ASSEMBLER PARA IBM-PC

Prof. Luciano Scandelari, PhD

SUMÁRIO

1. MOVIMENTAÇÃO DE DADOS

MOV MOVIMENTACAO DE DADOS

```
              MOV  DESTINO,ORIGEM
                   ---+--- --+---  ! REGISTRADOR
                      !      +---->! MEMORIA
                      !            ! DADO IMEDIATO
                      !
                      +----------->! REGISTRADOR
                                   ! MEMORIA
```

- OPERANDOS DE ORIGEM E DESTINO DEVEM TER A MESMA EXTENSAO.
- O REGISTRADO IP NAO PODE SER UTILIZADO.
- O REGISTRADOR CS NAO PODE SER DESTINO.
- NENHUM REGISTRADOR DE SEGMENTO PODE RECEBER DADO IMEDIATO.
- DESTINO E ORIGEM NAO PODEM SER AMBOS MEMORIA.
- O REGISTRADOR DE FLAGS NAO PODE SER UTILIZADO.
- ESTA INSTRUCAO NAO AFETA NENHUMA FLAG.
- EXISTEM SETE TIPOS DE INSTRUCOES MOV.

1)DADO_IMEDIATO_PARA_REGISTRADOR

```
MOV AX,1234H      ;MOVE WORD 1234 HEXA PARA AX
MOV AH,12H        ;MOVE BYTE 12 HEXA PARA AH
MOV AL,34H        ;MOVE BYTE 34 HEXA PARA AL
MOV SI,0A00H      ;MOVE WORD 0A00 HEXA PARA SI
MOV CX,120        ;MOVE 120 DECIMAL PARA CX
MOV BH,10100101B  ;MOVE 10100101 BINARIO PARA BH
```

2)DE_REGISTRADOR_PARA_REGISTRADOR

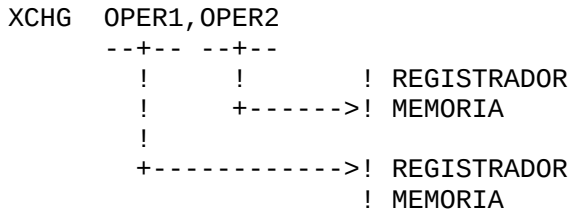
```
MOV AH,BL         ;MOVE CONTEUDO DE BL PARA AL
MOV CX,AX         ;MOVE CONTEUDO DE AX PARA CX
MOV SI,BX         ;MOVE CONTEUDO DE BX PARA SI
```

3)DE_MEMORIA/REGISTRADOR_PARA_REGISTRADOR/MEMORIA

```
MOV CL,[SI]           ;MOVE O BYTE CONTIDO NO OFFSET
                      ;DADO POR SI PARA CL
MOV BX,[DI+1000]      ;MOVE O WORD CONTIDO NO OFFSET
                      ;DI+1000 PARA O REG.BX
MOV POSICAO[BX+12],BH ;MOVE CONTEUDO DE BH PARA O
                      ;OFFSET BX+12 DE 'POSICAO'
MOV [BX+SI],DH        ;MOVE O BYTE CONTIDO EM DH
                      ;PARA O OFFSET DADO PELA SOMA
                      ;DOS CONTEUDOS DE BX E SI
```

```
4)DE_MEMORIA_OU_REGISTRADOR_PARA_REGISTRADOR_DE_SEGMENTO

        MOV ES,BX    ;MOVE VALOR DE BX PARA ES
        MOV DS,[SI]  ;MOVE CONTEUDO DO OFFSET [SI] PARA DS
        MOV SS,[BX]  ;MOVE CONTEUDO DO OFFSET [BX] PARA SS
                      *** O registrador CS nao pode ser
                          usado como destino.

5)DE_REGISTRADOR_DE_SEGMENTO_PARA_MEMORIA_OU_REGISTRADOR

        MOV AX,CS    ;MOVE CONTEUDO DE CS PARA AX
        MOV DX,ES    ;MOVE CONTEUDO DE ES PARA DX
        MOV [20],DS  ;MOVE CONTEUDO DE DS PARA O OFFSET
                     ;DE MEMORIA 20.
                      *** O registrador CS pode ser origem.


6)DO_ACUMULADOR/MEMORIA_PARA_MEMORIA/ACUMULADOR

        MOV [SI],AL    ;MOVE BYTE DE AL PARA OFFSET [SI]
        MOV [BX+DI],AX ;MOVE WORD DE AX PARA OFFSET DADO
                       ;PELA SOMA DE BX+DI
        MOV [210],AL   ;MOVE BYTE DE AL PARA OFFSET 210
        MOV AL,[BX+2]  ;MOVE CONTEUDO DO OFFSET [BX+2]
                       ;PARA O AL
        MOV AX,[SI]    ;MOVE O WORD DO OFFSET [SI] PARA AX
        MOV AL,[300]   ;MOVE O BYTE DO OFFSET [300] PARA AL

        MOV MEM1,AH    ;MOVE O CONTEUDO DE AH PARA O
                       ;OFFSET DECLARADO COMO 'MEM1'.
        MOV BL,MEM1    ;MOVE O CONTEUDO DO OFFSET DECLARADO
                       ;'MEM1' PARA O REGISTRADOR BL.


7)DADO_IMEDIATO_PARA_MEMORIA

        MOV byte ptr [SI],0Fh       ;MOVE O BYTE 0FH PARA
                                    ;O OFFSET DADO POR [SI]
        MOV word ptr [BX+200],3Fh   ;MOVE O WORD 003FH PARA
                                    ;O OFFSET [BX+200]
        MOV word ptr [1000h],0ABCDh ;MOVE O WORD ABCDH PARA
            ----+---                ;O OFFSET [1000H]
                !
                +---> BYTE PRT e' usado para referenciar
                !              um byte na memoria.
                !
                +---> WORD PTR e' usado para referenciar
                               um word na memoria.
```

XCHG TROCA DE DADOS ENTRE REGISTRADOR OU MEMORIA

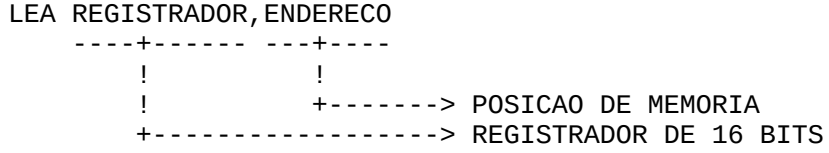


- ESTA INSTRUCAO PERMUTA (TROCA) OS CONTEUDOS DOS DOIS OPERANDOS.
- NAO E' PERMITIDO O USO DE REGISTRADORES DE SEGMENTO.
- O REGISTRADOR IP TAMBWM NAO PODE SER UTILIZADO.
- AMBOS OS OPERANDOS NAO PODEM SER MEMORIA.
- OS DOIS OPERANDOS DEVEM TER O MESMO TAMANHO.
- NAO AFETA NENHUMA FLAG.

EXEMPLOS:

```
XCHG AX,BX       ;TROCA OS CONTEUDOS DE AX E BX
XCHG BL,VARIAVEL ;TROCA O CONTEUDO DE BL COM O
                 ;CONTEUDO DO OFFSET 'VARIAVEL'
XCHG AL,BL       ;TROCA O CONTEUDO DE AL COM BL
```

LEA OBTEM O OFFSET DO ENDERECO EFETIVO DE UM DADO



- OBTEM O OFFSET OU DESVIO DO ENDERECO EFETIVO DE UMA VARIAVEL NA MEMORIA.
- O REGISTRADOR DEVE SER DE 16 BITS E NAO PODE SER CS NEM IP.
- OBSERVE AS INSTRUCOES A SEGUIR :

```
MOV BX,VAR1
MOV BX,OFFSET VAR1
LEA BX,VAR1
```

- A PRIMEIRA INSTRUCAO CARREGA O REGISTRADOR BX COM O VALOR DA VARIAVEL VAR1, ISTO E' , COM O CONTEUDO DAQUELA POSICAO DE MEMORIA.

- A SEGUNDA INSTRUCAO CARREGA O REGISTRADOR BX COM O OFFSET DA POSICAO DE MEMORIA VAR1.
- A TERCEIRA INSTRUCAO CARREGA O REGISTRADOR BX COM O OFFSET DA POSICAO DE MEMORIA VAR1.
- A DIFERENCA ENTRE MOV REG,OFFSET MEM E LEA REG,MEM SAO DUAS:

  1)A INSTRUCAO LEA CALCULA O OFFSET EM QUALQUER SITUACAO , INCLUSIVE COM O USO DE REGISTRADORES DE INDICES.
  2)A INSTRUCAO LEA E' MAIS LONGA E DEMORADA.

- PARA ENTENDERMOS A INSTRUCAO LEA ,VEJAMOS OS EXEMPLOS ABAIXO.

```
DADOS  SEGMENT
VAR1   DB       1AH          ;COLOCA 1 BYTE EM VAR1
VAR2   DW       0123H,4567H  ;COLOCA 2 WORDS EM VAR2
VAR3   DW       9876H        ;COLOCA 1 WORD EM VAR3
DADOS  ENDS
CODIGO SEGMENT
       ASSUME   CS:CODIGO,DS:DADOS
         .
         .
         .
       MOV AX,VAR3        ;CARREGA AX COM O WORD 9876H
       MOV AX,OFFSET VAR3 ;CARREGA AX COM O NUMERO 5
       LEA AX,VAR3        ;CARREGA AX COM O NUMERO 5
         .
         .
         .
CODIGO ENDS
```

LDS CARREGA UM PONTEIRO PARA DS E OUTRO REGISTRADOR

```
LDS REGISTRADOR,ENDERECO
    -----+----- ---+----
         !         !
         !         +-------> POSICAO DE MEMORIA
         +-----------------> REGISTRADOR DE 16 BITS
```

- CARREGA O PONTEIRO DE ENDERECO , EM 32 BITS.
- O REGISTRADOR DS RECEBE O SEGMENTO AO QUAL PERTENCE ENDERECO.
- O OUTRO REGISTRADOR DE 16 BITS ESPECIFICADO NA INSTRUCAO RECEBE O OFFSET DE ENDERECO.
- O REGISTRADOR DE 16 BITS PODE SER BP,SI,DI,AX,BX,CX OU DX.
- OBSERVE A INSTRUCAO ABAIXO :

```
                    ! AX <--- OFFSET DE VAR2
LDS AX,VAR2 ;       !
                    ! DS <--- SEGMENTO EM QUE ESTA' VAR2
```

LES CARREGA UM PONTEIRO PARA ES E OUTRO REGISTRADOR

```
          LES REGISTRADOR,ENDERECO
              -----+----- ---+----
                   !         !
                   !         +-------> POSICAO DE MEMORIA
                   +-----------------> REGISTRADOR DE 16 BITS
```

- CARREGA O PONTEIRO DE ENDERECO , EM 32 BITS.
- O REGISTRADOR ES RECEBE O SEGMENTO AO QUAL PERTENCE ENDERECO.
- O OUTRO REGISTRADOR DE 16 BITS ESPECIFICADO NA INSTRUCAO RECEBE O OFFSET DE ENDERECO.
- O REGISTRADOR DE 16 BITS PODE SER BP,SI,DI,AX,BX,CX OU DX.
- OBSERVE A INSTRUCAO ABAIXO :

```
                        ! SI <--- OFFSET DE VAR1
      LDS SI,VAR1 ;     !
                        ! ES <--- SEGMENTO EM QUE ESTA' VAR1
```

2. MANIPULAÇÃO DA PILHA

PUSH COLOCA UM WORD NA PILHA DE DADOS

```
          PUSH  OPERANDO
                ----+---
                    +-------> REGISTRADOR DE 16 BITS OU
                              POSICAO DE MEMORIA (16 BITS)
```

- A INSTRUCAO PUSH COLOCA UM WORD NA PILHA.
- UTILIZA-SE COLOCAR UM DADO NA PILHA PARA SALVA-LO TEMPORARIAMENTE OU ENTAO COMO UMA TECNICA DE PASSAGEM DE PARAMETROS PARA ROTINAS EM OUTRAS LINGUAGENS.
- NAO PODE COLOCAR VALOR IMEDIATO.
- ESTA INSTRUCAO OPERA DA SEGUINTE FORMA :

A) O REGISTRADOR SP E' DECREMENTADO DE 2.
B) O OPERANDO LSB (MENOS SIGNIFICATIVO) E' COLOCADO NA POSICAO SS:[SP].
C) O OPERANDO MSB (MAIS SIGNIFICATIVO) E' COLOCADO NA POSICAO SS:[SP+1].

- COMO EXEMPLO OBSERVE AS INSTRUCOES A SEGUIR :

```
          MOV  BX,1234H ; COLOCA 12H EM BH E 34H EM BL
          PUSH BX       ; SALVA BX NA PILHA NA SEGUINTE FORMA

          MEMORIA
         !       ! <---+--- SS(INICIO DE SEGMENTO DE PILHA)
         !       !     !
         !       !     !
         !  12H  ! <---+ SP (OFFSET DO TOPO DA PILHA)
         !  34H  !
         !       ! <.... Valor anterior do sp
```

- OUTROS EXEMPLOS :

```
            PUSH AX
            PUSH BX
            PUSH CX
            PUSH DX
            PUSH DS
            PUSH SS
            PUSH SP; COLOCA SP NA PILHA  COM  O  SEU  VALOR  JA'
                   ; DECREMENTADO.
            PUSH CS:[BP+SI+1] ; CARREGA NA PILHA O CONTEUDO  DAS
                              ; POSICOES DE MEMORIA NO  SEGMENTO
                              ; CS,COM OFFSET BP+SI+1 E BP+SI+2.
```

POP RETIRA UM WORD DA PILHA DE DADOS

```
             POP  OPERANDO
                  ----+---
                      +-------> REGISTRADOR DE 16 BITS OU
                                POSICAO DE MEMORIA (16 BITS)
```

- A INSTRUCAO POP RETIRA UM WORD DA PILHA.
- UTILIZA-SE COLOCAR UM DADO NA PILHA PARA SALVA-LO TEMPORARIAMENTE OU ENTAO COMO UMA TECNICA DE PASSAGEM DE PARAMETROS PARA ROTINAS EM OUTRAS LINGUAGENS.
- NAO PODE COLOCAR VALOR IMEDIATO.
- ESTA INSTRUCAO OPERA DA SEGUINTE FORMA :

A) O OPERANDO LSB (MENOS SIGNIFICATIVO) RECEBE O CONTEUDO DA POSICAO SS:[SP].
B) O OPERANDO MSB (MAIS SIGNIFICATIVO) RECEBE O CONTEUDO DA POSICAO SS:[SP+1].
C) O REGISTRADOR SP E' INCREMENTADO DUAS UNIDADES.

- COMO EXEMPLO OBSERVE AS INSTRUCOES A SEGUIR :

```
            MOV  BX,1234H ; COLOCA 12H EM BH E 34H EM BL
            MOV  CX,89ABH ; COLOCA 89H EM CH E ABH EM CL
            PUSH BX       ; COLOCA BX NA PILHA
            PUSH CX       ; COLOCA CX NA PILHA
            POP  BX       ; BX RECEBE 89ABH DA PILHA
            POP  CX       ; CX RECEBE 1234H DA PILHA , PORTANTO
                            HOUVE UMA INVERSAO DOS VALORES , PORQUE
                            O ULTIMO DADO A ENTRAR NA PILHA DEVERIA SER
                            O PRIMEIRO A SAIR.
```

3. ARITIMÉTICA

ADICAO E SUBTRACAO

```
    ADD OPERANDO1,OPERANDO2   ;  OPERANDO1 <-- OPERANDO1 + OPERANDO2
    ADC OPERANDO1,OPERANDO2   ;  OPERANDO1 <-- OPERANDO1 + OPERANDO2 + CF
    SUB OPERANDO1,OPERANDO2   ;  OPERANDO1 <-- OPERANDO1 - OPERANDO2
    SBB OPERANDO1,OPERANDO2   ;  OPERANDO1 <-- OPERANDO1 - OPERANDO2 - CF
        ----+---- ----+----
            !         !
            !         !      ! REGISTRADOR
            !         +----->! MEMORIA
            !                ! DADO IMEDIATO
            !
            !                ! REGISTRADOR
            +--------------->! MEMORIA
```

- OPERANDO1 E OPERANDO2 DEVEM SER DO MESMO TAMANHO.
- OPERANDO1 E OPERANDO2 NAO PODEM SER AMBOS MEMORIA.
- AFETAM TODOS OS BITS DE FLAG.

INCREMENTO E DECREMENTO

```
      INC OPERANDO  ;  OPERANDO <-- OPERANDO + 1
      DEC OPERANDO  ;  OPERANDO <-- OPERANDO - 1
          ----+---
              !         ! MEMORIA
              +-------->! REGISTRADOR
```

- ESTA INSTRUCAO ALTERA TODAS AS FLAGS MENOS O CF.

MULTIPLICAÇÃO E DIVISÃO

```
DIV OPERANDO-8bits  ; AX |OPERANDO-8bits
                         +--------------
                       AH   AL

DIV OPERANDO-16bits ; DX:AX |OPERANDO-16bits
                            +--------------
                         AX    DX

MUL OPERANDO-8bits   ; AL x OPERANDO-8bits → AX

MUL OPERANDO-16bits  ; AX x OPERANDO-16bits → DX:AX
```

- OPERANDO = posicao de memória ou registrador
- DIV : Todos os flags ficam indefinidos
- MUL : Afeta CF, OF, os demais ficam indefinidos

4. SALTOS

ROTULOS PARA DEMARCACAO DE SALTOS

```
            ! NEAR -> SALTO NO MESMO SEGMENTO DE CODIGO.
            !         EXECUCAO MAIS RAPIDA.
            !
            !         ROTULO:
            !                     OU
            !         ROTULO LABEL NEAR
   ROTULO --!
            ! FAR  -> SALTO EM SEGMENTO DIFERENTE DO DE ORIGEM.
            !
            !         ROTULO LABEL FAR
            !
```

REGISTRADORES DE CONDICAO (FLAGS)

- 6 BITS DE ESTADO (STATUS FLAGS).
- 3 BITS DE CONTROLE (CONTROL FLAGS).
- 7 BITS NAO USADOS.

BITS DE ESTADO → CARACTERISTICAS DO RESULATADO GERADO NA U.L.A.

- CF - TRANSPOSICAO (CARRY).
- AF - TRANSPOSICAO AUXILIAR - USADO EM OPERACOES BCD.
- OF - SOBRECARGA (OVERFLOW) - SIMILAR AO CF , POREM PARA ALGEBRICOS.
- PF - PARIDADE - SETADA QUANDO O RESULTADO CONTEM UM NUM.PAR DE '1'.
- SF - SINAL - SETADA QUANDO O RESULTADO E' NEGATIVO.
- ZF - ZERO - SETADA QUANDO O RESULTADO GERADO NA U.L.A. E' ZERO.

BITS DE CONTROLE → CONTROLAM AS OPERACOES DO PROCESSADOR

- ESTAS FLAGS AFETAM AS CONDICOES PARA A EXECUCAO DE DETERMINADAS INSTRUC.

- DF - DIRECAO - E' USADA NA MANIPULACAO DE STRINGS.
  ! DF=0 -> STRING PROCESSANDO PARA FRENTE.
  ! DF=1 -> STRING PROCESSANDO PARA TRAS.

- IF - INTERRUPCAO - HABILITA/DESABILITA INTERRUPCOES MASCARAVEIS.
  ! IF=0 -> INTERRUPCOES DESABILITADAS - NAO ACEITA INTERRUPCOES.
  ! IF=1 -> INTERRUPCOES HABILITADAS - ACEITA INTERRUPCOES.

- TF - TRACE FLAG - HABILITA/DESABILITA MODO PASSO A PASSO.
  ! TF=1 -> APOS CADA INSTRUCAO A CPU EXECUTA UM INT 01.
  ! TF=0 -> PROCESSAMENTO NORMAL.

INSTRUCOES DE MANIPULACAO DE FLAGS.

- STC - SET CARRY - FAZ CF=1.
- CLC - CLEAR CARRY - FAZ CF=0.
- CMC - COMPLEMENTA CARRY - FAZ CF=CF
- STD - SET DIRECTION - FAZ DF=1.
- CLD - CLEAR DIRECTION - FAZ DF=0.
- STI - SET INTERRUPT - FAZ IF=1.
- CLI - CLEAR INTERRUPT - FAZ IF=0. (NAO AFETA INTERRUPCOES DE SOFTWARE)

JMP SALTO INCONDICIONAL

```
              JMP ENDERECO
                  ---+----
                     !         ! ENDERECO IMEDIATO
                     +-------> ! IMEDIATO VIA REGISTRADOR
                               ! INDIRETO VIA REGISTRADOR
```

- ANTES DO DESVIO , O IP E' ATUALIZADO PARA A INSTRUCAO SEGUINTE.

A)DESVIO DIRETO

- CURTO : A DISTANCIA E' SOMADA AO IP (-128 A +127).
- INTRA-SEGMENTO : A DISTANCIA E' SOMADA AO IP (-32768 A +32767).
- INTERSEGMENTO : CS E IP RECEBEM OS VALORES CONTIDOS NA INSTRUCAO.

B)DESVIO INDIRETO

- INTRA-SEGMENTO : IP RECEBE O VALOR CONTIDO NO OPERANDO.
- INTERSEGMENTO : CS E IP RECEBEM OS VALORES CONTIDOS NO OPERANDO.

- CADA TIPO DE DESVIO E' DETERMINADO PELO TIPO DE OPERANDO:

OPERANDO	DESVIO
REGISTRADOR	INDIRETO INTRA-SEGMENTO
VARIAVEL WORD	INDIRETO INTRA-SEGMENTO
VARIAVEL DWORD	INDIRETO INTERSEGMENTO
ROTULO 'NEAR'	DIRETO INTRA-SEGMENTO OU CURTO
ROTULO 'FAR'	DIRETO INTERSEGMENTO

- A SELECAO ENTRE DESVIO DIRETO INTRA-SEGMENTO E DESVIO DIRETO CURTO E' CONTROLADA PELO PROGRAMADOR (ATRAVES DO ATRIBUTO SHORT) E PELO ASSEMBLER.
- SE O DESVIO E' FEITO 'PARA TRAS' O ASSEMBLER DETERMINA AUTOMATICAMENTE SE E' POSSIVEL UTILIZAR UM DESVIO CURTO.
- SE O DESVIO E' FEITO 'PARA FRENTE' , O ASSEMBLER RESERVA ESPACO PARA UM DESVIO INTERSEGMENTO (3 BYTES). SE VERIFICAR-SE O USO DE DESVIO CURTO , O ESPACO EXTRA E' PREENCHIDO COM UMA INSTRUCAO 'NOP'.

- EXEMPLOS:

```
END_NEAR    DW    ?
END_FAR           DD    ?

ROT_NEAR    LABEL NEAR
ROT_FAR           LABEL FAR

XIS:        JMP   ROT_NEAR
            JMP   ROT_FAR
            JMP   END_FAR
            JMP   END_NEAR
            JMP   XIS
            MOV   AX,OFFSET ROT_NEAR
            JMP   AX
            MOV   END_NEAR,OFFSET ROT_NEAR
            MOV   BX,OFFSET END_NEAR
            JMP   [BX]
```

J CONDICIONAL SALTA SE A CONDICAO FOR SATISFEITA

- O MAIOR DESVIO POSSIVEL E' DE +127 OU -128 BYTES.
- NORMALMENTE SAO UTILIZADOS APOS UMA INSTRUCAO DE COMPARACAO (CMP) OU APOS INSTRUCOES LOGICAS OU ARITIMETICAS (ALTERAM OS FLAGS).

```
  CONDICOES      SALTA SE:

  JE / JZ        ZF=1    - RESULTADO = 0.
 JNE / JNZ       ZF=0    - RESULTADO NAO E' 0.

       JS        SF=1    - RESULTADO NEGATIVO.
       JNS       SF=0    - RESULTADO POSITIVO.

       JO        OF=1    - HOUVE OVERFLOW.
       JNO       OF=0    - NAO HOUVE OVERFLOW.

       JC        CF=1    - HOUVE CARRY.
       JNC       CF=0    - NAO HOUVE CARRY.

       JP        PF=1    - PARIDADE E' PAR.
       JNP       PF=0    - PARIDADE NAO E' PAR.

       JCXZ      CX=0000H - SALTA SE CX E' ZERO.
```

CMP COMPARACAO ENTRE DOIS VALORES

```
          CMP OPERANDO1,OPERANDO2
              ----+----  ----+----
                  !          !         ! REGISTRADOR
                  !          +-------->! MEMORIA
                  !                    ! DADO IMEDIATO
                  !
                  !                    ! REGISTRADOR
                  +------------------->! MEMORIA
                                       ! DADO IMEDIATO
```

- NAO SE PODE COMPARAR DOIS LOCAIS DE MEMORIA NA MESMA INSTRUCAO.
- NAO SE PODE COMPARAR DOIS DADOS IMEDIATOS NA MESMA INSTRUCAO.

- OS OPERANDOS DEVEM TER O MESMO TAMANHO E PODE SER 8 OU 16 BITS.
- A COMPARACAO E' FEITA TAL COMO UMA SUBTRACAO , ONDE O RESULTADO NAO E' GUARDADO , E AS CARACTERISTICAS DO RESULTADO IRAO SETAR AS FLAGS.
- OS VALORES DOS OPERANDOS NAO SAO ALTERADOS.

SALTOS CONDICIONAIS PARA NUMEROS ARITIMETICOS

- OS ROTULOS DEVERAO SER PROXIMOS (-128 A +127 BYTES).
- DEVEM SER UTILIZADOS APOS UMA INSTRUCAO ARITIMETICA OU LOGICA , OU AINDA APOS UMA COMPARACAO , SEGUINDO A CONVENCAO ARITIMETICA SEM SINAL.

  - JB - SALTA SE MENOR = JNAE
  - JA - SALTA SE MAIOR = JNBE
  - JBE - SALTA SE MENOR OU IGUAL = JNA
  - JAE - SALTA SE MAIOR OU IGUAL = JNB
  - JE - SALTA SE IGUAL
  - JNE - SALTA SE NAO FOR IGUAL

SALTOS CONDICIONAIS PARA NUMEROS ALGEBRICOS

- OS ROTULOS DEVERAO SER PROXIMOS (-128 A +127 BYTES).
- DEVEM SER UTILIZADOS APOS UMA INSTRUCAO ARITIMETICA OU LOGICA , OU AINDA APOS UMA COMPARACAO , SEGUINDO A CONVENCAO ARITIMETICA COM SINAL.

  - JL - SALTA SE MENOR = JNGE
  - JG - SALTA SE MAIOR = JNLE
  - JLE - SALTA SE MENOR OU IGUAL = JNG
  - JGE - SALTA SE MAIOR OU IGUAL = JNL
  - JE - SALTA SE IGUAL
  - JNE - SALTA SE DIFERENTE

LOOP DECREMENTA CX E SALTA ENQUANTO NAO FOR ZERO

```
            LOOP      ROTULO
```

- ESTA INSTRUCAO EQUIVALE A -->
```
! DEC CX
! JNZ ROTULO
```

- MUITO UTILIZADA PARA ALGORITIMOS DE REPETICAO.

5. SUBROTINAS

CALL	CHAMADA DE SUBROTINA

```
                    CALL ROTULO
                    CALL REGISTRADOR
                    CALL MEMORIA
```

- ANTES DA CHAMADA DA SUBROTINA , O IP E' ATUALIZADO PARA APONTAR PARA A INSTRUCAO SEGUINTE AO CALL , QUE NADA MAIS E' QUE O ENDERECO DE RETORNO DA SUBROTINA.
- AS CHAMADAS PODEM SER DIRETAS OU INDIRETAS , INTRA-SEGMENTO OU INTERSEGMENTO.

1-CHAMADA_DIRETA

A-INTRA_SEGMENTO

-SUBTRAI 2 DO SP.
-COLOCA O IP NA PILHA.
-SOMA AO IP O DESLOCAMENTO ENTRE O DESTINO E A INSTRUCAO SEGUINTE AO CALL.

B-INTERSEGMENTO

-SUBTRAI 2 DE SP.
-COLOCA O CS NA PILHA.
-SUBTRAI 2 DE IP.
-COLOCA O IP NA PILHA.
-CARREGA CS E IP COM O ENDERECO DE DESTINO.

2-CHAMADA_INDIRETA

A-INTRA_SEGMENTO

-SUBTRAI 2 DE SP.
-COLOCA IP NA PILHA.
-CARREGA IP COM O CONTEUDO DO OPERANDO.

B-INTERSEGMENTO

-SUBTRAI 2 DE SP.
-COLOCA O CS NA PILHA.
-SUBTRAI 2 DE SP.
-COLOCA O CS NA PILHA.
-CARREGA CS E IP COM O CONTEUDO DO OPERANDO.

- O TIPO DE CHAMADA E' DETERMINADA PELO OPERANDO:

OPERANDO	CHAMADA
ROTULO NEAR	DIRETA INTRA-SEGMENTO
ROTULO FAR	DIRETA INTERSEGMENTO
REGISTRADOR	INDIRETA INTRA-SEGMENTO
VARIAVEL WORD	INDIRETA INTRA-SEGMENTO
VARIAVEL DWORD	INDIRETA INTERSEGMENTO

RET RETORNO DE SUBROTINA

```
                RET

- ESTA INSTRUCAO RETORNA DA SUBROTINA SEGUINDO OS SEGUINTES PASSOS:

     1.CARREGA IP COM O TOPO DA PILHA.
     2.SOMA 2 A SP.
     3.SE O RETORNO E' INTERSEGMENTO , ENTAO CARREGA CS COM O TOPO DA PILHA,
         E SOMA 2 AO SP.

- O TIPO DE RETORNO INTRA-SEGMENTO OU INTERSEGMENTO E' DETERMINADO PELA
  DIRETIVA PROC/ENDP QUE CERCAM A INSTRUCAO RET.
- E' IMPORTANTE O CASAMENTO DO TIPO DO CALL COM O TIPO DO RET PARA A CORRETA
  RESTAURACAO DE CS,IP E SP.
- COMO O ENDERECO DE RETORNO E' ARMAZENADO NA PILHA , E' IMPORTANTE A SUBROTINA
  NAO ALTERAR O CONTEUDO DO TOPO DA PILHA NA OCASIAO DO RET , POIS DO CONTRARIO
  O ENDERECO DE RETORNO SERIA PERDIDO OU TOMADO ERRADAMENTE.
```

6. STRINGS

```
DEFINICAO DO STRING :
---------------------

          STRING1 DB  40 DUP (?)
          CORDAO  DB  "ABCDEFGHIJKLMNOP"
          ORIGEM  DW  60 DUP (0)


    OBS: A VANTAGEM DE SE UTILIZAR A DIRETIVA DUP RESIDE NO
         FATO DE PODERMOS UTILIZAR A DIRETIVA LENGHT NOS
         LABELS QUE UTILIZARAM DUP.

    LENGHT      STRING1 ----> VALE 40 (BYTES)
    LENGHT  ORIGEM  ----> VALE 60 (WORDS)

    NAO PODEMOS UTILIZAR "LENGHT CORDAO".
```

MOVSB E MOVSW MOVE STRINGS DE BYTES OU WORDS

```
- ESTA INSTRUCAO MOVE 1 BYTE/WORD DA ORIGEM PARA O DESTINO , COM O AJUSTE
  AUTOMATICO DE SI E DI.


                ! DS:SI (DEFAULT) - OFFSET DI NO SEGMENTO DE DADOS.
         ORIGEM-!
                ! ES:SI  (ES EXPLICITO) - OFFSET SI NO SEGMENTO EXTRA.


                !
        DESTINO-! ES:DI (DEFAULT E UNICO) - OFFSET DI NO SEGMENTO EXTRA.
                !
```

```
  OPERACAO DO MOVSB :
  -------------------

     -COPIA O BYTE APONTADO POR SI PARA ES:DI.
     -SE DF=0 SOMA 1 A SI E DI.
     -SE DF=1 SUBTRAI 1 DE SI E DI.

  OPERACAO DO MOVSW :
  -------------------

     -COPIA O WORD APONTADO POR SI PARA ES:DI.
     -SE DF=0 SOMA 2 A SI E DI.
     -SE DF=1 SUBTRAI 2 DE SI E DI.


- PARA SETARMOS O VALOR DE DF (FLAG DE DIRECAO) USAMOS AS INSTRUCOES:

     STD  ; FAZ DF=1 - SENTIDO DIRETO NAS OPERACOES COM STRING.
     CLD  ; FAZ DF=0 - SENTIDO INVERSO NAS OPERACOES COM STRINGS.



- SE O SEGMENTO DE DESTINO TIVER QUE SER O MESMO DE ORIGEM , FAZEMOS ES=DS :

     ASSUME CS:CODIGO,SS:PILHA,DS:DADOS,ES:DADOS
     MOV AX,DADOS
     MOV DS,AX
     MOV ES,AX

- PARA INICIALIZAR SI E DI :

     LEA SI,STRING_DE_ORIGEM
     LEA DI,STRING_DE_DESTINO



- NO SENTIDO INVERSO , PARA CARREGARMOS SI E DI , DEVEMOS FAZER :

     LEA SI , ORIGEM + SIZE ORIGEM - TYPE ORIGEM
     LEA DI , DESTIN + SIZE DESTIN - TYPE DESTIN
     MOV CX , LENGHT ORIGEM
```

REP PREFIXO DE REPETICAO

```
                        REP INSTRUCAO


- PREFIXO DE INSTRUCAO.
- DEVE SER COLOCADO NO CAMPO DOS ROTULOS.
- REPETE "CX" VEZES A INSTRUCAO.
- ENQUANTO CX FOR DIFERENTE DE ZERO A INSTRUCAO E' EXECUTADA E CX E'
  DECREMENTADO DE UMA UNIDADE.
- OS FLAGS NAO SAO AFETADOS PELO PRFIXO REP,MAS SIM PELA INSTRUCAO QUE
  ESTA' SENDO REPETIDA.
- SE CX INICIALMENTE FOR IGUAL A ZERO A INSTRUCAO NAO E' EXECUTADA.
- O PREFIXO REP E' UTLIZADO EM CONJUNTO COM AS INSTRUCOES MOVS,STOS E LODS.
- PARA AS INSTRUCOES SCAS E CMPS UTILIZAM-SE OS PREFIXOS REPZ,REPNZ,
```

REPE E REPNE.

EXEMPLO:

```
        LEA SI,STRING_ORIGEM
        LEA DI,STRING_DESTINO
        MOV CX,LENGHT CORDAO_ORIGEM
        CLD
REP     MOVSB
```

CMPSB E CMPSW - COMPARA DOIS STRINGS DE BYTES/WORDS

- VALIDAS AS MESMAS CONSIDERACOES A RESPEITO DE ORIGEM E DESTINO CITADAS PARA A INSTRUCAO MOVSB E MOVSW.

OPERACAO DO CMPSB:

1.SETA OS FLAGS CONFORME O RESULTADO DA COMPARACAO (SUBTRACAO)
DO BYTE APONTADO POR SI E O BYTE APONTADO POR ES:DI.
2.SE DF=0 SOMA 1 A SI E DI.
3.SE DF=1 SUBTRAI 1 DE SI E DI.

OPERACAO DO CMPSW:

1.SETA OS FLAGS CONFORME O RESULTADO DA COMPARACAO (SUBTRACAO)
DO WORD APONTADO POR SI E O WORD APONTADO POR ES:DI.
2.SE DF=0 SOMA 2 A SI E DI.
3.SE DF=1 SUBTRAI 2 DE SI E DI.

REPNZ/REPNE PREFIXO DE REPETICAO ENQUANTO DIFERENTE

```
REPNZ  INSTRUCAO
REPNE  INSTRUCAO
```

- PREFIXO DE INSTRUCAO.
- DEVE SER COLOCADO NO CAMPO DOS ROTULOS.
- ENQUANTO CX FOR DIFERENTE DE ZERO A INSTRUCAO E' EXECUTADA E CX E' DECREMENTADO DE UMA UNIDADE.
- CASO O FLAG Z FIQUE ATIVO (ZF=1) APOS A EXECUCAO DA INSTRUCAO , A REPETICAO SERA' INTERROMPIDA.
- OS FLAGS NAO SAO AFETADOS PELO PRFIXO REP,MAS SIM PELA INSTRUCAO QUE ESTA' SENDO REPETIDA.
- SE CX INICIALMENTE FOR IGUAL A ZERO A INSTRUCAO NAO E' EXECUTADA.
- O PREFIXO E' UTLIZADO EM CONJUNTO COM AS INSTRUCOES SCAS E CMPS.

- UMA VEZ QUE CX E' DECREMENTADO ANTES DO TESTE DA FLAG DE ZERO E AS INSTRUCOES SCAS E CMPS AVANCAM (OU RECUAM) OS PONTEIROS (SI E DI) APOS A COMPARACAO , AO FINAL DA REPETICAO TEREMOS :

1. SE ZF=0 --> NAO FOI ENCONTRADO ELEMENTO NA CADEIA QUE RESULTASSE EM ZF=1. CX CONTEM ZERO E OS PONTEIROS APONTAM PARA A POSICAO SEGUINTE AO FINAL DAS CADEIAS.

2. SE ZF=1 --> FOI ENCONTRADO UM ELEMENTO NA CADEIA QUE RESULTOU EM
ZF=1. CX E OS PONTEIROS CORRESPONDEM `A POSICAO SEGUINTE
A ESTE ELEMENTO.

REPZ/REPE PREFIXO DE REPETICAO ENQUANTO IGUAL

```
REPZ  INSTRUCAO
REPE  INSTRUCAO
```

- PREFIXO DE INSTRUCAO.
- DEVE SER COLOCADO NO CAMPO DOS ROTULOS.
- ENQUANTO CX FOR DIFERENTE DE ZERO A INSTRUCAO E' EXECUTADA E CX E' DECREMENTADO DE UMA UNIDADE.
- CASO O FLAG Z FIQUE INATIVO (ZF=0) APOS A EXECUCAO DA INSTRUCAO , A REPETICAO SERA' INTERROMPIDA.
- OS FLAGS NAO SAO AFETADOS PELO PREFIXO REP,MAS SIM PELA INSTRUCAO QUE ESTA' SENDO REPETIDA.
- SE CX INICIALMENTE FOR IGUAL A ZERO A INSTRUCAO NAO E' EXECUTADA.
- O PREFIXO E' UTLIZADO EM CONJUNTO COM AS INSTRUCOES SCAS E CMPS.
- UMA VEZ QUE CX E' DECREMENTADO ANTES DO TESTE DA FLAG DE ZERO E AS INSTRUCOES SCAS E CMPS AVANCAM (OU RECUAM) OS PONTEIROS (SI E DI) APOS A COMPARACAO , AO FINAL DA REPETICAO TEREMOS :

1. SE ZF=1 --> NAO FOI ENCONTRADO ELEMENTO NA CADEIA QUE RESULTASSE
EM ZF=0. CX CONTEM ZERO E OS PONTEIROS APONTAM PARA
A POSICAO SEGUINTE AO FINAL DAS CADEIAS.

2. SE ZF=0 --> FOI ENCONTRADO UM ELEMENTO NA CADEIA QUE RESULTOU EM
ZF=0. CX E OS PONTEIROS CORRESPONDEM `A POSICAO SEGUINTE
A ESTE ELEMENTO.

SCASB E SCASB - PROCURA BYTE/WORD EM UM STRING

OPERACAO DO SCASB:

1.SETA OS FLAGS CONFORME O RESULTADO DA COMPARACAO (SUBTRACAO)
DE AL E O BYTE APONTADO POR ES:DI.
2.SE DF=0 SOMA 1 A SI E DI.
3.SE DF=1 SUBTRAI 1 DE SI E DI.

OPERACAO DO SCASW:

1.SETA OS FLAGS CONFORME O RESULTADO DA COMPARACAO (SUBTRACAO)
DE AX E O WORD APONTADO POR ES:DI.
2.SE DF=0 SOMA 2 A SI E DI.
3.SE DF=1 SUBTRAI 2 DE SI E DI.

```
EXEMPLO: PESQUISAR ATE' ENCONTRAR UM CARACTERE "X".
--------

          MOV   AL,'X'
          LEA   DI,STRING
          MOV   CX,LENGHT CORDAO
          CLD
    REPNE SCASB
          JNE   NAO_ACHOU
          JE    ACHOU
```

7. ENTRADA E SAÍDA

IN LEITURA DE PERIFERICO

```
          IN AL,ENDERECO_DO_PORT
          IN AL,DX
          IN AX,ENDERECO_DO_PORT
          IN AX,DX
```

- A INSTRUCAO IN LE UM BYTE OU UM WORD (DEPENDE DO PRIMEIRO OPERANDO) EM AL OU AX.O USO DE OUTROS REGISTRADORES NAO E' PERMITIDO.
- NO CASO DE ESPECIFICARMOS O ENDERECO COMO UM DADO IMEDIATO ,ESTE PODERA' SER ENTRE 0 A 255 . CASO UTILIZEMOS O DX PARA O ENDERECAMENTO , ESTE PODERA' ASSUMIR VALORES ENTRE 0 E 65535.

OUT ESCRITA EM PERIFERICO

```
          OUT AL,ENDERECO_DO_PORT
          OUT AL,DX
          OUT AX,ENDERECO_DO_PORT
          OUT AX,DX
```

- A INSTRUCAO OUT ESCREVE UM BYTE OU UM WORD (DEPENDE DO SEGUNDO OPERANDO) EM UM PERIFERICO.
- NO CASO DE ESPECIFICARMOS O ENDERECO COMO UM DADO IMEDIATO ,ESTE PODERA' SER ENTRE 0 A 255 . CASO UTILIZEMOS O DX PARA O ENDERECAMENTO , ESTE PODERA' ASSUMIR VALORES ENTRE 0 E 65535.

8. DIRETIVAS DO MONTADOR ASSEMBLER - MASM 5.0

As diretivas permitem ao programador definir a organizacao do codigo e dos dados do programa, quando forem carregados para execucao. As diretivas mais utilizadas sao as seguintes:

```
SEGMENT     ;Definicao de segmento
ENDS        ;Fim de segmento
END         ;Fim do programa fonte e ponto de entrada
GROUP       ;Agrupamento de Segmentos
ASSUME      ;Registradores de segmento
ORG         ;Endereco inicial dde segmento
EVEN        ;Alinhamento de segmento
PROC        ;Definicao de procedimento
ENDP        ;Fim de procedimento
```

E' importante frisar que as diretivas nao geram codigo executavel, elas servem exclusivamente para orientar o compilador a montar as instrucoes fornecidas de uma forma que se adeque ao seu projeto. Tambem devemos notar que muitos dos operadores sao opcionais.

SEGMENT

ENDS

Define qual segmento sera' utilizado para o codigo ou dados que forem colocados entre o SEGMENT e o ENDS. Um programa consiste em um ou mais segmentos. Normalmente utilizamos um para pilha, um para codigo e um para dados. Nada impede que utilizemos varios segmentos dentro de um programa.
O formato geral deste operador e' o seguinte:

```
nome  SEGMENT          [opcoes]
      .
      .
      .
nome    ENDS
```

As opcoes possiveis sao do tipo alinhamento, combinacao e classe. O tipo ALINHAMENTO pode ser: PARA, BYTE, WORD ou PAGE.

```
PARA  ;Determina que o segmento inicie em um endereco de memoria
      ;multiplo de 16 (paragrafo).
BYTE  ;O segmento pode iniciar em uma posicao qualquer
WORD  ;O segmento deve iniciar em um endereco par
PAGE  ;O segmento deve iniciar em uma pagina (multiplo de 16)
```

O tipo COMBINACAO determina se o segmento em questao deve ser combinado com outro segmento em tempo de link-edicao. Os tipos pssiveis sao STACK, COMMON, PUBLIC e AT. Os tipos COMMON e PUBLIC sao usados apenas quando programas compilados separadamente devem ser combinados pelo link-editor.

```
AT endereco ;Determina um endereco de memoria inicial para o
            ;segmento em questao. Normalmente este tipo de segmento
            ;nao contem codigo ou dados variaveis, e sao utilizados
            ;para representar codigo ou dados ja' existentes na
            ;memoria (ROM por exemplo). Desta forma, os labels e
            ;variaveis definidos dentro deste segmento podem ser
            ;utilizadas para acessar estes dados ou instrucoes.
STACK       ;Especifica que este segmento e' destinado a pilha. Esta
            ;diretiva so' e' necessaria para programas .exe
```

O tipo classe afeta a ordem dos segmentos e o lugar relativo deles na memoria apos a carga do programa. As classes possiveis sao "STACK", "CODE" e "DATA".

exemplo:

```
PILHA   SEGMENT  PARA STACK "STACK"
      DW     256 DUP(0)
PILHA ENDS
```

END PONTO_DE_ENTRADA

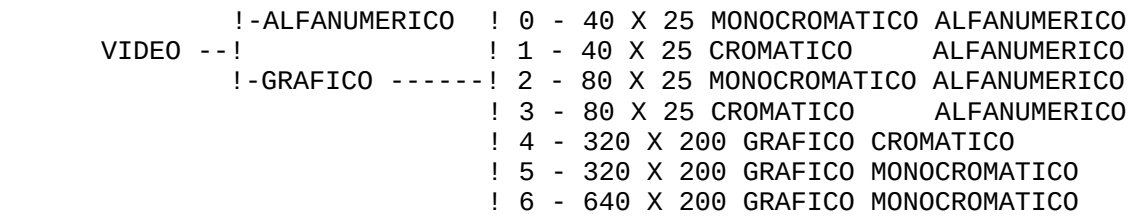
Este operador e' obrigatorio e tem duas finalidades:

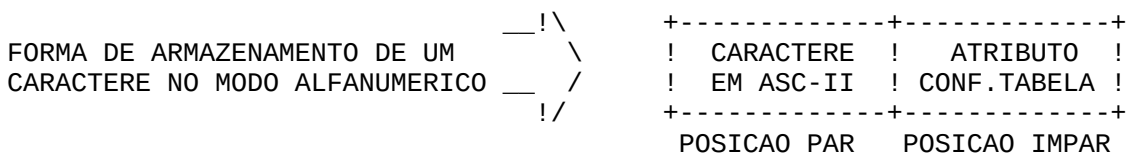
1- identifica o fim do programa fonte. Nenhuma linha apos esta diretiva sera' compilada.
2- o ponto de entrada, opcional, quando utilizado, identifica qual a procedure que sera' tomada como ponto de entrada no programa

GROUP

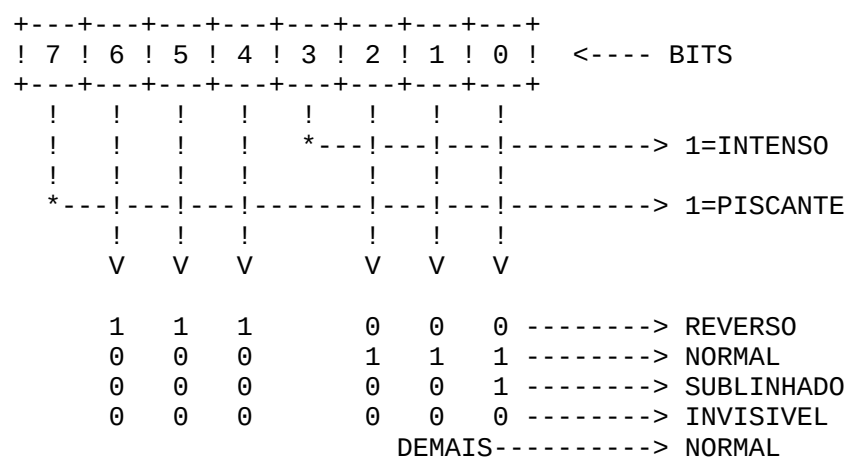
A diretiva GROUP, associa um nome de grupo com um ou mais segmentos, e faz com que todos os labels ou variaveis definidos em um dado segmento a ter um endereco que e' relativo ao inicio do grupo. Os segmentos que comporao um grupo nao estarao necessariamente contiguos. Isto significa que segmentos que nao fazem parte do grupo podem ser carregados entre segmentos que fazem parte.

9. MANIPULACAO DE VIDEO ATRAVES DO BIOS

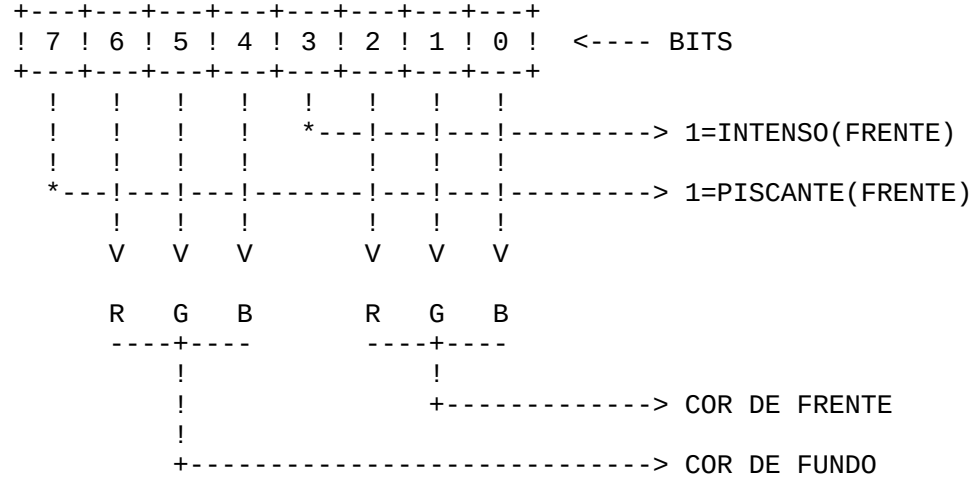




ATRIBUTO PARA O MODO ALFANUMERICO - MODO 7



ATRIBUTO PARA O VIDEO GRAFICO-ALFANUMERICO - MODOS 0,1,2 E 3



MODO DE VIDEO ALFANUMERICO - MODO 7

- IMPLEMENTADO PELA PLACA "MONOCHROME DISPLAY ADAPTER".
- CAPAZ DE GERAR VIDEO ALFANUMERICO MONOCROMATICO COM DEFINICAO DE CARACTERE DE 7X9 PONTOS EM UMA REGIAO DE 9X14 PONTOS.
- PROPRIO PARA LEITURA DE TEXTO COM BAIXA FADIGA VISUAL.
- A TELA OCUPA 2KBYTES DE MEMORIA .
- CADA CARACTERE OCUPA DUAS POSICOES - A POSICAO PAR DETERMINA O CARACTERE E A POSICAO IMPAR DETERMINA O ATRIBUTO.
- AS LINHAS SAO NUMERADAS DE 0 A 24.
- AS COLUNAS SAO NUMERADAS DE 0 A 79.
- O CARATERE DA LINHA (L) E COLUNA (C) FICA ARMAZENADO NO ENDERECO :

```
+----------------------------------------+
! ENDERECO(L,C) = 0B0000H + (160 X L) + C !
+----------------------------------------+
```

- ULTIMAMENTE,POR MOTIVOS DE CUSTOS, OS MICROS DE SEGUNDA LINHA (E ALGUNS DE PRIMEIRA), NAO ESTAO SENDO VENDIDOS COM ESTA PLACA.

MODO DE VIDEO GRAFICO - MODOS 0 A 6

- IMPLEMENTADO PELA PLACA "COLOR GRAPHICS ADAPTER".
- POSSUI QUATRO MODOS DE OPERACAO

1) ALFANUMERICO 25 LINHAS 40 CARACTERES 16 CORES (MODO 0 E 1).
2) ALFANUMERICO 25 LINHAS 80 CARACTERES 16 CORES (MODO 2 E 3).
3) GRAFICO , 200 LINHAS 320 PONTOS 4 CORES. (MODO 4 E 5).
4) GRAFICO , 200 LINHAS 640 PONTOS MONOCROMATICO (MODO 6).

10. PADRAO DE ESTRUTURACAO DE UM PROGRAMA EM ASSEMBLY 8088 (.EXE)

```
;COMENTARIOS INTRODUTORIOS
            TITLE <NOME>


PILHA       SEGMENT     STACK
            [SEGMENTO DE PILHA]
PILHA       ENDS


DADOS       SEGMENT
            [SEGMENTO DE DADOS]
DADOS       ENDS


EXTRA       SEGMENT
            [SEGMENTO EXTRA]
EXTRA       ENDS


CODIGO      SEGMENT
            ASSUME SS:PILHA,DS:DADOS,ES:EXTRA,CS:CODIGO

PRINC       PROC  FAR
            PUSH  DS          ;SALVA O SEGMENTO DO ENDERECO DE RETORNO
            MOV   AX,0            ;COM DESLOCAMENTO IGUAL A 0000
            PUSH  AX          ;NA PILHA
            MOV   AX,DADOS        ;AX=ENDERECO DO SEGMENTO DE DADOS
            MOV   DS,AX           ;INICIALIZA O SEGMENTO DE DADOS
            MOV   AX,EXTRA        ;AX=ENDERECO DO SEGMENTO EXTRA
            MOV   ES,AX           ;INICIALIZA O SEGMENTO EXTRA

                  .
                  .
                  .


            RET
PRINC       ENDP

PROC1       PROC
            [PROCEDIMENTO 1]
PROC1       ENDP

PROC2       PROC
            [PROCEDIMENTO 2]
PROC2       ENDP

CODIGO      ENDS
             END     PRINC
```

EXEMPLO DE UM PROGRAMA COMPLETO.

```
;FUNCAO DO PROGRAMA : APAGAR A TELA E MOSTRA UMA MENSAGEM.
;
;NOME DO PROGRAMA : CLS.

            TITLE CLS


PILHA             SEGMENT     STACK
                  DW      256 DUP (0)
PILHA             ENDS


DADOS             SEGMENT
                  DB      'ESTA E UMA MENSAGEM NA AREA DE DADOS',0DH,0AH,'$'
DADOS             ENDS


CODIGO            SEGMENT
                  ASSUME SS:PILHA,DS:DADOS,ES:DADOS,CS:CODIGO

PRINC             PROC    FAR
                  PUSH    DS              ;SALVA O SEGMENTO DE RETORNO
                  MOV     AX,0            ;COM DESLOCAMENTO IGUAL A 0000
                  PUSH    AX              ;NA PILHA
                  MOV     AX,DADOS        ;AX=ENDERECO DO SEGMENTO DE DADOS
                  MOV     DS,AX           ;INICIALIZA O SEGMENTO DE DADOS
                  MOV     AX,EXTRA        ;AX=ENDERECO DO SEGMENTO EXTRA
                  MOV     ES,AX           ;INICIALIZA O SEGMENTO EXTRA
                  CALL    MONOC           ;CHAMA O PROCEDIMENTO QUE SETA O MODO
                                           ;DE VIDEO E APAGA A TELA.
                  LEA     DX,MENSAGEM     ;APONTA PARA O STRING A SER IMPRESSO
                  MOV     AH,9            ;SELECIONA FUNCAO 9 DO DOS
                  INT     21H             ;CHAMA O DOS
                  RET
PRINT             ENDP


MONOC             PROC    NEAR
                  MOV     AH,0            ;MODO ALFANUMERICO 80x25 COLORIDO
                  MOV     AL,3            ;SELECIONA FUNCAO 3 DO BIOS
                  INT     10H             ;CHAMA O BIOS
                  MOV     AH,5            ;SELECIONA PAGINA 0
                  MOV     AL,0            ;SELECIONA FUNCAO 0 DO BIOS
                  INT     10H             ;CHAMA O BIOS
                  MOV     AH,11           ;SELECIONAR COR DA TELA
                  MOV     BH,0            ;COR DE BORDA=0
                  MOV     BL,0            ;INTENSIDADE DE FUNDO NORMAL
                  INT     10H
                  RET
MONOC             ENDP


CODIGO            ENDS
                  END     PRINC
```

SUB-ROTINA COMPILADA EXTERNAMENTE

```
;ESTE PROGRAMA ILUSTRA A CHAMADA DE UMA SUB-ROTINA COMPILADA EXTERNAMENTE
;
;A SUB-ROTINA DEVE SER DEFINIDA NO PROGRAMA ATRAVES DA DIRETIVA EXTRN
;E DEVE ESTAR DENTRO DO SEGMENTO DE CODIGO.
;
;
PILHA   SEGMENT   STACK
        DB        10 DUP ('PILHA')
PILHA   ENDS
CODIGO  SEGMENT
        ASSUME    CS:CODIGO,SS:PILHA
        EXTRN     MONOC:NEAR
PRINC   PROC      FAR
        PUSH      DS
        XOR       AX,AX
        PUSH      AX
        CALL      MONOC                   ;CHAMA A SUB-ROTINA
        RET
PRINC   ENDP
CODIGO  ENDS
        END       PRINC

-------------------------------------------------------------------------------

;SUB-ROTINA MONOC
;FUNCAO : SETAR O MODO DE VIDEO EM 80X25,COR DE BORDA NORMAL E COR DE
;         FUNDO NORMAL
;ENTRADAS               : NENHUMA
;REGISTRADORES AFETADOS : NENHUM
;SAIDAS                 : NENHUMA
;
;OBS : NOTE QUE ESTA SUB-ROTINA FOI ESCRITA ESPECIALMENTE PARA SER COMPILADA
;      EXTERNAMENTE,POIS NAO CONTEM SEGMENTO DE PILHA E O SEU NOME ESTA
;      SENDO REFERENCIADO PELA DIRETIVA 'PUBLIC'
;      A DEFINICAO DO SEGMENTO DE CODIGO TAMBEM CONTEM UMA DIRETIVA 'PUBLIC'
;      O PROCEDIMENTO ESTA COM O ROTULO 'NEAR',PARA COMPATIBILIZAR COM OS
;      PROGRAMAS '.COM' E PARA QUE A CHAMADA DA SUB-ROTINA SEJA MAIS
;      RAPIDA E OCUPE MENOS ESPACO.
;      A DIRETIVA END NAO DEVE APONTAR PARA NENHUM PONTO DE ENTRADA.
;
CODIGO  SEGMENT PUBLIC
        ASSUME  CS:CODIGO
        PUBLIC  MONOC
MONOC   PROC  NEAR
        MOV   AH,0        ;SELECIONAR MODO DE VIDEO
        MOV   AL,3        ;MODO 80X25 COLORIDO
        INT   10H         ;CHAMA A FUNCAO DE VIDEO_IO DO BIOS
        MOV   AH,5        ;SELECIONAR PAGINA DE VIDEO
        MOV   AL,0        ;PAGINA 0 DE VIDEO
        INT   10H         ;CHAMA A FUNCAO VIDEO_IO
        MOV   AH,11       ;SELECIONA COR DA TELA
        MOV   BH,0        ;PARAMETRO DA FUNCAO
        MOV   BL,0        ;COR BORDA=0,INTENSIDADE COR FUNDO=NORMAL
        INT   10H         ;CHAMA A FUNCAO DE VIDEO_IO DO BIOS
        RET
MONOC   ENDP
CODIGO  ENDS
        END
```

11. PROGRAMAS ".COM"

- UTILIZAM UM UNICO SEGMENTO (SEGMENTO DE CODIGO).
- O PSP FICA NOS 256 BYTES INICIAIS DO PROGRAMA.
- O PROGRAMA DEVE INICIAR NO ENDERECO 100H.
- A PILHA INICIA NO ENDERECO MAIS ALTO DO SEGMENTO (FIM DO SEGMENTO).
- TODAS AS SUBROTINAS DEVEM SER NEAR.
- UTILIZA RET PARA VOLTAR O DOS.
- PARA COMPILAR UM PROGRAMA "PROG.ASM" PARA O TIPO ".COM" :

    1. MASM PROG1;
    2. LINK PROG1;
    3. EXE2BIN PROG1 PROG1.COM

- UM PROGRAMA ".COM" TEM A SEGUINTE ESTRUTURA PADRAO:

```
CODIGO      SEGMENT
      ASSUME CS:CODIGO,SS:CODIGO,ES:CODIGO,SS:CODIGO
      ORG   100H
INICIO:     JMP   COMECO

      .
      .
      .
      ;AQUI FICA A AREA DE DADOS
      .
      .
      .

COMECO      PROC  NEAR
      .
      .
      .
      ;AQUI FICA A AREA DE PROGRAMA
      .
      .
      .
      RET         ;RETORNA PARA O DOS
COMECO      ENDP
      CODIGO      ENDS
      END   INICIO
```

- AS VANTAGENS DOS PROGRAMAS ".COM" SOBRE OS ".EXE" SAO DUAS :

    1.OCUPA MENOS ESPACO EM DISCO.
    2.E CARREGADO DO DISCO PARA A MEMORIA MAIS RAPIDAMENTE.
    3.DEPENDENDO DAS INSTRUCOES UTLIZADAS , O PROGRAMA PODE SER CARREGADO
      EM QUALQUER POSICAO DA MEMORIA.

- COMO DESVANTAGEM ESTA' O FATO DE PODERMOS UTILIZAR APENAS UM SEGMENTO E DESTA FORMA OS PROGRAMAS NAO PODEREM SER MAIOR DO QUE 64 KBYTES.