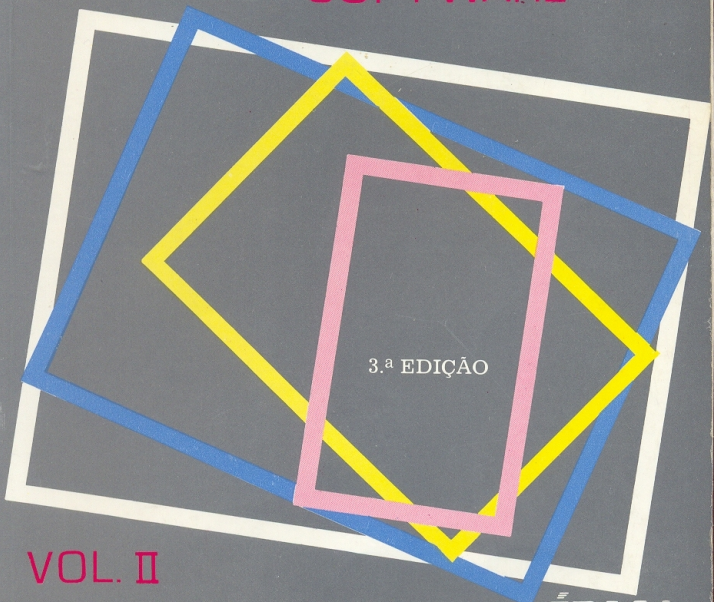
ENG.º LUIZ BENEDITO CYPRIANO

Z80

SOFTWARE

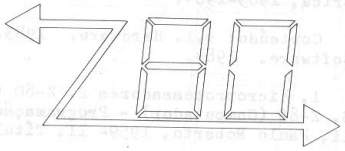


3.ª EDIÇÃO

VOL. II

ÉRICA

MICROPROCESSADOR



SOFTWARE

VOL. 2

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

C523m
v.1-2

Cipriano, Luís Benedito, 1957-
Microprocessador Z-80 / Luiz Benedito Cypriano, 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. Cardinali, Paulo Roberto, 1959- II. Título.

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

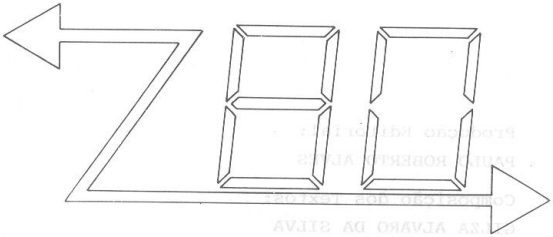
84-0796

Índices para catálogo sistemático:

1. Microprocessadores Z-80 : Engenharia eletrônica 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.)

ENGº LUIZ BENEDITO CYPRIANO

MICROPROCESSADOR



SOFTWARE

VOL. 2

4.ª EDIÇÃO

LIVROS ÉRICA EDITORA LTDA.

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

A meus Pais.

PREFÁCIO

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.

O Autor

PREFÁCIO

SUMÁRIO

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 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.

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 "Ø" 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 STAK 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 - 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.

2.1.2 - 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 (AØ a A15), sendo divididas em duas partes: linhas altas (A15 a A8), simbolizadas por (HI) e linhas baixas (A7 a AØ), 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 registrador de índice.

2.2 - Registrador

O registrador é um circuito lógico que tem a finalidade de reter a curto prazo um conjunto de BITS. No microprocessador 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 subdivididos, 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 versatilidade 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, embora nem todos os registradores apresentem esta versatilidade. Cada registrador apresenta, dependendo de sua aplicação, algumas dessas características.

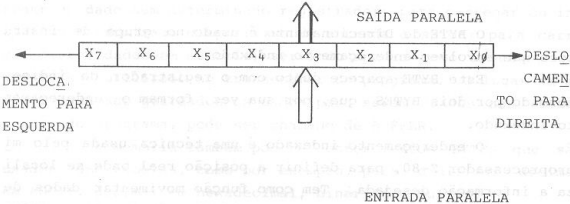


Fig. 2.1 - Registrador Geral.

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

GRUPO UM	GRUPO DOIS
Registrador B	Registrador B'
Registrador C	Registrador C'
Registrador D	Registrador D'
Registrador E	Registrador E'
Registrador H	Registrador H'
Registrador L	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.

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 $(\emptyset 1111\emptyset 1)_B$ 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.

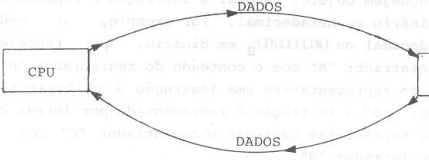


Fig. 3.1 - Transferência de Dados entre CPU e Memória.

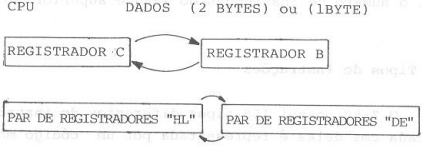


Fig. 3.2 - Transferência de Dados entre os Registradores Internos.

3.1.2 - Operações Aritméticas e Lógicas

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

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.

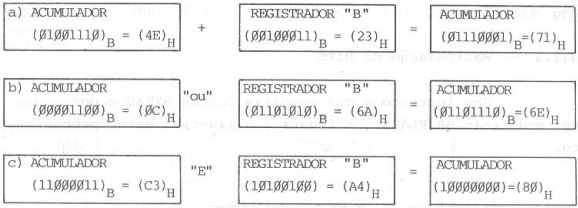


Fig. 3.3 - Exemplo de Operação Aritmética (a) e Lógica(b e c).

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.

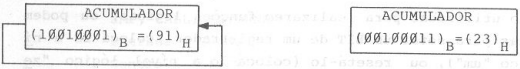
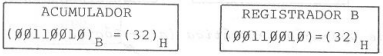


Fig. 3.4 - Rotação de um BIT para a Esquerda.

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.



FLAG "zero"= "1" porque os registradores são iguais.

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 seqüência 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.

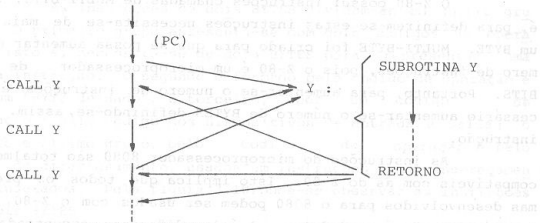


Fig. 3.7 - CALL e Retorno.

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.

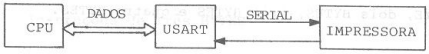


Fig. 3.8 Dispositivo de Entrada e Saída.

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, (IMØ, 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.

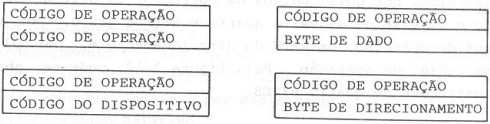


Fig. 3.10 - Instrução de dois BYTES.

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
C	001
D	010
E	011
H	100
L	101
A	111

LD B,C → (01000001B) → (41H)
(41H) Código de Operação da Instrução LD B,C

LD H,E → (01100011B) → (63H)
(63H) Código de Operação da Instrução LD H,E

"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"(ØA7H), quando o microprocessador executar a instrução LD B,E; o registrador "B", após a execução, passará a ter (ØA7H) 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.

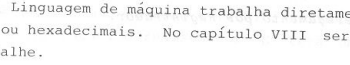


Fig. 4.4 - Endereçamento Imediato Estendido.

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.

Fig. 4.5 - Endereçamento Indireto por Registrador.

Na montagem dos mnemônicos do microprocessador Z-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)

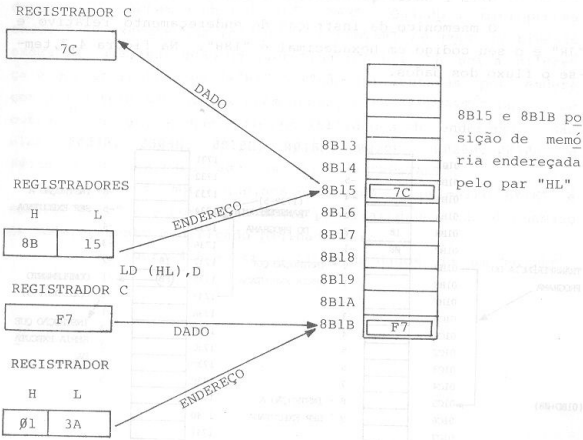


Fig. 4.6 - Fluxo de Dados.

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 à saltos em determinados 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.

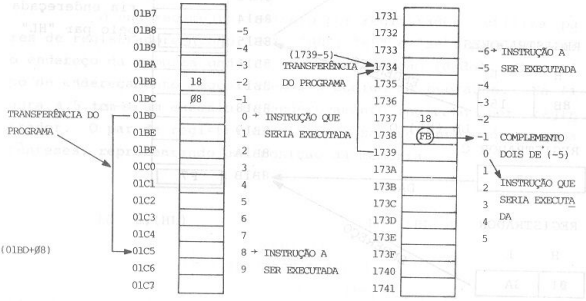


Fig. 4.7 - Fluxo de Dados de Endereçamento Relativo.

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 será executada, mas com a diferença de onde está o programa e para ra onde irá a instrução a ser executada. Esta diferença será a mesma em qualquer lugar onde se localize o programa. Outra vantagem deste endereçamento é na economia de um BYTE, pois instruções de saltos ocupam três BYTES, sendo que um BYTE corresponde ao código da instrução e os dois subseqüentes ao endereço de onde 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 endereçamento também é chamado de"RESTART". Quando o microprocessador Z-80 executa uma instrução de"RESTART",é como se tivesse executado uma instrução de subrotina (CALL). A única diferença é que as instruções de"RESTART"são apresentadas por endereços pré-determinados. Existem apenas oito "RESTARTS" ou, em outras palavras, apenas oito possibilidades de endereços. São eles: ØØØØH, ØØØ8H, ØØ1ØH, ØØ18H, ØØ2ØH, ØØ28H, ØØ3ØH e ØØ38H, 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".

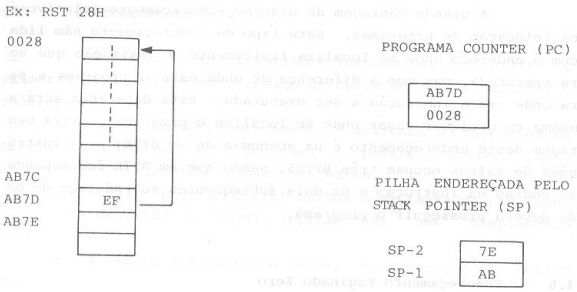


Fig. 4.8 - Fluxo de Dado de um "RESTART".

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.

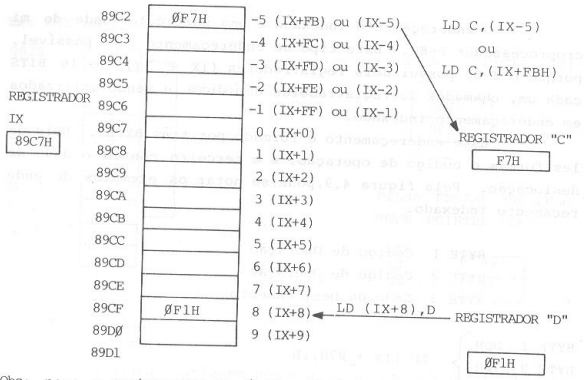
Este endereçamento é formado por três BYTES. Dois deles formam o código de operação, e o terceiro contém o dado de deslocação. Pela figura 4.9,pode-se notar os exemplos de endereçamento indexado.

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

BYTE 1 DDH
BYTE 2 7ØH } LD (IX + Ø7H),B
BYTE 3 Ø7H

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 "IY" 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 (Ø1); 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.

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.

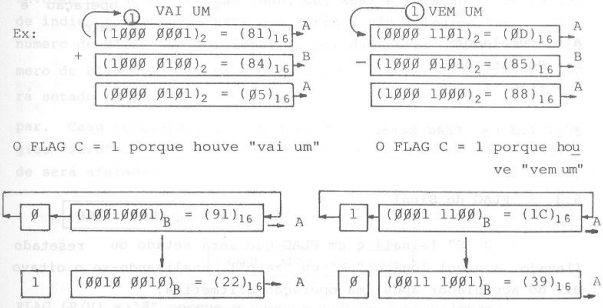
7	6	5	4	3	2	1	0	BITS
S	Z	*	H	*	P/V	N	C	DESCRIÇÃO

* 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.



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"				
	$(1ØØØ\ ØØØØ)_B = (8Ø)_H$	A		$(1ØØ1\ 1ØØ1)_B = (99)_H$	A
+	$(1ØØØ\ ØØØØ)_B = (8Ø)_H$	B	-	$(1ØØ1\ 1ØØ1)_B = (99)_H$	B
	$(ØØØØ\ ØØØØ)_B = (ØØ)_H$	A		$(ØØØØ\ ØØØØ)_B = (ØØ)_H$	A

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.

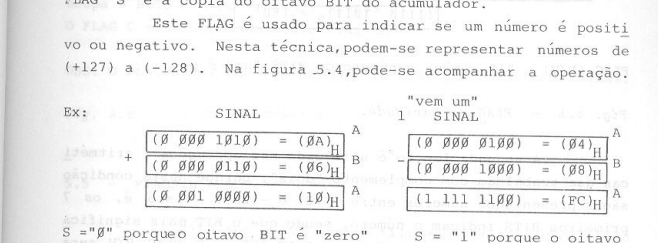


Fig. 5.4 - FLAG Sinal.

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.

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

$(011\emptyset\ 1\emptyset01)_B = (69)_H$ OR $(11\emptyset\emptyset\ \emptyset011)_B (C3)_H$

$(111\emptyset\ 1011)_B = (EB)_H$

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:

```
      1 "vai um"
      1   111   1001  número (- 7) em complemento dois
+     1   111   0001  número (-15) em complemento dois
    ---------------
      1   110   1010  número (-22) em complemento dois

    SINAL
```

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

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

Ex:

```
      1 "vai um"
      1   001  1100  número (- 100) em complemento dois
      1   100  1110  número (-50) em complemento dois
    ---------------
      0   110  1010  incorreto

    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 Algorítmos 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	1ØØØ	$F8_H$	A		Ø11Ø	Ø1ØØ	64_H	A
+	ØØØØ	1ØØ1	$Ø9_H$	B	-	Ø1ØØ	1ØØØ	48_H	B
	ØØØØ	ØØØ1	$Ø1_H$	A		ØØØ1	11ØØ	$1C_H$	A

H = "1" porque houve "vai um" no meio do BYTE.
N = "Ø" porque é adição

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

	Ø1ØØ	ØØ11	43_H	A		Ø1Ø1	Ø111	57_H	A
+	ØØ11	Ø1Ø1	35_H	B	-	Ø1ØØ	Ø11Ø	46_H	B
	Ø111	1ØØØ	78_H	A		ØØØ1	ØØØ1	11_H	A

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

N = "Ø" porque é adição

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

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.

CAPÍTULO 6 - INTERRUPÇÃO

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 lógi

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.

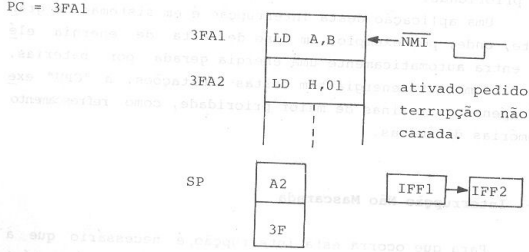
O "IFF1" será resetado, isto ocorre para que não haja um pedido de interrupção mascarada, enquanto estiver sendo executado.

Neste tipo de interrupção, o programa COUNTER (PC) será sempre carregado com o endereço (ØØ66H). 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 devem 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.



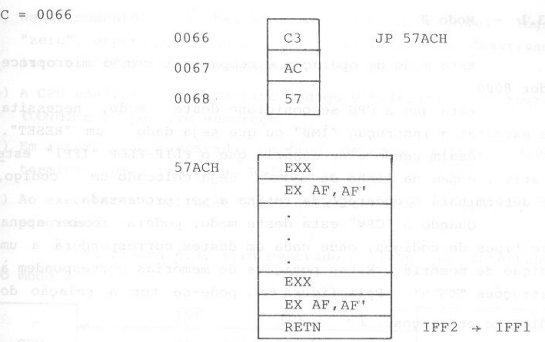


Fig. 6.1 - Fluxo da Interrupção não Mascarada.

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 "IMØ, 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 Ø, modo 1 e modo 2.

O modo Ø 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 Ø

Este modo de operação é compatível com o microprocessador 8080.

Para que a CPU se posicione deste modo, necessita-se executar a instrução "IMØ" 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 Ø

a) A CPU deverá estar no modo Ø e o FLIP-FLOP "IFF1" deverá estar em condições de receber uma interrupção.

b) A linha $\overline{INT}$ 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.

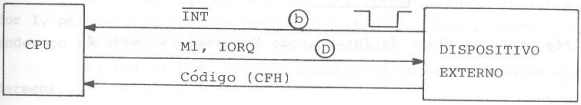
d) Neste momento, as linhas $\overline{M1}$ e $\overline{IORQ}$ irão para nível lógico "zero", esperando o código e a interrupção será desativada, isto é, "IFF1" será resetado.

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 Ø.



(b)	57ADH	LD H,2H	$\overline{INT}$	ativado pedido de interrupção mascarada
(g)	57AEH	AND H		

(c)	SP	AE
		57

(e) PC = 0008

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

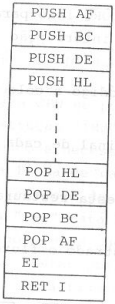


Fig. 6.3 - Fluxo da Interrupção Mascarada no modo 0.

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 $\overline{INT}$. 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 nomodo 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 um 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.

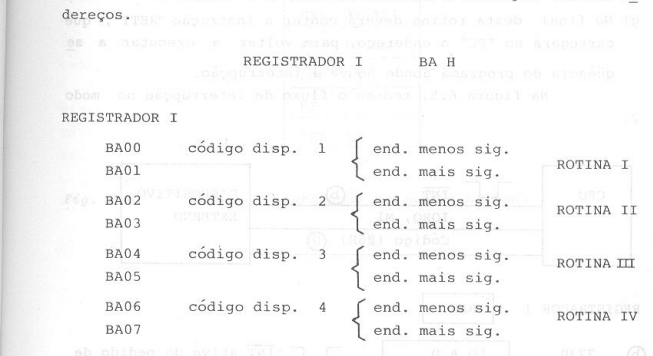


Fig. 6.4 - Montagem no Endereçamento no modo 2.

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.

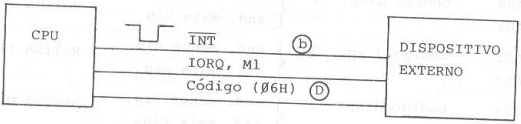
d) Neste momento o dispositivo enviará pela via de dados o seu código.

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 seqüência do programa aonde houve a interrupção.

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



REGISTRADOR I | BAH

(b) 7730
(h) 7731

LD A,D
LD A,01H

INT ativa do pedido de interrupção mascarada.

(c) SP

31
77

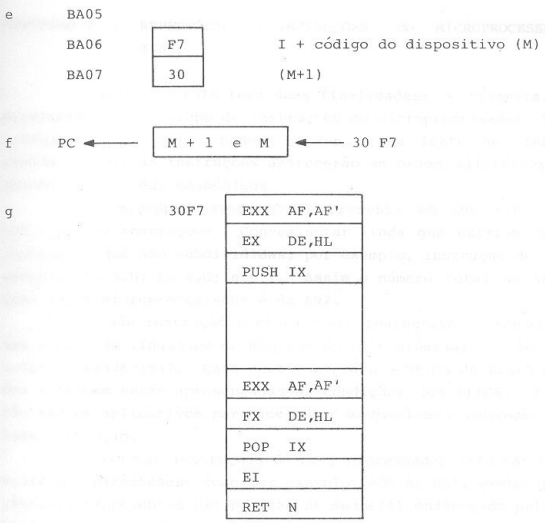


Fig. 6.5 - Fluxo da Interrupção Mascarada no modo 2.

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, baseando-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 carregamento (LD A,D; LD B,C; etc.). Assim o número total de instruções deste microprocessador é de 692.

Cada instrução conterá o seu respectivo mnemônico, seu código em linguagem de máquina na base binária, decimal, octal e hexadecimal. Cada caso conterá o sentido de fluxo de dados e também serão apresentadas as condições dos FLAGS. Existirão 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	Ø
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	2Ø6 n
Binário	11ØØ111Ø n

Parâmetros

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

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	ØØØ1	1Ø1Ø B
Dado	= 25 H	ØØ1Ø	Ø1Ø1 B
CARRY BIT	= Ø1 H	+ ØØØØ	ØØØ1 B
Acumulador	= 4Ø H	Ø1ØØ	ØØØØ

C = Ø Z = Ø P/V = Ø S = Ø N = Ø 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 acu

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

Código de Operação

Hexadecimal	8F
Octal	217
Decimal	143
Binário	1ØØØ1111

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	ØØ01	ØØ11
Acumulador	13	H	ØØ01	ØØ11
CARRY BIT	+ ØØ	H	+ ØØØØ	ØØØØ
Acumulador	26	H	ØØ1Ø	Ø11Ø

C = Ø Z = Ø P/V = Ø S = Ø N = Ø H = Ø

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 1ØØØ1 ← 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	ØlØØ	1Ø1Ø
Registrador "B"	5Ø	H	Ø1Ø1	ØØØØ
CARRY BIT	+ Ø1	H	+ ØØØØ	ØØØ1
Acumulador	9B	H	1ØØ1	1Ø11

C = Ø Z = Ø P/V = Ø S = 1 N = Ø H = Ø

Exemplo: ADC A,L

Acumulador	5C	H	Ø1Ø1	11ØØ
Registrador "L"	1D	H	ØØØ1	11Ø1
CARRY BIT	+ ØØ	H	ØØØØ	ØØØØ
Acumulador	79	H	Ø111	1ØØ1

C = Ø Z = Ø P/V = Ø S = Ø N = Ø 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	1ØØØ111Ø

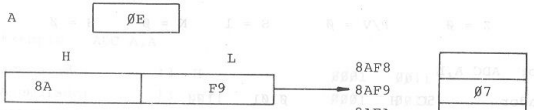
Parâmetros

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	ØE	H	ØØØØ	111Ø
Memória(8AF9H)	Ø7	H	ØØØØ	Ø111
CARRY BIT	+ Ø1	H	+ ØØØØ	ØØØ1
Acumulador	16	H	ØØØ1	Ø11Ø

C = Ø Z = Ø P/V = Ø S = Ø N = Ø 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.

Código de Operação

Hexadecimal	DD	8E	d
Octal	335	216	d
Decimal	221	142	d
Binário	11Ø111Ø1	1ØØØ111Ø	d

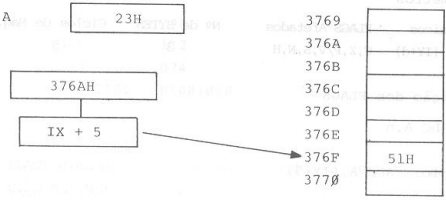
Parâmetros

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	ØØ1Ø	ØØ11
Memória(IX+5)	51 H	Ø1Ø1	ØØØ1
CARRY BIT	+ Ø1 H	ØØØØ	ØØØ1
Acumulador	75 H	Ø111	Ø1Ø1

C = Ø Z = Ø P/V = Ø S = Ø N = Ø H = Ø

6) ADC A,(IY+ d)

Operação Simbólica

AC ← 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

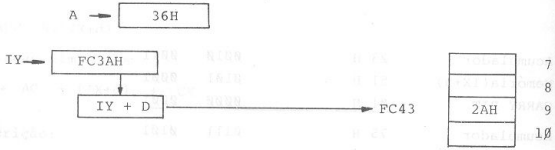
Parâmetros

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	ØØ11	Ø11Ø
Memória(IY+d)	2A H	ØØ1Ø	1Ø1Ø
CARRY BIT	+ ØØ H	+ ØØØØ	ØØØØ
Acumulador	6Ø H	Ø11Ø	ØØØØ

C = Ø Z = Ø P/V = Ø S = Ø N = Ø 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	111Ø11Ø1	Ø1ØØ1Ø1Ø

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		Ø11Ø	11Ø1	ØØØ1	11ØØ
Registrador	BC 127A H		ØØØ1	ØØ1Ø	Ø111	1Ø1Ø
CARRY BIT	ØØØ1 H	+	ØØØØ	ØØØØ	ØØØØ	ØØØ1
	7F97 H		Ø111	1111	1ØØ1	Ø111

C = Ø Z = Ø P/V = Ø S = Ø N = Ø H = Ø

8) ADC HL,DE

Operação Simbólica

HL ← HL + DE + 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 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	Ø9Ø
Binário	111Ø11Ø1	Ø1Ø11Ø1Ø

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:

				1	1		
Registrador	HL	6BA3	H	Ø11Ø	1Ø11	1Ø1Ø	ØØ11
Registrador	BC	Ø47A	H	ØØØØ	Ø1ØØ	Ø111	1Ø1Ø
CARRY BIT		+ ØØØØ	H	+ ØØØØ	ØØØØ	ØØØØ	ØØØØ
		7Ø1Ø	H	Ø111	ØØØØ	ØØØ1	11Ø1

C = Ø Z = Ø P/V = Ø S = Ø N = Ø H = 1

9) ADC HL,HL

Operação Simbólica

HL ← HL + HL + 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 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	111Ø11Ø1	Ø11Ø1Ø1Ø

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	Ø17A	H	ØØØØ	ØØØ1	Ø111	1Ø1Ø
Registrador	HL	Ø17A	H	ØØØØ	ØØØ1	Ø111	1Ø1Ø
CARRY BIT		+ ØØØØ	H	ØØØØ	ØØØØ	ØØØØ	ØØØØ
		Ø2F4	H	ØØØØ	ØØ1Ø	1111	Ø1ØØ

10) ADC HL,SP

Operação Simbólica

HL ← 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	111Ø11Ø1	Ø1111Ø1Ø

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 ← 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	1ØØØØ11Ø

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	ØØ11	Ø11Ø
Memória (76DAH)	+ A2 H	1Ø1Ø	ØØ1Ø
	Ø8 H	11Ø1	1ØØØ

C = Ø Z = Ø P/V = Ø S = 1 N = Ø H = Ø

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	11Ø111Ø1	1ØØØØ11Ø	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+ØAH)
(IX) = AB1Ø

Acumulador	1A H	ØØØ1	1Ø1Ø
Memória(IX+ØAH)	63 H	Ø11Ø	ØØ11
	7D H	Ø111	11Ø1

C = Ø Z = Ø P/V = Ø S = 1 N = Ø H = Ø

13) ADD A,(IY + d)

Operação Simbólica

A ← 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	2Ø6 d
Decimal	253	134 d
Binário	11111101	1ØØØØ11Ø 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 ← 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	3Ø6 n
Decimal	198 n
Binário	11ØØØ11Ø 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 1ØØØØ ← 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	ØlØØ	lØlØ
Registro "D"	+ 5Ø H	+ ØlØl	ØØØØ
Acumulador	9A H	lØØl	lØlØ

C = Ø Z = Ø P/V = Ø S = 1 N = Ø H = Ø

ADD A,H

Acumulador	5C H	Ø1Ø1	11ØØ
Registro "H"	+ 1D H	+ ØØØ1	11Ø1
Acumulador	79 H	Ø111	lØØ1

C = Ø Z = Ø P/V = Ø S = Ø N = Ø 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	Ø9
Octal	11
Decimal	9
Binário	ØØØØ1ØØ1

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	789ØH	Ø111	1ØØØ	1ØØ1	ØØØØ
Registrador	BC	Ø1FAH	ØØØØ	ØØØ1	1111	1Ø1Ø
Registrador	HL	7A8AH	Ø111	1Ø1Ø	1ØØØ	1Ø1Ø

C = Ø N = Ø H = Ø

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	ØØØ11ØØ1

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"	FFØØH	1111	1111	ØØØØ	ØØØØ
Registrador	"DE"	+ A17BH	+ 1Ø1Ø	ØØØ1	Ø111	1Ø11
Registrador	"HL"	AØ7BH	1Ø1Ø	ØØØØ	Ø111	1Ø11

C = 1 N = Ø 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. Está 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	ØØlØlØØl

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	ØØllØØl

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	ØØØØH	ØØØØ	ØØØØ	ØØØØ	ØØØØ
Registrador	SP	+ 7BAFH	+ Ø111	1Ø11	1Ø1Ø	1111
		7BAFH	Ø111	1Ø11	1Ø1Ø	1111

C = Ø N = Ø H = Ø

20) ADD IX, WW

SS representa os registradores BC, DE, IX, SP

Registrador SS	Código
BC	ØØ
DE	Ø1
IX	1Ø
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 11Ø111Ø1 ØØ ← WW → 1ØØ1

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	Ø111	1111	ØØØ1	1Ø1Ø
Registrador	DE	+ 1ØFØ	H	+ ØØØ1	ØØØØ	1111	ØØØØ
		9ØØA	H	1ØØ1	ØØØØ	ØØØØ	1Ø1Ø

C = Ø N = Ø H = 1

21) ADD IY,WW

SS representa os registradores BC, DE, IX, SP

Registradores WW	Códigos
BC	ØØ
DE	Ø1
IY	1Ø
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 llllll Ø1 ØØ ← WW ← 1ØØ1

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		1Ø11	ØØØ1	Ø111	1Ø1Ø
Registrador IY	+ B17A	H	+	1Ø11	ØØØ1	Ø111	1Ø1Ø
Registrador IY	62F4	H		Ø11Ø	ØØ1Ø	1111	Ø1ØØ

C = 1 N = Ø H = Ø

22) AND r

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

Registrador r	Código
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Operação Simbólica

A ← A ∧ 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 1Ø1ØØ ← _ 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	1ØØØ	1111
Registrador B	BØ	H	1Ø11	ØØØØ
Acumulador	8Ø	H	1ØØØ	ØØØØ

S = 1 Z = Ø H = 1 P/V = Ø N = Ø C = Ø

Acumulador	C8	H	11ØØ	1ØØØ
Registrador H	17	H	ØØØ1	Ø111
Acumulador	ØØ	H	ØØØØ	ØØØØ

S = Ø Z = 1 H = 1 P/V = 1 N = Ø C = Ø

23) AND 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 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	23Ø	n
Binário	111ØØ11Ø	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 ← A ∧ (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	1Ø1ØØ11Ø

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	Ø111 ØØ11
(3704H)=HL	3EH	ØØ11 111Ø
Acumulador	32H	ØØ11 ØØ1Ø

S = Ø Z = Ø H = 1 P/V = Ø N = Ø C = Ø

25) AND (IX+d)

Operação Simbólica

A ← A ∧ (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	11ØØ11Ø1	1ØØ1ØØ11Ø	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+1ØH)

Posição de memória IX = (7AC7H)
Posição de memória IX + 1ØH = (7AD7H)

Acumulador	7A H	Ø111 1Ø1Ø
Posição de memória(IX+1Ø)	F1 H	1111 ØØØ1
Acumulador	7Ø H	Ø111 ØØØØ

S = Ø Z = Ø H = 1 P/V = Ø N = Ø C = Ø

26) AND (IY+d)

Operação Simbólica

A ← A + (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" ← $\overline{r_b}$

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

11ØØ1Ø11 01 ← b→ ← r →

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 ØllllllØ

Z = Ø

BIT 3,Ø

Registrador D = B2H 1Ø11ØØ1Ø

Z = 1

28) BIT b, (HL)

Operação Simbólica

"FLAG Z" ← $\overline{M_b}$

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

11ØØ1Ø11

Ø1 ← _ b _ → 11Ø

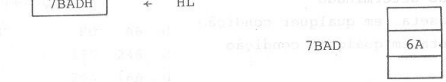
Parâmetros

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 Ø,(HL)



M (HL) = 6AH Ø11Ø1Ø1Ø

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 11Ø111Ø1 11ØØ1Ø11 d ØlbllØ

Parâmetros

Mnemônico	FLAG Afetados	Nº de BYTES	Ciclos de Máq.	Estados
BIT b,(HL)	Z,P/V,S,N,H	4	5	2Ø

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

11111Ø1 11ØØ1Ø11 d ØlbllØ

Parâmetros

Mnemônicos	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
BIT b,(IY+d)	Z,P/V,S,N,H	4	5	2Ø

Condição dos FLAGS

Ver BIT b,r

31) CALL cc,nn

cc	Condição	Código	FLAGS
NZ	Se não zero	ØØØ	Z
Z	Se zero	ØØ1	Z
NC	Se não CARRY	Ø1Ø	C

C	Se CARRY	ØÌÌ	C
PO	Se paridade ímpar	ÌØØ	P/V
PE	Se paridade par	ÌØÌ	P/V
P	Se sinal positivo	ÌÌØ	S
M	Se sinal negativo	ÌÌÌ	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-à no PC o endereço da subrotina, se as condições dos FLAGS forem favoráveis, caso contrário, a instrução será ignorada.

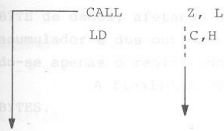
Parâmetros

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 ÌÌ ← cc → ÌØØ 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	2Ø5	nn
Binário	11ØØ11Ø1	nn

Parâmetros

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

33) CCF

Operação Simbólica

CY ← $\overline{CY}$

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	ØØ111111

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	Ø11Ø	11ØØ	
Registro B	- 61 H	- Ø11Ø	ØØØ1	
	ØB H	ØØØØ	1Ø11	Resultado ignorado

C = Ø Z = Ø P/V = Ø S = Ø N = Ø H = Ø

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	19Ø
Binário	1Ø111110

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	19Ø	d
Binário	11Ø111Ø1	1Ø111110	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 ≠ Ø
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 ØØØ4

ADFB	7A
ADFC	37
ADFD	47
ADFE	3F

CPD
P/V = Ø Z = Ø
HL ADFD
BC ØØØ3

CPD
P/V = Ø Z = Ø
HL ADFC
BC ØØØ2

CPD
P/V = Ø Z = 1
HL ADFC
BC ØØØ1

CPD
P/V = 1 Z = Ø
HL ADFB
BC ØØØØ

40) CPDR

Operação Simbólica

A - (HL)
HL ← HL + 1
BC ← BC - 1
PC ← PC - 2 se B ≠ Ø 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.

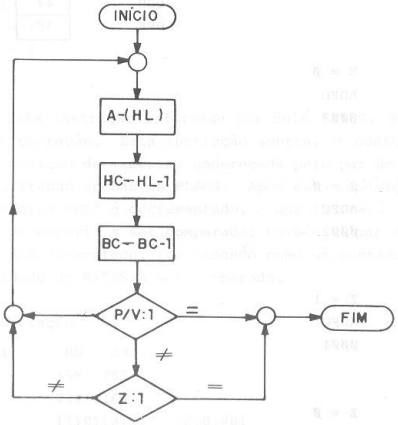


Fig. 7.1 - Estrutura da instrução CPDR.

Código de Operação

Hexadecimal	ED	B9
Octal	355	271
Decimal	237	185
Binário	111Ø11Ø1	1Ø111ØØ1

Parâmetros

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

BC = Ø e A ≠ (HL)
BC = Ø 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	111Ø11Ø1	1Ø1ØØØØ1

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 ≠ Ø 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	111Ø11Ø1	1Ø11ØØØ1

Parâmetros

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

BC ≠ Ø e A ≠ (HL)
BC = Ø ou A = (HL)

Condição dos FLAGS

Ver CPR

43) CPL

Operação Simbólica

$A \leftarrow \overline{A}$

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	ØØ1Ø1111

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	1111ØØ11
CPL		
Acumulador	ØCH	ØØØØ11ØØ

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
	Ø	Ø	Ø	Ø-9	Ø-9	ØØ	Ø
	Ø	Ø	Ø	Ø-8	A-F	Ø6	Ø
ADD	Ø	1	Ø	Ø-9	Ø-3	Ø6	Ø
ADC	Ø	Ø	Ø	A-F	Ø-9	6Ø	1
INC	Ø	Ø	Ø	9-F	A-F	66	1
	Ø	1	Ø	A-F	Ø-3	66	1
	1	Ø	Ø	Ø-2	Ø-9	ØØ	1
	1	Ø	Ø	Ø-2	A-F	FA	1
	1	1	Ø	Ø-3	Ø-3	A0	1
	Ø	Ø	1	Ø-9	Ø-9	ØØ	Ø
	Ø	1	1	6-F	Ø-8	FA	Ø
	1	Ø	1	Ø-9	7-F	AØ	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	ØØ1ØØ111

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 ← 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 ØØ ←_ r _→ 1Ø1

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 (8Ø) 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) ← (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 dados "d".

Código de Operação

Hexadecimal	DD	35	d
Octal	335	65	d
Decimal	221	53	d
Binário	11ØØ111Ø1	ØØ11Ø1Ø1	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) ← (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 de memória, endereçada pelo registrador de índice "IY", em conjunto com o BYTE de dados.

Código de Operação

Hexadecimal	FD	35	d
Octal	375	65	d
Decimal	253	53	d
Binário	11111Ø1	ØØ11Ø1Ø1	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	ØØ11Ø1Ø1

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 ØØ ← SS_ → 1Ø11

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	11Ø111Ø1	ØØ1Ø1Ø11

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	ØØ1Ø1Ø11

Parâmetros

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

52) DJNZ e

Operação Simbólica

B ← (B - 1)

B = Ø PC ← PC

B ≠ Ø PC ← (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.

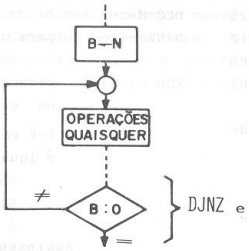


Fig. 7.3 - Estrutura da instrução DJNZ e.

Código de Operação

Hexadecimal	1Ø e-2	
Octal	2Ø e-2	
Decimal	16 e-2	
Binário	ØØØ1ØØØØ e-2	

Parâmetros

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

Se B = Ø
Se B ≠ Ø

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	Linguagem de máquina			
			Ø6	LD B,87 H	
			87		
e	-2	-4	1Ø	INC E	
	-1	-3	25	DEC H	
	0	-2	1Ø	DJNZ -4	para o Assembler
	1	-1	FB		
	2	Ø	85	XOR L	para linguagem de máquina e-2 = -6
		-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	11111Ø11

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 microprocessador 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	Ø8
Octal	1Ø
Decimal	8
Binário	ØØØØ1ØØØ

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	111ØlØ11

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).

Código de Operação

Hexadecimal	E3
Octal	343
Decimal	227
Binário	111ØØØ11

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 ← (SP)
IXH ← (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	11Ø111Ø1	111ØØØ11

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 ← (SP)
IYH ← (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	Ø111Ø11Ø

Parâmetros

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

61) IM Ø

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	ØØØØH
CFH	ØØØ8H
D7H	ØØ1ØH
DFH	ØØ18H
E7H	ØØ2ØH
EFH	ØØ28H
F7H	ØØ3ØH
FFH	ØØ38H

Fig. 7.4 - Código das posições dos saltos.

Código de Operação

Hexadecimal	ED	46
Octal	335	1Ø6
Decimal	237	Ø7Ø
Binário	111Ø11Ø1	ØlØØØ11Ø

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IM Ø	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 $\overline{INT}$ 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	111Ø11Ø1	ØlØlØ11Ø

Parâmetros

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

63) IM 2

Descrição:

Esta instruçao é formada por dois BYTES, sendo estes o código de operaçãc. 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 Ø

Código de Operação

Hexadecimal	ED	5E
Octal	355	136
Decimal	237	94
Binário	111Ø11Ø1	Ø1Ø1111Ø

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	11Ø11Ø11	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 ← (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 lllØ11Ø1 01r r ØØØ

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	ØØ11Ø1ØØ

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 ØØ ← r → 1ØØ

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) ← (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	Ø52	d
Binário	11Ø111Ø1	ØØ11Ø1ØØ	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) ← (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	ØØ110100	d

Parâmetros

Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
INC(IY 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 ØØ ← SS → ØØ11

Parâmetros

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

71) INC IX

Operação Simbólica

IX ← (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	11Ø111Ø1	ØØ1ØØØ11

Parâmetros

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

72) INC IY

Operação Simbólica

IY ← (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	11111Ø1	ØØ1ØØØ11

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	17Ø
Binário	111Ø11Ø1	1Ø1Ø1Ø1Ø

Parâmetros

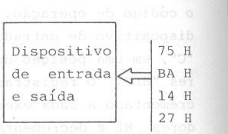
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
IND	Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

C Não modificado
Z Seta se B = Ø
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 Ø4H , 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 = Ø
 :
 .
IND
 :  FLAG Z = Ø          72D7    27
 .                      72D8    14
 .
IND                     72D9    BA
 .  FLAG Z = Ø          72DA    75
 :
 .
IND
   FLAG Z = 1
```



74) INDR

Operação Simbólica

(HL) ← (C)
HL ← (HL - 1)
B ← (B - 1)
PC ← PC - 2 se B ≠ Ø

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	111Ø11Ø1	1Ø111Ø1Ø

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 = Ø
se B ≠ Ø

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) ← (C)
HL ← (HL + 1)
B ← (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	111Ø11Ø1	1Ø1ØØØ1Ø

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 IND

76) INIR

Operação Simbólica

(HL) ← (C)
HL ← (HL + 1)
B ← (B - 1)
PC ← PC - 2 se B = Ø

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	111Ø11Ø1	1Ø11ØØ1Ø	

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 = Ø
se B ≠ Ø

Condição dos FLAGS

Ver INDR

77) JP CC,nn

CC Representam os FLAGS

CC	Condição	Código	FLAGS
NZ	Se não zero	ØØØ	Z
Z	Se zero	ØØ1	Z
NC	Se não CARRY	Ø1Ø	C
C	Se CARRY	Ø11	C
PO	Se paridade ímpar	1ØØ	P/V
PE	Se paridade par	1Ø1	P/V
P	Se sinal positivo	11Ø	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 → Ø1Ø 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	111Ø1ØØ1

Parâmetros

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

Exemplo:

```
        LXI     HL, CANAL
        JP      (HL)
        .
        .
        .
CANAL:  LD      C,A
        ADD     A,(HL)
```

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	11Ø111Ø1	111Ø1ØØ1

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
Decimal	253	233
Binário	11111Ø1	111Ø1ØØ1

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 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 incondicional, pois o salto independe de qualquer condição.

Código de Operação

Hexadecimal	C3	nn
Octal	3Ø3	nn
Decimal	195	nn
Binário	11ØØØØ11	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 primeiro o código de operação e o segundo um BYTE de dados. Esta instrução efetua um salto relativo, isto é, somando-se ou subtraindo-se do Programa COUNTER (PC), o salto ocorre incondicionalmente, não depende das condições dos FLAGS.

Condição de Operação

Hexadecimal	18	e-2
Octal	3Ø	e-2
Decimal	24	e-2
Binário	ØØØ11ØØØ	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	Ø,H		
Assem.	18 Ø7	JP	Ø9	0	
L.M.	41	LD	B,C		0
	7Ø	LD	(HL),B		
	CD AB 77	CALL	NC,77ABH		
	CB DC	SET	3,H		
	CB 27	SLA	A	9	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	ØØ	Z
Z	Se zero	Ø1	Z
NC	Se não CARRY	1Ø	C
C	Se CARRY	11	C

Operação Simbólica

PC ← (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 ØØ1 ← cc → ØØØ 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	Referência Linguagem de Máquina
	DD7717		LD (IX+17H),A		
	4F		LD (C,A) ←	-4	-6
	6D		LD (L,L)		
	E5		PUSH HL		
	AØ		AND B		
Assemb.	28	FA	JR Z,-4 →	Ø	Ø
L.M.	6Ø		LD H,B		

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

Linguagem máquina	(-6)	ØØØØØ11Ø
Complemento dois	(FAH)	11111Ø1Ø

84) LD A,(TT)

"TT" representa os registradores BC e DE

Registradores (TT)	Códigos
BC	Ø
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 ØØØ ← TT → 1Ø1Ø

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	F2
	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	ØØ111Ø1Ø	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	111Ø11Ø1	Ø1Ø1Ø111

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	111Ø1Ø11	Ø1Ø11111

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	Ø
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 ØØØ ← TT → ØØ1Ø

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 ØØ ← SS → ØØØ1 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,6AØ4H

Registrador BC → 6AØ4H

90) LD SS,(nn)

SS representa os registradores BC, DE, SP

Registrador SS	Código
BC	ØØ
DE	1Ø
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 111Ø11Ø1 01 ← SS → 1Ø11 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	ØØ11Ø11Ø 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	ØØ1Ø1Ø1Ø	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 Ø111Ø ← r _ →

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 ← 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	1Ø7
Decimal	237	71
Binário	111Ø1Ø11	Ø1ØØØ111

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	Ø
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 → 111Ø1 ØØ1ØØØØ1 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	Ø
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 → 111Ø1 ØØ1Ø1Ø1Ø 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	Ø
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 ll ← T →lllØl ØØ11Ø11Ø 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)		4A
(IX -3) = (IX+FD)		CD
(IX -2) = (IX+FE)		7E
(IX -1) = (IX+FF)		B9
	(IX+Ø)	14
	(IX+1)	39
	(IX+2)	FA
	(IX+3)	Ø1
	(IX+4)	ØF

LD (IX + 3),ØD7H

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

(IX + 2)	FA
(IX + 3)	D7
(IX + 4)	ØF

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 → 111Ø1 ØlllØ ← 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)	Ø4
(IY - 1) =	(IY + FF)	2Ø
	(IY + Ø)	IF
	(IY + 1)	CA
	(IY + 2)	ED

(IY + 3)	FØ
(IY + 4)	BA
(IY + 5)	1A

LD (IY-3),C Registrador C → Ø5H

Posição de (IX-3) após efetuar esta instrução ficará igual a "Ø5".

(IY - 4)	14
(IY - 3)	Ø5
(IY - 2)	Ø4

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	ØØ11ØØ1Ø	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 111Ø11Ø1 Ø1 ← SS → ØØ11 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 = FØA3H
Conteúdo de memória (14ØCH) = 14H
Conteúdo de memória (14ØDH) = 5AH

LD (14ØCH),BC

Conteúdo do par de registradores BC = FØA3H
Conteúdo de memória (14ØCH) = A3
Conteúdo de memória (14ØDH) = FØ

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).

Código de Operação

Hexadecimal	22	nn
Octal	42	nn
Decimal	33	nn
Binário	ØØ1ØØØ1Ø	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	Ø
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 →111Ø1	ØØ1ØØØ1Ø 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 circuíto memórias dinâmicas (RAM's).

Código de Operação

Hexadecimal	ED	4F
Octal	355	117
Decimal	237	79
Binário	111Ø11Ø1	Ø1ØØ1111

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 ØlL r L 11Ø

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 ← (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 ← T → 111Ø1 ØlL r L 11Ø 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)	A1
(IX - 4) = (IX + FC)	BA
(IX - 3) = (IX + FD)	ØF
(IX - 2) = (IX + FE)	DE
(IX - 1) = (IX + FF)	F1
(IX + Ø)	CA
(IX + 1)	14
(IX + 2)	34
(IX + 3)	44
(IX + 4)	AF
(IX + 5)	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 ØØ ← r → 11Ø n

Parâmetros

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

Exemplo:

Conteúdo dos registrador H = CØH

LD H,1CH

Conteúdo do registrador H = 1CH

106) LD r,r'

Operação Simbólica

r ← 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

Ø1 ←_ r _→ ←_ r'_→

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 ← 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	11111ØØ1

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 → 111Ø1 11111ØØ1

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	111Ø11Ø1	1Ø1Ø1ØØØ	

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:

Ø764	7E
Ø765	F1
Ø766	Ø3

Conteúdo do par de registradores BC Ø765 H
Conteúdo do par de registradores DE 1ØØD H
Conteúdo do par de registradores HL ACF1 H

ACF1 D5

LDD 1ØØD 14

Conteúdo do par de registradores BC Ø764
Conteúdo do par de registradores DE 1ØØC
Conteúdo do par de registradores HL ACFD

ACF1 D5

H = Ø P/V = 1 N = Ø 1ØØD D5

110) LDDR

Operação Simbólica

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

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	111Ø11Ø1	1Ø111ØØØ

Parâmetros

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

se BC ≠ Ø
se BC = Ø

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:

7313	1Ø
7314	C9
7315	48

Conteúdo do par de registradores BC ØØØ3H
Conteúdo do par de registradores DE A173H
Conteúdo do par de registradores HL 7315H

A171	FD
A172	37
A173	2A

LDDR

Conteúdo do par de registradores BC ØØØØH
Conteúdo do par de registradores DE A17ØH
Conteúdo do par de registradores HL 7312H

7313	1Ø
7314	C9
7315	48

A171	1Ø
A172	C9
A173	48

111) LDI

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 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) ← (HL)
BC ← BC - 1
DE ← DE + 1
HL ← HL + 1
PC ← PC - 2 se BC ≠ 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	BØ
Octal	355	26Ø
Decimal	237	176
Binário	111Ø11Ø1	1Ø11ØØØØ

Parâmetros

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

se BC ≠ Ø
se BC = Ø

Condição dos FLAGS

Ver LDDR

113) NEG

Operação Simbólica

A ← Ø - 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	1Ø4
Decimal	237	68

Binário lllØllØl ØlØØØlØØ

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 "8ØH" 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)
          ØØØØ   ØØØØ     (ØØH)
        - ØØ11   1Ø1Ø     (3AH)
          11ØØ   Ø11Ø     (C6H)
```

Acumulador = C6 H

C = 1 Z = Ø P/V = Ø 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	ØØ
Octal	ØØ

Decimal ØØ
Binário ØØØØØØØØ

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 1Ø11Ø ← _ 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      Ø111   1Ø1Ø
Conteúdo do registrador B = C2H   v  11ØØ   ØØ1Ø
                                    ------------
                                    1111   1Ø1Ø
OR B

Conteúdo acumulador   = FAH
Conteúdo registrador B = C2H
```

C = Ø Z = Ø P/V = 1 S = 1 N = Ø 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	1Ø11Ø11Ø

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 → 111Ø1 1Ø11Ø11Ø d

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 111Ø11Ø1 Ø1 ← r → ØØ1

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	111Ø11Ø1	1Ø1Ø1Ø11

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 = Ø
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	111Ø11Ø1	1Ø1ØØØ11

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 ≠ Ø

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	111Ø11Ø1	1Ø111Ø11

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 ≠ Ø
se B = Ø

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) ← (HL)
HL ← HL + 1
B ← B - 2
PC ← PC - 2 se B ≠ Ø

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	111Ø11Ø1	1Ø11ØØ11

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 = Ø
se B ≠ Ø

Condição dos FLAGS

Ver OTDR

123) POP DD

Operação Simbólica

DDL ← (SP)
DDH ← (SP + 1)
SP ← (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 ← DD → ØØØ1

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 AB18H

POP HL

Conteúdo do par de registradores HL = 4913H
Conteúdo do registrador SP = AB1AH

AB18	13
AB19	49
AB1A	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 → 111Ø1 111ØØØØ1

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 ← DD → Ø1Ø1

Parâmetros

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

Exemplo:

Conteúdo do par de registradores BC = Ø7F4H
Conteúdo do registrador SP = FABCH

PUSH HL

Conteúdo do par de registradores BC = Ø7F4H
Conteúdo do registrador SP = FABAH

"SP" FABA F4
FABB Ø7
FABC 7A
FABD 27

126) PUSH T

Operação Simbólica

(SP - 1) ← TH
(SP - 2) ← TL
SP ← 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 → 111Ø1 111ØØ1Ø1

Parâmetros

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

127) RES b,r

Operação Simbólica

$r_b \leftarrow Ø$

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 11ØØ1Ø11 1Ø ← 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 Ø111 1Ø1Ø

RES 6,B

Conteúdo do registrador B = 3AH ØØ11 1Ø1Ø

128) RES b,(HL)

Operação Simbólica

$(HL)_b \leftarrow \emptyset$

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 11ØØ1Ø11 1Ø ← b → 11Ø

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

$(T + d)_b \leftarrow \emptyset$

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 ← T → 111Ø1 11ØØ1Ø11 d 1Ø ← b → 11Ø

Parâmetros

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

130) RET

Operação Simbólica

```
PCL ← (SP)
PCH ← (SP - 1)
SP  ← (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.

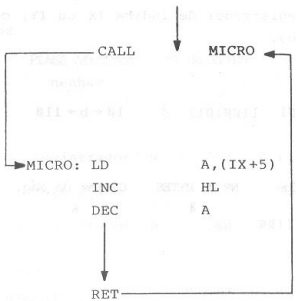
Código de Operação

Hexadecimal	C9
Octal	311
Decimal	2Ø1
Binário	11ØØ1ØØ1

Parâmetros

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

Exemplo:



131) RET CC

CC condição dos FLAGS

Condição CC		Código	FLAGS
NZ	Se não zero	ØØØ	Z
Z	Se zero	ØØ1	Z
NC	Se não CARRY	Ø1Ø	C
C	Se CARRY	Ø11	C
PO	Se paridade ímpar	1ØØ	P/V
PE	Se paridade par	1Ø1	P/V
P	Se sinal positivo	11Ø	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 ← (SP + 1)
SP ← (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	111Ø11Ø1	Ø1ØØ11Ø1

Parâmetros

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

133) RETN

Operação Simbólica

L PC ← (SP)
H PC ← (SP + 1)
SP ← (SP - 2)
IFF1 ← 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	1Ø5
Decimal	237	69
Binário	111Ø11Ø1	Ø1ØØØ1Ø1

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	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1

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 DØ.

Código de Operação

Binário	11ØØ1Ø11	ØØØ1Ø ← 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:

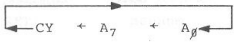
Conteúdo do registrador H DEH = 1Ø11 111Ø
CY = Ø

RL H
Conteúdo do registrador H 7CH = Ø111 11ØØ

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

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	ØØØ1Ø111

Parâmetros

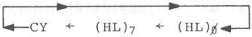
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RLA	C,N,H	1	1	4

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", exceto que o conteúdo de memória, endereçada pelo par "HL", é que sofrerá a operação.

Código de Operação

Hexadecimal	CB	16
Octal	313	26
Decimal	2Ø3	22
Binário	11ØØ1Ø11	ØØØ1Ø11Ø

Parâmetros

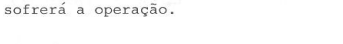
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RL (HL)	C,Z,P/V,S,N,H	2	4	15

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 ll ← T → llØl llØØlØll d ØØØlØllØ

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

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 "DØ" e os demais, deslocados de uma posição. O conteúdo do CARRY é afetado após a execução desta instrução.

Código de Operação

Binário llØØlØll ØØØlØ ← 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 → 73H → Ø111ØØ11

CY = 1

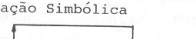
RLC B

Conteúdo do registrador B E6H → 111Ø Ø11Ø

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

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	Ø7
Octal	Ø7
Decimal	Ø7
Binário	ØØØØØ111

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	Ø6
Octal	313	6
Decimal	2Ø3	6
Binário	11ØØ1Ø11	ØØØØØ11Ø

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 endereça da pelo registrador de índice IX ou IY, com seu respectivo BYTE de dados, é quem sofre esta operação.

Código de Operação

Binário 11 ← T → 11Ø1 11ØØ1Ø11 d ØØØØØ11Ø

Parâmetros

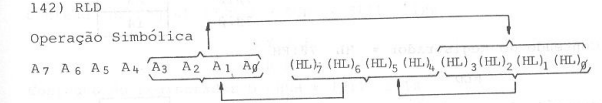
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RL (T + d)	Z,C,P/V,S,N,H	4	6	23

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	111Ø11Ø1	Ø11Ø1111

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

7Ø1E	6F
7Ø1F	A3
7Ø2Ø	14

Conteúdo do Registrador = HL 7Ø1FH

RLD

Conteúdo do acumulador 5AH

7Ø1E	6F
7Ø1F	3E
7Ø2Ø	14

Conteúdo do par de registradores HL 7Ø1FH

Z = Ø P/V = 1 S = Ø N = Ø H = Ø

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 "DØ" carregado no "CARRY".

Código de Operação

Binário 11ØØ1Ø11 ØØØ11 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 "DØ"
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 = Ø111 11ØØ

CY = 1
RR D

Conteúdo do registrador D → BEH = 1Ø11 111Ø

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

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	ØØØ11111

Parâmetros

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

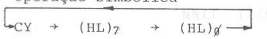
Condição dos FLAGS

C Carregamento com o valor do BIT "DØ"

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



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	2Ø3	3Ø
Binário	11ØØ1Ø11	ØØØ1111Ø

Parâmetros

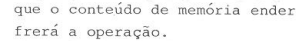
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RR (HL)	C,Z,P/V,S,N,H	2	4	15

Condição dos FLAGS

Ver RR r

146) RR (T + d)

Operação Simbólica



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 ll ← T → lllØl llØØlØll d ØØØlllØØ

Parâmetros

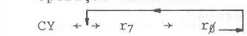
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RR (T + d)	Z,C,P/V,S,N,H	4	6	23

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 "DØ" 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 llØØlØll ØØØØl ←_ _r_ →

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 → A8H = 1Ø1Ø1ØØØ

CY = Ø

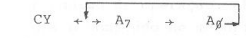
RLC C

Conteúdo do registrador C → 54H = ØlØl ØlØØ

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

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	ØF
Octal	17
Decimal	15
Binário	ØØØØllll

Parâmetros

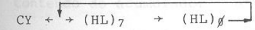
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRCA	C,N,H	1	1	4

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", exceto que o conteúdo de memória, endereçada pelo par "HL", é quem sofre a operação.

Código de Operação

Hexadecimal	CB	ØE
Octal	313	16
Decimal	2Ø3	14
Binário	11ØØ1Ø11	ØØØØ111Ø

Parâmetros

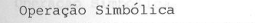
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRC (HL)	C,Z,P/V,S,N,H	2	4	15

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 → 11Ø1	11ØØ1Ø11	d	ØØØØ111Ø

Parâmetros

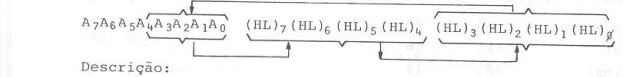
Mnemônico	FLAGS Afetados	Nº de BYTES	Ciclos de Máq.	Estados
RRC (T+d)	Z,C,P/V,S,N,H	4	6	23

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	1Ø3
Binário	111Ø11Ø1	Ø11ØØ111

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

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
Ø	ØØØ
8	ØØ1
10H	Ø1Ø
18H	Ø11
20H	1ØØ
28H	1Ø1
30H	11Ø
38H	111

Operação Simbólica

(SP-1) ← PCH
(SP-2) ← PCL
PCH ← Ø
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

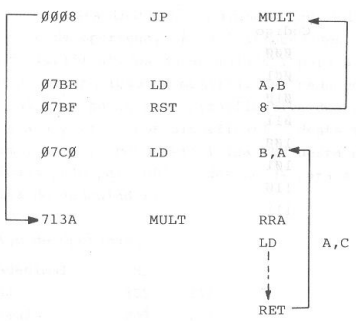
Binário 11 ← P → 111

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 ← 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

1ØØ11 ←_ 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 = Ø111 ØØØ1

Conteúdo do registrador B → 3AH = ØØ11 1Ø1Ø

Conteúdo do CARRY Ø → Ø1H = - ØØØØ ØØØ1

Resultado 36H = ØØ11 Ø11Ø

SBC A,B

Conteúdo do acumulador A → 36H

Conteúdo do registrador B → 3AH

C = Ø Z = Ø P/V = Ø S = Ø 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 11Ø1111Ø 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 ← 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 1ØØ1111Ø

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 ← 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 ← T → 111Ø1 1ØØ1111Ø 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 111Ø11Ø1 Ø1 ← SS → ØØ1Ø

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	ØØ11Ø111

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 11ØØ1Ø11 11 ← b→ ←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 → Ø111 11ØØ

SET 1,B

Conteúdo do registrador B = 7EH → Ø111 111Ø

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 11ØØ1Ø11 11 ← b → 11Ø

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 ← T → 111Ø1 11ØØ1Ø11 d 11 ← b → 11Ø

162) SLA r

Operação Simbólica

$CY \leftarrow r_7 \leftarrow r_Ø \leftarrow Ø$

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 Ø carrega-se "zero".

Código de Operação

Binário 11ØØ1Ø11 ØØ1ØØ ← r →

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 = ØØ11 Ø111

SLA H

Conteúdo do registrador H = 6E H Ø11Ø 111Ø

C = Ø Z = Ø P/V = Ø S = Ø N = Ø H = Ø

163) SLA (HL)

Operação Simbólica

CY ← $(HL)_7$ ← $(HL)_Ø$ ← Ø

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	2Ø3	38
Binário	11ØØ1Ø11	ØØ1ØØ11Ø

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 ← $(T + d)_7$ ← $(T + d)_0$ ← 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, 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 → 111011 11001011 d 00101110

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 → r_0 → 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 00101 ← r →

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 → 5CH ØlØl llØØ

SRA E

Conteúdo do registrador E → 2EH ØØlØ lllØ

C = Ø Z = Ø P/V = 1 S = Ø N= Ø H = Ø

166) SRA (HL)

Operação Simbólica

$(HL)_7 \rightarrow (HL)_{\emptyset} \rightarrow CY$

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	2Ø3	46
Binário	llØØlØll	ØØlØlllØ

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

$(T + d)_7 \rightarrow (T + d)_\emptyset \rightarrow CY$

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 ← T → 111Ø1 11ØØ1Ø11 d ØØ1Ø111Ø

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

$\emptyset \rightarrow r_7 \rightarrow r_\emptyset \rightarrow CY$

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 Ø carregado no CARRY BIT.

Código de Operação

Binário 11ØØ1Ø11 ØØ111 ← r →

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 = ØlH = ØØØØØØØl

SRL L

Conteúdo do registrador L = ØØH = ØØØØØØØØ

C = 1 Z = 1 P/V = 1 S = Ø N = Ø H = Ø

169) SRL (HL)

Operação Simbólica

$Ø \rightarrow (HL)_7 \rightarrow (HL)_Ø \rightarrow CY$

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	2Ø3	62
Binário	llØØlØll	ØØlllllØ

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

$\emptyset \rightarrow (T + d)_7 \rightarrow (T + d)_\emptyset \rightarrow CY$

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, endereça da 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 ← T → 111Ø1 11ØØ1Ø11 d ØØ111111Ø

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 ← AC − r

Descrição:

Esta instrução é formada por um único BYTE, sendo es te o código de operação. Esta instrução subtrai o conteúdo do acumulador com o registro (r) sendo o resultado guardado no acu mulador.

Código de Operação

Binário 1ØØ1Ø ← r →

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 → ØlllØØll

Conteúdo do registrador C = 3AH → - ØØllllØlØ

ØØllØØØl

SUB B

Conteúdo do acumulador = 39H

Conteúdo registrador C = 3AH

C = Ø Z = Ø P/V = Ø S = Ø 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	llØlØllØ	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 ← 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	15Ø
Binário	1ØØ1Ø11Ø

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 ← A − (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 IX e IY.

Código de Operação

Binário 11 ← T → 111Ø1 1ØØ1Ø11Ø 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 ← A ⊕ 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 1Ø1Ø1 ← 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 ØØ11 1Ø1Ø

Conteúdo do registrador E 8 7 + 1ØØØ Ø111

1Ø11 11Ø1

XOR E

Conteúdo do acumulador → = BDH

Conteúdo do registrador E = 87H

C = Ø Z = Ø P/V = 1 S = 1 N = Ø 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	111Ø111Ø	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 ← A ⊕ (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	1Ø1Ø111Ø

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 ← A ⊕ (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 ← T → 111Ø1 1Ø1Ø111Ø 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.

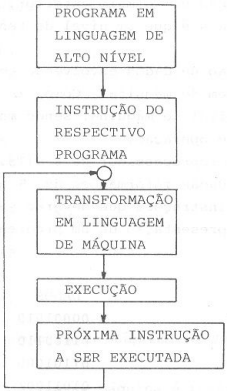


Fig. 8.3 - Programa Interpretador.

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 se transformar codificação binária em hexadecimal, divide-se em grupos de quatro elementos e, 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.

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 A1
	AND	ØFH	;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 lFA3H, lida com um conjunto de caracteres (LABEL), por exemplo JP LOOPl. Este artifício é útil para se seguir o fluxo de dados, e também quando se acrescenta ou se retira alguma instrução, pois o novo endereço do LABEL é automati

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,ØØ1ØØ11ØB; 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: (ØBA12H), (ØA3H).

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	Ø7BAH,	ØABH,	ØBFCØH
Binário	1ØØ1ØØB,	Ø111Ø1B,	1ØØ1ØØ11B
Octal	371O, 67ØO, 14O ou 371Q, 67ØQ, 14Q		
Decimal	77D, 6679ØD ou 77, 6679Ø		

```
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)	ØØØØ	7F	DEFB	7FH
	ØØØ1	ØF	DEFB	15
	ØØØ2	41	DEFB	41H
	ØØØ3	3A	DEFB	ØØ1111ØØB
	ØØØ4	Ø4	DEFB	Ø4H
	ØØØ5	1A	DEFB	1AH

```
        ØØØ6    33      DEFB    33H
        ØØØ7    ØA      DEFB    1Ø

2)      ØØØØ    CBØF    RRC     A
        ØØØ2    6Ø      LD      H,B
        ØØØ3    FA      DEFB    ØFAH
        ØØØ4    ØE20    LD      C,14H
        ØØØ6    CB85    RES     Ø,L
        ØØØ8    6C      LD      L,H
        ØØØ8    DD      DEFB    ØDDH
        ØØØA    2119AØ  LD      HL,ØAØ19H
```

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:

```
ØØØØ    3FØE    LD      A,14
=1ABO=  MULT    DEFL    1ABØH
ØØØ2    11Bø1A  LD      DE,MULT
ØØØ5    8Ø      ADD     A,B
ØØØ6    12      LD      (DE),A
=7B7E=  MULT    DEFL    7B7EH
ØØØ7    217E7B  LD      HL,MULT
ØØØA    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.

```
ØØØØ    4D 49 43 52     DEFM    "Microprocessador Z-80"
ØØØ4    4F 50 52 4F
ØØØ8    43 45 53 53
ØØØC    41 44 4F 52
ØØ1Ø    20 5A 38 30
ØØ11    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:

```
ØØØØ    78          LD      A,B
ØØØ1    21AØ7E      LD      HL,7EAØH
ØØØ4                DEES    7
ØØØB    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:

```
ØØØØ    B37A        DEFW        7AB3H
ØØØ2    Ø614        LD          B,14H
ØØØ4    8Ø          ADD         A,B
ØØØ5    14Ø7        DEFW        Ø714H
ØØØ7    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 determinar o endereço deste programa.

Exemplo:

```
                    ORG         2ØØØH
2ØØØ    DD2148ØØ    LD          IX,ØØ48H
2ØØ4    CBBF        RES         7 , A
2ØØ6    CBCØ        SET         Ø,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, exceto que não pode ser danificada no transcorrrer do programa.

Exemplo:

```
                    ORG         5ØØØH
=ØFFF=      FIM     EQU         ØFFFH
=7A=        PORT1   EQU         7AH
=8A=        PORT2   EQU         8AH
5ØØØ 31FFØF         LD          SP,FIM
5ØØ3 DB7A           IN          A,(PORT1)
5ØØ5 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     7BØØH
7BØØ        79          LD      A,B
                                          ⎧Representa
7BØ1        18Ø1        JR      Z,LOOP1-$ ⎨
7BØ3        44          LD      B,H       ⎩LOOP1-PC
7BØ4        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,0FFH
'000D  21800D           LD      HL,03456
'0010  C9               RET
```

Fig. 9.1 - Teste do Acumulador.

A instrução "CP ØØH" 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,0FFH
'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 = ØØØØØØ1Ø

```
   ØØØØ  ØØ1Ø ←— A
+  ØØØØ  ØØ1Ø ←— A
   ØØØØ  Ø1ØØ ←— A x 2
```

```
                    A
←— ØØØØ  ØØ1Ø ←0
   ØØØØ  Ø1ØØ ←A x 2
```

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,ØØH", 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 IGUAL A ZERO
'002C  AF               XOR     A ;ACUMULADOR IGUAL A ZERO

                        NAO ALTERA OS FLAGS
'002D  3E00             LD      A,00H ;ACUMULADOR IGUAL 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 = (Ø11Ø  111Ø) B    Deseja-se inverter "BIT 0" e o
                      "BIT 6"

'002F  EE41           XOR     01000001B

A = (ØØ1Ø  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       ;CONT. DE "B" NAO FOI AFETADO
'0033  CA3900'          JP      Z,TERMIN ;SE FOR ZERO VAI PARA TERMIN
                                         ;CASO CONTRARIO CONTINUE
'0036  0EFF             LD      C,0FFH  ;SETA O REGISTRO "C"
'0038  C9               RET
'0039  0E00     TERMIN: LD      C,000H  ;RESETA O REGISTRO "C"
'0038  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:

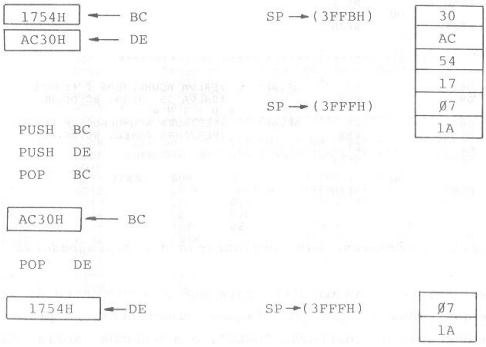


Fig. 9.7.A - Fluxo de dados de troca de registradores.

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  FB              EI
'0051  ED4D            RETI
                OU
'0053  ED45            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
                OU
'005C  ED45            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  5D       0014        LD      E,L
'0004  13       0015        INC     DE         ;INCREMENTA  "DE"
'0005  EDB0     0016        LDIR               ;CARREGA O CONT. DE MEM.' ENDER.
                0017                           ;PELO "HL" NA POS. DE MEM. END.
                0018                           ;PELO "DE"
'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,0FFH      ;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 a 255)$_{10}$, ou (Ø a FF)$_{16}$; portanto o "LOOP" poderá repetir apenas nesta faixa. Na figura 9.12, pode-se ter uma visão desta técnica.

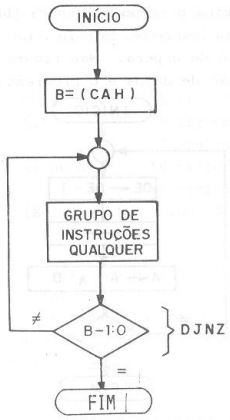


Fig. 9.12 - LOOP.

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.

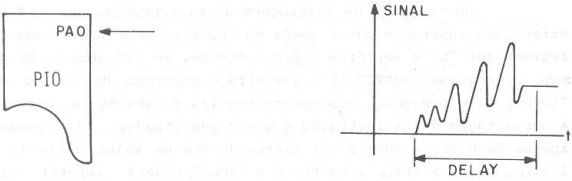


Fig. 9.13 - Aplicação da rotina "DELAY".

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.

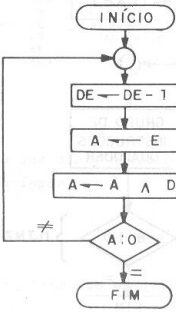


Fig. 9.14 - Fluxo da rotina "DELAY".

```
                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  1B       0010 DELAY:  DEC      DE          ;DECREMENTA O CONTADOR
'0001  7B       0011         LD       A,E
'0002  B2       0012         OR       D           ;VER. SE "D" E "E" SAO ZEROS
'0003  20F8     0013         JR       NZ,DELAY-$  ;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:

CLOCK = 2MHZ = 5×10^{-7} seg.

Tempo de execução = 0,5 seg.

$0,5 = 5 \times 10^{-7}$ (nn . 24 + 10)

$nn = (41666)_{10} = (A2B2)_{16}$

Portanto "DE" = (A2B2H)

OBS:. O máximo número que "nn" pode assumir é $(65535)_{10}$ ou $(FFFF)_{16}$; 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 BITØ), 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.

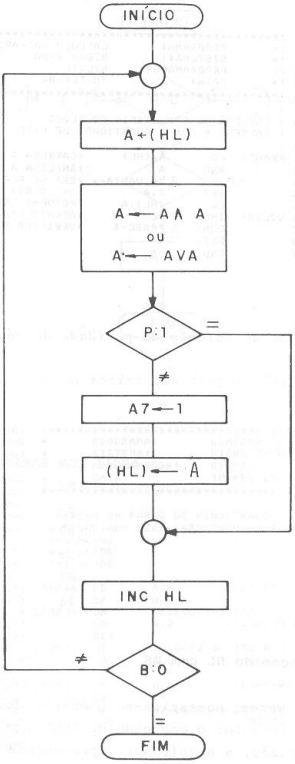


Fig. 9.17 - Fluxo do Cálculo da Paridade do Código ASCII.

```
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 ;
'0000  7E       0011 PRASC:  LD    A,(HL)     ;CARREGA A COM O DADO
'0001  A7       0012         AND   A          ;ANALISA A PARIDADE
'0002  EA0600   0013         JP    PE,VOLTA-$ ;SET SE FOR IMPAR
'0005  CBFF     0014         SET   7,A        ;SET O BIT 7
'0007  77       0015         LD    (HL),A     ;RETORNA COM O DADO
'0008  23       0016 VOLTA:  INC   HL         ;APONTA PARA O PROXIMO DADO
'0009  10F5     0017         DJNZ  PRASC-$    ;VERIFICA SE JA ACABOU
'0008  C9       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.

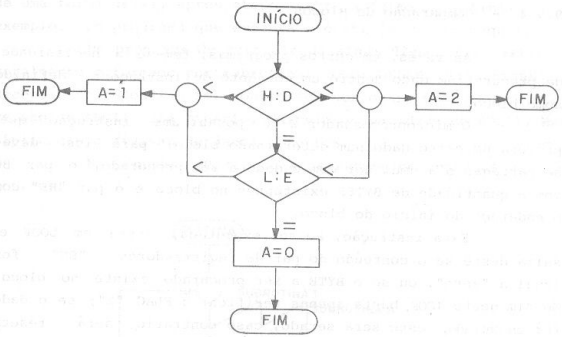


Fig. 9.18 - Fluxo da rotina que compara "HL" com "DE".

```
                        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  BA               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  ED81     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

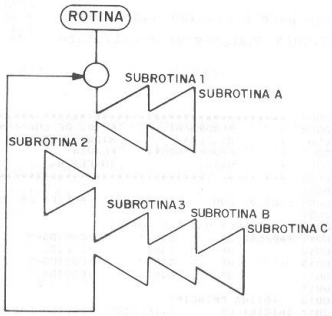
9.6.1 - Introdução

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" deverá 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-$
                    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
                    0036
'001F  C9           0037        RET
                    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.

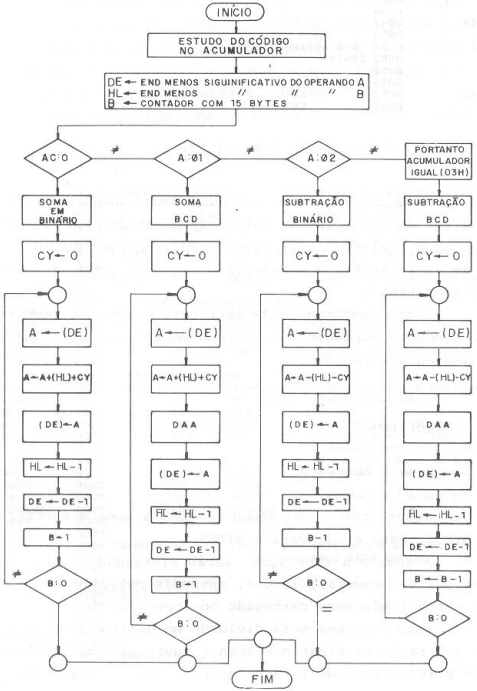


Fig. 9.22 - Fluxo da operação soma ou subtração.

```
                        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  11DE10           0020 INICIO:   LD      DE,OPERA+14     ;END. PRIM. BYTE DO OPER. A
'0003  21D010           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  2B               0029           DEC     HL              ;DECREMENTA OS PONTEIROS
'0011  1B               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  2B               0039           DEC     HL              ;DECREMENTA OS PONTEIROS
'001C  1B               0040           DEC     DE
'001D  10F8             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  2B               0050           DEC     HL              ;DECREMENTA OS PONTEIROS
'0029  2B               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  2B      '0060       DEC     HL          ;DECREMENTA OS PONTEIROS
'0033  2B      0061        DEC     HL
'0034  10F8    0062        DJNZ    LOOP4-$     ;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.

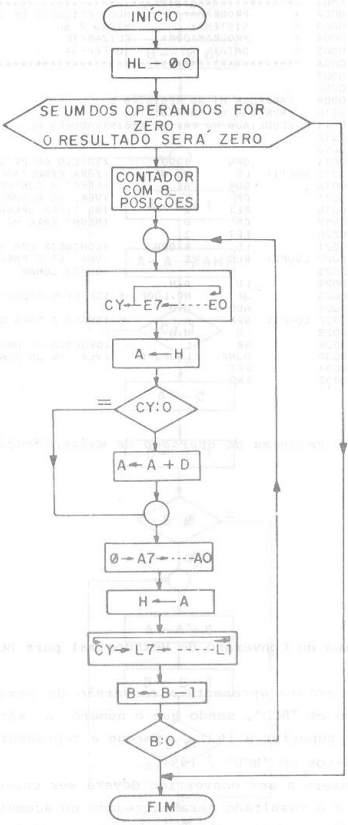


Fig. 9.23 - Fluxo da operação de multiplicação.

```
                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
>0514           0014          ORG     1300        ;INICIO DO PROGRAMA
'0514  210000    0015 MULT1:  LD      HL,00       ;ZERA RESULTADO
'0517  97        0016          SUB     A           ;RESET O CONTEUDO DO ACULM.
'0518  BB        0017          CP      E           ;VER. SE ALGUM OPERANDO E ZERO
'0519  C8        0018          RET     Z           ;SE TIVER OPERACAO TERMINADA
'051A  BA        0019          CP      D           ;MESMO PARA M2
'051B  C8        0020          RET     Z
'051C  0608      0021          LD      B,08H       ;CONTADOR COM 8 BITS
'051E  CB03      0022 LOOP2:   RLC     E           ;VER. SE E PARA SOMAR E DESL.
                 0023                              ;OU SO SOMAR
'0520  7C        0024          LD      A,H
'0521  3001      0025          JR      NC,LOOP1-$  ;SE CY=0 APENAS DESLOCA
'0523  82        0026          ADD     A,D
'0524  CB3F      0027 LOOP1:   SRL     A           ;DESLOCA PARA DIREITA
'0526  67        0028          LD      H,A
'0527  CB1D      0029          RR      L           ;DESLOCA DE UMA POSICAO "L"
'0529  10F3      0030          DJNZ    LOOP2-$     ;VER. SE JA ACABOU A OPERACAO
'052B  C9        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.

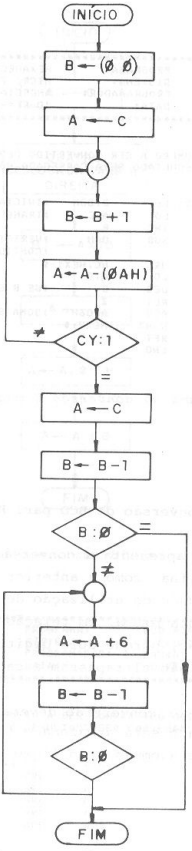


Fig. 9.24 - Fluxo da conversão de Hexadecimal para BCD.

```
                        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
                        0017                           ;CONTIDO NUMERO
'0006  30FB             0018         JR      NC,HEX1-$
'0008  79               0019         LD      A,C
'0009  05               0020         DEC     B         ;SE B E ZERO A CONVERSAO TERMINOU
'000A  C8               0021         RET     Z
'000B  C606             0022 HEX2:   ADD     A,06H     ;SOMA SEIS NO ACUMULADOR
'000D  1DFC             0023         DJNZ    HEX2-$
'000F  C9               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.

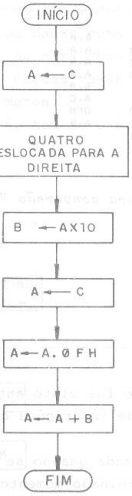


Fig. 9.25 - Fluxo da conversão BCD para Hexadecimal.

```
                0001 ;*****************************************************
                0002 ;*       PROGRAMA:     BCD PARA  HEXADECIMAL         *
                0003 ;*       SISTEMA:      MICRO Z-80                    *
                0004 ;*       PROGRAMADOR:  LAURA                         *
                0005 ;*       DATA:         10-FER-84                     *
                0006 ;*****************************************************
                0007 ;
                0008 ;
                0009 ; O NUMERO A SER CONVERTIDO DEVERA SER CARREGADO NO REG.
                0010 ; "C" O RESULTADO SER CARREGADO NO ACULMULADOR
                0011 ;
                0012 ;
'0000  79       0013 BCDHEX: LD      A,C              ;NUMERO A SER CONVERTIDO
'0001  CB3F     0014         SRL     A                ;RESET O DIG. MENOS SIGN.
'0003  CB3F     0015         SRL     A
'0005  CB3F     0016         SRL     A
'0007  CB3F     0017         SRL     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      ;DIGITO 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.

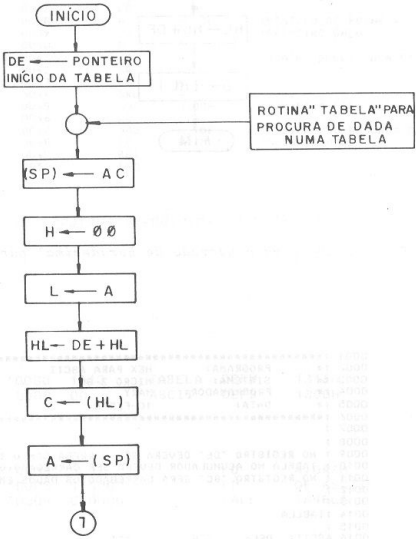
9.8.2 - Transformação de Hexadecimal para ASCII

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 Ø), 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 subrotina 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 rotina.



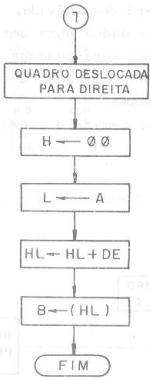


Fig. 9.26 - Fluxo para conversão de Hexadecimal para ASCII.

```
                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 ;
'0000  30       0016 ASCII:  DEFB    30H        ;"1"
'0001  31       0017         DEFB    31H        ;"2"
'0002  32       0018         DEFB    32H        ;"3"
```

```
'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.

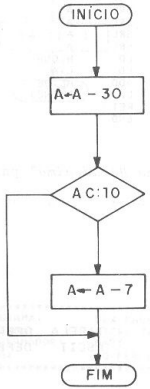


Fig. 9.27 - Fluxo da rotina.

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
	43	C
AC - 30 - 7	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-84                      *
                    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  D607         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)

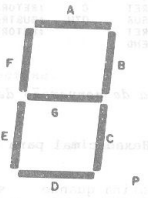


Fig. 9.28.A - DISPLAY de sete segmentos.

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 ;                  CODIGO      -     DIGITO
'0000  3F               0007 SETSEG   DEFB      03FH        ;"0"
'0001  06               0008          DEFB      006H        ;"1"
'0002  5B               0009          DEFB      05BH        ;"2"
'0003  4F               0010          DEFB      04FH        ;"3"
'0004  66               0011          DEFB      066H        ;"4"
'0005  6D               0012          DEFB      06DH        ;"5"
'0006  7D               0013          DEFB      07DH        ;"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 freqüência de entrada por 16 ou 255, e pelo seu registrador interno, que pode variar entre (255 a Ø).

Para cálculo do valor do registrador interno do CTC, basta apenas aplicar-se a fórmula dada a seguir:

$$R = \frac{Fc}{D \cdot Fp} \qquad \text{onde,}$$

R = Valor do registrador interno do CTC
Fc = Freqüência do CLOCK injetado no CTC
D = Fator (Prescale) 255 ou 16
Fp = Freqüência pedida

Exemplo:

Fc = 2,4756 MHZ
D = 256
FP = 100 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    RCTC0       ;INTERRUPCAO DO "CTC0"
                    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  0E81        0028         LD      C,CTCA0     ;ENDERECO DO "CTC0"
'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  3E41        0040         LD      A,41H       ;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 RCTC0:  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,FIM-$
```

```
'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:

$$TIMOUT = \frac{T}{T_{CTC}}$$

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

TIMOUT = 100 ms = $(10)_{10}$ = $(\emptyset A)_{16}$

```
LD  A,ØAH
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 PIO1AC  EQU     8FH        ;PORTO "A"
>0091            0006 PIO1BC  EQU     91H        ;PORTO "B"
                 0007
```

```
                        A)      MODO "ZERO"

'0000  3E0F             LD      A,00001111B ;CODIGO DE SAIDA
'0002  D38F             OUT     (PIO1AC),A
'0004  D391             OUT     (PIO1BC),A

                        B)      MODO "UM"

'0006  3E4F             LD      A,01001111B ;CODIGO DE ENTRADA
'0008  328F00           LD      (PIO1AC),A
'000B  329100           LD      (PIO1BC),A

                        C)      MODO "DOIS"

'000E  3E8F             LD      A,10001111B ;CODIGO DE BIDERECIONAL
'0010  328F00           LD      (PIO1AC),A
'0013  328F00           LD      (PIO1AC),A

                        D)      MODO    "TRES"

'0016  3ECF             LD      A,11001111B ;CONTROLE DO MODO TRES
'0018  D38F             OUT     (PIO1AC),A
'001A  D391             OUT     (PIO1BC),A
'001C  3E81             LD      A,10000001B ;BIT "7" E "0" COMO ENTRADA E
                                            ;E OS RESTANTES COMO SAIDA
'001E  D38F             OUT     (PIO1AC),A
'0020  3EF0             LD      A,11110000B ;BIT "7,6,5,4" COMO ENTRADA E
                                            ;OS RESTATE COMO SAIDA
'0022  D391             OUT     (PIO1BC),A
```

OBS: A nomenclatura PIO1AD, PIO1AC, PIO1BD, PIO1BC 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.

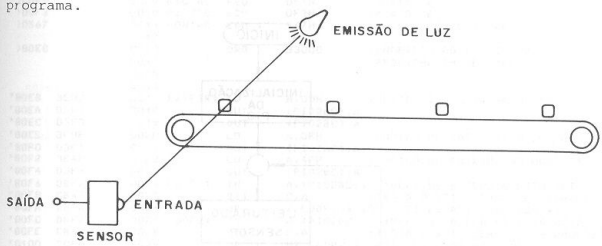


Fig. 9.32 - Estrutura para contagem de unidade em uma esteira.

A saída do sensor será acoplada a um "BIT" de um porto que, no caso, será o BIT "Ø" do porto B, e no porto A, teremos dois DISPLAYS, que contarão de Ø a 99 unidades.

Na figura 9.33, tem-se o esquema de ligação entre a CPU, PIO e o SENSOR.

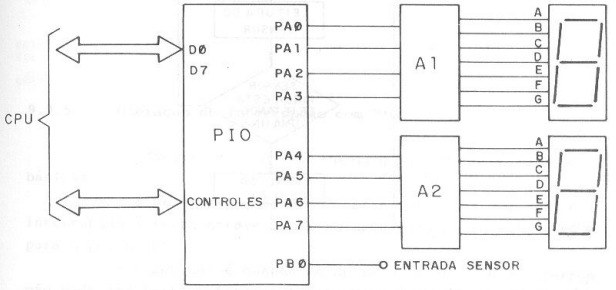


Fig. 9.33 - Esquema de ligação entre CPU, PIO e SENSOR.

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.

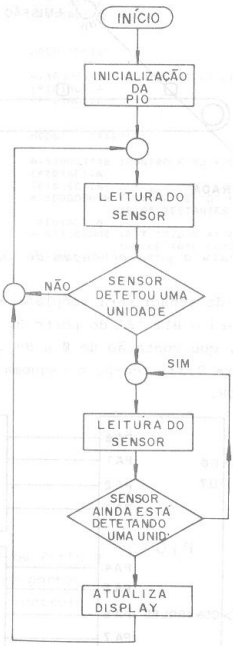


Fig. 9.34 - Fluxo de contagem de unidade.

```
                0001  ********************************************************
                0002  *       PROGRAMA:       PIO COM CONTADOR               *
                0003  *       SISTEMA:        MICRO Z-80                     *
                0004  *       PROGRAMADOR:    FERNANDA                       *
                0005  *       DATA:           10-FER-84                      *
                0006  ********************************************************
                0007
                0008
>00F1           0009 PIO2AC   EQU     0F1H        ;PORTO "A"
>00F3           0010 PIO2BC   EQU     0F3H        ;PORTO "B"
>0567           0011 CONTAD   EQU     567H        ;BUFFER DE CONTADOR
                0012
>80E8           0013          ORG     33000       ;ENDERECO DO INICIO DO
                0014                              ;PROGRAMA EM DECIMAL
                0015
                0016
'80E8  3E03     0017 EXTEIR:  LD      A,03H       ;DESABILITA A INTERRUPCAO
'80EA  D3F1     0018          OUT     (PIO2AC),A
'80EC  D3F3     0019          OUT     (PIO2BC),A
'80EE  3E0F     0020          LD      A,0FH       ;CODIGO DE SAIDA NO MODO "0"
'80F0  D3F1     0021          OUT     (PIO2AC),A
'80F2  3E4F     0022          LD      A,4FH       ;CODIGO DE ENTRADA NO MODO "1"
'80F4  D3F3     0023          OUT     (PIO2BC),A
'80F6  DBF2     0024 LOOP1:   IN      A,(PIO2BC-1) ;VER. SE O SENSOR ESTA ACIO.
'80F8  CB47     0025          BIT     0,A         ;TESTA O BIT "ZERO" DO ACUMUL.
'80FA  28FA     0026          JR      Z,LOOP1-$ ;SE ESTIVER ACIONADO CONT+1
'80FC  DBF2     0027 LOOP2:   IN      A,(PIO2BC-1) ;VER. SE O SENSOR ESTA ACIO.
'80FE  CB47     0028          BIT     0,A         ;TESTA O BIT "ZERO" DO ACUMULADOR
'8100  20FA     0029          JR      NZ,LOOP2-$ ;FICA EM LOOP ATE QUE O
                0030                              ;SENSOR SEJA DESATIVADO
'8102  3A6705   0031          LD      A,(CONTAD) ;BUFFER DO CONTADOR
'8105  C601     0032          ADD     A,01H       ;INCREMENTA CONTADOR
'8107  27       0033          DAA                 ;AJUSTA PARA DECIMAL
'8108  326705   0034          LD      (CONTAD),A ;GUARDA CONTADOR CORRIGIDO
'810B  D3F0     0035          OUT     (PIO2AC-1),A ;ATUALIZAR O PORTO
'810D  18E7     0036          JR      LOOP1-$
                0037          END
```

Fig. 9.36 - 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             0007            LD      A,01001111B ;DEFINE COMO MODO DE SAIDA
'0002  D3A4             0008            OUT     (0A4H),A
'0004  3E04             0009            LD      A,04H       ;VETOR DE INTERRUPCAO
'0006  D3A4             0010            OUT     (0A4H),A
'0008  3E87             0011            LD      A,10000111B ;HABILITA A INTERRUPACAO
'000A  D3A4             0012            OUT     (0A4H),A
                        0013
                        0014
                        0015
```

B) Interrupção pelo próprio BIT

```
                        0016
                        0017
                        0018
                        0019
'000C  3ECF             0020            LD      A,11001111B ;PALAVRA DE CONTR. MODO TRES
'000E  D378             0021            OUT     (78H),A
'0010  3E80             0022            LD      A,10000000B ;APENAS O BIT 7 COMO ENTRADA
'0012  D34E             0023            OUT     (7B),A
'0014  3EF4             0024            LD      A,0F4H      ;VETOR DE INTERRUPCAO
'0016  D34E             0025            OUT     (7B),A
'0018  3E87             0026            LD      A,10110111B ;BIT 7 HABILITA A INTERRUPCAO
                        0027                                ;BIT 6 INTER. EM UM UNICO BIT
                        0028                                ;BIT 5 LOGICA POSITIVA
                        0029                                ;BIT 4 MASCARA DA INTERRUPCAO
                        0030                                ;BIT 3,2,1,0 CONTROLE
'001A  D34E             0031            OUT     (7B),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 pinos "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.

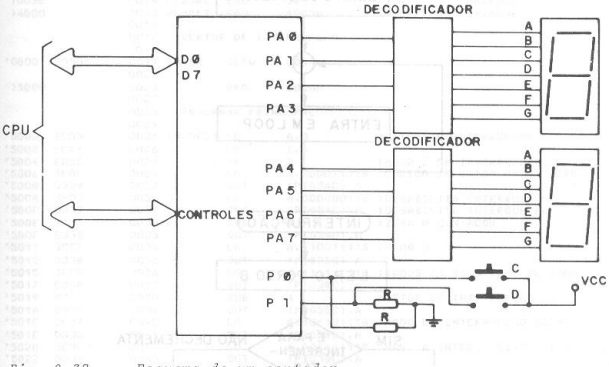


Fig. 9.37 - Esquema de um contador.

Nas figuras 9.38, pode-se acompanhar o fluxo e o programa do contador.

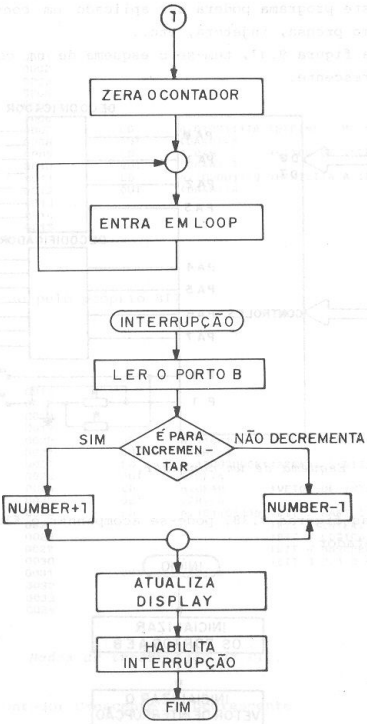


Fig. 9.38 - Fluxo do contador.

```
                        0001  ************************************************************
                        0002  *       PROGRAMA:       CONTADOR                           *
                        0003  *       SISTEMA:        MICRO Z-80                         *
                        0004  *       PROGRAMADOR:    CAROLINA                           *
                        0005  *       DATA:           10-FER-84                          *
                        0006  ************************************************************
                        0007
                        0008
                        0009    ENDERECAMENTO 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    VERTOR 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  D338             0035          OUT     (PIO3BC),A
'5015  3EFF             0036          LD      A,11111111B     ;TODOS OS BITS COMO ENTRADA
'5017  D338             0037          OUT     (PIO3BC),A
'5019  97               0038          SUB     A               ;VETOR DE INTERRUPCAO
'501A  D338             0039          OUT     (PIO3BC),A
'501C  3E87             0040          LD      A,10110111B     ;MODO DE INTERRUPCAO DO PIO
'501E  D338             0041          OUT     (PIO3BC),A
'5020  3E3F             0042          LD      A,00111111B     ;HAB. A INTER.  BITS 7 E 6
'5022  D338             0043          OUT     (PIO3BC),A
'5024  FB               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,(PIO38D)      ;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     (PIO3AD),A ;ATUALIZA DISPLY
'503E  F1        0062        POP     AF         ;RESTAURA ACUMULADOR E FLAGS
'503F  FB        0063        EI                 ;HABILITA INTERRUPCAO
'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
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
070	0100 0110	106	46
071	0100 0111	107	47
072	0100 1000	110	48
073	0100 1001	111	49
074	0100 1010	112	4A
075	0100 1011	113	4B
076	0100 1100	114	4C
077	0100 1101	115	4D
078	0100 1110	116	4E
079	0100 1111	117	4F
080	0101 0000	120	50
081	0101 0001	121	51
082	0101 0010	122	52
083	0101 0011	123	53
084	0101 0100	124	54
085	0101 0101	125	55
086	0101 0110	126	56
087	0101 0111	127	57
088	0101 1000	130	58
089	0101 1001	131	59
090	0101 1010	132	5A
091	0101 1011	133	5B
092	0101 1100	134	5C
093	0101 1101	135	5D
094	0101 1110	136	5E
095	0101 1111	137	5F
096	0110 0000	140	60
097	0110 0001	141	61
098	0110 0010	142	62
099	0110 0011	143	63
100	0110 0100	144	64
101	0110 0101	145	65
102	0110 0110	146	66
103	0110 0111	147	67
104	0110 1000	150	68
105	0110 1001	151	69
106	0110 1010	152	6A

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL
107	0110 1011	153	6B
108	0110 1100	154	6C
109	0110 1101	155	6D
110	0110 1110	156	6E
111	0110 1111	157	6F
112	0111 0000	160	70
113	0111 0001	161	71
114	0111 0010	162	72
115	0111 0011	163	73
116	0111 0100	164	74
117	0111 0101	165	75
118	0111 0110	166	76
119	0111 0111	167	77
120	0111 1000	170	78
121	0111 1001	171	79
122	0111 1010	172	7A
123	0111 1011	173	7B
124	0111 1100	174	7C
125	0111 1101	175	7D
126	0111 1110	176	7E
127	0111 1111	177	7F
128	1000 0000	200	80
129	1000 0001	201	81
130	1000 0010	202	82
131	1000 0011	203	83
132	1000 0100	204	84
133	1000 0101	205	85
134	1000 0110	206	86
135	1000 0111	207	87
136	1000 1000	210	88
137	1000 1001	211	89
138	1000 1010	212	8A
139	1000 1011	213	8B
140	1000 1100	214	8C
141	1000 1101	215	8D
142	1000 1110	216	8E
143	1000 1111	217	8F

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
181	1011 0101	265	B5
182	1011 0110	266	B6
183	1011 0111	267	B7
184	1011 1000	270	B8
185	1011 1001	271	B9
186	1011 1010	272	BA
187	1011 1011	273	BB
188	1011 1100	274	BC
189	1011 1101	275	BD
190	1011 1110	276	BE
191	1011 1111	277	BF
192	1100 0000	300	C0
193	1100 0001	301	C1
194	1100 0010	302	C2
195	1100 0011	303	C3
196	1100 0100	304	C4
197	1100 0101	305	C5
198	1100 0110	306	C6
199	1100 0111	307	C7
200	1100 1000	310	C8
201	1100 1001	311	C9
202	1100 1010	312	CA
203	1100 1011	313	CB
204	1100 1100	314	CC
205	1100 1101	315	CD
206	1100 1110	316	CE
207	1100 1111	317	CF
208	1101 0000	320	D0
209	1101 0001	321	D1
210	1101 0010	322	D2
211	1101 0011	323	D3
212	1101 0100	324	D4
213	1101 0101	325	D5
214	1101 0110	326	D6
215	1101 0111	327	D7
216	1101 1000	330	D8
217	1101 1001	331	D9

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

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

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

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		

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

RESTRUTURA DOS REGISTRADORES DO Z-80

D.1) Organização Interna Dos Dentro Unes do Z-80

ACUMULADOR "A"	F	D	H
FLAGS "F"	C	B	L

ACUMULADOR "A'"	B'	D'	H'
FLAGS "F'"	C'	E'	L'

REGISTRADOR "I"	PROGRAM COUNTER "PC"
REGISTRADOR DE INDEX "IX"	REGISTRADOR DE INDEX "IY"
STACK POINTER "SP"	PROGRAM 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

APÊNDICE E

INTERRUPÇÕES DO Z-80

E.1) NÃO MASCARADA ($\overline{NMI}$)

ENDEREÇO $(0066)_H$

E.2) MASCARADA (INT)

MODO Ø

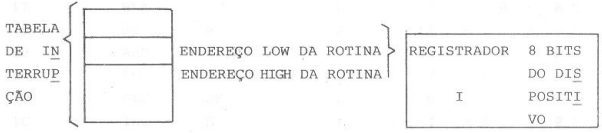
Código no DATA BUS, podendo até endereçar oito posições de memória.

($(ØØØØ)_H$, $(ØØØ8)_H$, $(ØØ1Ø)_H$, $(ØØ18)_H$, $(ØØ2Ø)_H$, $(ØØ28)_H$, $(ØØ3Ø)_H$, $(ØØ38)_H$).

E.3) MODO 1

ENDEREÇO $(ØØ38)_H$

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
$\overline{\text{NMI}}$	Ø	-	
RETN	IFF2	-	IFF1 ← IFF2
$\overline{\text{INT}}$	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
ØØ	NOP		1	4	- - - - - -
Ø1 YYXX	LD	BC,XXYY	3	1Ø	- - - - - -
Ø2	LD	(BC),A	1	7	- - - - - -
Ø3	INC	BC	1	6	- - - - - -
Ø4	INC	B	1	4	S Z H V Ø -
Ø5	DEC	B	1	4	S Z H V 1 -
Ø6 XX	LD	B,XX	2	7	- - - - - -
Ø7	RLCA		1	4	- - Ø - Ø C
Ø8	EX	AF,AF'	1	4	S Z H P/V N C
Ø9	ADD	HL,BC	1	11	- - X - Ø C
ØA	LD	A,(BC)	1	7	- - - - - -
ØB	DEC	BC	1	6	- - - - - -
ØC	INC	C	1	4	S Z H V Ø -
ØD	DEC	C	1	4	S Z H V 1 -
ØE XX	LD	C,XX	2	7	- - - - - -
ØF	RRCA		1	4	- - Ø - Ø C
1Ø e-2	DJNZ	e	2	8/13	- - - - - -
11 YYXX	LD	DE,XXYY	3	1Ø	- - - - - -
12	LD	(DE),A	1	7	- - - - - -
13	INC	DE	1	6	- - - - - -
14	INC	D	1	4	S Z H V Ø -
15	DEC	D	1	4	S Z H V 1 -
16 XX	LD	D,XX	1	4	- - - - - -
17	RLA		1	4	- - Ø - Ø C
18 e-2	JR	e	2	12	- - - - - -
19	ADD	HL,DE	1	11	- - X - Ø C
1A	LD	A,(DE)	1	7	- - - - - -
1B	DEC	DE	1	6	- - - - - -
1C	INC	E	1.	4	S Z H V Ø -
1D	DEC	E	1	4	S Z H V 1 -
1E XX	LD	E,XX	2	4	- - - - - -
1F	RRA		1	4	- - Ø - Ø C

HEXA	COP	OPER.	Nº BYTES	Nº EST.	FLAGS AFETADOS S,Z,H,P/V,N,C
2Ø e-2	JR	NZ,e	2	7/12	- - - - - -
21 YYXX	LD	HL,XXYY	3	1Ø	- - - - - -
22 ppqq	LD	(qqpp),HL	3	16	- - - - - -
23	INC	HL	1	6	- - - - - -
24	INC	H	1	4	S Z H V Ø -
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 - Ø C
2A ppqq	LD	HL,(qqpp)	3	16	- - - - - -
2B	DEC	HL	1	6	- - - - - -
2C	INC	L	1	4	S Z H V Ø -
2D	DEC	L	1	4	S Z H V 1 -
2E XX	LD	L,XX	2	4	- - - - - -
2F	CPL		1	4	- - 1 - 1 -
3Ø e-2	JR	NC,e	2	7/12	- - - - - -
31 YYXX	LD	SP,XXYY	3	1Ø	- - - - - -
32 ppqq	LD	(qqpp),A	3	13	- - - - - -
33	INC	SP	1	6	- - - - - -
34	INC	(HL)	1	11	S Z H V Ø -
35	DEC	(HL)	1	11	S Z H V 1 -
36 XX	LD	(HL),XX	2	1Ø	- - - - - -
37	SCF		1	4	- - Ø - Ø 1
38 e-2	JR	C,e	2	7/12	- - - - - -
39	ADD	HL,SP	1	11	- - X - Ø C
3A ppqq	LD	A,(qqpp)	3	13	- - - - - -
3B	DEC	SP	1	6	- - - - - -
3C	INC	A	1	4	S Z H V Ø -
3D	DEC	A	1	4	S Z H V 1 -
3E XX	LD	A,XX	2	7	- - - - - -
3F	CCF		1	4	- - H - Ø C
4Ø	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	-	-	-	-	-	-
7Ø	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	-	-	-	-	-	-
8Ø	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
9Ø	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
AØ	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
BØ	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
CØ	RET	NZ	1	5/11	-	-	-	-	- -
C1	POP	BC	1	1Ø	-	-	-	-	- -
C2 ppqq	JP	NZ,qqpp	3	1Ø	-	-	-	-	- -
C3 ppqq	JP	qqpp	3	1Ø	-	-	-	-	- -
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	1Ø	-	-	-	-	- -
CA ppqq	JP	Z,ppqq	3	1Ø	-	-	-	-	- -
CBØØ	RLC	B	2	8	S	Z	Ø	P	Ø C
CBØ1	RLC	C	2	8	S	Z	Ø	P	Ø C
CBØ2	RLC	D	2	8	S	Z	Ø	P	Ø C
CBØ3	RLC	E	2	8	S	Z	Ø	P	Ø C
CBØ4	RLC	H	2	8	S	Z	Ø	P	Ø C
CBØ5	RLC	L	2	8	S	Z	Ø	P	Ø C
CBØ6	RLC	(HL)	2	15	S	Z	Ø	P	Ø C
CBØ7	RLC	A	2	8	S	Z	Ø	P	Ø C
CBØ8	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	
CBØ9	RRC	C	2	8	S	Z	Ø	P	Ø	C	
CBØA	RRC	D	2	8	S	Z	Ø	P	Ø	C	
CBØB	RRC	E	2	8	S	Z	Ø	P	Ø	C	
CBØC	RRC	H	2	8	S	Z	Ø	P	Ø	C	
CBØD	RRC	L	2	8	S	Z	Ø	P	Ø	C	
CBØE	RRC	(HL)	2	15	S	Z	Ø	P	Ø	C	
CBØF	RRC	A	2	8	S	Z	Ø	P	Ø	C	
CB1Ø	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	
CB2Ø	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 AFETADOS						
					S	Z	H	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 -
CB8Ø	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	- - - - - -
CB9Ø	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	- - - - - -
CBAØ	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	-	-	-	-	-	-
CBAA	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	-	-	-	-	-	-
CBBB	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	-	-	-	-	-	-
CBFØ	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	Z,qqpp	3	1Ø/17	-	-	-	-	-	-
CD ppqq	CALL	qqpp	3	17	-	-	-	-	-	-
CE XX	ADC	A,XX	2	7	S	Z	H	V	Ø	C
CF	RST	08H	1	11	-	-	-	-	-	-
DØ	RET	NC	1	5/11	-	-	-	-	-	-
D1	POP	DE	1	1Ø	-	-	-	-	-	-
D2 ppqq	JP	NC,qqpp	3	1Ø	-	-	-	-	-	-
D3 pp	OUT	(pp),A	2	11	-	-	-	-	-	-
D4 ppqq	CALL	NC,ppqq	3	1Ø/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	1Ø	-	-	-	-	-	-	
DB pp	IN	A,(pp)	2	11	-	-	-	-	-	-	
DC ppqq	CALL	C,qqpp	3	1Ø/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	2Ø	-	-	-	-	-	-	
DD23	INC	IX	2	1Ø	-	-	-	-	-	-	
DD2A ppqq	LD	IX,(qqpp)	4	2Ø	-	-	-	-	-	-	
DD2B	DEC	IX	2	1Ø	-	-	-	-	-	-	
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 dØ6	RLC	(IX+d)	4	23	S	Z	Ø	P	Ø	C
DDCB dØE	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	-	-	-	-	-	-	
EØ	RET	PO	1	5/11	-	-	-	-	-	-	
E1	POP	HL	1	1Ø	-	-	-	-	-	-	
E2 ppqq	JP	PO,qqpp	3	1Ø	-	-	-	-	-	-	
E3	EX	(SP),HL	1	19	-	-	-	-	-	-	
E4 ppqq	CALL	PO,qqpp	3	1Ø/17	-	-	-	-	-	-	
E5	PUSH	HL	1	11	-	-	-	-	-	-	
E6 XX	AND	XX	2	7	S	Z	1	P	Ø	Ø	
E7	RST	20H	1	11	-	-	-	-	-	-	
E8	RET	PE	1	5/11	-	-	-	-	-	-	
E9	JP	(HL)	1	4	-	-	-	-	-	-	
EA ppqq	JP	PE,qqpp	3	1Ø	-	-	-	-	-	-	
EB	EX	DE,HL	1	4	-	-	-	-	-	-	
EC ppqq	CALL	PE,qqpp	3	1Ø/17	-	-	-	-	-	-	
ED 4Ø	IN	B,(C)	2	12	S	Z	H	P	Ø	-	
ED 48	IN	C,(C)	2	12	S	Z	H	P	Ø	-	
ED 5Ø	IN	D,(C)	2	12	S	Z	H	P	Ø	-	
ED 58	IN	E,(C)	2	12	S	Z	H	P	Ø	-	
ED 6Ø	IN	H,(C)	2	12	S	Z	H	P	Ø	-	
ED 68	IN	L,(C)	2	12	S	Z	H	P	Ø	-	
ED 78	IN	A,(C)	2	12	S	Z	H	P	Ø	-	
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	2Ø	-	-	-	-	-	-	
ED 53 ppqq	LD	(qqpp),DE	4	2Ø	-	-	-	-	-	-	
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	1Ø	S	Z	H	P/V	N	C	
F2 ppqq	JP	P,qqpp	3	1Ø	-	-	-	-	-	-	
F3	DI		1	4	-	-	-	-	-	-	
F4 ppqq	CALL	P,qqpp	3	1Ø/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	1Ø	-	-	-	-	-	-	
FB	EI		1	4	-	-	-	-	-	-	
FC ppqq	CALL	M,qqpp	3	1Ø/17	-	-	-	-	-	-	
FD Ø9	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	2Ø	-	-	-	-	-	-	
FD 23	INC	IY	2	1Ø	-	-	-	-	-	-	
FD 2A ppqq	LD	IY,(qqpp)	4	2Ø	-	-	-	-	-	-	
FD 2B	DEC	IY	2	1Ø	-	-	-	-	-	-	
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 7Ød	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 AEd	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 BEd	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	-	-	-	-	-	-
FDCBdBE	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	1Ø	-	-	-	-	-	-
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,BC
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        TESTO:      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

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
08                  EX      AF,AF'
EB                  EX      DE,HL
```

```
76                      HALT

ED46        INTER:      IM      0
ED56                    IM      1
ED5E                    IM      2

ED78        LEIPOR:     IN      A,(C)
DB30                    IN      A,(DIS)
ED40                    IN      B,(C)
ED48                    IN      C,(C)
ED50                    IN      D,(C)
ED58                    IN      E,(C)
ED60                    IN      H,(C)
ED68                    IN      L,(C)

34          INCR:       INC     (HL)
DD343E                  INC     (IX+3EH)
FD3418                  INC     (IY+01BH)
3C                      INC     A
04                      INC     B
0C                      INC     C
14                      INC     D
1C                      INC     E
24                      INC     H
2C                      INC     L

03          INCR16:     INC     BC
13                      INC     DE
23                      INC     HL
DD23                    INC     IX
FD23                    INC     IY
33                      INC     SP

EDAA        INCGRP:     IND
EDBA                    INDR
EDA2                    INI
EDB2                    INIR

E9          JUMP:       JP      (HL)
DDE9                    JP      (IX)
FDE9                    JP      (IY)
C3B902'                 JP      JUMP
DAB902'                 JP      C,JUMP
FAB902'                 JP      M,JUMP
D2B902'                 JP      NC,JUMP
C2B902'                 JP      NZ,JUMP
F2B902'                 JP      P,JUMP
EAB902'                 JP      PE,JUMP
E2B902'                 JP      PO,JUMP
CAB902'                 JP      Z,JUMP

1840        JUPREL:     JR      66
38D3                    JR      C,-43
30FA                    JR      NC,JUPREL-$
20F8                    JR      NZ,JUPREL-$
282E                    JR      Z,DIS

02          CARREG:     LD      (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
50                      LD      D,B
51                      LD      D,C
52                      LD      D,D
53                      LD      D,E
54                      LD      D,H
55                      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
60                      LD      H,B
61                      LD      H,C
62                      LD      H,D
63                      LD      H,E
64                      LD      H,H
65                      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
ED61                OUT     (C),H
ED69                OUT     (C),L
D330                OUT     (DIS),A

ED8B        SAIGRP: OTDR
EDB3                OTIR
EDAB                OUTD
EDA3                OUTI

F1          CARPIH: POP     AF
C1                  POP     BC
D1                  POP     DE
E1                  POP     HL
DDE1                POP     IX
FDE1                POP     IY

F5          DESPIH: PUSH    AF
C5                  PUSH    BC
D5                  PUSH    DE
E5                  PUSH    HL
DDE5                PUSH    IX
FDE5                PUSH    IY

CB86        RESO:   RES     0,(HL)
DDCB0586            RES     0,(IX+IND)
FDCB0586            RES     0,(IY+IND)
CB87                RES     0,A
CB80                RES     0,B
CB81                RES     0,C
CB82                RES     0,D
CB83                RES     0,E
CB84                RES     0,H
CB85                RES     0,L

CB8E        RES1:   RES     1,(HL)
DDCB058E            RES     1,(IX+IND)
FDCB058E            RES     1,(IY+IND)
CB8F                RES     1,A
CB88                RES     1,B
CB89                RES     1,C
CB8A                RES     1,D
CB8B                RES     1,E
CB8C                RES     1,H
CB8D                RES     1,L

CB96        RES2:   RES     2,(HL)
DDCB0596            RES     2,(IX+IND)
FDCB0596            RES     2,(IY+IND)
CB97                RES     2,A
CB97                RES     2,A
CB90                RES     2,B
CB91                RES     2,C
CB92                RES     2,D
CB93                RES     2,E
CB94                RES     2,H
CB95                RES     2,L
```

```
CB9E        RES3:   RES     3,(HL)
DDCB059E            RES     3,(IX+IND)
FDCB059E            RES     3,(IY+IND)
CB9F                RES     3,A
CB98                RES     3,B
CB99                RES     3,C
CB9A                RES     3,D
CB9B                RES     3,E
CB9C                RES     3,H
CB9D                RES     3,L

CBA6        RES4:   RES     4,(HL)
DDCB05A6            RES     4,(IX+IND)
FDCB05A6            RES     4,(IY+IND)
CBA7                RES     4,A
CBA0                RES     4,B
CBA1                RES     4,C
CBA2                RES     4,D
CBA3                RES     4,E
CBA4                RES     4,H
CBA5                RES     4,L

CBAE        RES5:   RES     5,(HL)
DDCB05AE            RES     5,(IX+IND)
FDCB05AE            RES     5,(IY+IND)
CBAF                RES     5,A
CBA8                RES     5,B
CBA9                RES     5,C
CBAA                RES     5,D
CBAB                RES     5,E
CBAC                RES     5,H
CBAD                RES     5,L

CBB6        RES6:   RES     6,(HL)
DDCB05B6            RES     6,(IX+IND)
FDCB05B6            RES     6,(IY+IND)
CBB7                RES     6,A
CBB0                RES     6,B
CBB1                RES     6,C
CBB2                RES     6,D
CBB3                RES     6,E
CBB4                RES     6,H
CBB5                RES     6,L

CBBE        RES7:   RES     7,(HL)
DDCB05BE            RES     7,(IX+IND)
FDCB05BE            RES     7,(IY+IND)
CBBF                RES     7,A
CBB8                RES     7,B
CBB9                RES     7,C
CBBA                RES     7,D
CBBB                RES     7,E
CBBC                RES     7,H
CBBD                RES     7,L

C9          RETOR:  RET
D8                  RET     C
F8                  RET     M
```

```
D0                      RET     NC
C0                      RET     NZ
F0                      RET     P
E8                      RET     PE
E0                      RET     PO
C8                      RET     Z

ED4D        RETINT:     RETI
ED45                    RETN

CB16        ROTL:       RL      (HL)
DDCB0516                RL      (IX+IND)
FDCB0516                RL      (IY+IND)
CB17                    RL      A
CB10                    RL      B
CB11                    RL      C
CB12                    RL      D
CB13                    RL      E
CB14                    RL      H
CB15                    RL      L

17                      RLA

CB06        ROTLC:      RLC     (HL)
DDCB0506                RLC     (IX+IND)
FDCB0506                RLC     (IY+IND)
CB07                    RLC     A
CB00                    RLC     B
CB01                    RLC     C
CB02                    RLC     D
CB03                    RLC     E
CB04                    RLC     H
CB05                    RLC     L

07                      RLCA

ED6F                    RLD

CB1E        ROLR:       RR      (HL)
DDCB051E                RR      (IX+IND)
FDCB051E                RR      (IY+IND)
CB1F                    RR      A
CB18                    RR      B
CB19                    RR      C
CB1A                    RR      D
CB1B                    RR      E
CB1C                    RR      H
CB1D                    RR      L

1F                      RRA

CB0E        ROLRC:      RRC     (HL)
DDCB050E                RRC     (IX+IND)
FDCB050E                RRC     (IY+IND)
CB0F                    RRC     A
CB08                    RRC     B
CB09                    RRC     C
CB0A                    RRC     D
```

```
CB0B                RRC     E
CB0C                RRC     H
CB0D                RRC     L

0F                  RRCA

ED67                RRD

C7          RESTAR: RST     0
CF                  RST     08H
D7                  RST     10H
DF                  RST     18H
E7                  RST     20H
EF                  RST     28H
F7                  RST     30H
FF                  RST     38H

9E          SUBCY:  SBC     A,(HL)
DD9E05              SBC     A,(IX+IND)
FD9E05              SBC     A,(IY+IND)
9F                  SBC     A,A
98                  SBC     A,B
99                  SBC     A,C
9A                  SBC     A,D
9B                  SBC     A,E
9C                  SBC     A,H
9D                  SBC     A,L
DE20                SBC     A,DADO

ED42        SUCY16: SBC     HL,BC
ED52                SBC     HL,DE
ED62                SBC     HL,HL
ED72                SBC     HL,SP

37                  SCF

CBC6        SET0:   SET     0,(HL)
DDCB05C6            SET     0,(IX+IND)
FDCB05C6            SET     0,(IY+IND)
CBC7                SET     0,A
CBC0                SET     0,B
CBC1                SET     0,C
CBC2                SET     0,D
CBC3                SET     0,E
CBC4                SET     0,H
CBC5                SET     0,L

CBCE        SET1:   SET     1,(HL)
DDCB05CE            SET     1,(IX+IND)
FDCB05CE            SET     1,(IY+IND)
CBCF                SET     1,A
CBC8                SET     1,B
CBC9                SET     1,C
CBCA                SET     1,D
CBCB                SET     1,E
CBCC                SET     1,H
CBCD                SET     1,L
```

```
CBD6          SET2:   SET     2,(HL)
DDCB05D6              SET     2,(IX+IND)
FDCB05D6              SET     2,(IY+IND)
CBD7                  SET     2,A
CBD0                  SET     2,B
CBD1                  SET     2,C
CBD2                  SET     2,D
CBD3                  SET     2,E
CBD4                  SET     2,H
CBD5                  SET     2,L

CBDE          SET3:   SET     3,(HL)
DDCB05DE              SET     3,(IX+IND)
DDCB05DE              SET     3,(IX+IND)
CBDF                  SET     3,A
CBD8                  SET     3,B
CBD9                  SET     3,C
CBDA                  SET     3,D
CBDB                  SET     3,E
CBDC                  SET     3,H
CBDD                  SET     3,L

CBE6          SET4:   SET     4,(HL)
DDCB05E6              SET     4,(IX+IND)
FDCB05E6              SET     4,(IY+IND)
CBE7                  SET     4,A
CBE0                  SET     4,B
CBE1                  SET     4,C
CBE2                  SET     4,D
CBE3                  SET     4,E
CBE4                  SET     4,H
CBE5                  SET     4,L

CBEE          SET5:   SET     5,(HL)
DDCB05EE              SET     5,(IX+IND)
FDCB05EE              SET     5,(IY+IND)
CBEF                  SET     5,A
CBE8                  SET     5,B
CBE9                  SET     5,C
CBEA                  SET     5,D
CBEB                  SET     5,E
CBEC                  SET     5,H
CBED                  SET     5,L

CBF6          SET6:   SET     6,(HL)
DDCB05F6              SET     6,(IX+IND)
FDCB05F6              SET     6,(IY+IND)
CBF7                  SET     6,A
CBF0                  SET     6,B
CBF1                  SET     6,C
CBF2                  SET     6,D
CBF3                  SET     6,E
CBF4                  SET     6,H
CBF5                  SET     6,L

CBFE          SET7:   SET     7,(HL)
DDCB05FE              SET     7,(IX+IND)
FDCB05FE              SET     7,(IY+IND)
```

```
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
```

```
A8              XOR     B
A9              XOR     C
AA              XOR     D
AB              XOR     E
AC              XOR     H
AD              XOR     L
EE20            XOR     DADO
```

APÊNDICE H

CORRESPONDÊNCIA DOS MNEMÔNICOS DO MICROPROCESSADOR 8080 COM O Z-80

8080		Z-80	
ACI	DADO	ADC	A,DADO
ADC	M	ADC	A,(HL)
ADC	REG	ADC	A,REG
ADD	M	ADD	A,(HL)
ADD	REG	ADD	A,REG
ADI	DADO	ADD	A,DADO
ANA	REG	AND	REG
ANA	M	AND	(HL)
ANI	DADO	AND	=DADO
CALL	ENDER	CALL	ENDER
CC	ENDER	CALL	C,ENDER
CM	ENDER	CALL	M,ENDER
CMA		CPL	
CMC		CCF	
CMP	M	CP	(HL)
CMP	REG	CP	REG
CNC	ENDER	CALL	NC,ENDER
CNZ	ENDER	CALL	NZ,ENDER
CP	ENDER	CALL	P,ENDER
CPE	ENDER	CALL	PE,ENDER
CPI	DADO	CP	DADO
CPO	ENDER	CP	PO,ENDER
CZ	ENDER	CP	Z,ENDER
DAA		DAA	
DAD	SS	ADD	HL,SS
DCR	M	DEC	(HL)
DCR	REG	DCR	REG
DCX	SS	DEC	SS
DI		DI	
EI		EI	
HLT		HALT	
IN	ENDER	IN	A,(ENDER)

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	PE

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

ÍNDICE DE FIGURAS

BIBLIOGRAFIA

Livros

ALLEN e BELTON, *Microcomputer System Software Linguagem*, 1ª ed., New York, Electrical End. Electronics Engineers, Inc, 1980.

ANDREWS, Michael, *Programming Microprocessor Interfaces For Control and Instrumentation*, 1ª ed., New Jersey, Prentice-Hall, Inc, 1982.

BARDEN, Willian Jr, *Z80 Microcomputer Design Projects*, 1ª ed., Indianapolis, Indiana, Howard W.Sams & Co. Inc, 1980.

CAPECE, Raymond P., *Microprocessors and Microcomputers*, 1ª ed., New York, McGraw-Hill Publications Co., 1982.

LEVENTHAL, Lance A., *Z80 Assembly Language Programming*, 1ª ed., Berkeley, California, Osborne/McGraw-Hill, 1979.

LEVENTHAL, Lance A., *Z80 Assembly Language Subroutines*, 1ª ed., Berkeley, California, Osborne/McGraw-Hill, 1983.

MILLER, Alan R., *8080/Z80 Assembly Language*, 1ª ed., New York, John Wiley & Sons Inc. Publishers, 1980.

NICHOLS, Elisabeth A., *Programación Del Microprocesador Z-80*, 1ª ed., Barcelona, Boixareu Editores, 1981.

OSBORNE, Adan, *Z80 Programming For Logic Design*, 1ª ed., Berkeley, California, Osborne/McGraw-Hill, 1978.

VISCONTI, Antonio C.J.F., *Microprocessador 8080 e 8085*, 3ª ed., São Paulo, Livros Érica Editora Ltda, 1983.

Manual

CARR, Joseph J., *Z-80 Users Manual*, 1ª ed., Reston, Virginia, Reston Publishing Company, Inc., 1980.

Z80 Microcomputer System Programming Manual Mostek (MD 78515).

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 $er ou Não $er
Lence	- Física - Vol. I
Filkauskas/Guedes	- O Plástico.