Microprocessador 8085

Contents

1. Arquitetura de um microprocessador de 8 bits:

A forma de operar, assim como a arquitetura de todos os microprocessadores de 8 bits, é muito similar. Apesar disso, é difícil encontrar um método didático que desenvolva de maneira geral a teoria necessária para compreender o funcionamento deste tipo de dispositivos.

Por este motivo consideramos que a melhor forma de abordar o estudo dos microprocessadores consiste em analisar um de forma mais ampla e extrapolar os conhecimentos adquiridos quando se deseje compreender a constituição e forma de operar de cada um dos dispositivos análogos ao examinado.

O microprocessador 8085 é um dispositivo que opera com uma *palavra de 8 bits* sendo capaz de endereçar, com suas *16 linhas*, até 64 K *posições de memória*. Fabricado em tecnologia NMOS, está constituído por 6.200 transistores. É um circuito integrado com encapsulamento dual in line de 40 pinos.

Para realizar as operações que lhe sejam encomendadas pelo programa de controle o 8085 é capaz de decodificar internamente 74 tipos de instruções diferentes. Assim, podemos dizer que o conjunto de instruções é constituído pelo número indicado.

A tensão de alimentação é única de 5 volts c.c., que são aplicados entre os terminais 40 (+) e 20 (-). Como todo microprocessador que toma parte de um sistema digital programável, necessita um clock ou oscilador para sincronizar as operações que realiza no sistema digital e trabalha com freqüência de 3.125 MHz.

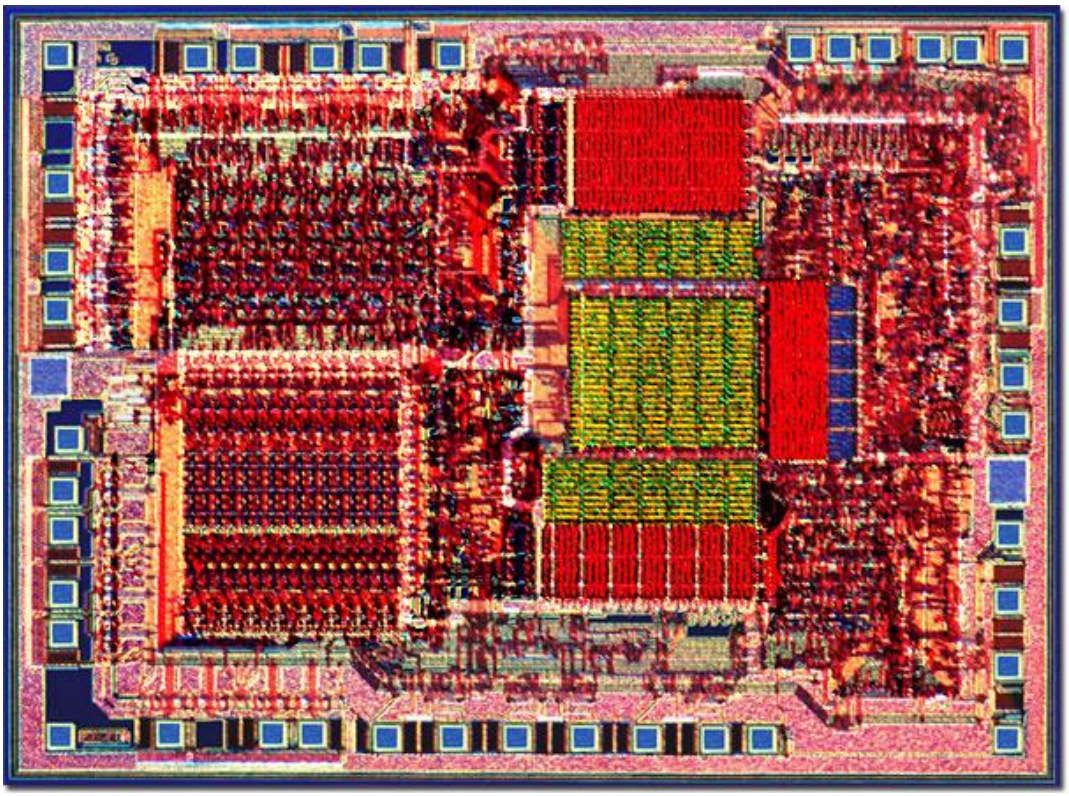


Figura 1: Core do 8085

Um nicho de aplicação para o versão rad-hard[1] do 8085 foi de processadores de dados on-board para instrumentos utilizados pela NASA e ESA em várias missões espaciais na década de 1990 e início de 2000, incluindo CRRES, Polar, FAST, Cluster, Hessi, Sojourner (rover) e Themis. A empresa suíça SAIA usou a 8085 e 8085-2 como os processadores de sua linha PCA1 de controladores lógicos programáveis, durante a década de 1980.

Figura 2: Encapsulamento do 8085

[1] Radiation Hardening: (endurecimento contra irradiação) é um sistema de fabricação aplicado a um dispositivo para protegê-lo contra a exposição a um ambiente contendo alta radiação gama e de nêutrons.

2. *Nomenclatura dos terminais do 8085:*

Figura 3: Pinagem do 8085

- ✓ **A8 – A15.** Estes terminais correspondem as saídas dos 8 bits mais significativos do bus de endereços do sistema em que está integrado o microprocessador.
- ✓ **AD0 – AD7.** É um conjunto de terminais de entrada/saída que realizam uma dupla função. Mediante um multiplexador, o microprocessador se comunica em primeiro lugar com os 8 bits menos significativos do bus de endereços e posteriormente com o bus bidirecional de dados.
- ✓ **ALE.** Address Latch Enable (Ativação do Latch biestável de endereços). É uma saída que determina se a informação presente nas linhas AD0 –AD7 corresponde a um dado ou a um endereço. É utilizada para disparar um registrador de 8 bits que memoriza a parte baixa de um endereço. Se estiver em nível lógico 1 carrega um endereço, se estiver em 0 carrega dados.
- ✓ **S_0 – S_1.** São saídas que informam o estado do bus de dados. Indicam o tipo de ciclo de máquina que o microprocessador realiza: busca, leitura, escrita ou parada.

- ✓ **RD. Read**. Ativa em nível baixo. Saída que é utilizada para indicar que a posição de memória ou dispositivo de E/S selecionado quer ser lido pela CPU e, além disso, que o bus de dados está disponível para realizar a transferência.
- ✓ **WR**. **Write.** Ativa a nível baixo. Saída que é utilizada para indicar que a informação presente no bus de dados pode ser escrita na posição de memória ou dispositivo de E/S endereçado.
- ✓ **READY**. Entrada que indica ao microprocessador (quando está em nível alto) que existem dados válidos sobre o bus de dados procedentes de memória ou dispositivos de E/S. Quando está em nível baixo, o microprocessador entra em um estado de espera. Este sinal é utiliza quando no sistema existe algum dispositivo que opera com um tempo maior que o marcado pelo clock do microprocessador.
- ✓ **HOLD**. Entrada que indica que outro processador ou controlador solicita o uso dos barramentos de endereço e dados. O microprocessador abandona o uso dos barramentos uma vez finalizado o ciclo de máquina em curso, colocando em estado de alta impedância as seguintes entradas e saídas: A8 – A15, AD0 – AD7, RD, WR e IO/M.
- ✓ **HLDA.** Sinal de saída gerado pelo microprocessador como resposta a entrada HOLD para indicar que os barramentos e demais terminais estão em estado de alta impedância.
- ✓ **INTR.** Interrupt Request (requisição de interrupção). Este sinal de entrada é utilizado pelos dispositivos externos, quando necessitam serem atendidos sem demora pelo microprocessador, interrompendo a seqüência do programa. Pode ocorrer que a interrupção não seja aceita. A aceitação ou não-aceitação da interrupção é controlada pelo programa principal.
- ✓ **INTA.** Ativa em nível baixo. Interrupt Acknowledge (reconhecimento da interrupção). É um sinal de saída. É utilizado para ativar (em lugar de RD) o dispositivo que gera a interrupção uma vez que tenha sido aceita.
- ✓ **TRAP.** É uma entrada de interrupção. Tem prioridade.
- ✓ **RESET IN.** Ativa em nível baixo. É uma entrada que põe em zero um registrador do microprocessador denominado Contador de Programa, os biestáveis que validam as interrupções e o sinal HLDA. São utilizados para indicar o programa.

- ✓ **RESET OUT.** É uma saída que indica o estado de RESET do microprocessador. Pode ser utilizada para por em zero os demais componentes do sistema.
- ✓ **IO/M.** É ativa em nível baixo. É uma saída que indica se a leitura ou escrita é realizada sobre a memória ou sobre dispositivos de E/S.
- ✓ **X_1 –X_2.** Estes terminais de entrada são utilizados para conectar um cristal de quartzo ou uma rede LC ou RC para estabilizar o gerador de clock interno. Também é possível conectar um clock externo.
- ✓ **SID e SOD.** Entrada e saída respectivamente, de dados em série.
- ✓ **CLK OUT.** Saída do sinal de clock do microprocessador para utilizá-lo como clock do sistema.

3. *Arquitetura do microprocessador*

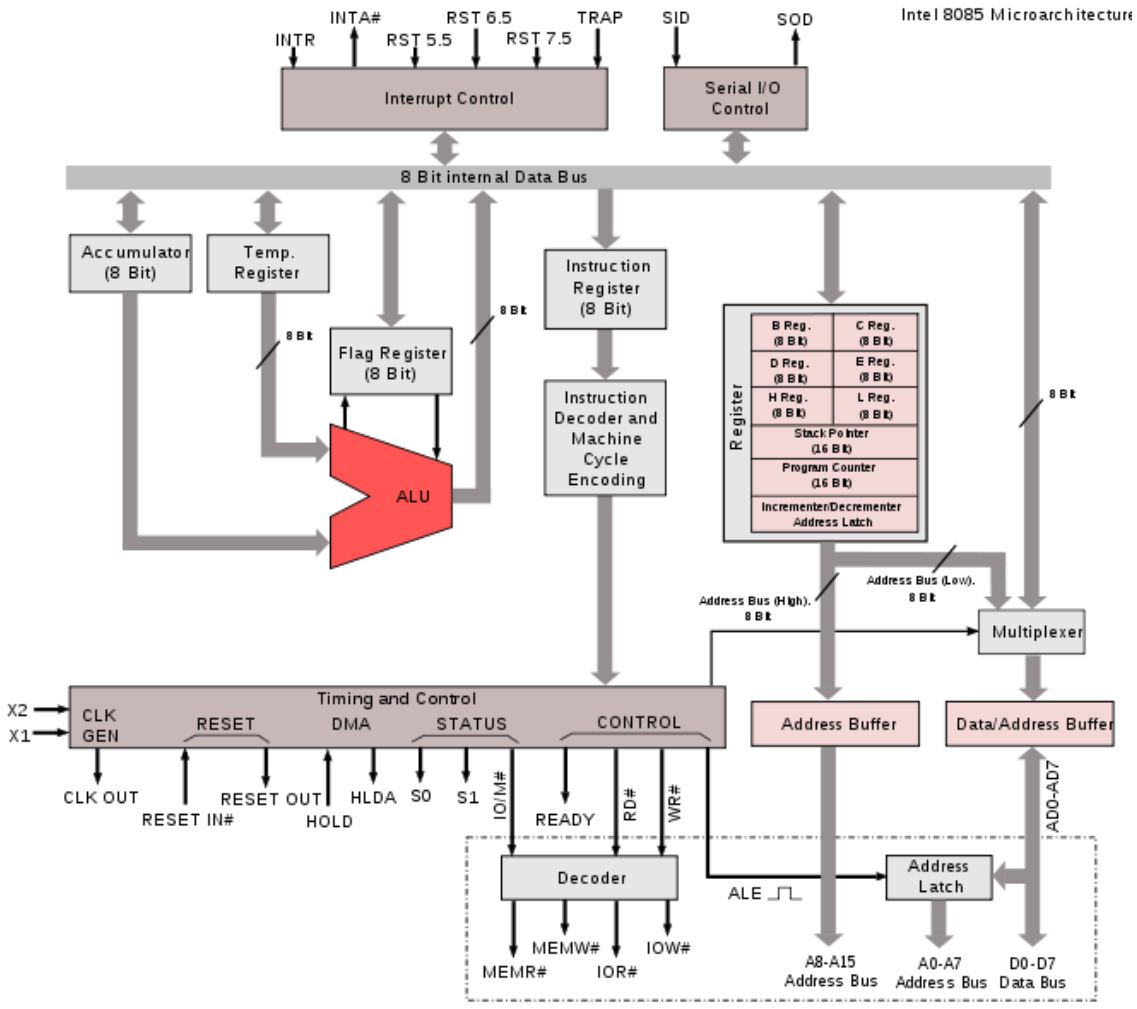


Figura 4: Arquitetura interna do 8085

O microprocessador 8085 é constituído basicamente por três grandes blocos: uma unidade de controle, um conjunto de registradores e uma unidade aritmético-lógica. É difícil separar uns elementos dos outros, devido a inter-relação que existe entre todos eles; não obstante, para facilitar sua compreensão, vamos examiná-los separadamente:

4.1. UNIDADE DE CONTROLE

A unidade de controle é um bloco de lógica cabeada dentro do CI. Esta parte do sistema controla e sincroniza as transferências de dados e as operações que são realizam com eles. A unidade de controle opera com o sinal de clock mestre do microprocessador e com os sinais de controle de outros elementos do sistema, com o objetivo de interromper a seqüência do programa e o bus de dados, através do decodificador de instruções com a informação já interpretada, para que a unidade possa responder com os sinais adequados.

Algumas saídas da unidade de controle são destinadas aos elementos externos do microprocessador (memória e dispositivos de E/S) para indicar, por exemplo, que a operação em curso se trata de uma leitura ou de una escrita, outras aos registradores internos da própria CPU.

Resumindo, a unidade de controle regula a função básica do microprocessador e consiste na busca e posterior execução de instruções. Esta operação é cíclica a não ser que seja interrompida mediante uma instrução de parada (HALT). No estado de busca se transfere uma instrução desde a memória até o microprocessador e no estado de execução se realiza esta operação.

Registradores internos do microprocessador. **O** 8085 têm uma serie de registradores de uso geral denominados B, C, D, E, H e L de 8 bits cada um, porém podem operar em pares (são agrupados para este fim B com C, D com E e H com L). São utilizados para transformações internas e assim obter uma maior flexibilidade e rapidez de operação. Existem outros registradores de uso especial. Estes são: o *Contador de Programa* (CP),

de 16 bits; o *Stack Pointer* ou ponteiro da pilha (SP), também de 16 bits; os *Registradores Temporários W-Z*, de 8 bits cada um, porém que podem operar unidos; o *Registrador de endereços*, de 16 bits, cujas saídas são os pinos do bus de endereços do microprocessador e o *Registrador de Instrução* (RI), de 8 bits.

O ***Contador de Programa*** é um registrador que memoriza o endereço da próxima instrução a ser executar ou o endereço de parte de uma instrução formada por mais de um byte. As instruções do 8085 e de todos os microprocessadores de 8 bits podem ser de 1, 2 ou 3 bytes, ordenados de forma sucessiva na memória. A unidade de controle incrementa automaticamente em um o conteúdo do CP a cada vez que termina um ciclo de busca, salvo quando aparece a instrução HALT (parada). O sinal de RESET, que é uma das entradas da unidade de controle, coloca o CP em zero e inicializa a execução do programa. Para obter a primeira instrução do programa, o endereço contido no CP [0000 (HEX)] é colocado no bus de endereços do sistema através do registrador de endereços.

A unidade de controle gera um sinal de leitura na memória. O dado endereçado é transferido ao microprocessador através do bus de dados e chega ao registrador de instrução. O primeiro byte de uma instrução é sempre o Código de Operação (CO). Este código de operação é interpretado pelo decodificador de instruções e passa para a unidade de controle, que gera uma seqüência de micro operações para que a instrução seja executada. As vezes, para que uma instrução possa ser executada, é necessária mais informação (segundo e terceiro byte de algumas instruções). Esta informação se encontra armazenada na memória na continuação do CO. Quando a unidade de controle, uma vez decodificado o CO, indica que a instrução requer bytes adicionais, o CP é incrementado em uma unidade e volta seu conteúdo novamente sobre o bus de endereços. O conteúdo da posição endereçada passa para um dos registradores temporários W-Z através do bus de dados. Estes registradores são utilizados, como seu nome indica, para armazenar temporariamente os bytes 2 e 3 (se existirem) de uma instrução, até que sejam passados a outros registradores internos ou externos do microprocessador. Pode ocorrer que os bytes segundo e terceiro da instrução sejam um endereço de memória ou um endereço de seleção de algum dispositivo de E/S.

Neste caso os 16 bits são armazenados temporariamente nos registradores W-Z. Na seqüência este conteúdo é colocado sobre CP para acessar o dispositivo correspondente.

As instruções nem sempre são executadas na ordem em que estão escritas na memória. Como já foi dito existe a possibilidade de efetuar saltos para seções do programa denominadas sub-rotinas. Para tal fim os sistemas com microprocessador utilizam uma pilha ou uma área reservada da memora principal. Quando, mediante uma instrução, se produz uma chamada para sub-rotina, o conteúdo de CP é deposita na pilha, ou seja, na posição de memória apontada pelo SP e a anterior (já que o conteúdo de CP é de 16 bits) até que seja executada a dita sub-rotina. Uma vez executada, o CP volta a ser carregado com o valor armazenado na pilha de memória para prosseguir com o programa principal. O motivo pelo qual é necessário uma pilha e não exclusivamente um registrador é a possibilidade de que existam sub-rotinas "aninhadas", ou seja, que desde o programa principal se chame uma sub-rotina e desta uma segunda e assim sucessivamente. Neste caso é necessário ir armazenando todas os endereços de início para posteriormente retornar.

4.2. UNIDADE ARITMÉTICO-LÓGICA.

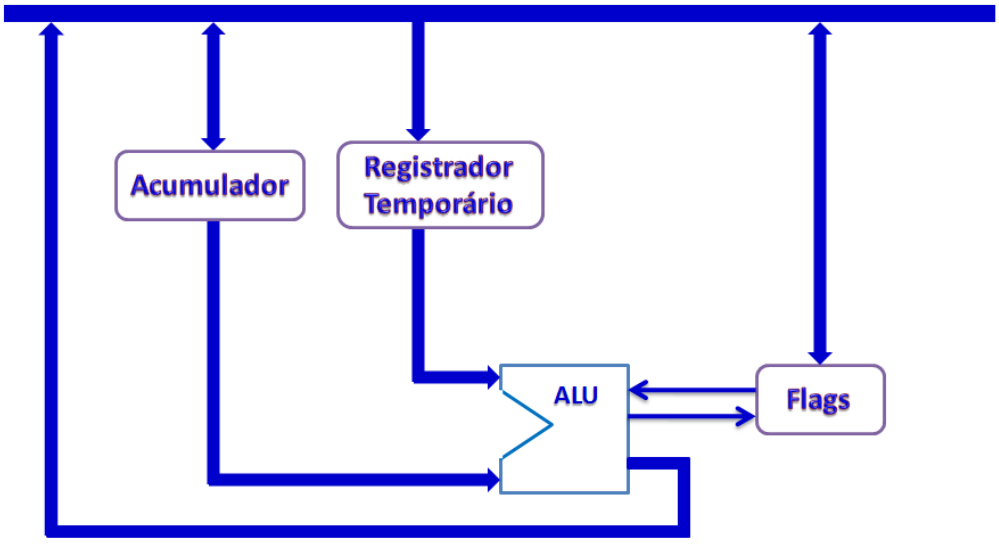


Figura 5: Arquitetura interna do 8085

A unidade de processamento completa do 8085 é formada por uma ALU (Unidade aritmético-lógica), que é um circuito combinacional capaz de realizar operações de soma e subtração, assim como operações lógicas, um registrador de 8 bits chamado Acumulador (A) e um conjunto de cinco biestáveis ou flags denominados registradores de estado, que oferecem informação relativa ao resultado das operações aritméticas ou lógicas que são realizadas.

Quando é realizada uma operação, o acumulador contém um operando e uno dos registradores temporários W-Z o outro. O resultado da operação é colocado no acumulador substituindo o dado ou operando que continha anteriormente.

A denominação e situação de cada flag é mostrada na **figura X**. CY é o sinalizador de "arraste" e seu conteúdo passa a ser 1 quando é produzido um "carry" ao ser realizada uma soma. AC é o sinalizador de "arraste" auxiliar. Utiliza-se quando são realizadas operações em BCD. Seu valor será 1 quando for produzido "arraste" ao efetuar a soma dos bits dos operandos que ocupam o quarto lugar partir da direita. Se for o flag de sinal o seu valor será 1 se o bit mais significativo do resultado for 1. Caso contrario, será zero. Z é o sinalizador de zero sendo colocado em 1 quando o resultado de uma operação tenha sido zero. Por último, P é o flag de paridade e seu valor será 1 quando o número de bits de uma palavra depositada no acumulador for par.

S	Z		AC		P		CY

Figura 6:

4. Código máquina e mnemônicos:

O conjunto de instruções de um microprocessador é constituído por todo o conjunto de expressões binárias que o decodificador de instruções é capaz de interpretar. É evidente que quanto maior for o número e mais variado o repertório maior será a

condição operacional do dispositivo e conseqüentemente, do sistema onde será integrado.

O microprocessador, como todo circuito digital, opera com uns e com zeros exclusivamente. Sem dúvida, o sistema de codificação binária é muito trabalhoso para ser utilizado como linguagem de programação, já que um dado ou palavra, em um circuito que utiliza um microprocessador como o 8085, é formado por 8 bits e um endereço por 16 bits. Além disso, tais expressões numéricas resultam muito extensas.

Por este motivo, os equipamentos mais elementares que são empregados para programar admitem para a escrita das instruções pelo menos a codificação hexadecimal. A tradução da linguagem binária, também denominado código máquina, é realizada com o correspondente decodificador hexadecimal/binário incorporado para o equipamento.

Logicamente este sistema oferece também grandes inconvenientes, tanto na fase de definição como na de depuração, quando o programa é complexo.

Com a finalidade de aproximar a escrita e leitura de programas para a linguagem natural, se decidiu expressar as instruções dos microprocessadores utilizando a abreviatura ou as siglas da palavra que define a operação que realiza. Devido a origem das empresas fabricantes deste tipo de dispositivos, estas palavras estão expressas em inglês. Assim, por exemplo, a abreviatura MOV, tem origem em MOVE e significa MOVER ou TRANSLADAR, é utiliza para transferência de dados entre registradores internos ou externos.

Estas expressões são denominadas **mnemônicos**, que significa lembrete, recordação, etc.. Portanto, as expressões do microprocessador 8085 são um conjunto de 74 mnemônicos compatíveis com as de outros dispositivos da INTEL, porém, desgraçadamente, diferentes, ainda que realizem a mesma operação, as de outros fabricantes.

Existem equipamentos de programação que admitem diretamente as instruções em forma de mnemônicos. Estes equipamentos dispõem de u programa denominado "**assembler"**, que traduz o programa escrito em linguagem montadora do sistema binário, que é o único código capaz de ser interpretado pela máquina.

Os equipamentos mais complexos, denominados genericamente sistemas de desenvolvimento, são capazes de traduzir em código máquina qualquer programa escrito em linguagens de alto nível, tais como Pascal, BASIC, etc. Estes equipamentos facilitam enormemente o desenvolvimento de software em um sistema digital.

5. Tipos de instruções:

O número de instruções que cada microprocessador pode interpretar e executar é diferente e específico de cada caso. Assim o 8085 foi projetado para admitir 74 instruções básicas, de tal forma que, se contemplarmos as variantes de algumas delas, este dispositivo estará preparado par realizar 246 operações distintas.
A instruções do 8085 podem ser classificadas em cinco grupos:

1. Transferência de dados entre registradores ou entre posições de memória e registradores.
2. Operações aritméticas.
3. Operações lógicas.
4. Bifurcações (salto).
5. Instruções de stack, E/S e controle de máquina.

6. Formato de instruções:

As instruções do 8085 e de outros microprocessadores semelhantes são constituídas por 1, 2 ou 3 bytes. Assim, pois existem instruções de 1, 2 e 3 bytes. Na **figura x** são mostrados os três bytes que podem formar uma instrução e o conteúdo de cada um deles.

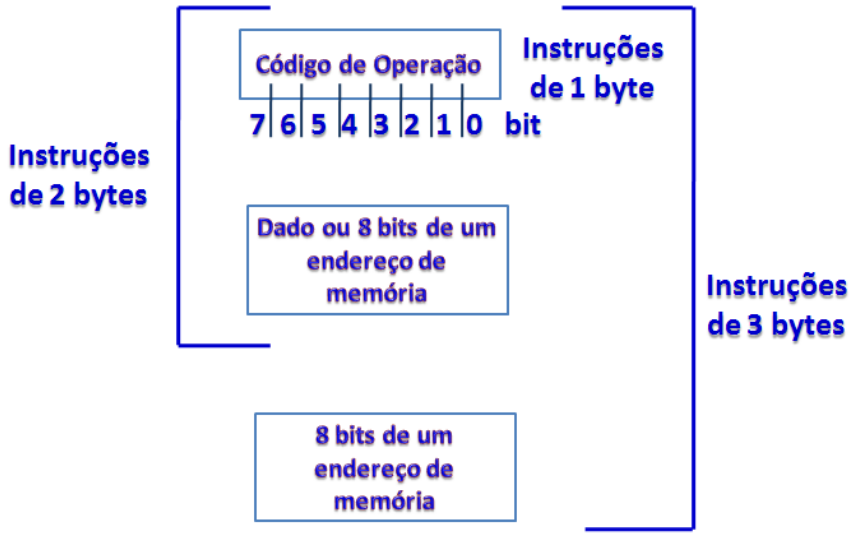


Figura 7: Formato das instruções

O primeiro byte sempre é utilizado para representar o CO de todas as instruções.

As instruções de 2 bytes são formadas pelo CO e dois bytes mais. O primeiro contém os 8 bits menos significativos de um endereço de memória e o outro os 8 bits mais significativos.

- ✓ Para os exercícios a seguir empregaremos as tabelas abaixo:

DDD ou SSS	Registrador
111	A
000	B
001	C
010	D
011	E
100	H
101	L

RP	Par de Registradores
00	BC
01	DE
10	HL
11	SP

Condição (XX)	CCC
NZ não zero (Z=0)	000
Z zero (Z=1)	001
NC não carry (CY =0)	010
C carry (CY=1)	011
PO paridade impar (P=0)	100
PE paridade par (P=1)	101
P positivo (S=0)	110
M negativo (S=1)	111

❖ **EXERCICIOS:**

1. Somar 2 números que se encontram nas posições de memória 006A e 006B e guardar o resultado na posição 0010. Escrever o programa em mnemônicos, hexadecimal e binário. Dados:

- LDA [ENDEREÇO]. Carrega o acumulador (A) com o conteúdo da posição de memória da qual se dá seu endereço.
- MOV B, A. O conteúdo de A passa para o registrador B
- ADD B. Soma ao acumulador o conteúdo do registrador B. O resultado fica no acumulador (A).
- STA [ENDEREÇO]. Guarda, armazena o conteúdo do acumulador na posição de memória da qual se dá seu endereço.

MNEMÔNICOS:

- ***LDA 006A***
- ***MOV B, A***
- ***LDA 006B***
- ***ADD B***
- ***STA 0010***

MNEMÔNICO	BINÁRIO	HEXADECIMAL
LDA	0011 1010	3 A
6 A	0110 1010	6 A
00	0000 0000	00
MOV B, A	0100 0111	4 7
LDA	0011 1011	3 A
6 B	0110 1011	6 B
00	0000 0000	0 0

ADD B	**1000 0000**	**8 0**
STA	**0011 0010**	**3 2**
10	**0001 0000**	**1 0**
00	**0000 0000**	**0 0**

2. Escrever um programa que carregue o registrador B com o dado 04, e carregue o registrador C com o dado 05. O programa deverá decrementar 4 vezes o conteúdo do registrador B e quando haouver feito isto decrementar 1 vez o registrador C. Dados:

- **MVI r, DADO**. O conteúdo da posição de memória apontada pelos registradores HL passa para o registrador r. O dado passa para o registrador r.
- **DCR r**. O conteúdo do registrador r é decrementado em uma unidade.

MNEMÔNICOS:

- ***MVI B, 04***
- ***MVI C, 05***
- ***DCR B***
- ***DCR B***
- ***DCR B***
- ***DCR B***
- ***DCR C***

MNEMÔNICO	BINÁRIO	HEXADECIMAL
MVI B	**0000 0110**	**0 6**
04	**0000 0100**	**0 4**
MVI C	**0000 1110**	**0 E**
0 5	**0000 0101**	**05**
DCR B	**0000 0101**	**0 5**
DCR B	**0000 0101**	**0 5**

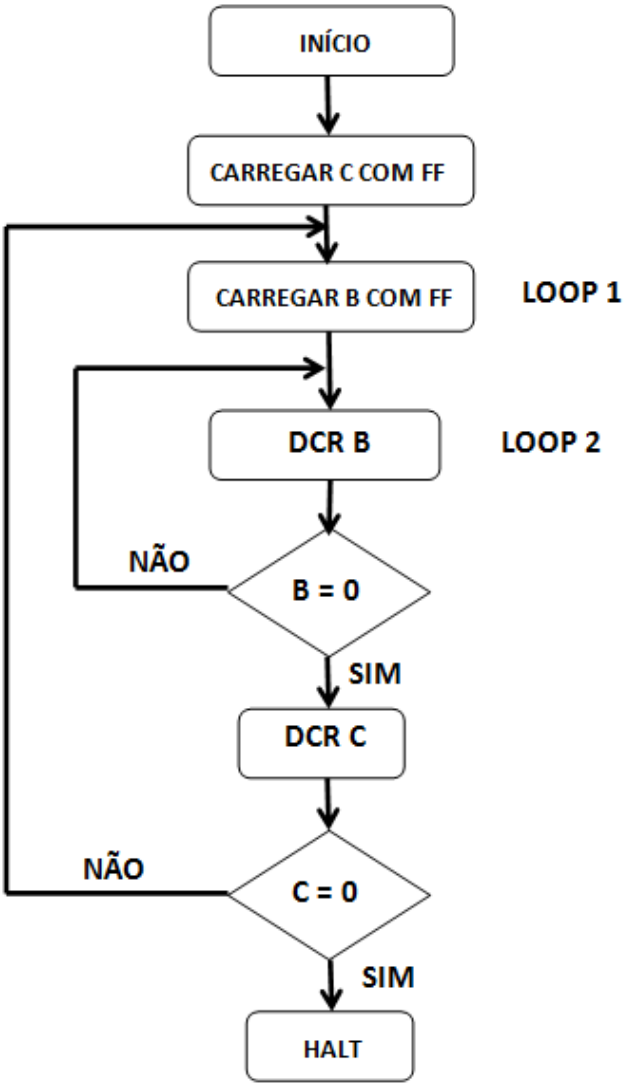
DCR B	**0000 0101**	**0 5**
DCR B	**0000 0101**	**0 5**
DCR C	**0000 1101**	**0 D**

3. Construir uma tabela com o programa anterior em hexadecimal e o endereço de memória onde se encontra cada instrução supondo que o programa se carrega a partir doa endereço 0000.

ENDEREÇO	BYTE
0000	**0 6**
0001	**0 4**
0002	**0 E**
0003	**0 5**
0004	**0 5**
0005	**0 5**
0006	**0 5**
0007	**0 5**
0008	**0 D**

4. Neste exercício introduziremos o conceito de salto condicional: Escrever um programa que carregue o registrador C com o dado FF carregue o registrador B com FF, decremente o registrador B até que seu conteúdo seja 0 e cada vez que isto ocorrer decremente o registrador C. Além disso, cada vez que seja decrementado o registrador C se deve carregar o registrador B com FF e decrementá-lo até 0. O programa terminará quando o registrador C chegue a 0. Dados:

- **JNZ [ENDEREÇO].** Se o resultado da última operação aritmético-lógica não for 0, saltar para o endereço de memória indicado.



MNEMÔNICO

- **MVI C, FF**
- **LOOP 1:MVI B, FF**
- **LOOP 2 DCR B**
- **JNZ LOOP 2**
- **DCR C**
- **JNZ LOOP 1**

MNEMÔNICO	BINÁRIO	ENDEREÇO	HEXADECIMAL
MVI C	0000 1110	0000	0 E
FF	1111 1111	0001	F F
MVI B	0000 0110	0002	0 6
FF	1111 1111	0003	F F
DCR B	0000 0101	0004	0 5
JNZ	1100 0010	0005	C 2
		0006	04
		0007	00
DCR C	0000 1101	0008	0 D
JNZ	1100 0010	0009	C 2
		000A	02
		000B	00

7. Seqüências e tempos do 8085.

A busca e execução de uma instrução se chama ciclo de instrução. O ciclo de instrução é formado por vários ciclos de máquina. A maior parte dos ciclos de máquina das instruções de um programa são referencias de leitura ou escrita para a memória ou para os dispositivos de E/S. Os diferentes tipos de ciclos de máquina no 8085 são:

1. Busca de CO.
2. Leitura da memória.
3. Escrita na memória.
4. Leitura de dispositivo de E/S.
5. Escrita em dispositivo de E/S
6. Resposta para interrupção.
7. Bus inativo.

- EXERCÍCIO. Quanto tempo demora em ser executada uma instrução MOV r_1, r_2 se o microprocessador trabalha a 4 MHz?

$$f = 4MHz = 4.10^6\ Hz$$
$$T = \frac{1}{f} = \frac{1}{4.10^6} = 0'25.10^{-6}\ S = 0'25\mu S$$
$$4\ \text{ciclos de reloj} \Rightarrow t = 4 \cdot 0'25 = 1\mu S$$

8. Modos de endereçamento:

As instruções do microprocessador 8085 fazem referencia aos dados, que explícita ou implicitamente contém, de diferentes maneiras.

Existem quatro maneiras de referenciar dados, denominados modos de endereçamento: **imediato, direto, por registradores e por registrador indireto.**

No endereçamento **imediato** a instrução contém os dados no byte ou os bytes seguintes do código de operação, por exemplo, **ADI 04.** O hexadecimal 04 (segundo byte da instrução) é somado ao acumulador.

A forma mais simples de endereçamento é a **direta**. Neste caso os bytes segundo e terceiro da instrução contém o endereço da posição de memória onde se encontra o dado, por exemplo, **LDA 003F**. O conteúdo da posição 003F passa para o acumulador.
No endereçamento **por registrador** a instrução especifica o registrador ou duplo registrador onde está o dado, por exemplo, **ADD B**. O conteúdo do registrador B é somado ao acumulador.

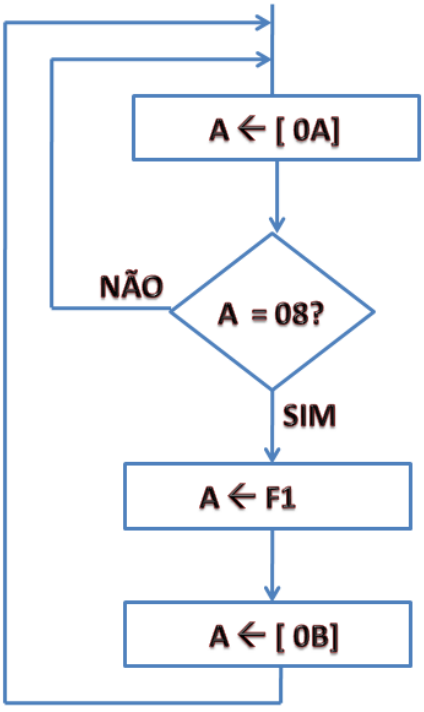
Por último, o modo de endereçamento **por registrador indireto** ou duplo registrador especificado na instrução contém o endereço de memória onde está o dado, por

exemplo, **LDAX B**. O conteúdo do endereço de memória, cujo endereço está no par de registradores B e C, se move para o acumulador.

- EXERCICIOS. Escrever um programa que leia a porta de entrada cujo endereço é 0A, verificar se o bit 3 do dado lido é 1 e o resto todo 0 e neste caso envie pela porta de saída cujo endereço é 0B o dado F1. A seqüência deverá ser repetida (ciclicamente).

  - Os endereços das portas de E/S são de 8 bits. Isto quer dizer que o 8085 pode endereçar 256 portas de E e 256 portas de saída.
  - IN [ENDEREÇO]. Le o conteúdo da porta endereçada.
  - OUT [ENDEREÇO]. Escreve na porta endereçada.
  - CPI DADO. Compara o conteúdo do acumulador com o dado.
  - JNZ. Salto condicional.
  - JMP. Salto incondicional.

7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0



INICIO: IN 0A; o conteúdo da porta 0A passa para o acumulador.

CPI 08; compara o acumulador com ol dado 08.

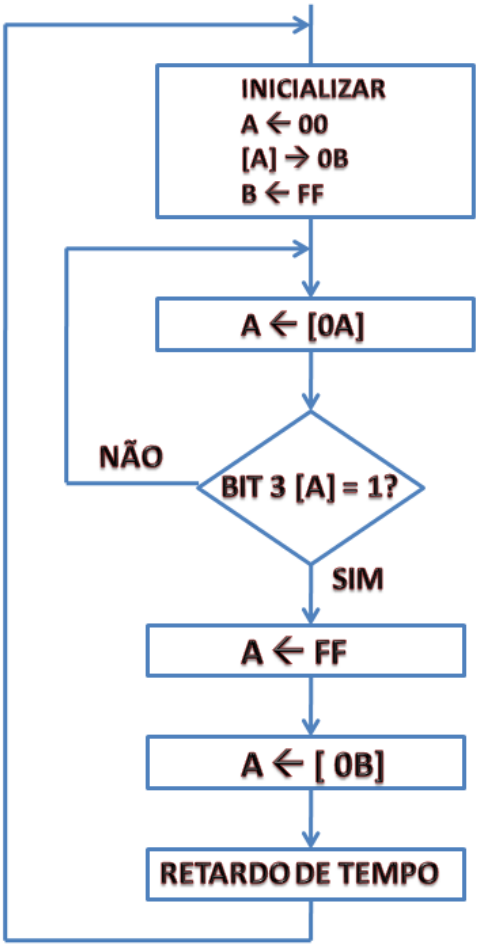
JNZ INICIO; salta se o bit de 0 não estiver ativado.

MVI A, F1; carrega o acumulador com F1.

OUT 0B; envia pela porta 0B o dado do acumulador.

JMP INICIO; salto incondicional para o endereço INICIO.

- EXERCICIO. Escrever um programa em que o bit 3 da porta de entrada 0A esteja em 1 e envie pela porta de saída 0B o dado FF e o mantenha um certo tempo nesta situação, transcorrido o tempo a situação da porta 0B deve voltar a 00.



INICIO: MVI A, 00; carrega o acumulador com 00

OUT 0B ; enviar o conteúdo do acumulador pela porta de saída 0B

MVI B, FF ; carrega o registrador B com FF

LOOP 1: IN 0A ; carrega o acumulador com o dado que existe na porta 0A

```
                ANI 08 ; testa se o bit 3 do acumulador está em 1.
                JZ  LOOP 1 ; saltar se o resultado for 0
                MVI A , FF ; carregar o acumulador com FF
                OUT 0B ; enviar o dado do acumulador para 0B
TEMPO:          DCR B ; decrementar B
                JNZ TEMPO ; saltar para tempo se o resultado não for 0
                JMP INICIO ; regressar ao estado inicial
```

Quanto tempo se mantém a porta 0B em FF? Sendo a freqüência de 4 MHz.

```
TEMPO:          DCR B (4)
                JNZ TEMPO (7/10)
                JMP INICIO (10)
INICIO: MVI A , 00 (7)
                OUT 0B  (10)
```

Nº de estados = (4 + 10) 254 + (4 + 7)+ 10 + 7 + 10 = 3594 estados

T=0,25μS ➔ t = 898,5 μS

NOTA: são 254 porque a última vez não volta a subir para DCR B e então neste momento somente são produzidos 7 pulsos de clock. Por isso temos de somar 4 + 7.

❖ EXERCICIO. Como podemos aumentar de uma forma simples este tempo?

Sempre que não buscamos diferenças de tempos muito grandes inserimos a instrução NOP

Como exemplo veremos o efeito de incluir 2 NOP em cada loop.

NOP ➔ 5 estados

Adicionamos (5 + 5) 255 = 2550

t adicionado = 637,5 μS

9. Explicação de cada instrução

Quando me refiro aos registradores "r" (seja r1,r2 ou simplesmente r) estamos falando de qualquer registrador próprio do 8085, A (Acumulador), os registradores B,C,D,E,H,L de 8 bits respectivamente. Estes registradores podem ser agrupados por pares BC,DE,HL (rp) e assim formar registradores de 16 bits para endereçar uma posição de memória diretamente(16 BITS), sendo o par HL o mais importante por ter instruções que trabalham diretamente com ele.

INSTRUÇÃO	CÓDIGO	DESCRIÇÃO
MOV r1, r2	**01DDRRR**	Carrega o registrador r1 (DDD) com o conteúdo de r2 (RRR)
MOV r, M	**01DDD110**	Carrega o registrador r (DDD) com o conteúdo do endereço apontado no par HL.
MOV M, r	**01110RRR**	Carrega o endereço apontado por HL com o conteúdo do registrador r (RRR).
MVI r	**00DDD110**	Carrega o registrador r de modo imediato com o 2º Byte da instrução.
MVI M	**00110110**	Carrega o endereço apontado por HL com o 2º Byte da instrução.
LXI rp	**00DD0001**	Carrega o par de registradores rp (serão BC, DE, HL) com o 2º e 3º Byte da instrução.
LDA	**00111010**	Carrega o acumulador com o conteúdo do endereço apontado no 2º e 3º Bytes da instrução.
STA	**00110010**	Carrega a posição apontada pelo 2º e 3º Bytes da instrução com o conteúdo do acumulador.
LHLD	**00101010**	Carrega os registradores HL com o conteúdo da posição apontada pelo 2º e 3º Bytes (carrega L) e pela apontada pelo 4º e 5º Bytes (carrega H) da instrução.
SHLD	**00100010**	Armazena o conteúdo de HL na posição que indica o 2º e 3º Bytes e o seguinte respectivamente.
LDAX rp	**00RR1010**	Carrega o acumulador com o conteúdo do endereço que aponta o par de registradores

		rp (RR) somente para os registradores pares BC e DE.
XCHG	**11101011**	Faz intercâmbio de conteúdo do par HL pelo conteúdo do par DE.

Todas as instruções:

INSTRUÇÃO	CÓDIGO (HEX)	INSTRUÇÃO	CÓDIGO (HEX)
MOV A, A	7F	MOV B, A	47
MOV A, B	78	MOV B, B	40
MOV A, C	79	MOV B, C	41
MOV A, D	7A	MOV B, D	42
MOV A, E	7B	MOV B, E	43
MOV A, H	7C	MOV B, H	44
MOV A, L	7D	MOV B, L	45
MOV A, M	7E	MOV B, M	46
MOV C, A	4F	MOV D, A	57
MOV C, B	48	MOV D, B	50
MOV C, C	49	MOV D, C	51
MOV C, D	4A	MOV D, D	52
MOV C, E	4B	MOV D, E	53
MOV C, H	4C	MOV D, H	54
MOV C, L	4D	MOV D, L	55
MOV C, M	4E	MOV D, M	56
MOV E, A	5F	MOV H, A	67
MOV E, B	58	MOV H, B	60
MOV E, C	59	MOV H, C	61
MOV E, D	5A	MOV H, D	62
MOV E, E	5B	MOV H, E	63
MOV E, H	5C	MOV H, H	64
MOV E, L	5D	MOV H, L	65
MOV E, M	5E	MOV H, M	66
MOV L, A	6F	MOV M, A	77
MOV L, B	68	MOV M, B	70
MOV L, C	69	MOV M, C	71

MOV L, D	**6A**	**MOV M, D**	**72**
MOV L, E	**6B**	**MOV M, E**	**73**
MOV L, H	**6C**	**MOV M, H**	**74**
MOV L, L	**6D**	**MOV M, L**	**75**
MOV L, M	**6E**	**MOV M, M**	**76**
MVI A, BYTE	**3E**	**LXI B, BYTE DUPLO**	**01**
MOV B, BYTE	**06**	**LXI D, BYTE DUPLO**	**11**
MOV C, BYTE	**0E**	**LXI H, BYTE DUPLO**	**21**
MOV D, BYTE	**16**	**LXI SP, BYTE DUPLO**	**31**
MOV E, BYTE	**1E**	**LDAX B**	**0A**
MOV H, BYTE	**26**	**LDAX D**	**1A**
MOV L, BYTE	**2E**	**LHLD ENDEREÇO**	**2A**
MOV M, BYTE	**36**	**LDA ENDEREÇO**	**3A**
STAX B	**02**		
STAX D	**12**	**XCHG**	**EB**
SHLD ENDEREÇO	**22**		
STA ENDEREÇO	**32**		
BYTE = VALOR ENTRE 00 E FF (O E 255 DECIMAIS)			
BYTE DUPLO = DOIS BYTES PARA FORMAR 16 BITS			
ENDEREÇO: 16 BITS OU 2BYTES.			

Instruções Aritméticas e Lógicas

INSTRUÇÃO	CÓDIGO	DESCRIÇÃO
ADD r	**10000RRR**	Soma ao acumulador o registrador r e o resultado fica no Acumulador.
ADD M	**10000110**	Soma ao acumulador o conteúdo da posição apontada por HL e o resultado no Acumulador.
ADI	**11000110**	Soma ao acumulador o 2º byte da instrução e resultado no Acumulador.
ADC r	**10001RRR**	Soma ao acumulador o registrador r e o Carry, resultado no Acumulador.
ADC M	**10001110**	Soma o Acumulador ao conteúdo da posição apontada por HL e o carry, resultado no Acumulador.
ACI	**11001110**	Soma ao Acumulador o 2º byte da instrução

		e o carry e resultado no Acumulador.
SUB r	**10010RRR**	Subtrai do Acumulador o conteúdo do registrador r e o deixa no Acumulador.
SUB M	**10010110**	Subtrai do Acumulador o conteúdo da posição que aponta HL e o resultado no Acumulador.
SUI	**11010110**	Subtrai do Acumulador o 2º byte da instrução.
SBB r	**10011RRR**	Subtrai do acumulador o registradoe r + o carry.
SBB M	**10011110**	Subtrai do Acumulador o conteúdo da posição que aponta HL.
SBI	**11011110**	Subtrai do acumulador o 2º byte + o carry.
INR r	**00RRR100**	Incrementa em 1 o registrador r.(Z;S;P;AC)
INR M	**00110100**	Incrementa em 1 o conteúdo da posição que aponta HL (Z;S;P;AC)
DCR r	**00RRR101**	Decrementa em 1 o registrador r (Z;S;P;AC)
DCR M	**00110101**	Decrementa em 1 o conteúdo da posição que aponta HL (Z;S;P;AC)
INX rp	**00RR0011**	Incrementa em 1 o par rp de registradores. BC,DE,HL
DCX rp	**00RR1011**	Decrementa em 1 o par rp de registradores. BC,DE,HL
DAD rp	**00RR1001**	Soma a HL o par de registradores rp (CY a vezes).
DAA	**00100111**	Os 8 bits do acumulador são ajustados para BCD=decimal. (flags)

INSTRUÇÃO	CÓDIGO (HEX)	INSTRUÇÃO	CÓDIGO (HEX)
ADD A	**87**	**ADC A**	**8F**
ADD B	**80**	**ADC B**	**88**
ADD C	**81**	**ADC C**	**89**
ADD D	**82**	**ADC D**	**8A**
ADD E	**83**	**ADC E**	**8B**
ADD H	**84**	**ADC H**	**8C**
ADD L	**85**	**ADC L**	**8D**
ADD M	**86**	**ADC M**	**8E**
SUB A	**97**	**SBB A**	**9F**
SUB B	**90**	**SBB B**	**98**
SUB C	**91**	**SBB C**	**99**
SUB D	**92**	**SBB D**	**9A**

SUB E	**93**	**SBB E**	**9B**
SUB H	**94**	**SBB H**	**9C**
SUB L	**95**	**SBB L**	**9D**
SUB M	**96**	**SBB M**	**9E**
INR A	**3C**	**DCR A**	**3D**
INR B	**04**	**DCR B**	**05**
INR C	**0C**	**DCR C**	**0D**
INR D	**14**	**DCR D**	**15**
INR E	**AC**	**DCR E**	**1D**
INR H	**24**	**DCR H**	**25**
INR L	**2C**	**DCR L**	**2D**
INR M	**34**	**DCR M**	**35**
INX B	**03**	**DCX B**	**0B**
INX D	**13**	**DCX D**	**1B**
INX H	**23**	**DCX H**	**2B**
INX SP	**33**	**DCX SP**	**3B**
		SP = STACK POINTER	
ADI BYTE	**C6**	**DAD B**	**D9**
ACI BYTE	**CE**	**DAD D**	**19**
SUI BYTE	**D6**	**DAD H**	**29**
SBI BYTE	**DE**	**DAD SP**	**39**
DAA	**27**		

INSTRUÇÃO	CÓDIGO	DESCRIÇÃO
ANA r	**10100RRR**	AND entre o Acumulador e o registrador r (flags, CY=0,AC=1)
ANA M	**10100110**	AND entre o acumulador e o conteúdo da posição apontada por HL (flags idem).
ANI	**11100110**	AND entre o Acumulador e o 2º byte da instrução (flags idem).
XRA r	**10101RRR**	OR Exclusiva entre o acumulador e o registro r (flags,CY e AC=0)
XRA M	**10110110**	OR Exclusiva entre o Acumulador e o conteúdo da posição de HL(flags idem).
XRI	**11101110**	OR Exclusiva entre o acumulador e o 2º Byte da instrução(flags idem).
ORA r	**10110RRR**	OR entre o Acumulador e o registrador

		r(flags idem).
ORA M	**10110110**	OR entre o Acumulador e o conteúdo da posição apontada por HL(flags idem).
ORI	**11110110**	OR entre acumulador e o 2º byte da instrução(idem).
CMP r	**10111RRR**	Compara o acumulador com o registrador r não alterando o conteúdo do acumulador(sei A=1 Z=1,se A<r CY=1,flags).
CMP M	**10111110**	Compara o acumulador com o conteúdo da posição apontada por HL (flags idem).
CPI	**11111110**	Compara o acumulador com o 2º byte da instrução.(idem).
RLC	**00000111**	Rotação do conteúdo do acumulador um lugar a esquerda(flags, CY e bit 0= valor bit 7 do Acumulador).
RRC	**00001111**	Rotação do conteúdo do acumulador um lugar a direita (flags, CY e bit 7= bit 0 do Acumulador)
RAL	**00010111**	Rotação do Acumulador um lugar a esquerda intercalando o CY (CY).
RAR	**00011111**	Idem anterior, porém a direita (CY).
CMA	**00101111**	Complementa o conteúdo e do acumulador bit a bit.
CMC	**00111111**	Complementa o conteúdo do carry CY(CY).
STC	**00110111**	Fixa CY em 1 (CY).

INSTRUÇÃO	CÓDIGO (HEX)	INSTRUÇÃO	CÓDIGO (HEX)
ANA A	**A7**	**XRA A**	**AF**
ANA B	**A0**	**XRA B**	**A8**
ANA C	**A1**	**XRA C**	**A9**
ANA D	**A2**	**XRA D**	**AA**
ANA E	**A3**	**XRA E**	**AB**
ANA H	**A4**	**XRA H**	**AC**
ANA L	**A5**	**XRA L**	**AD**
ANA M	**A6**	**XRA M**	**AE**
ORA A	**B7**	**CMP A**	**BF**
ORA B	**B0**	**CMP B**	**B8**

ORA C	**B1**	**CMP C**	**B9**
ORA D	**B2**	**CMP D**	**BA**
ORA E	**B3**	**CMP E**	**BB**
ORA H	**B4**	**CMP H**	**BC**
ORA L	**B5**	**CMP L**	**BD**
ORA M	**B6**	**CMP M**	**BE**
CMA	**2F**	**RLC**	**07**
STC	**37**	**RRC**	**0F**
CMC	**3F**	**RAL**	**17**
		RAR	**1F**
ANI BYTE	**E6**		
XRI BYTE	**EE**		
ORI BYTE	**F6**		
CPI BYTE	**FE**		

INSTRUÇÃO	CÓDIGO	DESCRIÇÃO
JMP	**11000011**	Salto para a posição indicada pelo 2º e 3º byte da instrução, 2º byte=parte baixa, 3º=alta.

INSTRUÇÃO	Flags	CÓDIGO	DESCRIÇÃO
JZ	**Z**	**11FF1010**	Salto para a posição indicada no 2 e 3º bytes da instrução se o flag (FF) tiver valor 1.
JC	**CY**		
JPE	**P**		
JM	**S**		
JNZ	**Z**	**11FF0010**	Idem se o valor do Flag for zero.
JNC	**CY**		
JPO	**P**		
JP	**S**		
PCHL		**11101001**	Salto para a posição endereçada por HL.

INSTRUÇÃO	CÓDIGO (HEX)	INSTRUÇÃO	CÓDIGO (HEX)
JMP dir.	C3	JPO dir.	E2
JNZ dir.	C2	JPE dir.	EA
jz dir.	CA	JP dir.	F2
JNC dir.	D2	JM dir.	FA
JC dir.	DA	PCHL	E9

INSTRUÇÃO	Flags	CÓDIGO	DESCRIÇÃO
CALL		11001101	Salto para a posição endereçada pelo 2º e 3º byte da instrução. Guarda no STACK o conteúdo do P.C.(Contador de programa).
CZ	Z		
CC	CY		
CPE	P		
CM	S		
CNZ	Z	11FF1100	Idem anterior, porém se o Flag for zero. Guarda o P.C. na STACK.
CNC	CY		
CPO	P		
CP	S		
RET		11001001	Retorno ao ultimo endereço armazenado na STACK.
RZ	Z	11FF1000	Retorno ao ultimo endereço armazenado na STACK somente se o Flag indicado valer 1.
RC	CY		
RPE	P		
RM	S		
RNZ	Z	11FF0000	Idem caso anterior, porém se o Flag valer 0.
RNC	CY		
RPO	P		
RP	S		

INSTRUÇÃO	CÓDIGO (HEX)	INSTRUÇÃO	CÓDIGO (HEX)
CALL dir.	CD	RET	C9
CNZ dir.	C4	RNZ	C0
CZ dir.	CC	RZ	C8
CNC dir.	D4	RNC	D0
CC dir.	DC	RC	D8
CPO dir.	E4	RPO	E0
CPE dir.	EC	RPE	E8
CP dir.	F4	RP	F0
CM dir.	FC	RM	F8

Outras Instruções

INSTRUÇÃO	CÓDIGO	DESCRIÇÃO
PUSH rp	11RR0101	Guarda o par de registradores rp na Stack.
PUSH PSW	11110101	Guarda o conteúdo do Acumulador e o registrador de Flags na Stack.
POP rp	11RR0001	Carrega no par de registradores rp o conteúdo das duas últimas posições do Stack.
POP PSW	11110001	Carrega o registrador de Flags e o Acumulador com as duas últimas posições da Stack.
XTHL	11100011	Intercambio do conteúdo das posições endereçadas pelop Stack pelo conteúdo dos registradores H e L.
SPHL	11111001	Traslada o conteúdo do par HL ao Ponteiro do Stack. Deve ser executado ao iniciar o sistema para posicionar o Stack na zona escolhida da memoria.
IN port	11011011	Carrega o Acumulador com o conteúdo do canal I/O endereçado pelo 2º byte da instrução.
OUT port	11010011	Carrega o canal I/O endereçado pelo 2º byte da instrução com o Acumulador.
EI	11111011	Habilita as interrupções (não afeta a interrupção TRAP)
DI	11110011	Desabilita as interrupções (não afeta a TRAP)

HLT	**01110110**	Para a entrada da CPU até que seja realizado um Reset, uma interrupção valida ou um HOLD.
NOP	**00000000**	Não opera, somente serve para temporização.
SIM	**00110000**	Fixa o registrador de interrupções e a linha SOD com o valor do Acumulador.
		Bit 7 do acumulador= sai pela linha SOD.
		Bit 6 " " = em 1 habilita a linha SOD.
		Bit 5 não é utilizado.
		Bit 4 " " = em 1 apaga o FF RST 7,5
		Bit 3 = em 1 permite modificar este registrador.
		Bit 2 = em 1 proibe a interrupção 7,5
		Bit 1 = em 1 proibe a " 6,5
		Bit 0 = a 1 proibe a " 5,5
RIM	**00100000**	Le o estado da linha SID e o estado das interrupções carregando-o no Acumulador:
		Bit 7 do Acumulador com o que tiver a linha SID.
		Bit 6 " : em 1 indica que a interrupção 7,5 éa solicitada para executar.
		Bit 5 " : idem anterior porém com a 6,5
		Bit 4 " : idem anterior, porém com a 5,5.
		Bit 3 " : em 1 indica que as interrupções INTR foram habilitadas.
		Bit 2 " : Estado da interrupção 7,5.
		Bit 1 " : Estado da interrupção 6,5.
		Bit 0 ": Estado da interrupção 5,5.
RST	**11AAA111**	Guarda o P.C. na Stack e o carrega com a dir:00000000AAA000 Quando se produz a interrupção INTR, depois de aceitá-la com INTA deve-se introduzir pelo bus de dados o código AAA que será axial:
		AAA RST Endereço de salto(em (HEX.)
		000 0 0000
		001 1 0008
		010 2 0010
		011 3 0018
		100 4 0020
		101 5 0028

	110	6	0030
	111	7	0038

INSTRUÇÃO	CÓDIGO (HEX)	INSTRUÇÃO	CÓDIGO (HEX)
PUSH B	**C5**	**POP B**	**C1**
PUSH D	**D5**	**POP D**	**D1**
PUSH H	**E5**	**POP H**	**E1**
PUSH PSW	**F5**	**POP PSW**	**F1**
XTHL	**E3**	**SPHL**	**F9**
OUT Byte	**D3**	**IN Byte**	**DB**
DI	**F3**	**EI**	**FB**
NOP	**D0**	**HLT**	**78**
RIM	**20**	**SIM**	**30**
RST 0	**C7**	**RST 4**	**E7**
RST 1	**CF**	**RST 5**	**EF**
RST 2	**D7**	**RST 6**	**F7**
RST 3	**DF**	**RST 7**	**FF**

Os FLAGS

Os (flag (sinalizadores) são registradores internos da CPU que indicam o estado de uma operação depois de tê-la realizado) estes são controlados pela ALU. Vejamos quem são eles:

FLAGS	DESCRIÇÃO
S	Quando um resultado for negativo ficará em 1.
Z	Quando uma operação tiver sido zero este é colcoado em 1
P	Indica a paridade de uma operação, se for par P=1
CY	É colocado em 1 quando é produzido um carry em uma operação de 8 bits
	Sempre o primeiro byte que se carrega (por exemplo, na instrução JMP dir.) é o mais baixo e o segundo o mais alto, portanto ao lado das instruções que levan 2º e 3º bytes estes indicarão o endereço porém de forma inversa.