APOSTILA DAS INSTRUÇÕES ASSEMBLER DO 8088

Departamento de Eletroeletrônica

Disciplina: Microprocessadores II, Arquitetura de Sistemas Digitais

Professores: Cláudio Leão, Orlando Rocha, Sérgio Campos

Índice

1 - Modos de endereçamento do 8086

Os modos de endereçamento do 8086 podem ser divididos em dois tipos distintos:

1. Modos de endereçamento de programa
2. Modos de endereçamento de dados

1.1 - Modos de endereçamento de programa

Quando deseja-se que a CPU realize uma busca de instrução, o endereço da localização de memória da qual a instrução é obtida é calculado como a soma de um offset armazenado no ponteiro de instrução IP, com o conteúdo do registrador CS. Normalmente, o conteúdo do IP é incrementado à medida que as instruções são executadas. Entretanto, as instruções de desvio no fluxo de execução de um programa podem modificá-lo de uma das três maneiras abaixo:

1. **Endereçamento relativo a programa:** Um deslocamento de 8 ou 16 bits fornecido pela instrução na forma de uma dado é somado ao IP como um número binário sinalizado. Essa operação não altera o conteúdo de CS, sendo uma operação intra-segmento (dentro do próprio segmento corrente).
2. **Endereçamento direto:** Um endereço de 16 bits obtido da própria instrução na forma de um dado é colocado no IP, nesse caso trata-se de uma operação intra-segmento. Se o endereço especificado for de 32 bits, os primeiros dois bytes carregarão o IP e os dois seguintes o CS, nesse caso ocorre uma operação inter-segmento (fora do segmento de código corrente).
3. **Endereçamento indireto:** Um dos modos de endereçamento de dados (que serão descritos a seguir) podem ser utilizados para ler um dado da memória. Entretanto, esse dado é interpretado como um endereço de memória pelas instruções de desvio ou chamada de subrotina. Se o dado de 16 bits é acessado, este será carregado no registrador IP (operação intra-segmento). Se no entanto, dois dados de 16 bits são acessados, o primeiro será carregado em IP e o segundo em CS, realizando assim uma operação inter-segmento.

1.2 - Modos de endereçamento de dados

Seis opções estão disponíveis para o endereçamento de dados

* **Modo Imediato**

Neste tipo de endereçamento, um dos operandos está presente no byte seguinte ao código da instrução (op-code). Se bytes de endereçamento seguem o op-

code, então o dado a ser transferido de maneira imediata virá logo após os bytes de endereçamento.

Exemplo: ADD AX,1000h : Soma 1000h ao conteúdo do registrador AX

* **Modo Direto**

O modo de endereçamento direto é feito somando-se os dois bytes seguintes ao op-code, ao CS, para compor um novo endereço linear.

Exemplo: suponha que o registrador DS contenha B000h. A instrução MOV DX,[8000h], carregará em DX o conteúdo da posição de memória DS:8000, cujo endereço linear é B8000h.

***Modo Direto Indexado**

Nesse modo de endereçamento utiliza-se os registradores SI e DI como indexadores, somando-se um deslocamento de 8 ou 16 bits a um desses registradores de forma a gerar um endereço efetivo (offset).

```
Exemplo:  MOV  AL,[SI+10h]
          MOV  [DI-1000h],DX
```

***Modo Implícito**

O modo implícito é uma versão simplificada do modo de endereçamento direto indexado. A única diferença entre eles é que, nesse modo, não é especificado um deslocamento.

```
Exemplo:  MOV  AL,[SI]
          MOV  [DI],DX
```

***Modo Relativo a Dados (utilizando o registrador de segmento DS)**

Nesse modo o conteúdo do registrador BX é utilizado para formar a base para o endereço linear. Todos os modos de endereçamento descritos até aqui, com exceção do modo imediato, podem ser utilizados também no modo relativo.

```
Exemplo:  Modo relativo direto            MOV  AX,[BX+1000h]
          Modo relativo implícito         MOV  DH,[BX+SI]
          Modo relativo direto indexado   MOV  [BX+DI+1000h],DL
```

***Modo Relativo a Stack**

Nesse modo o conteúdo do registrador BP é utilizado para formar a base para o endereço linear de acesso a dados gravados na pilha.

```
Exemplo:    MOV  BP,SP
            MOV  AX,[BP+04h]
            MOV  [BP+DI+10h],BX
```

2 - Tipos de Dados (8086)

Os principais tipos de dados são os bytes e as words. Um byte são 8 bits, sendo estes numerados de 0 a 7, o bit 0 é o menos significativo (LSB - least significant bit). A word é uma seqüência de dois bytes que ocupam dois endereços consecutivos quaisquer. Uma word tem, portanto, 16 bits os quais são numerados de 0 a 15. Novamente, o bit 0 é o menos significativo (LSB). O byte que contém o bit zero da word é chamado de low byte (byte baixo) e o que contém o bit 15 é chamado de high byte (byte alto). No 8086 e nos demais microprocessadores da família INTEL, o low byte é armazenado no endereço mais baixo e seu endereço é também o endereço da word. O endereço do high byte é utilizado somente quando a metade superior da word estiver endereçada separadamente da parte baixa.

Como o 8086 possui um barramento de 16 bits, as comunicações entre o microprocessador e a memória ocorrem como transferências de words alinhadas a endereços pares; o microprocessador converte transferências entre endereços não alinhados a words, em múltiplas transferências alinhadas. As operações desalinhadas reduzem a velocidade do processamento em função de ciclos extras de barramento.

Embora os bytes e words sejam os principais tipos de dados para o 8086, o mesmo tem a possibilidade de interpretar estes operandos de maneiras diversas. Algumas instruções especializadas reconhecem os seguintes tipos de dados:

- Inteiro: Número binário sinalizado armazenado em uma word ou byte. Todas as operações assumem a representação em complemento de dois. O bit de sinal é o 7 se for um byte ou o 15 se for uma word. Se negativo, este bit será igual a 1, caso positivo será 0. Em um byte pode-se ter os valores compreendidos entre -128 a +127; na word pode-se representar de -32768 a +32767.
- Ordinal: Número binário sem sinal armazenado em um byte ou uma word. Se estiver contido em um byte, estará na faixa de 0 a 255, se numa word, entre 0 a 65535.
- String: Seqüência contínua de bytes ou words. Uma string pode conter de 0 a (2^{20})-1 bytes.

- BCD: Representação de um dígito decimal codificado em binário (Binary Coded Decimal) na faixa de 0 a 9. Um decimal não compactado é expresso por um byte sem bit de sinal. Cada dígito é armazenado em um byte. A magnitude do número é o valor binário do nibble de ordem mais baixa (4 bits menos significativos do byte); os dígitos somente podem assumir valores entre 0 e 9. O nibble de ordem mais alta (4 bits mais significativos do byte), deve estar zerado para as operações de multiplicação e de divisão, e podem conter qualquer valor nas somas e subtrações.
- BCD Compactado (packed BCD): Representação de dígitos decimais codificados em binário, cada um na faixa de 0 a 9. Um dígito é armazenado nos 4 bits menos significativos do byte e outro nos 4 bits mais significativos; portanto cada byte comporta dois dígitos.

3 - Instruções Assembly

Neste capítulo serão apresentadas as instruções assembly do 8086, que se aplicam a qualquer microprocessador da família 80XXX da Intel

3.1 - Instruções de movimentação de dados

Neste grupo de instruções estão aquelas que permitem manipular dados do tipo byte, word, doubleword entre memória e registradores do microprocessador. São de três tipos:

Instruções de movimentação de uso geral

Instruções para manipulação de stack

Instruções de conversão de tipo

3.1.1 - Instruções de movimentação de uso geral

a) MOV - Move

Propósito: Copiar o conteúdo do operando-fonte para o operando-destino. O conteúdo do fonte não é afetado

Formato: MOV destino,fonte

Flags: Nenhum afetado

Exemplos: Há 7 tipos de instruções MOV:

1 - Do acumulador para a memória

```
MOV  [SI], AL
MOV  [BX+DI],AX
MOV  [400], AL
```

2 - Da memória para o acumulador

```
MOV  AL,[BX+2]
MOV  AX,[SI]
MOV  AL,[400]
```

3 - Da memória ou registrador para registrador de segmento

```
MOV  ES,BX
MOV  DS,[SI]
MOV  SS,[BX]
```

Obs: O registrador CS não pode ser usado como destino.

4 - Do registrador de segmento para a memória ou registrador

```
MOV  AX,CS
MOV  CX,SS
```

```
MOV   [500],DS
```

5 - De registrador para registrador / da memória ou registrador para registrador / de registrador para a memória

```
MOV   CL,DH
MOV   AL,AH
MOV   SI,BX
MOV   DI,DX
MOV   CL,[SI]
MOV   DX,[SI+1000]
MOV   [BX+DI],DH
MOV   TABELA[BX+3],CL
```

6 - Dado imediato para registrador

```
MOV   AX,99H
MOV   BL,99H
MOV   CX,0FFFFH
MOV   DI,61CH
```

7 - Dado imediato para a memória

```
MOV BYTE PTR[SI],0FH
MOV WORD PTR[BX+1234],1157H
```

b) LDS - Carrega registrador de segmento

Propósito: Transferir quatro bytes consecutivos de um operando fonte para um par de registradores de 16 bits. O operando-fonte precisa estar na memória. Um dos registradores é aquele especificado na instrução, o outro é o DS.

Formato: LDS destino, fonte

Flags: Nenhum afetado

Exemplo: Considere que o registrador SI contenha o valor 5334H, BX contenha 0FFFFH e que as posições de memória a partir da indicada por SI, dentro do segmento de dados, contenha os seguintes bytes 21h, 12h, DFh e 2Ch. Após a instrução

LDS BX, dword ptr[SI] o registrador DS conterá o valor 1221H e o registrador BX conterá o valor 2CDFH.

c) LES - Carrega registrador de segmento extra

Propósito: Transferir quatro bytes consecutivos de um operando fonte para um par de registradores de 16 bits. O operando-fonte precisa estar na memória. Um dos registradores é aquele especificado na instrução, o outro é o ES.

Formato: LES destino, fonte

Flags: Nenhum afetado

Exemplo: Considere que o registrador BX contenha o valor 5334H, DI contenha 0FEF3H e que as posições de memória a partir da indicada por BX, dentro do segmento de dados, contenha os seguintes bytes E3h, 2Dh, 24h e ACh. Após a instrução LES DI, dword ptr[BX] o registrador ES conterá o valor 2DE3H e o registrador DI conterá o valor AC24H.

3.1.2 - Instruções para manipulação de stack

d) PUSH - Grava word na pilha

Propósito: Colocar, na área de memória usada como pilha, o conteúdo de um registrador ou posição de memória.

Modo de operação:

1 - Decrementa o SP

2 - Move o byte mais alto do registrador ou posição de memória para a locação indicada por SS:SP

3 - Decrementa o SP

4 - Move o byte mais baixo do operando para a locação indicada por SS:SP

Formato: PUSH fonte

Flags: Nenhum afetado

```
Exemplo:  PUSH SI       ; Grava na pilha a word armazenada em SI
          PUSH ES       ; Grava na pilha a word armazenada em ES
          PUSH [BX+SI]; Grava na pilha a word armazenada na posição de
memória                 apontada por (BX+SI)
```

e) POP - Copia word da pilha para registrador ou memória

Propósito: Retirar uma palavra armazenada no topo da pilha, colocando-a no registrador ou posição de memória especificada.

Modo de operação:

1 - Retira o byte no endereço de memória determinado por SS:SP, movendo-o para o byte de mais baixa ordem do operando destino.

2 - Incrementa o SP em 1

3 - Retira o byte da posição indicada por SS:SP e o move para o byte de alta ordem do operando destino.

4 - Incrementa o SP em 1

Formato: POP destino

Flags: Nenhum afetado

```
Exemplo:  POP  AX    ; Grava em AL a word no topo da pilha
          POP  DS    ; Grava em DS a word no topo da pilha
          POP  [DI]  ; Grava na posição de memória apontada por DI a word no topo
                       da pilha
```

3.1.3 - Instruções de conversão de tipo

f) CBW - Converte byte para word

Propósito: Propagar o bit de sinal do valor presente no registrador AL (bit mais significativo) para o registrador AH. Assim, se o valor em AL for positivo, a instrução armazena 00H em AH e, em caso contrário, 0FFH.

Formato: CBW

Flags: Nenhum

g) CWD - Converte word em doubleword

Propósito: Propagar o bit de sinal do valor presente no registrador AX (bit mais significativo) para o registrador DX. Assim, se o valor em AX for positivo, a instrução armazena 0000H em DX e, em caso contrário, 0FFFFH. Essa instrução é utilizada antes da execução de uma divisão sinalizada do conteúdo de AX por outro operando de 16 bits, de modo a preencher o DX com o valor do sinal em AX.

Formato: CWD

Flags: Nenhum

3.2 - Instruções aritméticas binárias

As instruções aritméticas binárias dos processadores da família 8086/286/386/486/586... operam com dados codificados em binário. As operações incluem adição, subtração, multiplicação, divisão, incremento, decremento, comparação e mudança de sinal. São suportados tanto números inteiros sinalizados

como os não sinalizados. Os operandos fontes podem ser valores imediatos, registradores ou memória. Operandos destino podem ser registradores ou memória (exceto quando o operando fonte estiver na memória).

As instruções aritméticas alteram os flags ZF, CF, SF e OF para reportar o tipo de resultado produzido pela execução da mesma. O tipo de instrução a ser utilizada para o teste do flag dependerá do dado estar sendo interpretado como valor sinalizado ou não. O flag CF contém informações importantes para inteiros sinalizados. O flag ZF é relevante para ambos, e será setado sempre que um resultado for 00H

As instruções aritméticas operam dados de 8, 16 ou 32 bits. Os flags são utilizados para refletir o tamanho da operação. Por exemplo, se o resultado de uma operação de ADD com dados de 8 bits for maior do que 255 (decimal), o flag CF será igual a 1. Se o número inteiro é sem sinal, o flag CF pode ser testado após uma instrução aritmética binária para determinar se a operação requer carry ou borrow que será propagado para o próximo estágio da operação.

As instruções de incremento e decremento não alteram o indicador CF, isto permite o uso destas instruções para a atualização de contadores no controle de loops sem alteração dos reportes referentes aos resultados da aritmética.

Os flags SF e OF reportam a aritmética sinalizada. O flag SF possui o mesmo valor do bit de sinal de um resultado. O flag OF será igual a 1 em qualquer destes casos:

- Um carry foi gerado do MSB para o bit de sinal mais o bit de sinal não gerou um carry para o CF, ou seja, o resultado foi maior do que o maior número positivo que pode ser representado no formato complemento de dois.
- Um carry foi gerado do bit de sinal para o MSB mas nenhum carry foi gerado para o bit de sinal, ou seja, o resultado foi menor do que o menor valor negativo que pode ser representado no formato complemento de dois.

Estes flags de status podem ser testados por qualquer tipo de instrução condicional.

3.2.1 - Instruções de adição e subtração

h) ADD - Adição

Propósito: Executar a adição normal entre dois operandos, um fonte e outro destino, devolvendo o resultado no destino.

Formato: ADD destino,fonte

Flags: Afetados AF, CF, OF, PF, SF, ZF.

```
Exemplos:  ADD  AL,04H              ; Adiciona 04H ao conteúdo de AL
           ADD  byte ptr[300],05H   ; Adiciona 05H ao conteúdo da posição de
                                      memória DS:0300
           ADD  AX,SI               ; Adiciona o conteúdo de SI ao conteúdo de AX
           ADD  DL,[300]            ; Adiciona o byte gravado em DS:0300 ao
                                      conteúdo de DL
           ADD  [SI+BX+2],AL        ; Adiciona o conteúdo de AL ao conteúdo
da                                    posição de memória DS:SI+BX+02
```

i) ADC - Adição com carry *

Propósito: Executar a adição normal entre dois operandos, um fonte e outro destino, mais o valor do flag carry existente antes da operação, devolvendo o resultado no destino.

Formato: ADC destino,fonte

Flags: Afetados AF, CF, OF, PF, SF, ZF.

```
Exemplos:  ADC  AL,04H              ; Adiciona 04H ao conteúdo de AL e então soma
                                      o conteúdo do flag carry
           ADC  byte ptr[300], 05H  ; Adiciona 05H ao conteúdo da posição de
                                      memória DS:0300 e então soma o conteúdo do
                                      flag carry
           ADC  AX,SI               ; Adiciona o conteúdo de SI ao conteúdo de AX
                                      e então soma o conteúdo do flag carry
           ADC  DL,[300]            ; Adiciona o byte gravado em DS:0300 ao
                                      conteúdo de DL e então soma o conteúdo do flag
                                      carry
           ADC  [SI+BX+2], AL       ; Adiciona o conteúdo de AL ao conteúdo
da                                    posição de memória DS:SI+BX+02 e então
                                      soma o conteúdo do flag carry
```

(*) O flag carry será setado, após a execução da instrução ADC, caso uma das condições abaixo ocorra:

a) Há um "vai um" na etapa da soma dos operandos ***destino*** e ***fonte***, ou

b) Há um "vai um" na etapa da soma do flag carry (existente antes da operação), com o resultado de (***destino + fonte***).

j) SUB - Subtração

Propósito: Executar uma subtração entre dois operandos, um fonte e outro destino, devolvendo o resultado no destino.

Formato: SUB destino,fonte

Flags: Afetados AF, CF, OF, PF, SF, ZF.

```
Exemplos:  SUB  BL,04H              ; Subtrai 04H do conteúdo de BL
           SUB  byte ptr[DI],05H    ; Subtrai 05H do conteúdo da posição de
                                    memória DS:DI
           SUB  AX,CX               ; Subtrai o conteúdo de CX do conteúdo de AX
           SUB  BX,[SI]                  ; Subtrai a word gravada em DS:SI do
conteúdo                                  de BX
           SUB  [BX+2],DL           ; Subtrai o conteúdo de DL do conteúdo da
                                    posição de memória DS:BX+02
```

j) SBB - Subtração com carry

Propósito: Executar a subtração entre dois operandos, um fonte e outro destino, menos o valor do flag carry existente antes da operação, devolvendo o resultado no destino.

Formato: SBB destino,fonte

Flags: Afetados AF, CF, OF, PF, SF, ZF.

```
Exemplos:  SBB  BL,04H              ; Subtrai 04H do conteúdo de BL e então subtrai
                                    do resultado o conteúdo do flag carry
           SBB  byte ptr[DI],05H    ; Subtrai 05H do conteúdo da posição de
                                    memória DS:DI e então subtrai do resultado o
                                    conteúdo do flag carry
```

```
        SBB   AX,CX          ; Subtrai o conteúdo de CX do conteúdo de AX
                             e então subtrai do resultado o conteúdo do flag
                             carry
        SBB   BX,[SI]            ; Subtrai a word gravada em DS:SI do
conteúdo                         de BX e então subtrai do resultado o
conteúdo                         do flag carry
        SBB   [BX+2],DL      ; Subtrai o conteúdo de DL do conteúdo da
                             posição de memória DS:BX+02 e então subtrai
                             do resultado o conteúdo do flag carry
```

k) INC - Incrementa destino em 1

Propósito: Soma 1 ao conteúdo de um operando, que pode ser registrador ou posição de memória.

Formato: INC destino

Flags: Afetados AF, OF, PF, SF, ZF.

```
Exemplos: INC   BL           ; Soma 1 ao conteúdo de BL
          INC   byte ptr [SI] : Soma 1 ao byte gravado na posição de memória
                              apontada por DS:SI
```

Obs: Registradores de segmento não podem ser usados como operando.

l) DEC - Decrementa destino em 1

Propósito: Subtrai 1 do conteúdo de um operando, que pode ser registrador ou posição de memória.

Formato: DEC destino

Flags: Afetados AF, OF, PF, SF, ZF.

```
Exemplos: DEC   BX           ; Subtrai 1 do conteúdo de BX
          DEC   word ptr [0100]   : Subtrai 1 da word gravada na posição de
memória                           apontada por DS:0100
```

Obs: Registradores de segmento não podem ser usados como operando.

3.2.2 - Instruções de comparação e mudança de sinal

m) CMP - Compara dois operandos

Propósito: Efetuar uma subtração entre dois operandos, alterando os flags e descartando o resultado da operação.

Formato: CMP destino,fonte

Flags: Afetados AF, CF, OF, PF, SF, ZF.

```
Exemplos:  CMP  AL,57H       ; Compara o conteúdo de AL com o valor 57H
           CMP  DI,BX        ; Compara os conteúdos de DI e BX
           CMP  [SI],AX      ; Compara uma word gravada na posição DS:SI, com
o                            conteúdo de AX
           CMP  CH,[SI+BX+3]  ; Compara o conteúdo de CH com o byte
armazenado na                posição de memória DS:SI+BX+3
```

n) NEG - Calcula o complemento de dois do destino

Propósito: Efetuar o complemento de dois de um operando, que pode ser registrador ou posição de memória..

Formato: NEG destino

Flags: Afetados AF, CF, OF, PF, SF, ZF.

Exemplo: Considere que o registrador CX contenha o valor 0005 (+5). Após a execução da instrução NEG CX, o registrador conterá o valor FFFB (-5).

3.2.3 - Instruções de multiplicação e divisão

o) MUL - Multiplicação sem sinal

Propósito: Efetuar uma multiplicação não sinalizada entre o conteúdo do acumulador (AL ou AX) pelo operando fonte, devolvendo o resultado no acumulador AX (caso a operação tenha envolvido AL com um operando de 8 bits), ou em DX e AX (caso a operação tenha envolvido AX e um operando de 16 bits).

Formato: MUL fonte

Flags: Afetados CF, OF, indefinidos AF, PF, SF, ZF.

```
Exemplos:  MOV  AL, 83H
           MOV  CL, 44H
           MUL  CL           ; realiza o produto entre AL e CL, resultado em AX

           MOV  AX,1234H
           MOV  BX,6000H
           MUL  BX           ; realiza o produto de 16 bits entre AX e BX, resultado
                             em DX(parte alta) e AX(parte baixa)
```

p) IMUL - Multiplicação com sinal

Propósito: Efetuar uma multiplicação sinalizada entre o conteúdo do acumulador (AL ou AX) pelo operando fonte, devolvendo o resultado no acumulador AX (caso a operação tenha envolvido AL com um operando de 8 bits), ou em DX e AX (caso a operação tenha envolvido AX e um operando de 16 bits).

Como são sinalizados , os operandos podem ter valores na faixa de -128 a +127 (operação de 8 bits) ou entre -32768 a +32767 (operação de 16 bits).

Formato: IMUL fonte

Flags: Afetados CF, OF, indefinidos AF, PF, SF, ZF.

```
Exemplos: MOV  AL, 83H
          MOV  CL, 44H
          IMUL CL          ; realiza o produto entre AL e CL, resultado em AX

          MOV  AX,1234H
          MOV  BX,6000H
          IMUL BX          ; realiza o produto de 16 bits entre AX e BX, resultado
                           em DX(parte alta) e AX(parte baixa)
```

g) DIV - Divisão sem sinal

Propósito: Efetuar uma divisão não sinalizada entre o conteúdo do acumulador (AL ou AX) pelo operando fonte, devolvendo o resultado no acumulador AX (caso a operação tenha envolvido AX com um operando de 8 bits), ou em DX e AX (caso a operação tenha envolvido DS:AX e um operando de 16 bits). Na operação de 8 bits, o quociente da operação de divisão fica em AL e o resto da divisão em AH. Na operação de 16 bits, o quociente da operação fica em AX e o resto da divisão em DX.

Formato: DIV fonte

Flags: indefinidos AF, CF, OF, PF, SF, ZF.

```
Exemplos: DIV  BL   ; Dividendo em AX, divisor em BL. O quociente estará em AL
e                   o resto em AH
          DIV  BX   ; Dividendo em DX:AX, divisor em BX. O quociente estará
em                  AX e o resto em DX
```

No primeiro exemplo, considere que o conteúdo do acumulador AX seja 800H e que BL contenha 30H. Após a execução da instrução, AL conterá 2AH e AH o valor 20H

r) IDIV - Divisão com sinal

Propósito: Efetuar uma divisão sinalizada entre o conteúdo do acumulador (AL ou AX) pelo operando fonte, devolvendo o resultado no acumulador AX (caso a operação tenha envolvido AX com um operando de 8 bits), ou em DX e AX (caso a operação tenha envolvido DS:AX e um operando de 16 bits). Na operação de 8 bits, o quociente da operação de divisão fica em AL e o resto da divisão em AH. Na operação de 16 bits, o quociente da operação fica em AX e o resto da divisão em DX.

Se os operandos tiverem sinais diferentes, sendo um positivo e o outro negativo, a divisão será feita de forma que o resto tenha o mesmo sinal do dividendo.

Formato: IDIV fonte

Flags: indefinidos AF, CF, OF, PF, SF, ZF.

```
Exemplos:  IDIV  DH                  ; Dividendo em AX, divisor em DH. O quociente
                                       estará em AL e o resto em AH
           IDIV  word ptr[BX+100]    ; Dividendo em DX:AX, divisor na posição de
                                       memória apontada por (BX+100). O quociente
                                       estará em AX e o resto em DX
```

3.3 - Instruções de aritmética decimal

A aritmética decimal é realizada combinando-se as instruções da aritmética binária com as da aritmética decimal. As instruções da aritmética decimal são utilizadas em uma das seguintes formas:

- Para ajustar os resultados de uma instrução da aritmética binária, realizada anteriormente, a fim de se produzir um valor decimal válido, compactado ou não.
- Para ajustar as entradas de um a operação binária subsequente para que a mesma produza um resultado decimal, compactado ou não. estas instruções operam somente nos registradores AL ou AH, associados ao flag AF.

3.3.1 - Instruções de alteração para BCD compactado

s) DAA - Ajuste decimal após adição

Propósito: Corrigir o resultado presente no acumulador AL, após uma soma entre dois valores BCD compactados (um dígito em cada nibble), para obter um par de dígitos BCD compactados.

O procedimento dessa instrução é:

1. Se os quatro bits menos significativos de AL estão entre A e F, ou se o flag AF está ligado, somar 06H ao acumulador AL e ligar o flag AF.
2. Se os quatro bits mais significativos de AL estão entre A e F, então somar 60H ao acumulador AL e ligar o flag CF.

Formato: DAA

Flags: Afetados AF, CF, PF, SF, ZF, indefinido OF.

Exemplo: Suponha que AL contenha o valor 39H e AH o valor 53H. Após a seqüência

```
ADD  AL,AH
DAA
```

o acumulador AL conterá o valor 92H, e não 8CH.

t) DAS - Ajuste decimal após subtração

Propósito: Corrigir o resultado presente no acumulador AL, após uma subtração entre dois valores BCD compactados (um dígito em cada nibble), para obter um par de dígitos BCD compactados.

O procedimento dessa instrução é:

1. Se os quatro bits menos significativos de AL estão entre A e F, ou se o flag AF está ligado, subtrair 06H do conteúdo do acumulador AL e ligar o flag AF.
2. Se os quatro bits mais significativos de AL estão entre A e F, ou se o flag CF está ligado, então subtrair 60H do conteúdo do acumulador AL e ligar o flag CF.

Formato: DAS

Flags: Afetados AF, CF, PF, SF, ZF.

Exemplo: Suponha que AL contenha o valor 84H e DL o valor 16H. Após a seqüência

```
SUB  AL,DL
DAS
```

o acumulador AL conterá o valor 68H, e não 6EH.

3.3.2 - Instruções de alteração para BCD não compactado

u) AAA - Ajuste ASCII após adição

Propósito: Corrigir o resultado presente no acumulador AL, após uma soma de dois dígitos ASCII.

O procedimento dessa instrução é:

1. Se os quatro bits menos significativos de AL estão entre A e F e o flag AF=0, então execute o passo 3.
2. Se os quatro bits menos significativos de AL estão entre A e F, ou se o flag AF=1, então somar 06H ao acumulador AL, somar 01H ao conteúdo de AH e ligar o flag AF.
3. Apaga os quatro bits mais significativos do registrador AL.

Formato: AAA

Flags: Afetados AF, CF; indefinidos, PF, SF, ZF, OF.

Exemplo: Suponha que o acumulador AX contenha o valor 0437H e que CL contenha o valor 34H. Após a seqüência

```
ADD  AL,CL
AAA
```

o acumulador AX conterá o valor 0501H.

v) AAS - Ajuste ASCII após subtração

Propósito: Corrigir o resultado presente no acumulador AL, após uma subtração entre dois dígitos ASCII.

O procedimento dessa instrução é:

1. Se os quatro bits menos significativos de AL estão entre A e F e o flag AF=0, então execute o passo 3.
2. Se os quatro bits menos significativos de AL estão entre A e F, ou se o flag AF=1, então subtraia 06H do conteúdo do acumulador AL, subtraia 01H do conteúdo de AH e ligue o flag AF.
3. Apaga os quatro bits mais significativos do registrador AL.
4. Sinaliza o flag CF com o mesmo valor do AF.

Formato: AAS

Flags: Afetados AF, CF; indefinidos, PF, SF, ZF, OF.

Exemplo: Considera que o acumulador AX contenha o valor 0935H. Após a seqüência

```
SUB  AL,36H
AAS
```

o acumulador AX conterá o valor 0809H.

x) AAM - Ajuste ASCII após multiplicação

Propósito: Corrigir o resultado da multiplicação de dois números decimais não compactados. A instrução deve seguir-se a multiplicação de dois operandos decimais não compactados para produzir um resultado decimal válido. A instrução fará o seguinte: divide o valor em AL por 10, armazenando o resultado em AH e o resto em AL.

Formato: AAM

Flags: Afetados PF, SF, ZF; indefinidos, AF, CF, OF. (os flags são afetados de acordo com o resultado em AL.)

Exemplo: Suponha que o acumulador AL contenha o valor 09 e que BL contenha o valor 08. Após a seqüência

```
MUL  BL
AAM
```

o acumulador AX conterá o valor 0702H. Observe que se a esse valor somarmos 3030H, teremos em AX dois dígitos numéricos em ASCII.

w) AAD - Ajuste ASCII antes de uma divisão

Propósito: Modifica o numerador nos registradores AH ou AL a fim de prepará-lo para a divisão de dois operandos decimais não compactados, o que resultará num número decimal não compactado válido. O registrador AH deve conter o dígito superior e o AL o dígito inferior o resultado desta instrução é armazenado no registrador AL. O registrador AH é zerado.

Procedimento executado pela instrução:

1. Multiplica o valor de AH por 10.
2. Soma o conteúdo de AH a AL.
3. Zera o registrador AH.

Formato: AAD

Flags: Afetados PF, SF, ZF; indefinidos, AF, CF, OF. (os flags são afetados de acordo com o resultado em AL.)

Exemplo: Suponha que o acumulador AX contenha o valor 0902. Após a instrução

```
AAD
```

o acumulador AX conterá o valor 005CH.

3.4 - Instruções Lógicas

As instruções lógicas possuem dois operandos. Os operandos fontes podem ser valores imediatos, registradores gerais, ou memória. Os operandos destino podem ser registradores gerais ou memória (exceto quando o operando fonte também for memória). As instruções lógicas modificam o estado dos flags. Estão organizadas em:

- Instruções para operações booleanas.
- Instruções de teste de bit.
- Instruções de deslocamentos e rotações.

3.4.1 - Instruções para operações booleanas

As operações lógicas são realizadas pelas instruções AND, OR, XOR e NOT

y) NOT - Operação lógica NOT

Propósito: Efetuar o complemento de 1 do operando, que pode ser registrador ou memória. Todos os bits do operando são complementados.

Formato: NOT destino

Flags: Nenhum afetado

```
Exemplo:    NOT   word ptr[SI]
            NOT   CL
```

suponha que no último caso o valor inicial de CL seja 1FH. Após a instrução, CL assumiu o valor E0H.

z) AND - Operação lógica AND

Propósito: Executar a função lógica AND (“E”) entre cada bit de um operando-fonte e seu correspondente em um operando-destino, colocando o resultado no operando-destino.

Formato: AND destino,fonte

Flags: Afetados: CF, OF, PF, SF, ZF; indefinido: AF

```
Exemplo:    AND AL,08H          ; Realiza a operação lógica entre o conteúdo de AL e o
                                  valor 08H, bit-a-bit.
            AND byte ptr[BX],06H ; Realiza a operação lógica entre o conteúdo da
posição                           de memória apontada por BX e o valor 06H, bit-
a-bit.
```

```
        AND AX,CX       ; Realiza a operação lógica entre o conteúdo de AX e
                          CX, bit-a-bit.
```

a1) OR - Operação lógica OR

Propósito: Executar a função lógica OR ("OU") entre cada bit de um operando-fonte e seu correspondente em um operando-destino, colocando o resultado no operando-destino.

Formato: OR destino,fonte

Flags: Afetados: CF, OF, PF, SF, ZF; indefinido: AF

```
Exemplo:  OR BX,0801H       ; Realiza a operação lógica entre o conteúdo de
BX e o                        valor 0801H, bit-a-bit.
          OR [BX],DL        ; Realiza a operação lógica entre o conteúdo da
posição                       de memória apontada por BX e o conteúdo do
                              registrador DL.
          OR BL,AL          ; Realiza a operação lógica entre o conteúdo de BL e
                              AL, bit-a-bit.
```

b1) XOR - Operação lógica XOR

Propósito: Executar a função lógica XOR ("OU EXCLUSIVO") entre cada bit de um operando-fonte e seu correspondente em um operando-destino, colocando o resultado no operando-destino.

Formato: XOR destino,fonte

Flags: Afetados: CF, OF, PF, SF, ZF; indefinido: AF

```
Exemplo:  XOR AL,AH         ; Realiza a operação lógica entre o conteúdo de AL e o
                              conteúdo de AH.
          XOR DL,[BX]       ; Realiza a operação lógica entre o conteúdo de
DL e o                        conteúdo da posição de memória apontada por
BX.
          XOR CX,1234H      ; Realiza a operação lógica entre o conteúdo de CX e
                              o dado imediato 1234H.
```

3.4.2 - Instrução de teste de bit

c1) TEST - Comparação lógica

Propósito: Efetuar o teste de determinados bits de um operando, para determinar se os mesmos estão ligados ou desligados, mas sem afeta-los.

Formato: TEST destino,fonte

Flags: Afetados: CF, OF, PF, SF, ZF; indefinido: AF

Exemplo: TEST AL,AH ; Verifica que bits no registrador AL estão ativos, de acordo com a máscara gravada em AH.

TEST DL,[BX] ; Verifica que bits no registrador DL estão ativos, de acordo com a máscara gravada na posição de memória apontada por BX.

TEST CX,1234H ; Verifica que bits no registrador CX estão ativos, de acordo com a máscara especificada na instrução (1234H).

3.4.3 - Instruções de rotação e deslocamento

d1) RCL - Rotacione para a esquerda através do carry

Propósito: Rotacionar o conteúdo da locação de memória ou registrador especificado, para a esquerda, através do flag carry. O número de vezes a rotacionar é 1, ou aquele contido no registrador CL. O conteúdo de CL não é afetado pela instrução, quando usado como contador.

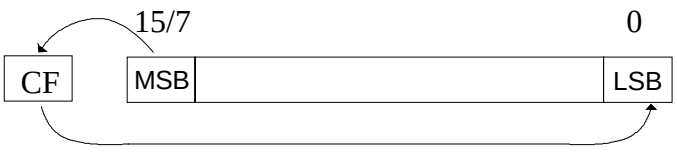
Formato: RCL destino,1 ou
RCL destino, CL

Flags: Afetados: CF, OF

Exemplos: RCL CL,1 ; Rotaciona o conteúdo do registrador CL, uma posição para a esquerda.

RCL AX,CL ; Rotaciona o conteúdo do registrador AX, o número de vezes especificado em CL,

RCL byte ptr[SI],CL ; Rotaciona o conteúdo da posição de memória apontada por SI, o número de vezes especificado em CL.



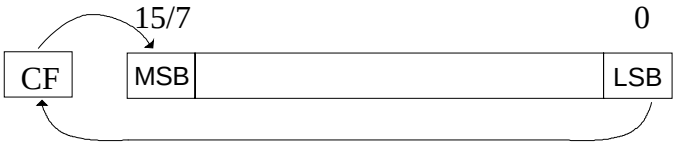
e1) RCR - Rotacione para a direita através do carry

Propósito: Rotacionar o conteúdo da locação de memória ou registrador especificado, para a direita, através do flag carry. O número de vezes a rotacionar é 1, ou aquele contido no registrador CL. O conteúdo de CL não é afetado pela instrução, quando usado como contador.

Formato: RCR destino,1 ou
RCR destino, CL

Flags: Afetados: CF, OF

Exemplos: RCR AL,1 ; Rotaciona o conteúdo do registrador AL, uma posição para a direita.
RCR BH,CL ; Rotaciona o conteúdo do registrador BH, o número de vezes especificado em CL,
RCR byte ptr[300],CL ; Rotaciona o conteúdo da posição de memória DS:0300, o número de vezes especificado em CL.



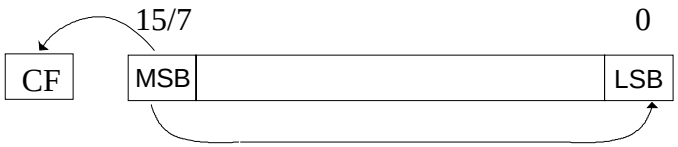
f1) ROL - Rotacione para a esquerda

Propósito: Rotacionar o conteúdo da locação de memória ou registrador especificado, para a esquerda. O número de vezes a rotacionar é 1, ou aquele contido no registrador CL. O conteúdo de CL não é afetado pela instrução, quando usado como contador.

Formato: ROL destino,1 ou
ROL destino, CL

Flags: Afetados: CF, OF

Exemplos: ROL DX,1 ; Rotaciona o conteúdo do registrador DX, uma posição para a esquerda.
ROL BH,CL ; Rotaciona o conteúdo do registrador BH, o número de vezes especificado em CL,
ROL byte ptr[BX],1 ; Rotaciona o conteúdo da posição de memória DS:BX, uma posição para a esquerda.



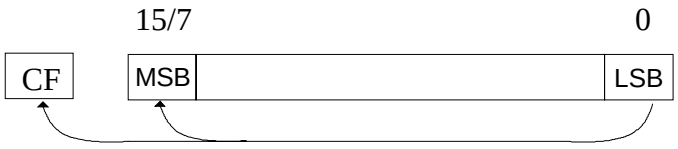
g1) ROR - Rotacione para a direita

Propósito: Rotacionar o conteúdo da locação de memória ou registrador especificado, para a direita. O número de vezes a rotacionar é 1, ou aquele contido no registrador CL. O conteúdo de CL não é afetado pela instrução, quando usado como contador.

Formato: ROR destino,1 ou
ROR destino, CL

Flags: Afetados: CF, OF

Exemplos de:
```
ROR AL,CL          ; Rotaciona o conteúdo do registrador AL, o número de vezes especificado em CL.
ROR BH,1           ; Rotaciona o conteúdo do registrador BH, uma posição para a direita.
ROR byte ptr[DI],1 ; Rotaciona o conteúdo da posição de memória apontada por DI, no segmento de dados, uma posição para a direita.
```



h1) SAL/SHL - Deslocamento aritmético lógico / Deslocamento lógico

Propósito: Deslocar o conteúdo da locação de memória ou registrador especificado à esquerda. Com o deslocamento o bit que estava na posição mais significativa é levado para o flag CF, cujo valor anterior é perdido, e um bit 0 é carregado na posição do bit menos significativo. O número de vezes a deslocar é 1, ou aquele contido no registrador CL. O conteúdo de CL não é afetado pela instrução, quando usado como contador.

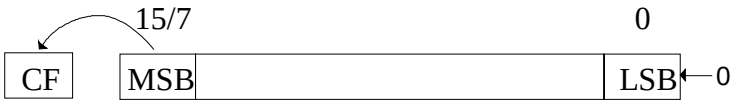
Formato: SAL destino,1 ou
SAL destino, CL

Flags: Afetados: CF, OF, PF, SF, ZF; indefinido AF.

Exemplos: SAL DX,1 ; Desloca o conteúdo do registrador DX, uma posição para a esquerda.

SAL BH,CL ; Desloca o conteúdo do registrador BH, o número de vezes especificado em CL,

SAL byte ptr[BX],1 ; Desloca o conteúdo da posição de memória DS:BX, uma posição para a esquerda.



h1) SAR - Deslocamento aritmético à direita

Propósito: Deslocar o conteúdo da locação de memória ou registrador especificado à direita. Com o deslocamento o bit que estava na posição mais significativa é mantido, e o bit 0 é carregado no flag CF. O número de vezes a deslocar é 1, ou aquele contido no registrador CL. O conteúdo de CL não é afetado pela instrução, quando usado como contador.

Formato: SAR destino,1 ou

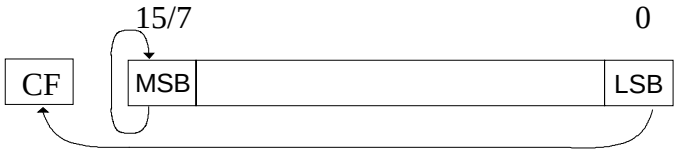
SAR destino, CL

Flags: Afetados: CF, OF, PF, SF, ZF; indefinido AF.

Exemplos: SAR AL,CL ; Desloca o conteúdo do registrador AL, o número de vezes especificado em CL.

SAR BH,1 ; Desloca o conteúdo do registrador BH, uma posição para a direita.

SAR byte ptr[DI],1 ; Desloca o conteúdo da posição de memória apontada por DI, no segmento de dados, uma posição para a direita.



i1) SHR - Deslocamento lógico à direita

Propósito: Deslocar o conteúdo da locação de memória ou registrador especificado à direita, inserindo um bit 0 na posição do bit mais significativo do operando e levando o bit 0 (menos significativo)para o flag CF, cujo valor inicial é perdido. O número de

vezes a deslocar é 1, ou aquele contido no registrador CL. O conteúdo de CL não é afetado pela instrução, quando usado como contador.

Formato: SHR destino,1 ou
SHR destino, CL

Flags: Afetados: CF, OF, PF, SF, ZF; indefinido AF.

Exemplos: SHR AL,CL ; Desloca o conteúdo do registrador AL, o número de vezes especificado em CL.
SHR BH,1 ; Desloca o conteúdo do registrador BH, uma posição para a direita.
SHR byte ptr[DI],1 ; Desloca o conteúdo da posição de memória apontada por DI, no segmento de dados, uma posição para a direita.

3.5 - Instruções de transferência de controle

Os microprocessadores 80XXX permitem que o controle do fluxo de execução seja transferido de forma condicional ou incondicional. O modo condicional de transferência é feito a partir do estado ou de combinação de estados dos flags. As transferências incondicionais são sempre executadas.

3.5.1 - Instruções de transferência de controle incondicional

As instruções que realizam a transferência incondicional para um endereço destino em um segmento de código são: JMP, CALL, RET, INT e IRET. O endereço destino pode estar no mesmo segmento de códigos (transferência near - próxima) ou em um segmento de códigos diferente (transferência far - distante).

j1) JMP - Desviar

Propósito: Provocar um desvio incondicional no fluxo de processamento, transferindo a execução para o operando alvo.

O desvio pode ser feito dentro ou fora do atual segmento de código. No caso de desvios dentro do próprio segmento, denominados desvios intra-segmento, apenas o registrador IP é afetado pela instrução, sendo alterado para o valor endereço-alvo. Quando a distância entre o valor atual e o endereço-alvo, para onde se dá o desvio

incondicional está entre -128 a +127 bytes, pode-se especificar um desvio-curto (JMP SHORT), também chamado desvio incondicional relativo curto, com a vantagem de o deslocamento para o desvio ser expresso por um único byte, contendo um valor sinalizado que, ao ser somado ao valor de IP gerará o endereço-alvo. Se a distância relativa para onde se dará o desvio for maior que +127 ou -128, então o endereço relativo gerado será um valor sinalizado de 16 bits, o que possibilita desvios por todo o segmento corrente.

```
Exemplo:    label1:     -----------
                        -----------
            atual: JMP label1
```

A outra forma de desvio incondicional é o desvio inter-segmento. Neste caso os valores do CS e IP precisam ser alterados para que ocorra a migração de um segmento para outro. O código que identifica um desvio incondicional para outro segmento deve ser seguido por 4 bytes: os dois primeiros contendo o novo valor do IP e os dois seguintes o novo valor do CS.

```
Exemplo:            -----------
            atual: JMP novo_longe    ----> Segmento de código atual
                    -----------


                    -----------
    novo_longe: MOV AX,10H       ----> Segmento de código novo
                    -----------
```

As formas de especificar um desvio incondicional, vistas até aqui, são ditas diretas, pois o endereço da posição-alvo que deseja-se atingir é explicitamente especificado na instrução, quer com um valor numérico, quer por um “label”. A outra forma de executar-se um desvio incondicional é a indireta, onde o valor do endereço-alvo está armazenado em registrador(es) ou em posição(ões) de memória, o qual será movido para o registrador IP, no caso de desvio intra-segmento ou para os registradores IP e CS, no caso de desvio inter-segmento.

Formato: JMP alvo

Flags: Nenhum afetado

```
Exemplos:   JMP   desvio_curto; Direto, intra-segmento, relativo.
            JMP   desvio_intra ; Direto, intra-segmento.
```

```
JMP   desvio_inter ; Direto, inter-segmento.
JMP   CX           ; Indireto, intra-segmento, o endereço-alvo está
                     armazenado em CX.
JMP   dword ptr[BX]    ; Indireto, inter-segmento, o endereço-alvo está
                         armazenado nas 4 posições de memória a partir
de                       DS:BX.
```

k1) CALL - Chama uma procedure

Propósito: Chamar uma subrotina, alterando o fluxo normal de execução, processar uma sub-rotina e ao fim da mesma, permitir que a execução retorne ao ponto do programa imediatamente posterior à instrução de chamada da sub-rotina.

A sub-rotina chamada pode residir no mesmo segmento de código corrente ou não. Tem-se assim as chamadas de subrotinas intra-segmento e inter-segmento, respectivamente.

A sub-rotina pode ser chamada por via direta ou indireta. Chamadas diretas podem ser realizadas a “labels”(nomes associados a endereços de memória) dentro de segmentos de código, enquanto que, por via indireta, usa-se registradores que contêm em si, ou nas posições de memórias apontadas por eles, o valor do endereço onde se inicia a sub-rotina.

Para guardar o endereço da instrução a ser executada ao retornar da sub-rotina, o processador salva na pilha o conteúdo do IP, no caso de chamadas a subrotinas no mesmo segmento de código. Nas chamadas de sub-rotinas inter-segmento, além do valor de IP, também é necessário salvar o conteúdo do CS, pois ele será alterado para provocar o desvio para outro segmento. O processador primeiro salva o IP e só então o CS.

Formato: CALL alvo

Flags: Nenhum afetado

```
Exemplos: CALL  sub_proxima       ; Direta, intra-segmento
          CALL  sub_distante; Direta, inter-segmento
          CALL  BX          ; Indireta, intra-segmento
          CALL  dword ptr [SI]    ; Indireta, inter-segmento
```

l1) RET - Retorno de uma procedure

Propósito: Encerrar uma sub-rotina, transferindo o fluxo do processamento para a instrução seguinte à chamada da subrotina. O endereço desta instrução, para onde deve ser dado o retorno, foi colocado na pilha pela instrução CALL. Portanto, para encerrar uma sub-rotina "near", que reside no mesmo segmento, a instrução RET deverá apenas retirar uma palavra do topo da pilha e movê-la para o registrador IP. Para endereçar uma procedurre "far", que reside em um outro segmento de código, a instrução RET deverá retirar as duas palavras que foram armazenadas na pilha pela instrução CALL inter-segmento, restaurando os antigos valores de CS e IP.

Operação da instrução

- Retirar uma palavra do topo da pilha e move-la para o IP.
- Incrementar o SP em 2.

Se a rotina é do tipo inter-segmento, então:

- Retirar outra palavra do topo da pilha e move-la para o CS.
- Incrementar o SP em 2.

Formato: RET

Flags: Nenhum afetado

m1) INT - Interrupção

Propósito: Alterar o fluxo normal de execução do programa, desviando-se para uma rotina de interrupção, cujo vetor (endereço formado por CS:IP) está presente numa tabela nos primeiros 1024 bytes da memória.

A tabela de vetores de interrupção tem 256 entradas, cada qual com 4 bytes (os dois primeiros contendo o valor do IP e os dois seguintes o valor do CS), que indicam o endereço de uma rotina na memória.

Procedimento da instrução:

- Decrementa o SP em 2 e salva o registrador de flags na pilha
- Apaga os bits IF e TF, desabilitando futuras interrupções de hardware
- Decrementa o SP em 2, salva o CS na pilha e coloca a palavra de mais alta ordem do vetor selecionado em CS
- Decrementa o SP em 2, salva o IP na pilha e carrega-o com a palavra de baixa ordem dentro do vetor selecionado. A próxima instrução a ser executada será então a primeira dentro da rotina de interrupção.

Formato: INT tipo

Flags: Afetados: IF e TF

n1) IRET - Retorno de uma interrupção

Propósito: Retornar de uma rotina de tratamento de interrupção. Esta instrução simplesmente recupera da pilha o conteúdo dos registradores que foram automaticamente salvos, na ocorrência da interrupção.

Operação da instrução

- Retirar uma palavra do topo da pilha e move-la para o IP.
- Incrementar o SP em 2.
- Retirar outra segunda palavra do topo da pilha e move-la para o CS.
- Incrementar o SP em 2.
- Retirar outra terceira palavra do topo da pilha e move-la para o registrador de flags.
- Incrementar o SP em 2.

Formato: IRET

Flags: Todos são afetados

3.5.2 - Instruções de transferência de controle condicional

As instruções de controle condicional são desvios (jumps) que transferem a execução do programa caso o estado de um flag seja o mesmo que o especificado pela instrução.

o1) J(condição) - Desvio curto se condição é satisfeita

Propósito: Desviar o fluxo do processamento para o operando-alvo, se uma condição testada for encontrada. A distância do endereço-alvo, para onde se quer direcionar o desvio, deve estar na faixa de -128 a +127 bytes da próxima instrução.

Observação: *Above* e *below* referem-se a relação entre dois valores não sinalizados e *greater* e *less*, à relação entre dois valores sinalizados

Desvios condicionais não sinalizados		
Mnemônico	Estado dos flags	Descrição
JA/JNBE	(CF ou ZF)=0	Acima/Não abaixo ou igual
JAE/JNB	CF=0	Acima ou igual/Não abaixo
JB/JNAE	CF=1	Abaixo/Não acima ou igual
JBE/JNA	(CF ou ZF)=1	Abaixo ou igual/Não acima
JC	CF=1	Carry
JE/JZ	ZF=1	Igual/Zero
JNC	CF=0	Não carry
JNE/JNZ	ZF=0	Não igual/Não zero
JNP/JPO	PF=0	Sem paridade/Paridade ímpar
Desvios condicionais sinalizados		
JG/JNLE	((SF ⊕ OF) ou ZF)=0	Maior/Não menor ou igual

JGE/JNL	(SF ⊕ OF)=0	Maior ou igual/Não menor
JL/JNGE	(SF ⊕ OF)=1	Menor/Não maior ou igual
JLE/JNG	((SF ⊕ OF) ou ZF)=1	Menor ou igual/Não maior
JNO	OF=0	Sem estouro (overflow)
JNS	SF=0	Sem sinal (não negativo)

Formato: J(condição) alvo_curto

Flags: Nenhum afetado

3.5.3 - Instruções de LOOP

As instruções de loop são desvios condicionais que se utilizam do registrador CX como contador do número de vezes que um bloco de instruções deve ser executado. Todas as instruções de loop sempre decrementam o conteúdo do registrador CX quando executadas e terminam se esta operação zerar o mesmo. Em quatro das cinco formas possíveis, o flag ZF pode ser utilizado para finalizar o loop mesmo que o conteúdo do registrador CX não seja zero.

m1) LOOP - Retornar até a contagem ser completada

Propósito: Decrementar o conteúdo do registrador contador CX e provocar um desvio no fluxo do processamento para o endereço-alvo, se o valor em CX ainda não for zero. A distância para o desvio deve estar na faixa de 128 bytes para trás ou 127 para a frente, em relação ao endereço da instrução que segue o loop.

Formato: LOOP alvo

Flags: Nenhum afetado

n1) LOOPE/LOOPZ Retornar se igual / zero

Propósito: Decrementar o conteúdo do registrador contador CX e provocar um desvio no fluxo do processamento para o endereço-alvo, se o valor em CX ainda não for zero e se o flag ZF=1. A distância para o desvio deve estar na faixa de 128 bytes para trás ou 127 para a frente, em relação ao endereço da instrução que segue o loop.

Formato: LOOPE alvo ou
LOOPZ alvo

Flags: Nenhum afetado

o1) LOOPNE/LOOPNZ Retornar se diferente / não zero

Propósito: Decrementar o conteúdo do registrador contador CX e provocar um desvio no fluxo do processamento para o endereço-alvo, se o valor em CX ainda não for zero e se o flag ZF=0. A distância para o desvio deve estar na faixa de 128 bytes para trás ou 127 para a frente, em relação ao endereço da instrução que segue o loop.

Formato: LOOPNE alvo ou
LOOPNZ alvo

Flags: Nenhum afetado

3.6 - Operações com string

As operações com string manipulam grande quantidade de dados em memória, como por exemplo, uma cadeia de caracteres. As instruções são:

- MOVS - Move string
- CMPS - Compara string
- SCAS - Scan (busca) string
- LODS - Load (carrega) string
- STOS - Store (armazena) string

O processador possui dois registradores especiais para manipulação de strings. São eles:

- SI - Registrador de índice de fonte
- DI - Registrador de índice de destino

Após a execução de uma operação string, estes registradores são automaticamente incrementados ou decrementados em acordo com o valor do flag de direção. O incremento ou decremento, será de acordo com o tamanho do elemento da string, o qual pode ser um byte ou uma word.

p1) MOVS/MOVSB/MOVSW - Move string de bytes ou words

Propósito: Mover o byte ou word endereçada pelo conteúdo de SI, dentro do segmento de dados, para a posição de memória indicada pelo conteúdo de DI, dentro do segmento extra. Após a operação, ambos os registradores de índice (SI e DI) são automaticamente incrementados ou decrementados, caso o flag de direção esteja resetado ou setado, respectivamente. O valor do incremento/decremento é de 1 ou 2, dependendo se a operação envolve dados do tipo byte ou word, respectivamente.

Diante desta instrução, pode-se utilizar o prefixo REP, que provocará uma repetição contínua, de acordo com o número de vezes especificado em CX, permitindo assim a movimentação de blocos de memória.

Formato: MOVSB ou MOVSW

Flags: Nenhum afetado

```
Exemplo:   MOV SI,1000H     ; Endereça a área de origem
           MOV DI,1500H     ; Endereça a área de destino
           MOV CX,200H      ; Contador em CX
           CLD              ; Reseta o flag de direção
           REP  MOVSW             ; Move 200H words
```

g1) CMPS/CMPSB/CMPSW - Compara duas string do tipo byte ou word

Propósito: Comparar, efetuando uma subtração entre o byte ou word endereçada por DI dentro do segmento extra e o byte ou word endereçada por SI dentro do segmento de dados. Após a subtração os flags são alterados de acordo com o resultado mas o mesmo é desprezado. A instrução incrementa/decrementa automaticamente os registradores de índice SI e DI desde que o flag de direção esteja resetado ou setado, respectivamente. O incremento/decremento será de 1 ou 2, caso os dados comparados sejam do tipo byte ou word, respectivamente.

Formato: CMPSB ou CMPSW

Flags: Afetados AF, CF, OF, PF, SF, ZF.

Exemplos: Considere que o registrador SI contenha o valor 1255H, DI contenha o valor 0761H, DF=0, a posição apontada por SI dentro do segmento de dados contenha a palavra 9988H e a posição apontada por DI dentro do segmento extra contenha a palavra 9988H. Após a execução de CMPSW, o registrador SI conterá o valor 1257H, DI terá 0763H e os flags CF e ZF, os mais relevantes de serem avaliados após a instrução, estarão sinalizando:

ZF = 1: os conteúdos dos operandos são iguais

CF = 0: O elemento indicado por DI é menor ou igual ao elemento indicado por SI

r1) SCAS/SCASB/SCASW - Busca byte ou word

Propósito: Comparar o elemento indicado por DI, dentro do segmento extra, com o conteúdo do acumulador. A instrução efetua a subtração do byte ou palavra indicada por DI do conteúdo do acumulador AL ou AX, afetando porém somente os flags, e não os operandos. Após a comparação, o registrador DI é automaticamente incrementado ou decrementado, dependendo do estado do flag de direção. O valor do incremento ou decremento é 1 ou 2, dependendo se a comparação foi realizada entre operandos do tipo byte ou word, respectivamente.

Diante desta instrução, pode-se utilizar o prefixo REP, que provocará uma repetição contínua, de acordo com o número de vezes especificado em CX, permitindo assim repetir a comparação em uma seqüência de elementos, até encontrar um que seja igual ou diferente ao conteúdo do acumulador.

Formato: SCASB ou SCASW

Flags: Afetados: AF, CF, OF, PF, SF, ZF.

```
Exemplo:    MOV  CX,100          ; Grava em CX 0100H
            MOV  DI,400          ; Grava em DI 0400H, endereço inicial de pesquisa
            MOV  AL,FF     ; Grava em AL FFH, o dado a ser comparado
            CLD            ; Reseta o flag de direção, DI será incrementado
            REPNZ SCASB    ; Compara e repete enquanto CX ≠0 e ZF=0
```

s1) STOS/STOSB/STOSW - Grava na memória um byte ou word armazenado no acumulador

Propósito: Mover o conteúdo do acumulador para a posição de memória indicada pelo registrador DI, dentro do segmento extra de dados. STOSB move o conteúdo de AL para a posição indicada por ES:DI e STOSW move o conteúdo de AX para a posição de memória indicada por ES:DI. Após a movimentação dos dados, o registrador DI é automaticamente incrementado ou decrementado, dependendo do estado do flag de direção. O valor do incremento ou decremento é 1 ou 2, dependendo se foi movido um byte ou uma word, respectivamente.

Diante desta instrução, pode-se utilizar o prefixo REP, que provocará uma repetição contínua, de acordo com o número de vezes especificado em CX, permitindo assim que um bloco de memória seja preenchido com um dado valor.

Formato: STOSB ou STOSW

Flags: Nenhum afetado

```
Exemplo:    CLD              ; Reseta o flag de direção
            MOV DI,1000H     ; Endereço inicial da área destino
            MOV CX, 300H     ; Contador em CX
            MOV AX, FFFFH    ; Valor a preencher as posições a partir de ES:DI
        REP STOSW            ; Preenche 300H words com FFFFH
```

t1) LODS/LODSB/LODSW - Grava no acumulador um byte ou word copiado da memória

Propósito: Mover um byte ou word da posição de memória indicada pelo registrador SI, dentro do segmento de dados para o acumulador. LODSB move para AL um byte na posição indicada por DS:SI e LODSW move para AX uma word na posição de memória indicada por DS:SI. Após a movimentação dos dados, o registrador SI é automaticamente incrementado ou decrementado, dependendo do estado do flag de direção. O valor do incremento ou decremento é 1 ou 2, dependendo se foi movido um byte ou uma word, respectivamente.

Formato: LODSB ou LODSW

Flags: Nenhum afetado

Exemplo: Considere que o registrador SI contenha o valor 1234H, DF=0 e que a posição de memória indicada por SI, dentro do segmento de dados, contenha a palavra 1100H. Após a instrução LODSW , o registrador AX conterá o valor 1100H e o registrador SI conterá o valor 1236H, indicando a próxima palavra a ser acessada.

3.6.1 - Prefixos de repetição

Os prefixos de repetição, especificam mais de uma execução de uma mesma instrução de string. Quando acompanha uma instrução de string, a operação é repetida até que uma das condições de término, especificada no prefixo, seja satisfeita

Para cada repetição, a operação pode ser suspensa por uma interrupção ou exceção. Após ter sido atendida a exceção ou a interrupção, a operação reinicializa do ponto onde foi suspensa. Desta forma, a resposta do sistema a uma interrupção não fica comprometida pela execução de uma instrução com string.

u1) REP/REPZ ou REPE/REPNZ ou REPNE - Repita operação de string

Propósito: Prefixos usados para repetir a instrução string seguinte pelo número de vezes especificado em CX.

Diante das instruções MOVS, LODS e STOS, pode-se usar o prefixo REP, que decrementa o registrador CX (sem afetar os flags) e repete a instrução string enquanto CX ≠ 0.

Diante das instruções CMPS e SCAS, pode-se usar os prefixos REPZ ou REPNZ. REPZ decrementa o registrador CX (sem afetar os flags) e repete a instrução string enquanto CX ≠ 0 e ZF =1. O prefixo REPNZ decrementa o contador CX (sem afetar os flags) e provoca a repetição da instrução string enquanto CX ≠ 0 e ZF=0.

Formato: REP instrução_string
REPZ instrução_string
REPNZ instrução_string

Flags: Serão aqueles afetados pela instrução_string

Exemplo: Considere que se queira preencher um bloco de memória, com 512 bytes de extensão, com o valor inicial 20H.

```
CLD             ; Reseta o flag de direção
MOV  CX,200             ; Contador em CX
MOV  DI,1000    ; Endereça a área de memória
MOV  AX,2020    ; Dados em AH e AL
REP  STOSW      ; Preenche 200 palavras a partir de ES:1000 (ES:DI)
```

3.7 - Instruções de controle de flag

3.7.1 - Instruções de controle do flag de direção, flag carry e de interrupção

As instruções de controle de carry são úteis em conjunto com as instruções de rotações com carry com o RCL e RCR. Pode-se a partir delas, inicializar o conteúdo do flag carry que será carregado em um operando.

v1) CLC - Reseta o flag carry

Propósito: Resetar o bit CF no registrador de flags

Formato: CLC

Flags: Afetado: CF

x1) CMC - Complementa o flag carry

Propósito: Complementar o estado do bit CF no registrador de flags. Se o valor de CF era 1, ele passará a 0, e vice-versa.
Formato: CMC
Flags: Afetado: CF

w1) STC - Seta o flag carry

Propósito: Setar o bit CF no registrador de flags
Formato: STC
Flags: Afetado: CF

y1) CLD - Reseta o flag de direção

Propósito: Resetar o bit DF no registrador de flags, forçando assim, o incremento dos registradores de índice após a execução de uma das instruções string.
Formato: CLD
Flags: Afetado: DF

z1) STD - Seta o flag de direção

Propósito: Setar o bit DF no registrador de flags, forçando assim, o decremento dos registradores de índice após a execução de uma das instruções string.
Formato: STD
Flags: Afetado: DF

a2) CLI - Reseta o flag de interrupção

Propósito: Resetar o bit IF no registrador de flags, desabilitando as interrupções mascaráveis externas, que aparecem na linha INTR do processador.
Formato: CLI
Flags: Afetado: IF

b2) STI - Seta o flag de interrupção

Propósito: Setar o bit IF no registrador de flags, habilitando as interrupções mascaráveis externas, que aparecem na linha INTR do processador.
Formato: STI
Flags: Afetado: IF

3.7.2 - Instruções de transferência de flags

Apenas os flags CF, DF e IF, possuem instruções especificas que permitem a alteração de seus estados diretamente. Para acessar os demais indicadores as instruções de transferência de flags devem ser utilizadas para movimentar o conteúdo do registrador de flags (F) para o registrador AH ou a pilha, onde então, poderão ser alterados e em seguida retornados para o registrador de flags (F).

c2) LAHF - Carregar o conteúdo do registrador de flags em AH

Propósito: Mover os bits dos registradores de flags SF, ZF, AF, PF e CF para o registrador AH, mantendo a mesma posição relativa que ocupam na parte baixa no registrador de flags. Os demais bits não são movidos..

Formato: LAHF

Flags: Nenhum afetado

d2) SAHF - Armazena conteúdo de AH no registrador de flags

Propósito: Transferir bits específicos do registrador AH, mantendo a mesma posição relativa, para a parte baixa do registrador de flags. Os demais bits não são movidos.

(SF) = bit 7 de AH

(ZF) = bit 6 de AH

(AF) = bit 4 de AH

(PF) = bit 2 de AH

(CF) = bit 0 de AH

Formato: SAHF

Flags: Afetados: AF, CF, PF, SF, ZF.

e2) PUSHF - Grava o conteúdo do registrador de flags na pilha

Propósito: Colocar, na área de memória usada como pilha, o conteúdo do registrador de flags.

Modo de operação: Idêntico ao da instrução PUSH

Formato: PUSHF

Flags: Nenhum afetado

f2) POPF - Copia word da pilha para o registrador de flags.

Propósito: Retirar a palavra no topo da pilha e movê-la para o registrador de flags. Sua operação é idêntica a da instrução POP.

Formato: POPF

Flags: Todos são afetados

3.8 - Instruções de acesso a portas de entrada e saída

g2) IN - Lê byte ou word de uma porta de E/S

Propósito: Transferir dados (byte ou word) de uma porta de entrada para o acumulador AL ou AX.

A porta é identificada com um valor de 8 bits que permite a acesso a portas cujos endereços variam de 0 a 255, ou com o uso de DX que, sendo de 16 bits, permite a especificação de endereços de portas que variam de 0 a 65535.

Formato: IN acumulador, porta ou
IN acumulador, DX

Flags: Nenhum afetado

```
Exemplo:  IN AL, 38H  ; Transfere para AL, o byte na porta de entrada 38H.
          IN AX,DX    ; Transfere para AX, a word na porta de entrada especificada
                        pelo registrador DX.
```

h2) OUT - Escreve byte ou word em uma porta de E/S

Propósito: Transferir um byte ou uma word, presente no acumulador AL ou AX, respectivamente, para uma porta de saída.

A porta é identificada com um valor de 8 bits que permite a acesso a portas cujos endereços variam de 0 a 255, ou com o uso de DX que, sendo de 16 bits, permite a especificação de endereços de portas que variam de 0 a 65535.

Formato: OUT porta, acumulador ou
OUT DX, acumulador

Flags: Nenhum afetado

```
Exemplo:  OUT 61H,AL  ; Transfere para a porta de saída 61H, o byte em AL.
          OUT DX,AL   ; Transfere para a porta de saída especificada em DX, o byte
                        gravado em AL.
```

3.9 - Instruções de uso geral

Algumas instruções não se encaixam em nenhuma das categorias apresentadas até agora. No entanto, como são importantes, são reunidas nesse bloco de instruções de uso geral.

i2) LEA - Carrega endereço efetivo em um registrador

Propósito: Carregar o *offset* de um endereço ou deslocamento de um operando na memória, para um registrador de 16 bits, especificado na própria instrução.

Formato: LEA destino, fonte

Flags: Nenhum afetado

```
Exemplos:   mensagem   DB     'Curso Assembler $'
            -
            LEA    DX, mensagem
            LEA    SI,[BX+DI+5]
```

No primeiro caso, o registrador DX conterá o valor do endereço associado ao "label" *mensagem*; no segundo, o registrador SI conterá o valor do endereço de memória do operando indicado pela somatória de BX e DI com 5.

j2) XCHG - Trocar

Propósito: Permutar o conteúdo de dois operandos, que podem ser dois registradores ou um registrador e uma posição de memória..

Formato: XCHG destino,fonte

Flags: Nenhum afetado

```
Exemplos:   XCHG AL,CH          ; Permuta o conteúdo de AL e CH
            XCHG        SI,DI           ; Permuta o conteúdo de SI e DI
            XCHG AL,[BX+2]  ; Permute o conteúdo de AL com o da posição de
                            memória apontada por (BX+2)
            XCHG [SI],AX        ; Permuta o conteúdo da posição apontada por SI com
o                               conteúdo de AL
```

k2) XLAT - Converter

Propósito: Converter um valor presente em AL acessando uma tabela previamente endereçada por BX, com no máximo 256 valores, usando o valor em AL como índice desta tabela. O Byte assim endereçado é então colocado no próprio registrador AL.

Formato: XLAT

Flags: Nenhum afetado

Exemplo: Considere que no endereço DS:0300 de memória, haja uma tabela com os seguintes valores:

12 14 16 18 20 22 24 26 28 30 32 34...

Após a seqüência de instruções

```
MOV BX,0300H
MOV AL,04H
XLAT
```

o registrador AL conterá o valor 20. Isto é, o valor inicial 04 foi convertido para 20 (elemento de índice 04). O primeiro elemento da tabela tem índice 00.