B7	OR	A	1	4	S Z 1	P Ø Ø
B8	CP	B	1	4	S Z H	V 1 C
B9	CP	C	1	4	S Z H	V 1 C
BA	CP	D	1	4	S Z H	V 1 C
BB	CP	E	1	4	S Z H	V 1 C
BC	CP	H	1	4	S Z H	V 1 C
BD	CP	L	1	4	S Z H	V 1 C
BE	CP	(HL)	1	7	S Z H	V 1 C
BF	CP	A	1	4	S Z H	V 1 C
C0	RET	NZ	1	5/11	- - -	- - -
C1	POP	BC	1	10	- - -	- - -
C2 ppqq	JP	NZ,qqpp	3	10	- - -	- - -
C3 ppqq	JP	qqpp	3	10	- - -	- - -
C4 ppqq	CALL	NZ,qqpp	3	10/17	- - -	- - -
C5	PUSH	BC	1	11	- - -	- - -
C6 XX	ADD	A,XX	2	7	S Z H	V Ø C
C7	RST	00H	1	11	- - -	- - -
C8	RET	Z	1	5/11	- - -	- - -
C9	RET		1	10	- - -	- - -
CA ppqq	JP	Z,ppqq	3	10	- - -	- - -
CB00	RLC	B	2	8	S Z Ø	P Ø C
CB01	RLC	C	2	8	S Z Ø	P Ø C
CB02	RLC	D	2	8	S Z Ø	P Ø C
CB03	RLC	E	2	8	S Z Ø	P Ø C
CB04	RLC	H	2	8	S Z Ø	P Ø C
CB05	RLC	L	2	8	S Z Ø	P Ø C
CB06	RLC	(HL)	2	15	S Z Ø	P Ø C
CB07	RLC	A	2	8	S Z Ø	P Ø C
CB08	RRC	B	2	8	S Z Ø	P Ø C

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS
					S Z H P/V N C	
CB09	RRC	C	2	8	S Z Ø P Ø C	
CB0A	RRC	D	2	8	S Z Ø P Ø C	
CB0B	RRC	E	2	8	S Z Ø P Ø C	
CB0C	RRC	H	2	8	S Z Ø P Ø C	
CB0D	RRC	L	2	8	S Z Ø P Ø C	
CB0E	RRC	(HL)	2	15	S Z Ø P Ø C	
CB0F	RRC	A	2	8	S Z Ø P Ø C	
CB10	RL	B	2	8	S Z Ø P Ø C	
CB11	RL	C	2	8	S Z Ø P Ø C	
CB12	RL	D	2	8	S Z Ø P Ø C	
CB13	RL	E	2	8	S Z Ø P Ø C	
CB14	RL	H	2	8	S Z Ø P Ø C	
CB15	RL	L	2	8	S Z Ø P Ø C	
CB16	RL	(HL)	2	15	S Z Ø P Ø C	
CB17	RL	A	2	8	S Z Ø P Ø C	
CB18	RR	B	2	8	S Z Ø P Ø C	
CB19	RR	C	2	8	S Z Ø P Ø C	
CB1A	RR	D	2	8	S Z Ø P Ø C	
CB1B	RR	E	2	8	S Z Ø P Ø C	
CB1C	RR	H	2	8	S Z Ø P Ø C	
CB1D	RR	L	2	8	S Z Ø P Ø C	
CB1E	RR	(HL)	2	15	S Z Ø P Ø C	
CB1F	RR	A	2	8	S Z Ø P Ø C	
CB20	SLA	B	2	8	S Z Ø P Ø C	
CB21	SLA	C	2	8	S Z Ø P Ø C	
CB22	SLA	D	2	8	S Z Ø P Ø C	
CB23	SLA	E	2	8	S Z Ø P Ø C	
CB24	SLA	H	2	8	S Z Ø P Ø C	
CB25	SLA	L	2	8	S Z Ø P Ø C	
CB26	SLA	(HL)	2	15	S Z Ø P Ø C	
CB27	SLA	A	2	8	S Z Ø P Ø C	
CB28	SRA	B	2	8	S Z Ø P Ø C	
CB29	SRA	C	2	8	S Z Ø P Ø C	
CB2A	SRA	D	2	8	S Z Ø P Ø C	
CB2B	SRA	E	2	8	S Z Ø P Ø C	
CB2C	SRA	H	2	8	S Z Ø P Ø C	

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS S Z H	AFETADOS P/V N C
CB2D	SRA	L	2	8	S Z Ø	P Ø C
CB2E	SRA	(HL)	2	8	S Z Ø	P Ø C
CB2F	SRA	A	2	8	S Z Ø	P Ø C
CB38	SRL	B	2	8	S Z Ø	P Ø C
CB39	SRL	C	2	8	S Z Ø	P Ø C
CB3A	SRL	D	2	8	S Z Ø	P Ø C
CB3B	SRL	E	2	8	S Z Ø	P Ø C
CB3C	SRL	H	2	8	S Z Ø	P Ø C
CB3D	SRL	L	2	8	S Z Ø	P Ø C
CB3E	SRL	(HL)	2	15	S Z Ø	P Ø C
CB3F	SRL	A	2	8	S Z Ø	P Ø C
CB4Ø	BIT	0,B	2	8	X Z 1	X Ø -
CB41	BIT	0,C	2	8	X Z 1	X Ø -
CB42	BIT	0,D	2	8	X Z 1	X Ø -
CB43	BIT	0,E	2	8	X Z 1	X Ø -
CB44	BIT	0,H	2	8	X Z 1	X Ø -
CB45	BIT	0,L	2	8	X Z 1	X Ø -
CB46	BIT	0,(HL)	2	12	X Z 1	X Ø -
CB47	BIT	0,A	2	8	X Z 1	X Ø -
CB48	BIT	1,B	2	8	X Z 1	X Ø -
CB49	BIT	1,C	2	8	X Z 1	X Ø -
CB4A	BIT	1,D	2	8	X Z 1	X Ø -
CB4B	BIT	1,E	2	8	X Z 1	X Ø -
CB4C	BIT	1,H	2	8	X Z 1	X Ø -
CB4D	BIT	1,L	2	8	X Z 1	X Ø -
CB4E	BIT	1,(HL)	2	12	X Z 1	X Ø -
CB4F	BIT	1,A	2	8	X Z 1	X Ø -
CB5Ø	BIT	2,B	2	8	X Z 1	X Ø -
CB51	BIT	2,C	2	8	X Z 1	X Ø -
CB52	BIT	2,D	2	8	X Z 1	X Ø -
CB53	BIT	2,E	2	8	X Z 1	X Ø -
CB54	BIT	2,H	2	8	X Z 1	X Ø -
CB55	BIT	2,L	2	8	X Z 1	X Ø -
CB56	BIT	2,(HL)	2	12	X Z 1	X Ø -
CB57	BIT	2,A	2	8	X Z 1	X Ø -
CB58	BIT	3,B	2	8	X Z 1	X Ø -
CB59	BIT	3,C	2	8	X Z 1	X Ø -

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS					
					S	Z	H	P/V	N	C
CB5A	BIT	3,D	2	8	X	Z	1	X	Ø	-
CB5B	BIT	3,E	2	8	X	Z	1	X	Ø	-
CB5C	BIT	3,H	2	8	X	Z	1	X	Ø	-
CB5D	BIT	3,L	2	8	X	Z	1	X	Ø	-
CB5E	BIT	3,(HL)	2	12	X	Z	1	X	Ø	-
CB5F	BIT	3,A	2	8	X	Z	1	X	Ø	-
CB6Ø	BIT	4,B	2	8	X	Z	1	X	Ø	-
CB61	BIT	4,C	2	8	X	Z	1	X	Ø	-
CB62	BIT	4,D	2	8	X	Z	1	X	Ø	-
CB63	BIT	4,E	2	8	X	Z	1	X	Ø	-
CB64	BIT	4,H	2	8	X	Z	1	X	Ø	-
CB65	BIT	4,L	2	8	X	Z	1	X	Ø	-
CB66	BIT	4,(HL)	2	12	X	Z	1	X	Ø	-
CB67	BIT	4,A	2	8	X	Z	1	X	Ø	-
CB68	BIT	5,B	2	8	X	Z	1	X	Ø	-
CB69	BIT	5,C	2	8	X	Z	1	X	Ø	-
CB6A	BIT	5,D	2	8	X	Z	1	X	Ø	-
CB6B	BIT	5,E	2	8	X	Z	1	X	Ø	-
CB6C	BIT	5,H	2	8	X	Z	1	X	Ø	-
CB6D	BIT	5,L	2	8	X	Z	1	X	Ø	-
CB6E	BIT	5,(HL)	2	12	X	Z	1	X	Ø	-
CB6F	BIT	5,A	2	8	X	Z	1	X	Ø	-
CB7Ø	BIT	6,B	2	8	X	Z	1	X	Ø	-
CB71	BIT	6,C	2	8	X	Z	1	X	Ø	-
CB72	BIT	6,D	2	8	X	Z	1	X	Ø	-
CB73	BIT	6,E	2	8	X	Z	1	X	Ø	-
CB74	BIT	6,H	2	8	X	Z	1	X	Ø	-
CB75	BIT	6,L	2	8	X	Z	1	X	Ø	-
CB76	BIT	6,(HL)	2	12	X	Z	1	X	Ø	-
CB77	BIT	6,A	2	8	X	Z	1	X	Ø	-
CB78	BIT	7,B	2	8	X	Z	1	X	Ø	-
CB79	BIT	7,C	2	8	X	Z	1	X	Ø	-
CB7A	BIT	7,D	2	8	X	Z	1	X	Ø	-
CB7B	BIT	7,E	2	8	X	Z	1	X	Ø	-
CB7C	BIT	7,H	2	8	X	Z	1	X	Ø	-
CB7D	BIT	7,L	2	8	X	Z	1	X	Ø	-

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS					
					S	Z	H	P/V	N	C
CB7E	BIT	7, (HL)	2	12	X	Z	1	X	Ø	-
CB7F	BIT	7, A	2	8	X	Z	1	X	Ø	-
CB80	RES	Ø, B	2	8	-	-	-	-	-	-
CB81	RES	Ø, C	2	8	-	-	-	-	-	-
CB82	RES	Ø, D	2	8	-	-	-	-	-	-
CB83	RES	Ø, E	2	8	-	-	-	-	-	-
CB84	RES	Ø, H	2	8	-	-	-	-	-	-
CB85	RES	Ø, L	2	8	-	-	-	-	-	-
CB86	RES	Ø, (HL)	2	15	-	-	-	-	-	-
CB87	RES	Ø, A	2	8	-	-	-	-	-	-
CB88	RES	1, B	2	8	-	-	-	-	-	-
CB89	RES	1, C	2	8	-	-	-	-	-	-
CB8A	RES	1, D	2	8	-	-	-	-	-	-
CB8B	RES	1, E	2	8	-	-	-	-	-	-
CB8C	RES	1, H	2	8	-	-	-	-	-	-
CB8D	RES	1, L	2	8	-	-	-	-	-	-
CB8E	RES	1, (HL)	2	15	-	-	-	-	-	-
CB8F	RES	1, A	2	8	-	-	-	-	-	-
CB90	RES	2, B	2	8	-	-	-	-	-	-
CB91	RES	2, C	2	8	-	-	-	-	-	-
CB92	RES	2, D	2	8	-	-	-	-	-	-
CB93	RES	2, E	2	8	-	-	-	-	-	-
CB94	RES	2, H	2	8	-	-	-	-	-	-
CB95	RES	2, L	2	8	-	-	-	-	-	-
CB96	RES	2, (HL)	2	15	-	-	-	-	-	-
CB97	RES	2, A	2	8	-	-	-	-	-	-
CB98	RES	3, B	2	8	-	-	-	-	-	-
CB99	RES	3, L	2	8	-	-	-	-	-	-
CB9A	RES	3, D	2	8	-	-	-	-	-	-
CB9B	RES	3, E	2	8	-	-	-	-	-	-
CB9C	RES	3, H	2	8	-	-	-	-	-	-
CB9D	RES	3, L	2	8	-	-	-	-	-	-
CB9E	RES	3, (HL)	2	15	-	-	-	-	-	-
CB9F	RES	3, A	2	8	-	-	-	-	-	-
CBA0	RES	4, B	2	8	-	-	-	-	-	-
CBA1	RES	4, C	2	8	-	-	-	-	-	-

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS					
					S	Z	H	P/V	N	C
CBA2	RES	4,D	2	8	-	-	-	-	-	-
CBA3	RES	4,E	2	8	-	-	-	-	-	-
CBA4	RES	4,H	2	8	-	-	-	-	-	-
CBA5	RES	4,L	2	8	-	-	-	-	-	-
CBA6	RES	4,(HL)	2	15	-	-	-	-	-	-
CBA7	RES	4,A	2	8	-	-	-	-	-	-
CBA8	RES	5,B	2	8	-	-	-	-	-	-
CBA9	RES	5,C	2	8	-	-	-	-	-	-
CBA A	RES	5,D	2	8	-	-	-	-	-	-
CBAB	RES	5,E	2	8	-	-	-	-	-	-
CBAC	RES	5,H	2	8	-	-	-	-	-	-
CBAD	RES	5,L	2	8	-	-	-	-	-	-
CBAE	RES	5,(HL)	2	15	-	-	-	-	-	-
CBAF	RES	5,A	2	8	-	-	-	-	-	-
CBB0	RES	6,B	2	8	-	-	-	-	-	-
CBB1	RES	6,C	2	8	-	-	-	-	-	-
CBB2	RES	6,D	2	8	-	-	-	-	-	-
CBB3	RES	6,E	2	8	-	-	-	-	-	-
CBB4	RES	6,H	2	8	-	-	-	-	-	-
CBB5	RES	6,L	2	8	-	-	-	-	-	-
CBB6	RES	6,(HL)	2	15	-	-	-	-	-	-
CBB7	RES	6,A	2	8	-	-	-	-	-	-
CBB8	RES	7,B	2	8	-	-	-	-	-	-
CBB9	RES	7,C	2	8	-	-	-	-	-	-
CBBA	RES	7,D	2	8	-	-	-	-	-	-
BBBB	RES	7,E	2	8	-	-	-	-	-	-
CBBC	RES	7,H	2	8	-	-	-	-	-	-
CBBD	RES	7,L	2	8	-	-	-	-	-	-
CBBE	RES	7,(HL)	2	15	-	-	-	-	-	-
CBBF	RES	7,A	2	8	-	-	-	-	-	-
CBC0	SET	Ø,B	2	8	-	-	-	-	-	-
CBC1	SET	Ø,C	2	8	-	-	-	-	-	-
CBC2	SET	Ø,D	2	8	-	-	-	-	-	-
CBC3	SET	Ø,E	2	8	-	-	-	-	-	-
CBC4	SET	Ø,H	2	8	-	-	-	-	-	-
CBC5	SET	Ø,L	2	8	-	-	-	-	-	-

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS				
					S	Z	H	P/V	N	C
CBC6	SET	Ø, (HL)	2	15	-	-	-	-	-	-
CBC7	SET	Ø, A	2	8	-	-	-	-	-	-
CBC8	SET	1, B	2	8	-	-	-	-	-	-
CBC9	SET	1, C	2	8	-	-	-	-	-	-
CBCA	SET	1, D	2	8	-	-	-	-	-	-
CBCB	SET	1, E	2	8	-	-	-	-	-	-
CBCC	SET	1, H	2	8	-	-	-	-	-	-
CBCD	SET	1, L	2	8	-	-	-	-	-	-
CBCE	SET	1, (HL)	2	15	-	-	-	-	-	-
CBCF	SET	1, A	2	8	-	-	-	-	-	-
CBDØ	SET	2, B	2	8	-	-	-	-	-	-
CBD1	SET	2, C	2	8	-	-	-	-	-	-
CBD2	SET	2, D	2	8	-	-	-	-	-	-
CBD3	SET	2, E	2	8	-	-	-	-	-	-
CBD4	SET	2, H	2	8	-	-	-	-	-	-
CBD5	SET	2, L	2	8	-	-	-	-	-	-
CBD6	SET	2, (HL)	2	15	-	-	-	-	-	-
CBD7	SET	2, A	2	8	-	-	-	-	-	-
CBD8	SET	3, B	2	8	-	-	-	-	-	-
CBD9	SET	3, C	2	8	-	-	-	-	-	-
CBDA	SET	3, D	2	8	-	-	-	-	-	-
CBDB	SET	3, E	2	8	-	-	-	-	-	-
CBDC	SET	3, H	2	8	-	-	-	-	-	-
CBDD	SET	3, L	2	8	-	-	-	-	-	-
CBDE	SET	3, (HL)	2	15	-	-	-	-	-	-
CBDF	SET	3, A	2	8	-	-	-	-	-	-
CBEØ	SET	4, B	2	8	-	-	-	-	-	-
CBE1	SET	4, C	2	8	-	-	-	-	-	-
CBE2	SET	4, D	2	8	-	-	-	-	-	-
CBE3	SET	4, E	2	8	-	-	-	-	-	-
CBE4	SET	4, H	2	8	-	-	-	-	-	-
CBE5	SET	4, L	2	8	-	-	-	-	-	-
CBE6	SET	4, (HL)	2	15	-	-	-	-	-	-
CBE7	SET	4, A	2	8	-	-	-	-	-	-
CBE8	SET	5, B	2	8	-	-	-	-	-	-
CBE9	SET	5, C	2	8	-	-	-	-	-	-

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS
						S Z H P/V N C
CBEA	SET	5,D	2	8	- - - -	- - - -
CBEB	SET	5,E	2	8	- - - -	- - - -
CBEC	SET	5,H	2	8	- - - -	- - - -
CBED	SET	5,L	2	8	- - - -	- - - -
CBEE	SET	5,(HL)	2	15	- - - -	- - - -
CBEF	SET	5,A	2	8	- - - -	- - - -
CBF0	SET	6,B	2	8	- - - -	- - - -
CBF1	SET	6,C	2	8	- - - -	- - - -
CBF2	SET	6,D	2	8	- - - -	- - - -
CBF3	SET	6,E	2	8	- - - -	- - - -
CBF4	SET	6,H	2	8	- - - -	- - - -
CBF5	SET	6,L	2	8	- - - -	- - - -
CBF6	SET	6,(HL)	2	15	- - - -	- - - -
CBF7	SET	6,A	2	8	- - - -	- - - -
CBF8	SET	7,B	2	8	- - - -	- - - -
CBF9	SET	7,C	2	8	- - - -	- - - -
CBFA	SET	7,D	2	8	- - - -	- - - -
CBFB	SET	7,E	2	8	- - - -	- - - -
CBFC	SET	7,H	2	8	- - - -	- - - -
CBFD	SET	7,L	2	8	- - - -	- - - -
CBFE	SET	7,(HL)	2	15	- - - -	- - - -
CBFF	SET	7,A	2	8	- - - -	- - - -
CC ppqq	CALL	2,qqpp	3	10/17	- - - -	- - - -
CD ppqq	CALL	qqpp	3	17	- - - -	- - - -
CE XX	ADC	A,XX	2	7	S Z H V	0 C
CF	RST	08H	1	11	- - - -	- - - -
D0	RET	NC	1	5/11	- - - -	- - - -
D1	POP	DE	1	10	- - - -	- - - -
D2 ppqq	JP	NC,qqpp	3	10	- - - -	- - - -
D3 pp	OUT	(pp),A	2	11	- - - -	- - - -
D4 ppqq	CALL	NC,ppqq	3	10/17	- - - -	- - - -
D5	PUSH	DE	1	11	- - - -	- - - -
D6,XX	SUB	XX	2	7	S Z H V	1 C
D7	RST	10H	1	11	- - - -	- - - -
D8	RET	C	1	5/11	- - - -	- - - -
D9	EXX		1	4	- - - -	- - - -

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS
					S Z H P/V N C	
DA ppqq	JP	C,qqpp	3	10	- - - -	- - -
DB pp	IN	A,(pp)	2	11	- - - -	- - -
DC ppqq	CALL	C,qqpp	3	10/17	- - - -	- - -
DD09	ADD	IX,BC	2	15	- - X -	- Ø C
DD19	ADD	IX,DE	2	15	- - X -	- Ø C
DD29	ADD	IX,IX	2	15	- - X -	- Ø C
DD39	ADD	IX,SP	2	15	- - X -	- Ø C
DD21 YYXX	LD	IX,XXYY	4	14	- - - -	- - -
DD22 ppqq	LD	(qqpp),IX	4	20	- - - -	- - -
DD23	INC	IX	2	10	- - - -	- - -
DD2A ppqq	LD	IX,(qqpp)	4	20	- - - -	- - -
DD2B	DEC	IX	2	10	- - - -	- - -
DD34 d	INC	(IX+d)	3	23	S Z H V	Ø -
DD36 d XX	LD	(IX+d),XX	4	19	- - - -	- - -
DD46 d	LD	B,(IX+d)	3	19	- - - -	- - -
DD4E d	LD	C,(IX+d)	3	19	- - - -	- - -
DD56 d	LD	D,(IX+d)	3	19	- - - -	- - -
DD5E d	LD	E,(IX+d)	3	19	- - - -	- - -
DD66 d	LD	H,(IX+d)	3	19	- - - -	- - -
DD6E d	LD	L,(IX+d)	3	19	- - - -	- - -
DD7E d	LD	A,(IX+d)	3	19	- - - -	- - -
DD7Ø d	LD	(IX+d),B	3	19	- - - -	- - -
DD71 d	LD	(IX+d),C	3	19	- - - -	- - -
DD72 d	LD	(IX+d),D	3	19	- - - -	- - -
DD73 d	LD	(IX+d),E	3	19	- - - -	- - -
DD74 d	LD	(IX+d),H	3	19	- - - -	- - -
DD75 d	LD	(IX+d),L	3	19	- - - -	- - -
DD77 d	LD	(IX+d),A	3	19	- - - -	- - -
DD86 d	ADD	A,(IX+d)	3	19	S Z H V	Ø C
DD8E d	ADC	A,(IX+d)	3	19	S Z H V	Ø C
DD96 d	SUB	(IX+d)	3	19	S Z H V	1 C
DD9E d	SBC	(IX+d)	3	19	S Z H V	1 C
DDA6 d	AND	(IX+d)	3	19	S Z 1 P	Ø Ø
DDAE d	XOR	(IX+d)	3	19	S Z 1 P	Ø Ø
DDB6 d	OR	(IX+d)	3	19	S Z 1 P	Ø Ø
DDBE d	CP	(IX+d)	3	19	S Z H V	1 C

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS
					S Z H	P/V N C
DDCB d06	RLC	(IX+d)	4	23	S Z Ø	P Ø C
DDCB d0E	RRC	(IX+d)	4	23	S Z Ø	P Ø C
DDCB d16	RL	(IX+d)	4	23	S Z Ø	P Ø C
DDCB d1E	RR	(IX+d)	4	23	S Z Ø	P Ø C
DDCB d26	SLA	(IX+d)	4	23	S Z Ø	P Ø C
DDCB d2E	SRA	(IX+d)	4	23	S Z Ø	P Ø C
DDCB d3E	SRL	(IX+d)	4	23	S Z Ø	P Ø C
DDCB d46	BIT	0, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d4E	BIT	1, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d56	BIT	2, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d5E	BIT	3, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d66	BIT	4, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d6E	BIT	5, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d76	BIT	6, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d7E	BIT	7, (IX+d)	4	2Ø	X Z 1	X Ø -
DDCB d86	RES	0, (IX+d)	4	23	- - -	- - -
DDCB d8E	RES	1, (IX+d)	4	23	- - -	- - -
DDCB d96	RES	2, (IX+d)	4	23	- - -	- - -
DDCB d9E	RES	3, (IX+d)	4	23	- - -	- - -
DDCB dA6	RES	4, (IX+d)	4	23	- - -	- - -
DDCB dAE	RES	5, (IX+d)	4	23	- - -	- - -
DDCB dB6	RES	6, (IX+d)	4	23	- - -	- - -
DDCB dBE	RES	7, (IX+d)	4	23	- - -	- - -
DDCB dC6	SET	0, (IX+d)	4	23	- - -	- - -
DDCB dCE	SET	1, (IX+d)	4	23	- - -	- - -
DDCB dD6	SET	2, (IX+d)	4	23	- - -	- - -
DDCB dDE	SET	3, (IX+d)	4	23	- - -	- - -
DDCB dE6	SET	4, (IX+d)	4	23	- - -	- - -
DDCB dEE	SET	5, (IX+d)	4	23	- - -	- - -
DDCB dF6	SET	6, (IX+d)	4	23	- - -	- - -
DDCD dFF	SET	7, (IX+d)	4	23	- - -	- - -
DD E1	POP	IX	2	14	- - -	- - -
DD E3	EX	(SP), IX	2	23	- - -	- - -
DD E5	PUSH	IX	2	15	- - -	- - -
DD E9	JP	(IX)	2	8	- - -	- - -
DD F9	LD	SP, IX	2	1Ø	- - -	- - -

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS				
					S	Z	H	P/V	N C
DE XX	SBC	A,XX	2	7	S	Z	H	V	1 C
DF	RST	18H	1	11	-	-	-	-	- -
E0	RET	PO	1	5/11	-	-	-	-	- -
E1	POP	HL	1	10	-	-	-	-	- -
E2 ppqq	JP	PO,qqpp	3	10	-	-	-	-	- -
E3	EX	(SP),HL	1	19	-	-	-	-	- -
E4 ppqq	CALL	PO,qqpp	3	10/17	-	-	-	-	- -
E5	PUSH	HL	1	11	-	-	-	-	- -
E6 XX	AND	XX	2	7	S	Z	1	P	0 0
E7	RST	20H	1	11	-	-	-	-	- -
E8	RET	PE	1	5/11	-	-	-	-	- -
E9	JP	(HL)	1	4	-	-	-	-	- -
EA ppqq	JP	PE,qqpp	3	10	-	-	-	-	- -
EB	EX	DE,HL	1	4	-	-	-	-	- -
EC ppqq	CALL	PE,qqpp	3	10/17	-	-	-	-	- -
ED 40	IN	B,(C)	2	12	S	Z	H	P	0 -
ED 48	IN	C,(C)	2	12	S	Z	H	P	0 -
ED 50	IN	D,(C)	2	12	S	Z	H	P	0 -
ED 58	IN	E,(C)	2	12	S	Z	H	P	0 -
ED 60	IN	H,(C)	2	12	S	Z	H	P	0 -
ED 68	IN	L,(C)	2	12	S	Z	H	P	0 -
ED 78	IN	A,(C)	2	12	S	Z	H	P	0 -
ED 41	OUT	(C),B	2	12	-	-	-	-	- -
ED 49	OUT	(C),C	2	12	-	-	-	-	- -
ED 51	OUT	(C),D	2	12	-	-	-	-	- -
ED 59	OUT	(C),E	2	12	-	-	-	-	- -
ED 61	OUT	(C),H	2	12	-	-	-	-	- -
ED 69	OUT	(C),L	2	12	-	-	-	-	- -
ED 79	OUT	(C),A	2	12	-	-	-	-	- -
ED 42	SBC	HL,BC	2	15	S	Z	X	V	1 C
ED 52	SBC	HL,DE	2	15	S	Z	X	V	1 C
ED 62	SBC	HL,HL	2	15	S	Z	X	V	1 C
ED 72	SBC	HL,SP	2	15	S	Z	X	V	1 C
ED 43 ppqq	LD	(qqpp),BC	4	20	-	-	-	-	- -
ED 53 ppqq	LD	(qqpp),DE	4	20	-	-	-	-	- -
ED 44	NEG		2	8	S	Z	H	V	1 C

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS
					S Z H P/V N C	
ED 45	RETN		2	14	- - - -	- - -
ED 46	IM	Ø	2	8	- - - -	- - -
ED 56	IM	1	2	8	- - - -	- - -
ED 5E	IM	2	2	8	- - - -	- - -
ED 47	LD	I, A	2	9	- - - -	- - -
ED 4A	ADC	HL, BC	2	15	S Z H V	Ø C
ED 5A	ADC	HL, DE	2	15	S Z H V	Ø C
ED 6A	ADC	HL, HL	2	15	S Z H V	Ø C
ED 7A	ADC	HL, SP	2	15	S Z H V	Ø C
ED 4B ppqq	LD	BC, (qqpp)	4	2Ø	- - - -	- - -
ED 5B ppqq	LD	DE, (qqpp)	4	2Ø	- - - -	- - -
ED 4D	RETI		2	14	- - - -	- - -
ED 4F	LD	R, A	2	9	- - - -	- - -
ED 57	LD	A, I	2	9	S Z Ø	IFF1 Ø -
ED 5F	LD	A, R	2	9	S Z Ø	IFF1 Ø -
ED 67	RRD		2	18	S Z Ø P	Ø -
ED 6F	RLD		2	18	S Z Ø P	Ø -
ED AØ	LDI		2	16	- - Ø P/V	Ø -
ED A1	CPI		2	16	S Z H P/V	1 -
ED A2	INI		2	15	X Z X X	1 X
ED A3	OUTI		2	15	X Z X X	1 X
ED A8	LDD		2	16	- - Ø P/V	Ø -
ED A9	CPD		2	16	S Z H P/V	1 -
ED AA	IND		2	15	X Z X X	1 X
ED AB	OUTD		2	15	X Z X X	1 X
ED BØ	LDIR		2	16/21	X X Ø Ø	Ø -
ED B1	CPIR		2	16/21	S Z H P/V	1 -
ED B2	INIR		2	16/21	X 1 X X	1 X
ED B3	OTIR		2	16/21	X 1 X X	1 X
ED B8	LDDR		2	16/21	- - Ø Ø	Ø -
ED B9	CPDR		2	16/21	S Z H P/V	1 -
ED BA	INDR		2	16/21	X 1 X X	1 X
ED BB	OTDR		2	16/21	X 1 X X	1 X
EE XX	XOR	XX	2	7	S Z Ø P	Ø Ø
EF	RST	28H	1	11	- - - -	- - -
FØ	RET	P	1	5/11	- - - -	- - -

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS	AFETADOS
					S Z H P/V N C	
F1	POP	AF	1	10	S Z H P/V N C	
F2 ppqq	JP	P,qqpp	3	10	- - - -	- - - -
F3	DI		1	4	- - - -	- - - -
F4 ppqq	CALL	P,qqpp	3	10/17	- - - -	- - - -
F5	PUSH	AF	1	11	- - - -	- - - -
F6 XX	OR	XX	2	7	S Z Ø P Ø Ø	
F7	RST	30H	1	11	- - - -	- - - -
F8	RET	M	1	5/11	- - - -	- - - -
F9	LD	SP,HL	1	6	- - - -	- - - -
FA ppqq	JP	M,qqpp	3	10	- - - -	- - - -
FB	EI		1	4	- - - -	- - - -
FC ppqq	CALL	M,qqpp	3	10/17	- - - -	- - - -
FD 09	ADD	IY,BC	2	15	- - H -	Ø C
FD 19	ADD	IY,DE	2	15	- - H -	Ø C
FD 29	ADD	IY,IX	2	15	- - H -	Ø C
FD 39	ADD	IY,SP	2	15	- - H -	Ø C
FD 21 YYXX	LD	IY,XXYY	4	14	- - - -	- - - -
FD 22 ppqq	LD	(qqpp),IY	4	20	- - - -	- - - -
FD 23	INC	IY	2	10	- - - -	- - - -
FD 2A ppqq	LD	IY,(qqpp)	4	20	- - - -	- - - -
FD 2B	DEC	IY	2	10	- - - -	- - - -
FD 34d	INC	(IY+d)	3	23	S Z H V	Ø -
FD 36d XX	LD	(IY+d),XX	4	19	- - - -	- - - -
FD 46d	LD	B,(IY+d)	3	19	- - - -	- - - -
FD 4Ed	LD	C,(IY+d)	3	19	- - - -	- - - -
FD 56d	LD	D,(IY+d)	3	19	- - - -	- - - -
FD 5Ed	LD	E,(IY+d)	3	19	- - - -	- - - -
FD 66d	LD	H,(IY+d)	3	19	- - - -	- - - -
FD 6Ed	LD	L,(IY+d)	3	19	- - - -	- - - -
FD 7Ed	LD	A,(IY+d)	3	19	- - - -	- - - -
FD 70d	LD	(IY+d),B	3	19	- - - -	- - - -
FD 71d	LD	(IY+d),C	3	19	- - - -	- - - -
FD 72d	LD	(IY+d),D	3	19	- - - -	- - - -
FD 73d	LD	(IY+d),E	3	19	- - - -	- - - -
FD 74d	LD	(IY+d),H	3	19	- - - -	- - - -
FD 75d	LD	(IY+d),L	3	19	- - - -	- - - -

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS					
					S	Z	H	P/V	N	C
FD 77d	LD	(IY+d), A	3	19	-	-	-	-	-	-
FD 86d	ADD	A, (IY+d)	3	19	S	Z	H	V	Ø	C
FD 8Ed	ADC	A, (IY+d)	3	19	S	Z	H	V	Ø	C
FD 96d	SUB	(IY+d)	3	19	S	Z	H	V	1	C
FD 9Ed	SBC	(IY+d)	3	19	S	Z	H	V	1	C
FD A6d	AND	(IY+d)	3	19	S	Z	Ø	P	Ø	Ø
FD AE d	XOR	(IY+d)	3	19	S	Z	Ø	P	Ø	Ø
FD B6d	OR	(IY+d)	3	19	S	Z	Ø	P	Ø	Ø
FD BE d	CP	(IY+d)	3	19	S	Z	H	V	1	C
FDCBdØ6	RLC	(IY+d)	4	23	S	Z	Ø	P	Ø	C
FDCBdØE	RRC	(IY+d)	4	23	S	Z	Ø	P	Ø	C
FDCBd16	RL	(IY+d)	4	23	S	Z	Ø	P	Ø	C
FDCBd1E	RR	(IY+d)	4	23	S	Z	Ø	P	Ø	C
FDCBd26	SLA	(IY+d)	4	23	S	Z	Ø	P	Ø	C
FDCBd2E	SRA	(IY+d)	4	23	S	Z	Ø	P	Ø	C
FDCBd3E	SRL	(IY+d)	4	23	S	Z	Ø	P	Ø	C
FDCBd46	BIT	Ø, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd4E	BIT	1, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd56	BIT	2, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd5E	BIT	3, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd66	BIT	4, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd6E	BIT	5, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd76	BIT	6, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd7E	BIT	7, (IY+d)	4	2Ø	X	Z	1	X	Ø	-
FDCBd86	RES	Ø, (IY+d)	4	23	-	-	-	-	-	-
FDCBd8E	RES	1, (IY+d)	4	23	-	-	-	-	-	-
FDCBd96	RES	2, (IY+d)	4	23	-	-	-	-	-	-
FDCBd9E	RES	3, (IY+d)	4	23	-	-	-	-	-	-
FDCBdA6	RES	4, (IY+d)	4	23	-	-	-	-	-	-
FDCBdAE	RES	5, (IY+d)	4	23	-	-	-	-	-	-
FDCBdB6	RES	6, (IY+d)	4	23	-	-	-	-	-	-
FDCBdB E	RES	7, (IY+d)	4	23	-	-	-	-	-	-
FDCBdC6	SET	Ø, (IY+d)	4	23	-	-	-	-	-	-
FDCBdCE	SET	1, (IY+d)	4	23	-	-	-	-	-	-
FDCBdD6	SET	2, (IY+d)	4	23	-	-	-	-	-	-
FDCBdDE	SET	3, (IY+d)	4	23	-	-	-	-	-	-

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS					
					S	Z	H	P/V	N	C
FDCBdE6	SET	4,(IY+d)	4	23	-	-	-	-	-	-
FDCBdEE	SET	5,(IY+d)	4	23	-	-	-	-	-	-
FDCBdF6	SET	6,(IY+d)	4	23	-	-	-	-	-	-
FDCBdFE	SET	7,(IY+d)	4	23	-	-	-	-	-	-
FD E1	POP	IY	2	14	-	-	-	-	-	-
FD E3	EX	(SP),IY	2	23	-	-	-	-	-	-
FD E5	PUSH	IY	2	15	-	-	-	-	-	-
FD E9	JP	(IY)	2	8	-	-	-	-	-	-
FD F9	LD	SP,IY	2	10	-	-	-	-	-	-
FE XX	CP	XX	2	7	S	Z	H	V	1	C
FF	RST	38H	1	11	-	-	-	-	-	-

OBS:.

XX	Um BYTE DE DADOS
XXYY	Dois BYTES de dados, onde (XX) representa o BYTE mais significativo e (YY) o BYTE menos significativo.
qqpp	Posicionamento de memória, onde (qq) representa o BYTE mais significativo e (pp) representa o BYTE menos significativo.
qq	Posicionamento de dispositivo de entrada e saída.
d	Endereçamento indexado.
X	Indica condição de FLAG não determinada
-	Indica FLAG não afetado

APÊNDICE G

CÓDIGO DE OPERAÇÃO POR ORDEM ALFABÉTICA

		ORG	100H	
AA		DEFB	0AAH	
43		DEFB	67	
00		DEFB	00H	
	CONV	DEFL	5678H	
31323334		DEFM	'1234567890 POIU YTTR EWQQ	
35363738				
39302050				
4F495520				
59545452				
20455751				
5120				
	NNOMU	DEFS	4	
CBAD		DEFW	0ADCBH	
	CONV	DEFL	990H	
	END	EQU	0543H	
	IND	EQU	05H	
	DADO	EQU	20H	
	DIS	EQU	30H	
8E	SOMCY:	ADC	A, (HL)	
DD8E05		ADC	A, (IX+IND)	
FD8E05		ADC	A, (IY+IND)	
8F		ADC	A, A	
88		ADC	A, B	
89		ADC	A, C	
8A		ADC	A, D	
8B		ADC	A, E	
8C		ADC	A, H	
CE20		ADC	A, DADO	
ED4A	SOCY16:	ADC	HL, BC	
ED5A		ADC	HL, DE	
ED6A		ADC	HL, HL	
ED7A		ADC	HL, SP	
86	SOMA:	ADD	A, (HL)	
DD86FB		ADD	A, (IX-IND)	
FD86FB		ADD	A, (IY-IND)	
87		ADD	A, A	
80		ADD	A, B	
81		ADD	A, C	
82		ADD	A, D	
83		ADD	A, E	
84		ADD	A, H	
C620		ADD	A, DADO	
09	SOM16:	ADD	HL, BC	
19		ADD	HL, DE	

29	ADD	HL,HL
39	ADD	HL,SP
DD09	ADD	IX,8C
DD19	ADD	IX,DE
DD29	ADD	IX,IX
DD39	ADD	IX,SP
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD29	ADD	IY,IY
FD39	ADD	IY,SP

A6	LOGAND:	AND	(HL)
DDA605		AND	(IX+IND)
FDA605		AND	(IY+IND)
A7		AND	A
A0		AND	B
A1		AND	C
A2		AND	D
A3		AND	E
A4		AND	H
A5		AND	L
E620		AND	DADO

CB46	TEST0:	BIT	0,(HL)
DDCBFB46		BIT	0,(IX-IND)
FDCBFB46		BIT	0,(IY-IND)
CB47		BIT	0,A
CB40		BIT	0,B
CB41		BIT	0,C
CB42		BIT	0,D
CB43		BIT	0,E
CB44		BIT	0,H
CB45		BIT	0,L

CB4E	TEST1:	BIT	1,(HL)
DDCB054E		BIT	1,(IX+IND)
FDCB054E		BIT	1,(IY+IND)
CB4F		BIT	1,A
CB48		BIT	1,B
CB49		BIT	1,C
CB4A		BIT	1,D
CB4B		BIT	1,E
CB4C		BIT	1,H
CB4D		BIT	1,L

CB56	TEST2:	BIT	2,(HL)
DDCBFB56		BIT	2,(IX-IND)
FDCBFB56		BIT	2,(IY-IND)
CB57		BIT	2,A
CB50		BIT	2,B
CB51		BIT	2,C
CB52		BIT	2,D
CB53		BIT	2,E
CB54		BIT	2,H
CB55		BIT	2,L

CB5E	TEST3:	BIT	3,(HL)
DDCB055E		BIT	3,(IX+IND)

FDCB055E		BIT	3,(IY+IND)		
CB5F		BIT	3,A		
CB58		BIT	3,B		
CB59		BIT	3,C		
CB5A		BIT	3,D		
CB5B		BIT	3,E		
CB5C		BIT	3,H		
CB5D		BIT	3,L		
ED40					
CB66	TEST4:	BIT	4,(HL)		
DDCBFB66		BIT	4,(IX-IND)		
FDCBFB66		BIT	4,(IY-IND)		
CB67		BIT	4,A		
CB60		BIT	4,B		
CB61		BIT	4,C		
CB62		BIT	4,D		
CB63		BIT	4,E		
CB64		BIT	4,H		
CB65		BIT	4,L		
CB6E	TEST5:	BIT	5,(HL)		
DDCB056E		BIT	5,(IX+IND)		
FDCB056E		BIT	5,(IY+IND)		
CB6F		BIT	5,A		
CB68		BIT	5,B		
CB69		BIT	5,C		
CB6A		BIT	5,D		
CB6B		BIT	5,E		
CB6C		BIT	5,H		
CB6D		BIT	5,L		
CB76	TEST6:	BIT	6,(HL)		
DDCBFB76		BIT	6,(IX-IND)		
FDCBFB76		BIT	6,(IY-IND)		
CB77		BIT	6,A		
CB70		BIT	6,B		
CB71		BIT	6,C		
CB72		BIT	6,D		
CB73		BIT	6,E		
CB74		BIT	6,H		
CB75		BIT	6,L		
CB7E	TEST7:	BIT	7,(HL)		
DDCB057E		BIT	7,(IX+IND)		
FDCB057E		BIT	7,(IY+IND)		
CB7F		BIT	7,A		
CB78		BIT	7,B		
CB79		BIT	7,C		
CB7A		BIT	7,D		
CB7B		BIT	7,E		
CB7C		BIT	7,H		
CB7D		BIT	7,L		
CD2D02'	SUBROT:	CALL	SUBROT		
DC2D02'		CALL	C,SUBROT		
FC2D02'		CALL	M,SUBROT		
D42D02'		CALL	NC,SUBROT		
C42D02'		CALL	NZ,SUBROT		

F42D02'	CALL	P, SUBROT		
EC2D02'	CALL	PE, SUBROT		
E42D02'	CALL	PO, SUBROT		
CC2D02'	CALL	Z, SUBROT		
3F	CCF			
BE	COMPAR:	CP	(HL)	
DDBE4E		CP	(IX+78)	
FDBE34		CP	(IY+52)	
BF		CP	A	
B8		CP	B	
B9		CP	C	
BA		CP	D	
BB		CP	E	
BC		CP	H	
BD		CP	L	
FE20		CP	DADO	
EDA9	CMPGRP:	CPD		
EDB9		CPDR		
EDA1		CPI		
EDB1		CPIR		
2F		CPL		
27		DAA		
35	DECR:	DEC	(HL)	
DD35B2		DEC	(IX-78)	
FD35CC		DEC	(IY-52)	
3D		DEC	A	
05		DEC	B	
0D		DEC	C	
15		DEC	D	
1D		DEC	E	
25		DEC	H	
2D		DEC	L	
0B	DECR16:	DEC	BC	
1B		DEC	DE	
2B		DEC	HL	
DD2B		DEC	IX	
FD2B		DEC	IY	
3B		DEC	SP	
F3		DI		
1021		DJNZ	23H	
FB		EI		
E3		EX	(SP), HL	
DDE3		EX	(SP), IX	
FDE3		EX	(SP), IY	
0B		EX	AF, AF'	
EB		EX	DE, HL	

ED46 INTER: IM
 ED56 IM
 ED5E IM

 ED78 LEIPOR: IN
 DB30 IN
 ED40 IN
 ED48 IN
 ED50 IN
 ED58 IN
 ED60 IN
 ED68 IN

 34 INCR: INC
 DD343E INC
 FD341B INC
 3C INC
 04 INC
 0C INC
 14 INC
 1C INC
 24 INC
 2C INC

 03 INCR16: INC
 13 INC
 23 INC
 DD23 INC
 FD23 INC
 33 INC

 EDAA INCRP: IND
 ED8A INDR
 EDA2 INI
 EDB2 INIR

 E9 JUMP: JP
 DDE9 JP
 FDE9 JP
 C3B902' JP
 DAB902' JP
 FAB902' JP
 D2B902' JP
 C2B902' JP
 F2B902' JP
 EAB902' JP
 E2B902' JP
 CAB902' JP

 1840 JUPREL: JR
 38D3 JR
 30FA JR
 20F8 JR
 282E JR

 02 CARREG: LD

0
 1
 2

 A, (C)
 A, (DIS)
 B, (C)
 C, (C)
 D, (C)
 E, (C)
 H, (C)
 L, (C)

 (HL)
 (IX+3EH)
 (IY+01BH)
 A
 B
 C
 D
 E
 H
 L

 BC
 DE
 HL
 IX
 IY
 SP

 (HL)
 (IX)
 (IY)
 JUMP
 C, JUMP
 M, JUMP
 NC, JUMP
 NZ, JUMP
 P, JUMP
 PE, JUMP
 PQ, JUMP
 Z, JUMP

 66
 C, -43
 NC, JUPREL-\$
 NZ, JUPREL-\$
 Z, DIS

 (BC), A

12		LD	(DE),A		
77		LD	(HL),A		
70		LD	(HL),B		
71		LD	(HL),C		
72		LD	(HL),D		
73		LD	(HL),E		
74		LD	(HL),H		
75		LD	(HL),L		
3620		LD	(HL),DADO		
DD7705	CARIND:	LD	(IX+IND),A		
DD7022		LD	(IX+34),B		
DD71DE		LD	(IX-34),C		
DD726C		LD	(IX+06CH),D		
DD7394		LD	(IX-06CH),E		
DD7505		LD	(IX+IND),L		
DD7405		LD	(IX+IND),H		
DD360520		LD	(IX+IND),DADO		
FD77FB		LD	(IY-IND),A		
FD707F		LD	(IY+127),B		
FD7180		LD	(IY-128),C		
FD720A		LD	(IY+0AH),D		
FD73F6		LD	(IY-0AH),E		
FD75FB		LD	(IY-IND),L		
FD74FB		LD	(IY-IND),H		
FD36FB20		LD	(IY-IND),DADO		
324305	CARDIR:	LD	(END),A		
ED434305		LD	(END),BC		
ED534305		LD	(END),DE		
224305		LD	(END),HL		
DD224305		LD	(END),IX		
FD224305		LD	(END),IY		
ED734305		LD	(END),SP		
0A	ACUMUL:	LD	A,(BC)		
1A		LD	A,(DE)		
7E		LD	A,(HL)		
DD7E05		LD	A,(IX+IND)		
FD7EFB		LD	A,(IY-IND)		
3A4305		LD	A,(END)		
7F		LD	A,A		
78		LD	A,B		
79		LD	A,C		
7A		LD	A,D		
7B		LD	A,E		
7C		LD	A,H		
7D		LD	A,L		
3E20		LD	A,DADO		
ED5F		LD	A,R		
46	REGB:	LD	B,(HL)		
DD46FB		LD	B,(IX-IND)		
FD4605		LD	B,(IY+IND)		
47		LD	B,A		
40		LD	B,B		
41		LD	B,C		

42		LD	B,D		
43		LD	B,E		
44		LD	B,H		
45		LD	B,L		
0620		LD	B,DADO		
ED4B4305	CA16BC:	LD	BC,(END)		
014305		LD	BC,END		
4E	REGC:	LD	C,(HL)		
DD4E05		LD	C,(IX+IND)		
FD4E05		LD	C,(IY+IND)		
4F		LD	C,A		
48		LD	C,B		
49		LD	C,C		
4A		LD	C,D		
4B		LD	C,E		
4C		LD	C,H		
4D		LD	C,L		
0E20		LD	C,DADO		
56	REGD:	LD	D,(HL)		
DD56FB		LD	D,(IX-IND)		
FD56FB		LD	D,(IY-IND)		
57		LD	D,A		
58		LD	D,B		
59		LD	D,C		
5A		LD	D,D		
5B		LD	D,E		
5C		LD	D,H		
5D		LD	D,L		
1620		LD	D,DADO		
ED5B4305	CR16DE:	LD	DE,(END)		
114305		LD	DE,END		
5E	REGE:	LD	E,(HL)		
DD5E05		LD	E,(IX+IND)		
FD5E05		LD	E,(IY+IND)		
5F		LD	E,A		
58		LD	E,B		
59		LD	E,C		
5A		LD	E,D		
5B		LD	E,E		
5C		LD	E,H		
5D		LD	E,L		
1E20		LD	E,DADO		
66	REGH:	LD	H,(HL)		
DD6605		LD	H,(IX+IND)		
FD6605		LD	H,(IY+IND)		
67		LD	H,A		
68		LD	H,B		
69		LD	H,C		
6A		LD	H,D		
6B		LD	H,E		
6C		LD	H,H		
6D		LD	H,L		

2620		LD	H,DADO		
2A4305	CA16HL:	LD	HL,(END)		
214305		LD	HL,END		
ED47		LD	I,A		
DD2A4305	CA16IX:	LD	IX,(END)		
DD214305		LD	IX,END		
FD2A4305	CA16IY:	LD	IY,(END)		
FD214305		LD	IY,END		
6E	REGL:	LD	L,(HL)		
DD6E05		LD	L,(IX+IND)		
FD6EF8		LD	L,(IY-IND)		
6F		LD	L,A		
68		LD	L,B		
69		LD	L,C		
6A		LD	L,D		
6B		LD	L,E		
6C		LD	L,H		
6D		LD	L,L		
2E20		LD	L,DADO		
ED4F		LD	R,A		
ED7B4305	CA16SP:	LD	SP,(END)		
F9		LD	SP,HL		
DDF9		LD	SP,IX		
FDF9		LD	SP,IY		
314305		LD	SP,END		
EDA8	CARGRP:	LDD			
EDB8		LDDR			
EDA0		LDI			
EDB0		LDIR			
ED44		NEG			
00		NOP			
B6	LOGOR:	OR	(HL)		
DDB605		OR	(IX+IND)		
FDB605		OR	(IY+IND)		
B7		OR	A		
B0		OR	B		
B1		OR	C		
B2		OR	D		
B3		OR	E		
B4		OR	H		
B5		OR	L		
F620		OR	DADO		
ED79	SAIPOR:	OUT	(C),A		
ED41		OUT	(C),B		
ED49		OUT	(C),C		
ED51		OUT	(C),D		

ED59	OUT	(C),E	238	18839	8883
ED61	OUT	(C),H	238		8883
ED69	OUT	(C),L	238		8883
D330	OUT	(DIS),A	238		8883
ED88	SAIGRP:	OTDR	238		8883
ED83		OTIR	238		8883
EDAB		OUTD	238		8883
EDA3		OUTI	238		8883
F1	CARPIH:	POP	238	18839	8883
C1		POP	238		8883
D1		POP	238		8883
E1		POP	238		8883
DDE1		POP	238		8883
FDE1		POP	238		8883
F5	DESPIH:	PUSH	238		8883
C5		PUSH	238		8883
D5		PUSH	238		8883
E5		PUSH	238		8883
DDE5		PUSH	238		8883
FDE5		PUSH	238		8883
CB86	RES0:	RES	238		8883
DDCB0586		RES	238		8883
FDCB0586		RES	238		8883
CB87		RES	238		8883
CB80		RES	238		8883
CB81		RES	238		8883
CB82		RES	238		8883
CB83		RES	238		8883
CB84		RES	238		8883
CB85		RES	238		8883
CB8E	RES1:	RES	238		8883
DDCB058E		RES	238		8883
FDCB058E		RES	238		8883
CB8F		RES	238		8883
CB88		RES	238		8883
CB89		RES	238		8883
CB8A		RES	238		8883
CB8B		RES	238		8883
CB8C		RES	238		8883
CB8D		RES	238		8883
CB96	RES2:	RES	238		8883
DDCB0596		RES	238		8883
FDCB0596		RES	238		8883
CB97		RES	238		8883
CB97		RES	238		8883
CB90		RES	238		8883
CB91		RES	238		8883
CB92		RES	238		8883
CB93		RES	238		8883
CB94		RES	238		8883
CB95		RES	238		8883

CB9E	RES3:	RES	3,(HL)		
DDCB059E		RES	3,(IX+IND)	100	9203
FDCB059E		RES	3,(IY+IND)	100	1203
CB9F		RES	3,A	100	9203
CB98		RES	3,B	100	0200
CB99		RES	3,C		
CB9A		RES	3,D	8016 :9901A0	8803
CB9B		RES	3,E	4010	2803
CB9C		RES	3,H	0700	8A03
CB9D		RES	3,L	1700	0A03
CBA6	RES4:	RES	4,(HL)	909	12
DDCB05A6		RES	4,(IX+IND)	909	10
FDCB05A6		RES	4,(IY+IND)	909	01
CBA7		RES	4,A	909	13
CBA0		RES	4,B	909	0003
CBA1		RES	4,C	909	1303
CBA2		RES	4,D		
CBA3		RES	4,E	8204	23
CBA4		RES	4,H	8204	20
CBA5		RES	4,L	8204	20
CBAE	RES5:	RES	5,(HL)	8204	2300
DDCB05AE		RES	5,(IX+IND)	8204	2303
FDCB05AE		RES	5,(IY+IND)		
CBAF		RES	5,A	838	8803
CBA8		RES	5,B	838	000300
CBA9		RES	5,C	838	000303
CBAA		RES	5,D	838	8803
CBAB		RES	5,E	838	0803
CBAC		RES	5,H	838	1803
CBAD		RES	5,L	838	0803
CBB6	RES6:	RES	6,(HL)	838	8803
DDCB05B6		RES	6,(IX+IND)	838	8803
FDCB05B6		RES	6,(IY+IND)		
CBB7		RES	6,A	838	3803
CBB0		RES	6,B	838	000300
CBB1		RES	6,C	838	000303
CBB2		RES	6,D	838	8803
CBB3		RES	6,E	838	8803
CBB4		RES	6,H	838	9803
CBB5		RES	6,L	838	0803
CBBE	RES7:	RES	7,(HL)	838	8803
DDCB05BE		RES	7,(IX+IND)	838	0803
FDCB05BE		RES	7,(IY+IND)		
CBBF		RES	7,A	838	9803
CBB8		RES	7,B	838	000300
CBB9		RES	7,C	838	000303
CBBA		RES	7,D	838	9803
CBBB		RES	7,E	838	9803
CBBC		RES	7,H	838	0903
CBBD		RES	7,L	838	0903
C9	RETOR:	RET		838	5903
D8		RET	C	838	5903
F8		RET	M	838	5903

D0
 C0
 F0
 E8
 E0
 C8

RET
 RET
 RET
 RET
 RET
 RET

NC
 NZ
 P
 PE
 PO
 Z

ED4D
 ED45

RETINT: RETI
 RETN

CB16
 DDCB0516
 FDCB0516
 CB17
 CB10
 CB11
 CB12
 CB13
 CB14
 CB15

ROTL: RL
 RL
 RL
 RL
 RL
 RL
 RL
 RL
 RL

(HL)
 (IX+IND)
 (IY+IND)
 A
 B
 C
 D
 E
 H
 L

17

RLA

CB06
 DDCB0506
 FDCB0506
 CB07
 CB00
 CB01
 CB02
 CB03
 CB04
 CB05

ROTLC: RLC
 RLC
 RLC
 RLC
 RLC
 RLC
 RLC
 RLC
 RLC

(HL)
 (IX+IND)
 (IY+IND)
 A
 B
 C
 D
 E
 H
 L

07

RLCA

ED6F

RLD

CB1E
 ODCB051E
 FDCB051E
 CB1F
 CB18
 CB19
 CB1A
 CB1B
 CB1C
 CB1D

ROLR: RR
 RR
 RR
 RR
 RR
 RR
 RR
 RR
 RR

(HL)
 (IX+IND)
 (IY+IND)
 A
 B
 C
 D
 E
 H
 L

1F

RRA

CB0E
 DDCB050E
 FDCB050E
 CB0F
 CB08
 CB09
 CB0A

ROLRC: RRC
 RRC
 RRC
 RRC
 RRC
 RRC
 RRC

(HL)
 (IX+IND)
 (IY+IND)
 A
 B
 C
 D

CBD6	SET2:	SET	2,(HL)	T38	3783
DDCB05D6		SET	2,(IX+IND)	T38	3783
FDCB05D6		SET	2,(IY+IND)	T38	3783
CBD7		SET	2,A	T38	3783
CBD0		SET	2,B	T38	3783
CBD1		SET	2,C	T38	3783
CBD2		SET	2,D	T38	3783
CBD3		SET	2,E	T38	3783
CBD4		SET	2,H	T38	3783
CBD5		SET	2,L	T38	3783
C8DE	SET3:	SET	3,(HL)	A38	3783
DDCB05DE		SET	3,(IX+IND)	A38	3783
DDCB05DE		SET	3,(IX+IND)	A38	3783
C8DF		SET	3,A	A38	3783
C8D8		SET	3,B	A38	3783
C8D9		SET	3,C	A38	3783
C8DA		SET	3,D	A38	3783
C8DB		SET	3,E	A38	3783
C8DC		SET	3,H	A38	3783
C8DD		SET	3,L	A38	3783
CBE6	SET4:	SET	4,(HL)	A38	3783
DDCB05E6		SET	4,(IX+IND)	A38	3783
FDCB05E6		SET	4,(IY+IND)	A38	3783
CBE7		SET	4,A	A38	3783
CBE0		SET	4,B	A38	3783
CBE1		SET	4,C	A38	3783
CBE2		SET	4,D	A38	3783
CBE3		SET	4,E	A38	3783
CBE4		SET	4,H	A38	3783
CBE5		SET	4,L	A38	3783
CBE6	SET5:	SET	5,(HL)	A38	3783
DDCB05EE		SET	5,(IX+IND)	A38	3783
FDCB05EE		SET	5,(IY+IND)	A38	3783
CBEF		SET	5,A	A38	3783
CBE8		SET	5,B	A38	3783
CBE9		SET	5,C	A38	3783
CBEA		SET	5,D	A38	3783
CBEB		SET	5,E	A38	3783
CBEC		SET	5,H	A38	3783
CBED		SET	5,L	A38	3783
CBF6	SET6:	SET	6,(HL)	A38	3783
DDCB05F6		SET	6,(IX+IND)	A38	3783
FDCB05F6		SET	6,(IY+IND)	A38	3783
CBF7		SET	6,A	A38	3783
CBF0		SET	6,B	A38	3783
CBF1		SET	6,C	A38	3783
CBF2		SET	6,D	A38	3783
CBF3		SET	6,E	A38	3783
CBF4		SET	6,H	A38	3783
CBF5		SET	6,L	A38	3783
CBFE	SET7:	SET	7,(HL)	A38	3783
DDCB05FE		SET	7,(IX+IND)	A38	3783
FDCB05FE		SET	7,(IY+IND)	A38	3783

CBFF		SET	7,A		
CBF8		SET	7,B		
CBF9		SET	7,C		
CBFA		SET	7,D		
CBFB		SET	7,E		
CBFC		SET	7,H		
CBFD		SET	7,L		
CB26	ROLLA:	SLA	(HL)		
DDCB0526		SLA	(IX+IND)		
FDCB0526		SLA	(IY+IND)		
CB27		SLA	A		
CB20		SLA	B		
CB21		SLA	C		
CB22		SLA	D		
CB23		SLA	E		
CB24		SLA	H		
CB25		SLA	L		
CB2E	ROLRA:	SRA	(HL)		
DDCB052E		SRA	(IX+IND)		
FDCB052E		SRA	(IY+IND)		
CB2F		SRA	A		
CB28		SRA	B		
CB29		SRA	C		
CB2A		SRA	D		
CB2B		SRA	E		
CB2C		SRA	H		
CB2D		SRA	L		
CB3E	ROLRL:	SRL	(HL)		
DDCB053E		SRL	(IX+IND)		
FDCB053E		SRL	(IY+IND)		
CB3F		SRL	A		
CB38		SRL	B		
CB39		SRL	C		
CB3A		SRL	D		
CB3B		SRL	E		
CB3C		SRL	H		
CB3D		SRL	L		
96	SUBTR:	SUB	(HL)		
DD9605		SUB	(IX+IND)		
FD9605		SUB	(IY+IND)		
97		SUB	A		
97		SUB	A		
90		SUB	B		
91		SUB	C		
92		SUB	D		
93		SUB	E		
94		SUB	H		
95		SUB	L		
D620		SUB	DADO		
AE	LOGXOR:	XOR	(HL)		
DDAE05		XOR	(IX+IND)		
FDAE05		XOR	(IY+IND)		
AF		XOR	A		

AB	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
EE20	XOR	DADO

CORRESPONDÊNCIA DOS MNEMÔNICOS DO MICROPROCESSADOR 8080 COM O Z-80

319

8080		Z-80
INR	M	INC (HL)
INR	REG	INC REG
INX	SS	INC SS
JC	ENDER	JP C, ENDER
JM	ENDER	JP M, ENDER
JMP	ENDER	JP ENDER
JNC	ENDER	JP NC, ENDER
JNZ	ENDER	JP NZ, ENDER
JP	ENDER	JP P, ENDER
JPE	ENDER	JP PE, ENDER
JPO	ENDER	JP PO, ENDER
JZ	ENDER	JP Z, ENDER
LDA	ENDER	LD A, (ENDER)
LDAX	B	LD A, (BC)
LDAX	D	LD A, (DE)
LHLD	ENDER	LD HL, (ENDER)
LXI	B, DADO	LD BC, DADO
LXI	D, DADO	LD DE, DADO
LXI	H, DADO	LD HL, DADO
LXI	SP, DADO	LD SP, DADO
MOV	M, REG	LD (HL), REG
MOV	REG, M	LD REG, (HL)
MOV	REG, REG	LD REG, REG
MVI	M, DADO	LD (HL), DADO
MVI	REG, DADO	LD REG, DADO
NOP		NOP
ORA	M	OR M
ORA	REG	OR REG
ORI	DADO	OR DADO
OUT	ENDER	OUT (ENDER), A
PCHL		JP (HL)
PUSH	B	PUSH BC
PUSH	D	PUSH DE
PUSH	H	PUSH HL
PUSH	PSW	PUSH AF
POP	B	POP BC
POP	D	POP DE

8080		Z-80
POP	H	POP HL
POP	PSW	POP PF
RAL		RLA
RAR		RRA
RC		RET C
RET		RET
RLC		RLCA
RM		RET M
RNC		RET NC
RNZ		RET NZ
RP		RET P
RPE		RET PE
RPO		RET PO
RRC		RRCA
RST	0	RST 0
RST	1	RST 8
RST	2	RST 10H
RST	3	RST 18H
RST	4	RST 20H
RST	5	RST 28H
RST	6	RST 30H
RST	7	RST 38H
RZ		RET 2
SBB	M	SUB (HL)
SBB	REG	SUB REC
SBI	DADO	SBC DADO
SHLD	ENDER	LD (ENDER), HL
SPHL		LD SP, HL
STA	ENDER	LD (ENDER), A
STAX	B	LD (BC), A
STAX	D	LD (DE), A
STC		SCF
SUB	M	SUB (HL)
SUB	REG	SUB REG
SUI	DADO	SUB DADO
XC HC		EX DE, HL
XRA	M	XOR (HL)

XRA	REG	XOR	REG
XRI	DADO	XOR	DADO
XTHL		EX	(SP), HL

OBS:.

SS Representa BC
DE
HL
SP

REG Representa B
C
D
E
H
L
A

DADO: Movimentação de dados de 16 BITS ou 8 BITS.

ENDER: Endereçamento de memória e dispositivo externo

ÍNDICE REMISSIVO

- Aritmética, 23
ASCII, 275
BYTE de Dados, 18
BYTE de Direcionamento, 19
BYTE de Endereçamento, 19
BYTE Dispositivo, 18
CARRY, 43
CHIP, 14
Código de Operação, 207
Comparação, 228
Compilador, 202
Computador, 14
Contador, 255
Controle Interno, 27
Conversão, 238, 240, 242, 244, 247
DELAY, 224, 250
Delimitadores, 207
Dispositivo I/O, 250
Divisão, 217
Endereçamento Imediato, 31
Endereçamento Imediato Estendido, 33
Endereçamento, 31
Endereçamento Implícito, 38
Endereçamento Indexado, 38
Endereçamento Indireto por Registrador, 34
Endereçamento Registrador, 32
Endereçamento Relativo, 36
Entrada e Saída de Dados, 27
Endereçamento Paginado Zero, 37
EQUATE, 212
FLAG, 43, 44, 45, 46, 47, 48
Fluxo de Dados, 39
HALF-CARRY, 48
HALF-BORROW, 47
HARDWARE, 13
IMØ, IM1, IM2, 51
Instruções, 23, 59
Interpretador, 203
Interrupção, 49
LABEL, 206
Linguagem, 201
Linguagem Assembly, 205
Linguagem Máquina, 203
Lógica, 23
LOOP, 222
Manipulação de BITS, 26
Mascarada, 51
Memória, 17
Microprocessador, 14
Movimentação de Dados, 33
Multiplicação, 216, 237
Não Mascarada, 49
Número de BYTE de uma Instrução, 28
Operações Aritméticas e Lógicas, 24
ORG, 212
OVERFLOW, 45
Paridade, 45, 222, 226
Programação, 215
Pseudo-Instrução, 208
Registrador, 20
Restauração, 220
Retornos, 26
Rotação e Deslocamento, 25, 26
Rotinas, 230
Saltos, 26
Sinal, 44
SOFTWARE, 13
Soma, 233
Subtração, 233
Tabelas Exponenciais, 273
TITLE, 209

Transferência de Blocos, 25

Transferência de Dados, 24

Troca de Informação, 24

Troca de Registradores, 219

Zero, 44

REG

DADO

INFORMAÇÃO

Aritmética, 23

ASCII, 275

Byte de Dados, 19

Byte de Deslocamento, 19

Byte de Deslocamento, 19

Byte Deslocativo, 19

CARRY, 83

CHIP, 14

Código de Instrução, 205

Comparação, 278

Condição, 205

Computador, 14

Contador, 255

Control - Instrução, 23

Conversão, 238, 240, 42, 244, 245

DELAY, 204, 206

Definição, 205

Diagrama, 204, 206

Divisão, 217

Endereçamento Direto, 21

Endereçamento Indireto Registrador, 21

Endereçamento, 21

Endereçamento Indireto, 21

Endereçamento Indireto, 21

Endereçamento Indireto por Registrador, 21

End, 14

Endereçamento Indireto, 21

Endereçamento Indireto, 21

Endereçamento Indireto, 21

Endereçamento Indireto, 21

EXACT, 215

EXACT, 215, 216, 217, 218, 219

Fluxo de Dados, 14

Fluxo de Dados, 14

Fluxo de Dados, 14

HARWARE, 14

TITLE, 205

Tabela Explicativa, 273

Título, 217

Título, 217

Título, 217

Título, 217

Título, 217

Título, 217

Título e Subtítulo, 217

Título, 217

Título, 217

Título-Indicador, 217

Título, 217

Título, 217

Título, 217

Título, 217

Título, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título, 217

Título, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título, 217

Título, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título e Subtítulo, 217

Título, 217

ÍNDICE DE FIGURAS

- Fig. 2.1 - Registrador Geral, 20
- Fig. 3.1 - Transferência de Dados entre CPU e Memória, 24
- Fig. 3.2 - Transferência de Dados entre os Registradores Internos, 24
- Fig. 3.3 - Exemplo de Operação Aritmética e Lógica, 25
- Fig. 3.4 - Rotação de um BIT para a Esquerda, 26
- Fig. 3.5 - Manipulação de BITS, 26
- Fig. 3.6 - Saltos, 26
- Fig. 3.7 - CALL e Retorno, 27
- Fig. 3.8 - Dispositivo de Entrada e Saída, 27
- Fig. 3.9 - Instrução de um BYTE, 29
- Fig. 3.10 - Instrução de dois BYTES, 29
- Fig. 3.11 - Instrução de três BYTES, 30
- Fig. 3.12 - Instrução de quatro BYTES, 30
- Fig. 4.1 - Endereçamento por Registrador, 33
- Fig. 4.2 - Movimentação de Dados, 33
- Fig. 4.3 - Formato de Endereçamento Imediato Estendido, 33
- Fig. 4.4 - Endereçamento Imediato Estendido, 34
- Fig. 4.5 - Endereçamento Indireto por Registrador, 34
- Fig. 4.6 - Fluxo de Dados, 35
- Fig. 4.7 - Fluxo de Dados de Endereçamento Relativo, 36
- Fig. 4.8 - Fluxo de Dado de um RESTART, 38
- Fig. 4.9 - Endereçamento Indexado, 39
- Fig. 4.10 - Fluxo de Dados do Endereçamento Indexado, 40
- Fig. 4.11 - Formato da Instrução de Endereçamento de BIT, 41
- Fig. 5.1 - Registrador de FLAG, 43
- Fig. 5.2 - FLAG CARRY, 44
- Fig. 5.3 - FLAG Zero, 44
- Fig. 5.4 - FLAG Sinal, 45
- Fig. 5.5 - FLAG de Paridade, 46
- Fig. 5.6 - FLAG de OVERFLOW, 47
- Fig. 5.7 - FLAG HALF-CARRY e HALF-BORROW, 48

- Fig. 6.1 - Fluxo da Interrupção Não Mascarada, 51
- Fig. 6.2 - Relação dos Códigos e Endereços no Modo "Zero", 52
- Fig. 6.3 - Fluxo da Interrupção Mascarada no Modo 0, 54
- Fig. 6.4 - Montagem no Endereçamento no Modo 2, 55
- Fig. 6.5 - Fluxo da Interrupção Mascarada no Modo 2, 57

- Fig. 7.1 - Estrutura da Instrução CPDR, 96
- Fig. 7.2 - Ajuste para BCD, 100
- Fig. 7.3 - Estrutura da Instrução DJNZ e, 105

- Fig. 8.1 - Programa Assembler, 201
- Fig. 8.2 - Programa Compilador, 202
- Fig. 8.3 - Programa Interpretador, 203
- Fig. 8.4 - Linguagem de Máquina em Binário, 204
- Fig. 8.5 - Linguagem de Máquina em Hexadecimal, 205
- Fig. 8.6 - Estrutura da Linguagem Assembly, 206

- Fig. 9.1 - Teste do Acumulador, 216
- Fig. 9.2 - Teste do Acumulador, 216
- Fig. 9.2.A - Multiplica por dois o Conteúdo do Acumulador, 217
- Fig. 9.3 - Divide por dois o Conteúdo do Acumulador, 217
- Fig. 9.4 - Zerar o Conteúdo do Acumulador, 218
- Fig. 9.5 - Inverter um BIT do Conteúdo do Acumulador, 218
- Fig. 9.6 - Verificação do Conteúdo de um Registrador, 218
- Fig. 9.7 - Programa de Troca de Registradores, 219
- Fig. 9.7.A - Fluxo de Dados de Troca de Registradores, 219
- Fig. 9.8 - Programa para Restauração dos Registradores, 220
- Fig. 9.9 - WAIT por SOFTWARE, 221
- Fig. 9.10 - Zera o Conteúdo de Memória, 221
- Fig. 9.11 - Programa que Verifica se o Conteúdo do Acumulador é Par ou Ímpar, 222
- Fig. 9.12 - LOOP, 223
- Fig. 9.13 - Aplicação da Rotina "DELAY", 224
- Fig. 9.14 - Fluxo da Rotina "DELAY", 224
- Fig. 9.15 - Programa de DELAY, 225
- Fig. 9.16 - Programa DELAY, 225
- Fig. 9.17 - Fluxo do Cálculo da Paridade do Código ASCII, 227

- Fig. 9.17.A - Programa do Cálculo da Paridade do Código ASCII, 228
- Fig. 9.18 - Fluxo da Rotina que Compara "HL" com "DE", 229
- Fig. 9.18.A - Programa que Compara "HL" com "DE", 229
- Fig. 9.19 - Programa para Comparação de Blocos, 230
- Fig. 9.20 - Estrutura de uma Rotina, 231
- Fig. 9.21 - Estrutura de Subrotinas, 233
- Fig. 9.21.A - Estrutura da Operação de Soma e Subtração, 234
- Fig. 9.22 - Fluxo da Operação Soma ou Subtração, 234
- Fig. 9.22.A - Programa da Operação de Soma e Subtração, 236
- Fig. 9.23 - Fluxo da Operação de Multiplicação, 237
- Fig. 9.23.A - Programa da Operação de Multiplicação, 238
- Fig. 9.24 - Fluxo da Conversão de Hexadecimal para BCD, 239
- Fig. 9.24.A - Programa da Conversão Hexadecimal para BCD, 240
- Fig. 9.25 - Fluxo da Conversão BCD para Hexadecimal, 241
- Fig. 9.25.A - Programa para Conversão "BCD" para Hexadecimal, 242
- Fig. 9.26 - Fluxo para Conversão de Hexadecimal para ASCII, 244
- Fig. 9.26.A - Programa Hexadecimal para ASCII, 245
- Fig. 9.27 - Fluxo da Rotina, 246
- Fig. 9.27.A - Códigos do ASCII, 247
- Fig. 9.27.B - Programa de Conversão de ASCII para Hexadecimal, 247
- Fig. 9.28 - Código de Sete Segmentos, 248
- Fig. 9.28.A - DISPLAY de Sete Segmentos, 248
- Fig. 9.29 - Tabela para DISPLAY de Sete Segmentos, 249
- Fig. 9.29.A - Fluxo para Transformação de Hexadecimal em Sete Segmentos, 249
- Fig. 9.29.B - Programa Hexadecimal para BCD, 250
- Fig. 9.30 - Programa do Contador, 253
- Fig. 9.31 - Programação da PIO, 254
- Fig. 9.32 - Estrutura para contagem de Unidade em uma Esteira, 255
- Fig. 9.33 - Esquema de Ligação entre CPU, PIO, SENSOR, 255
- Fig. 9.34 - Fluxo de Contagem de Unidade, 256
- Fig. 9.35 - Programa de Contagem de Unidade, 257
- Fig. 9.36 - Modos de Interrupção da PIO, 258
- Fig. 9.37 - Esquema de um Contador, 259
- Fig. 9.38 - Fluxo do Contador, 260
- Fig. 9.38.A - Programa Contador Decrescente, 262

BIBLIOGRAFIA

Livros

- ALLEN e BELTON, *Microcomputer System Software Linguagem*, 1^a ed., New York, Electrical Eng. Electronics Engineers, Inc, 1980.
- ANDREWS, Michael, *Programming Microprocessor Interfaces For Control and Instrumentation*, 1^a ed., New Jersey, Prentice-Hall, Inc, 1982.
- BARDEN, Willian Jr, *Z80 Microcomputer Design Projects*, 1^a ed., Indianapolis, Indiana, Howard W.Sams & Co. Inc, 1980.
- CAPECE, Raymond P., *Microprocessors and Microcomputers*, 1^a ed., New York, McGraw-Hill Publications Co., 1982.
- LEVENTHAL, Lance A., *Z80 Assembly Language Programming*, 1^a ed., Berkeley, California, Osborne/McGraw-Hill, 1979.
- LEVENTHAL, Lance A., *Z80 Assembly Language Subroutines*, 1^a ed., Berkeley, California, Osborne/McGraw-Hill, 1983.
- MILLER, Alan R., *8080/Z80 Assembly Language*, 1^a ed., New York, John Wiley & Sons Inc. Publishers, 1980.
- NICHOLS, Elisabeth A., *Programación Del Microprocesador Z-80*, 1^a ed., Barcelona, Boixareu Editores, 1981.
- OSBORNE, Adan, *Z80 Programming For Logic Design*, = 1^a ed., Berkeley, California, Osborne/McGraw-Hill, 1978.
- VISCONTI, Antonio C.J.F., *Microprocessador 8080 e 8085*, 3^a ed., São Paulo, Livros Érica Editora Ltda, 1983.

Manual

CARR, Joseph J., *Z-80 Users Manual*, 1st ed., Reston, Virginia,
Reston Publishing Company, Inc., 1980.

Z80 Microcomputer System Programming Manual Mostek (MD 78515).

14100

ALLEN, E. R., *Microcomputer System Design*, 1st ed.,
New York, Electrical Eng. Electronics Engineers, Inc., 1980.

ANDREWS, Michael, *Programming Microprocessors*, 1st ed.,
New York, McGraw-Hill, Inc., 1982.

BARDEN, William D., *Z80 Microcomputer Design*, 1st ed.,
Indianapolis, Howard W. Sams & Co., Inc., 1982.

CAPRON, Raymond E., *Microprocessors and Microcomputers*, 1st ed.,
New York, McGraw-Hill Publications Co., 1982.

LEVINTHAL, Isaac A., *Z80 Assembly Language Programming*, 1st ed.,
Berkeley, California, Osborne/McGraw-Hill, 1982.

LEVINTHAL, Isaac A., *Z80 Assembly Language Programming*, 1st ed.,
Berkeley, California, Osborne/McGraw-Hill, 1982.

MILNER, Alan R., *Z80 Assembly Language Programming*, 1st ed.,
New York, John Wiley & Sons Inc. Publishers, 1980.

NICHOLS, Elizabeth A., *Microprocessors and Microcomputers*, 1st ed.,
New York, McGraw-Hill, 1982.

OSBORN, Alan, *Microprogramming for Logic Design*, 1st ed.,
Berkeley, California, Osborne/McGraw-Hill, 1982.

ROBERTS, Anthony J., *Microprocessors and Microcomputers*, 1st ed.,
New York, McGraw-Hill, 1982.

Impresso em off-set por
EDITORA SANTUÁRIO
Rua Pe. Claro Monteiro, 342
Aparecida - São Paulo
Fone DDD (0125) 36-2140
com filmes fornecidos pelo editor.

PUBLICAÇÕES ÉRICA

LINGUAGENS E APLICATIVOS

- | | |
|----------|------------------------------------|
| Santos | - Aplicativos |
| Arsonval | - Basic para Computadores Pessoais |
| Ideali | - Sistema Operacional CP/M - 80 |
| Natale | - Probasic - Programação em Basic |

MICROPROCESSADORES

- | | |
|--------------------|--|
| Visconti | - Microprocessadores 8080 e 8085 - Hardware - Vol. 1 |
| Visconti | - Microprocessadores 8080 e 8085 - Software - Vol. 2 |
| Cypriano/Cardinali | - Microprocessadores Z-80 - Hardware - Vol. 1 |
| Cypriano | - Microprocessadores Z-80 - Software - Vol. 2 |

ELETRÔNICA

- | | |
|------------------|---|
| Lando/Serg | - Amplificador Operacional |
| Capuano/Idoeta | - Elementos de Eletrônica Digital |
| Almeida | - Eletrônica Industrial |
| Almeida | - Eletrônica de Potência |
| Cipelli/Sandrini | - Teoria e Desenvolvimento de Projetos de Circuitos Eletrônicos |
| Azevedo | - TTL/CMOS - Circuitos Integrados - Vol. 1 |
| Azevedo | - TTL/CMOS - Circuitos Integrados - Vol. 2 |

TELECOMUNICAÇÕES

- | | |
|-------|---|
| Smit | - Rádio Propagação |
| Gomes | - Telecomunicações - Transmissão e Recepção AM FM |
| | - Sistemas Pulsados |

OUTROS

- | | |
|-------------------|-------------------------------|
| Sandrini | - Profissional Ser ou Não Ser |
| Lence | - Física - Vol. I |
| Filkauskas/Guedes | - O Plástico. |