

# Instruções de movimentação de Dados

*Professor*

Dr. Jorge Leonid Aching Samatelo  
[jlasm001@gmail.com](mailto:jlasm001@gmail.com)

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Opcode	Mod reg rm	Lo Desloc	Hi Desloc	Lo data	Hi data

Byte 1		Byte 2				Byte 3		Byte 4		Byte 5		Byte 6	
1	1	0	0	0	1	1	1	0	0	0	0	0	0
OPCODE		D	W	MOD	REG	R/M							
C		7		8		0							

R/M	MOD=00	MOD=01	MOD=10
000	[BX+SI]	[BX+SI+D8]	[BX+SI+D16]
001	[BX+DI]	[BX+DI+D8]	[BX+DI+D16]
010	[BP+SI]	[BP+SI+D8]	[BP+SI+D16]
011	[BP+DI]	[BP+DI+D8]	[BP+DI+D16]
100	[SI]	[SI+D8]	[SI+D16]
101	[DI]	[DI+D8]	[DI+D16]
110	END.DIRETO	[BP+D8]	[BP+D16]
111	[BX]	[BX+D8]	[BX+D16]

Byte 1		Byte 2		Byte 3	
1	0	1	0	0	1
OPCODE		D	W		
A		3			

# Índice

- ☐ Instruções de movimentação de dados
- ☐ Laboratório.

# Instruções de movimentação de dados

# Instruções de movimentação de dados

## Instrução **MOV**

### ❑ **Que faz:**

- Transferência de dados entre posições de memória, registradores e o acumulador (instrução mais usada no endereçamento de dados).

### ❑ **Sintaxe:**

`MOV <destino>, <fonte>`

- Os operadores devem possuir mesmo tamanho.
- Os dois operandos não podem ser posições de memória.

# Instruções de movimentação de dados

## Instrução **MOV**

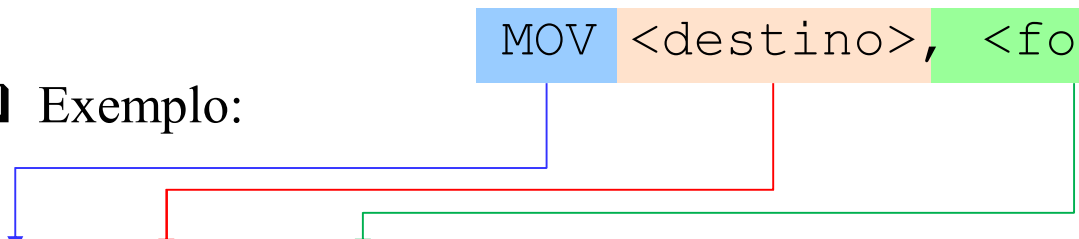
### ❑ Que faz:

- Transferência de dados entre posições de memória, registradores e o acumulador (instrução mais usada no endereçamento de dados).

### ❑ Sintaxe:

MOV <destino>, <fonte>

### ❑ Exemplo:



MOV	AX,	0010h	; AX <= 0010H
MOV	AH,	41h	; AH <= 41H
MOV	AH,	BL	; AH <= BL

# Instruções de movimentação de dados

## Instrução **MOV**

### ❑ Possíveis combinações dos operandos:

- A instrução **MOV** transferem BYTES ou WORDS de dados entre registradores ou entre registradores e memória.

#### ❖ **REG/MEM** $\leftarrow$ **REG**

- **MOV REG, REG**

- Exemplo: **MOV AX, BX**

- **MOV MEM, REG**

- Exemplo: **MOV [BX], AL**

**BX** contém um valor de deslocamento (atua como um apontador a uma posição de memória)

- ❑ **REG**: registrador
- ❑ **MEM**: memória
- ❑ **INMED**: número imediato
- ❑ **SEGREG**: registrador de segmento
- ❑ **REG16**: registrador de 16 bits

# Instruções de movimentação de dados

## Instrução **MOV**

### ❑ Possíveis combinações dos operandos:

- A instrução **MOV** transferem BYTES ou WORDS de dados entre registradores ou entre registradores e memória.

❖ **REG ← MEM**

- **MOV REG, MEM**

- Exemplo: **MOV CH, [40FFh]**

- ❑ **REG**: registrador
- ❑ **MEM**: memória
- ❑ **INMED**: número imediato
- ❑ **SEGREG**: registrador de segmento
- ❑ **REG16**: registrador de 16 bits

# Instruções de movimentação de dados

## Instrução **MOV**

### ❑ Possíveis combinações dos operandos:

- A instrução **MOV** transferem BYTES ou WORDS de dados entre registradores ou entre registradores e memória.

❖ **MEM**  $\nleftrightarrow$  **MEM**

- Exemplo:

- **MOV** [BX], [AX] ; instrução inválida

- A solução é:

- **MOV** **CX**, [AX] ; REG  $\leftarrow$  MEM
  - **MOV** [BX], **CX** ; MEM  $\leftarrow$  REG

- ❑ REG: registrador
- ❑ MEM: memória
- ❑ INMED: número imediato
- ❑ SEGREG: registrador de segmento
- ❑ REG16: registrador de 16 bits



# Instruções de movimentação de dados

## Instrução **MOV**

### ❑ Possíveis combinações dos operandos:

- A instrução **MOV** transferem BYTES ou WORDS de dados entre registradores ou entre registradores e memória.

#### ❖ **REG/MEM ← INMED**

- **MOV REG, INMED**

- Exemplo: **MOV BX, FFFFh**

- **MOV MEM, INMED**

- Exemplo: **MOV BYTE PTR [DI], 0**

Por que usar o operador **PTR**?

- ❑ **REG**: registrador
- ❑ **MEM**: memória
- ❑ **INMED**: número imediato
- ❑ **SEGREG**: registrador de segmento
- ❑ **REG16**: registrador de 16 bits

# Instruções de movimentação de dados

## Operador **PTR**

- ❑ Acrônimo de **PoinTeR**, define uma referencia à memória de um tipo de dado em particular, para que o programa ensamblador selecione a instrução correta. Por exemplo, seja a instrução:

```
MOV [SI],5
```

- ❑ O registrador SI aponta:
  - Um BYTE?:  $[SI] \leftarrow 05H$
  - Um WORD?:  $[SI] \leftarrow 0005H$

- ❑ Para clarificar usamos o operador PTR:

- O registrador SI aponta um BYTE?:  $[SI] \leftarrow 05H$

```
MOV BYTE PTR[SI],5;no byte apontado por SI é armazenado  
;o valor 05H
```

- O registrador SI aponta um WORD?:  $[SI] \leftarrow 0005H$

```
MOV WORD PTR[SI],5;no word apontado por SI é armazenado  
;o valor 0005H
```

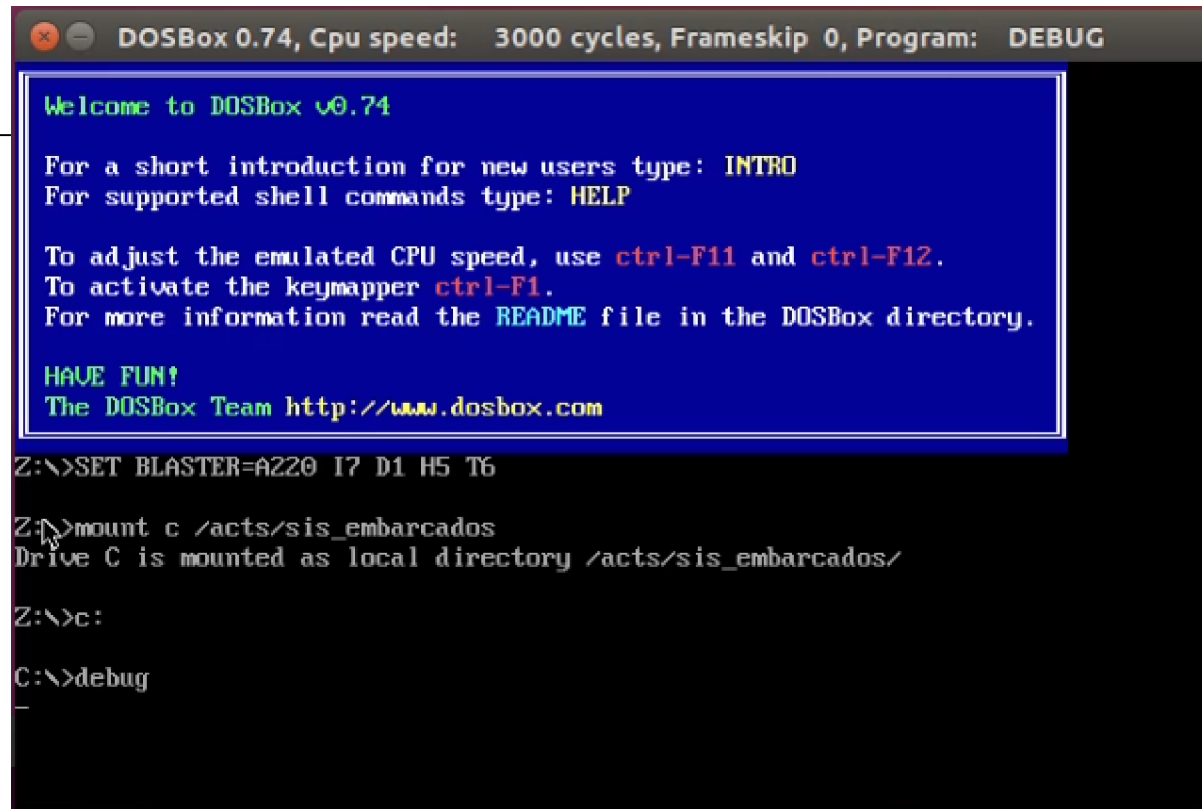
# Laboratório

# Laboratório

Entrar do programa DEBUG:

- ❑ (a) Digite DEBUG na linha de comando do **DOSBox** para chamar o programa DEBUG. Um caracter – vai aparecer como *prompt* do DEBUG;

```
Z:\>mount c /acts/sis_embarcados↵
Z:\>c:↵
C:\>debug↵
-
```



The screenshot shows a DOSBox 0.74 window with the title bar "DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG". The main window has a blue background with white text. It displays a welcome message: "Welcome to DOSBox v0.74", followed by instructions: "For a short introduction for new users type: INTRO", "For supported shell commands type: HELP", "To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.", "To activate the keymapper ctrl-F1.", and "For more information read the README file in the DOSBox directory." Below this, it says "HAVE FUN!" and "The DOSBox Team http://www.dosbox.com". The command prompt shows the following commands and responses: "Z:\>SET BLASTER=A220 I7 D1 H5 T6", "Z:\>mount c /acts/sis\_embarcados" (response: "Drive C is mounted as local directory /acts/sis\_embarcados/"), "Z:\>c:", and "C:\>debug" (response: "-").

# Laboratório

1.

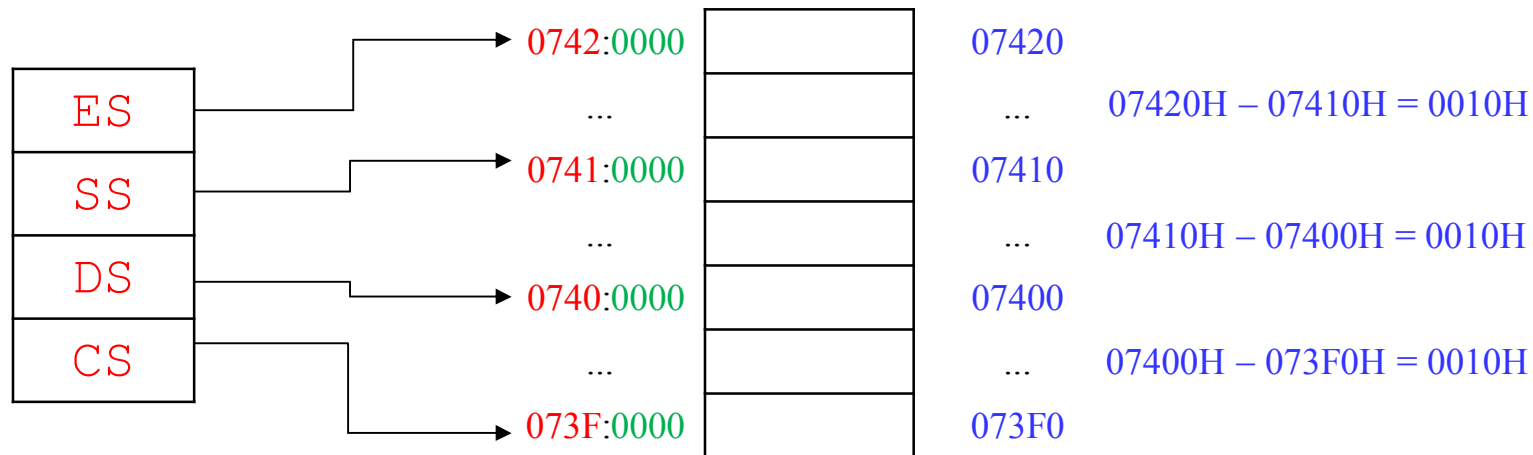
- ❑ Usando o comando **R** do debug,
    - mantenha **CS** apontando para o segmento estabelecido pelo debug.
    - Modifique **DS**, **SS** e **ES** para que apontem para parágrafos adjacentes a partir do segmento CS (Diferença de 16 bytes na memória = 1 parágrafo).
-

# Laboratório

1.

❑ Usando o comando **R** do debug,

- mantenha **CS** apontando para o segmento estabelecido pelo debug.
- Modifique **DS**, **SS** e **ES** para que apontem para parágrafos adjacentes a partir do segmento CS (Diferença de 16 bytes na memória = 1 parágrafo).

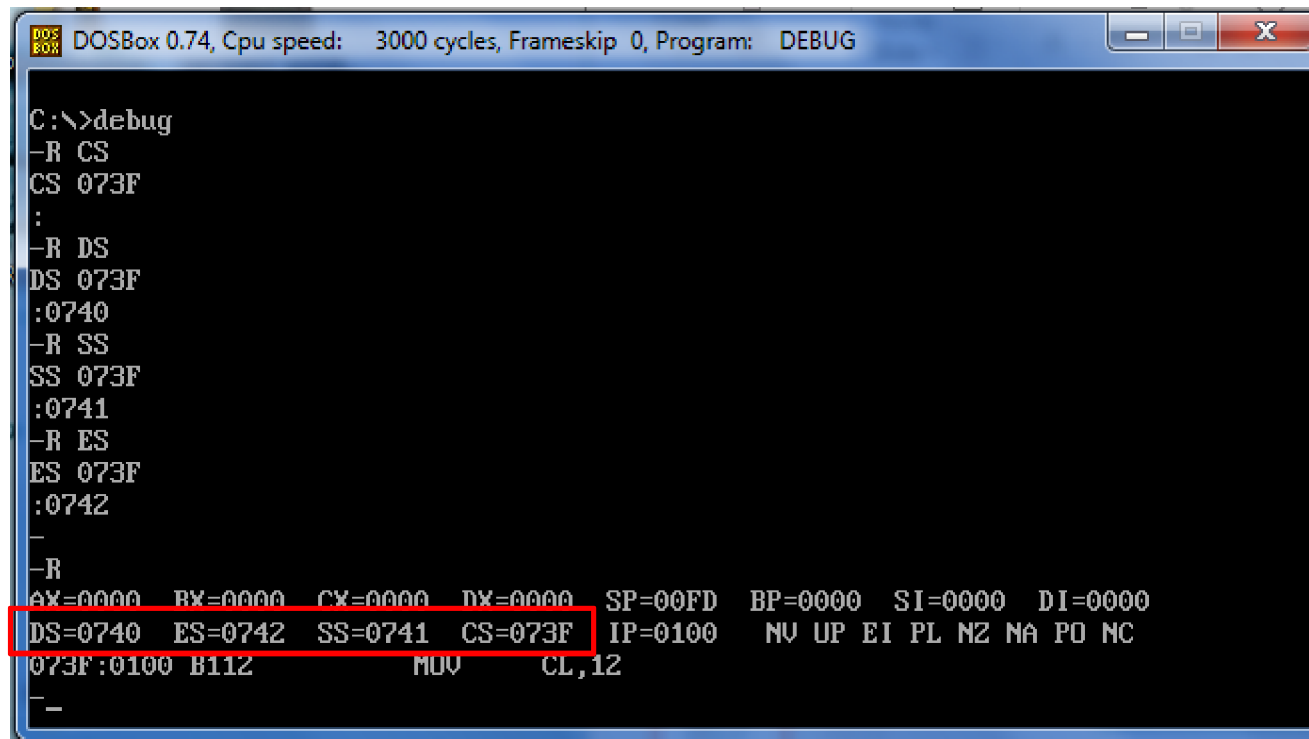


$$\boxed{\text{SEGMENTO}} \boxed{0H} + \boxed{\text{OFFSET}} = \boxed{\text{ENDEREÇO FÍSICO}}$$

# Laboratório

1.

- ❑ Usando o comando **R** do debug,
  - mantenha **CS** apontando para o segmento estabelecido pelo debug.
  - Modifique **DS**, **SS** e **ES** para que apontem para parágrafos adjacentes a partir do segmento CS (Diferença de 16 bytes na memória = 1 parágrafo).



```
DOS BOX DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>debug
-R CS
CS 073F
:
-R DS
DS 073F
:0740
-R SS
SS 073F
:0741
-R ES
ES 073F
:0742
-
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0742 SS=0741 CS=073F IP=0100  NV UP EI PL NZ NA PO NC
073F:0100 B112      MOV     CL,12
-
```

# Laboratório

2.

- ❑ Na posição 100H do segmento de código, codifique as instruções indicadas e usando o comando `U`, verifique os bytes gerados para cada instrução e compare com os campos definidos no formato da instrução típica.

- `MOV CL,[12H] ; REG <= MEM`
- `MOV CL,12H ; REG <= INMED`
- `MOV WORD PTR[BX],1200H ; MEM <= INMED`
- `MOV WORD PTR[BX],AX ; MEM <= REG`

---

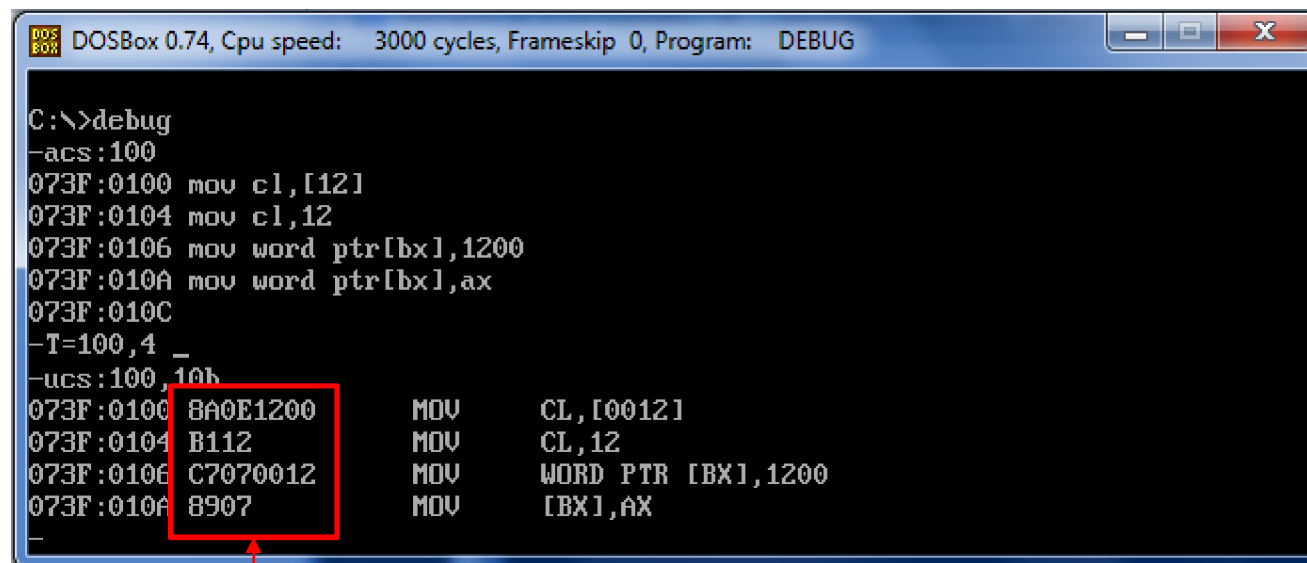


# Laboratório

2.

- ❑ Na posição 100H do segmento de código, codifique as instruções indicadas e usando o comando U, verifique os bytes gerados para cada instrução e compare com os campos definidos no formato da instrução típica.

- MOV CL,[12H] ; REG <= MEM
- MOV CL,12H ; REG <= INMED
- MOV WORD PTR[BX],1200H ; MEM <= INMED
- MOV WORD PTR[BX],AX ; MEM <= REG



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>debug
-acs:100
073F:0100 mov cl,[12]
073F:0104 mov cl,12
073F:0106 mov word ptr[bx],1200
073F:010A mov word ptr[bx],ax
073F:010C
-T=100,4 _
-ucs:100,10h
073F:0100 8A0E1200 MOV CL,[0012]
073F:0104 B112 MOV CL,12
073F:0106 C7070012 MOV WORD PTR [BX],1200
073F:010A 8907 MOV [BX],AX
```

Código de máquina das instruções

# Laboratório

## 3. Exercício

- ❑ Escreva um programa que some dois números de 32 bits localizados na memória nos offsets 1000H e 2000H, colocando o resultado no offset 3000H.
- ❑ Utilize a instrução ADC para somar propagando o transporte da soma anterior. Lembre-se que na memória o byte menos significativo dos números está no menor offset.
- ❑ *IMPORTANTE*
  - Definir valores adequados para os registradores CS e SS, de modo que, não se tenha a superposição de ambos blocos de memória (tome em conta que, cada bloco de memória será no máximo de 64Kbytes).

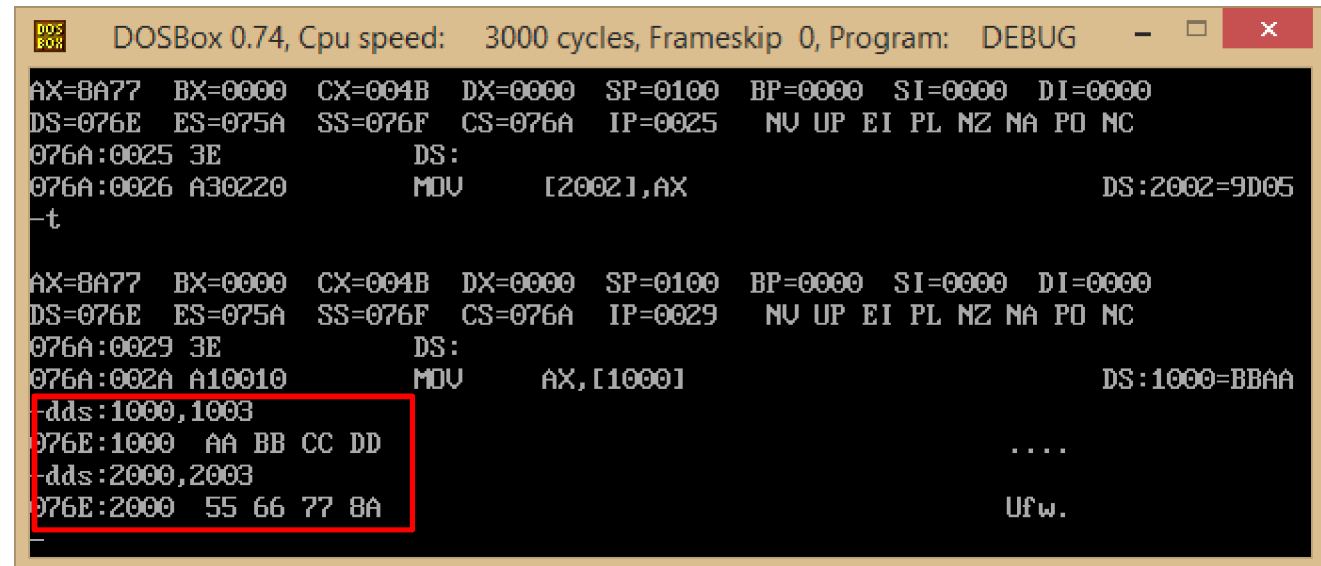
# Laboratório

## 3. Exercício: Dica 1: logica para somar números de 32bits

### ❑ Passo 1: Cargando os dados no segmento de dados

DS:2003H	8A
DS:2002H	77
DS:2001H	66
DS:2000H	55
DS:1003H	DD
DS:1002H	CC
DS:1001H	BB
DS:1000H	AA

BYTE



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
AX=8A77 BX=0000 CX=004B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=076E ES=075A SS=076F CS=076A IP=0025 NV UP EI PL NZ NA PO NC
076A:0025 3E DS:
076A:0026 A30220 MOV [2002],AX DS:2002=9D05
-t
AX=8A77 BX=0000 CX=004B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=076E ES=075A SS=076F CS=076A IP=0029 NV UP EI PL NZ NA PO NC
076A:0029 3E DS:
076A:002A A10010 MOV AX,[1000] DS:1000=BBAA
-dds:1000,1003
076E:1000 AA BB CC DD ....
-dds:2000,2003
076E:2000 55 66 77 8A Ufw.
```

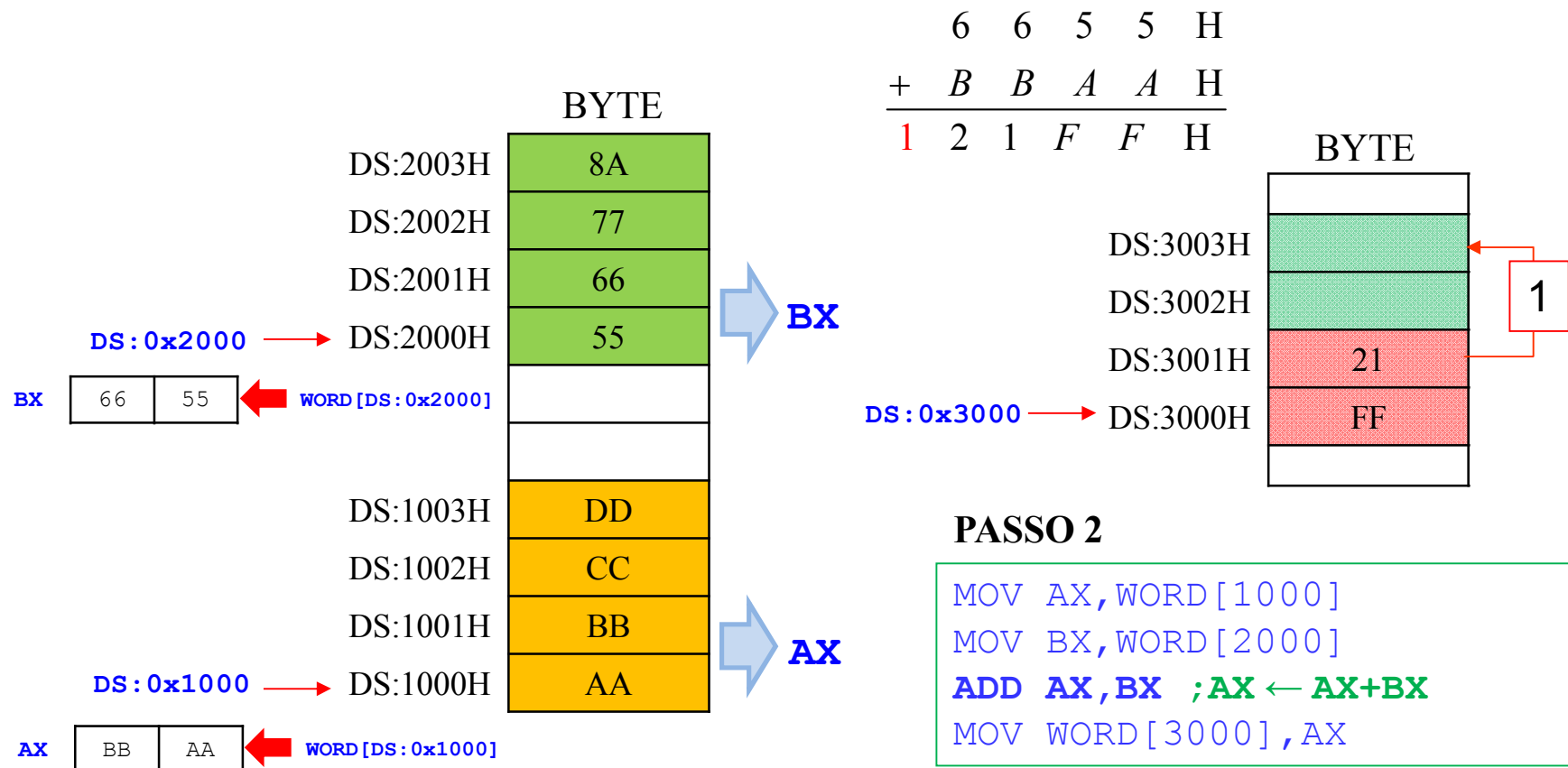
### PASSO 1

```
MOV AX,BBAA
MOV WORD[1000],AX
MOV AX,DDCC
MOV WORD[1002],AX
MOV AX,6655
MOV WORD[2000],AX
MOV AX,8A77
MOV WORD[2002],AX
```

# Laboratório

## 3. Exercício: Dica 1: logica para somar números de 32bits

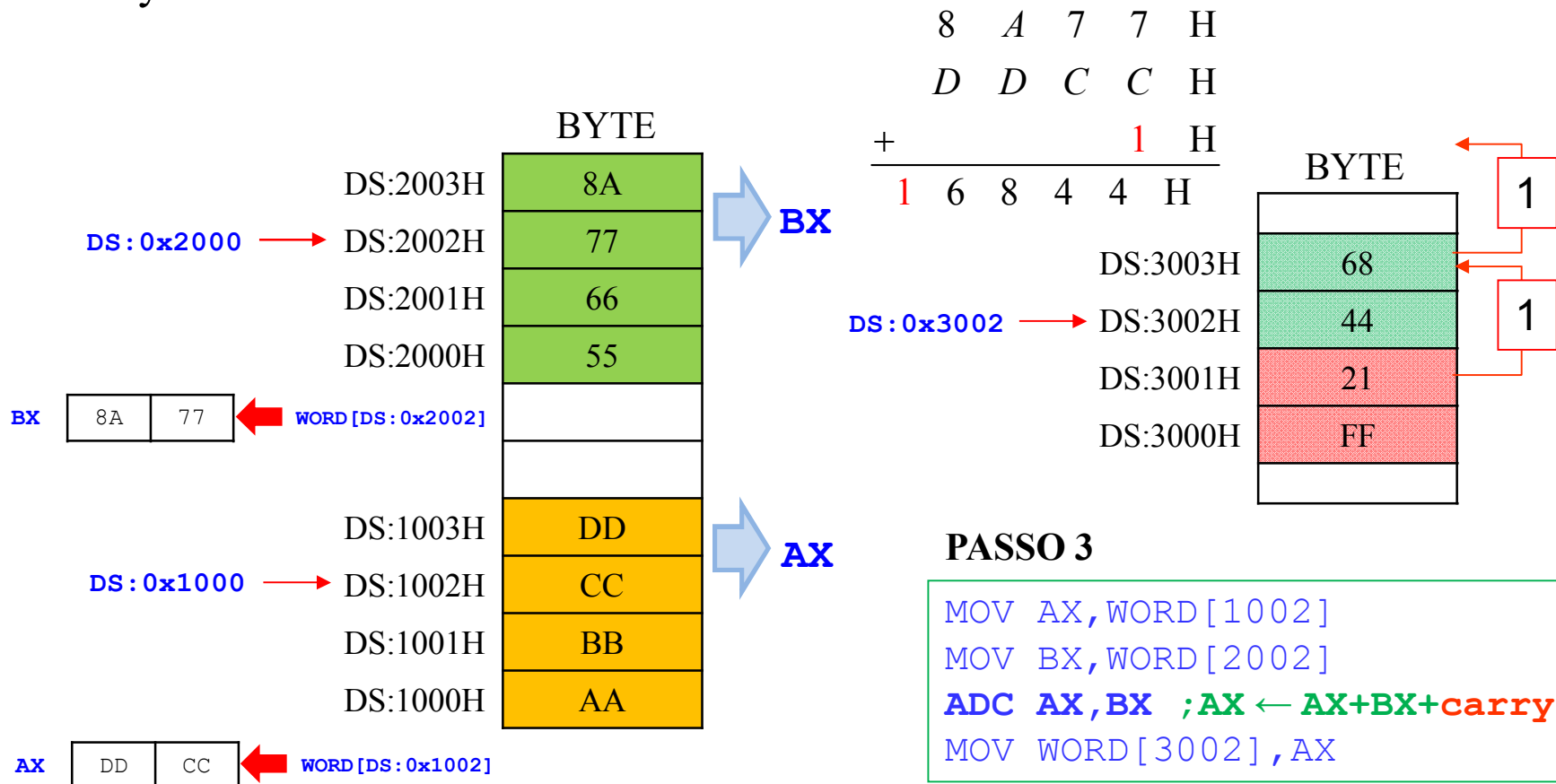
❑ **Passo 2:** Usando as instruções **MOV**, e **ADD** são sumados os dois primeiros bytes.



# Laboratório

## 3. Exercício: Dica 1: logica para somar números de 32bits

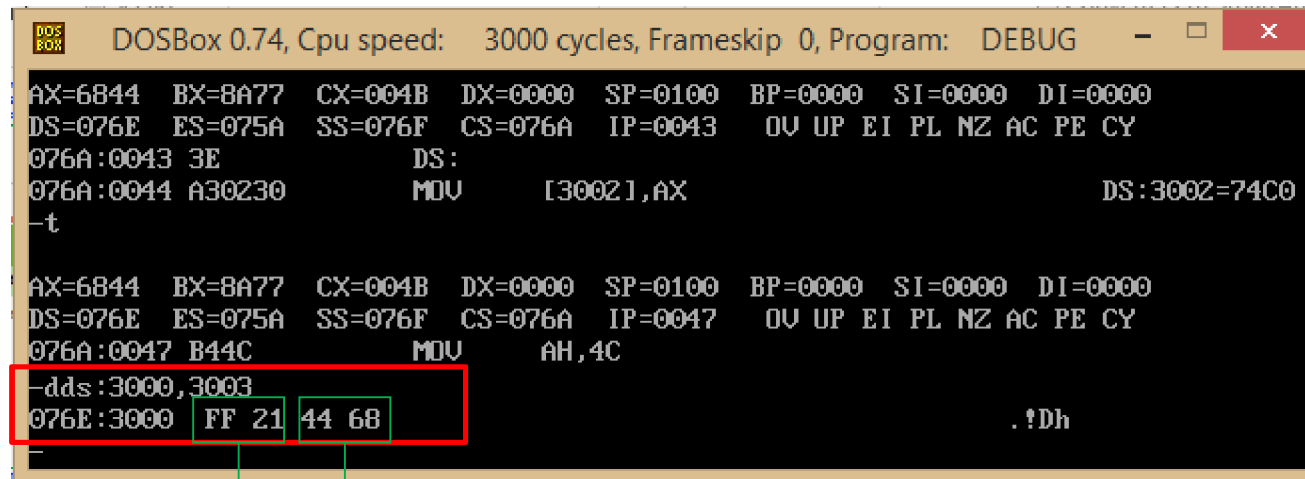
❑ **Passo 3:** Usando as instruções **MOV**, e **ADC** são sumados os seguintes dois bytes.



# Laboratório

## 3. Exercício: Dica 1: logica para somar números de 32bits

- ❑ **Passo 4:** Comprovando o valor da soma usando o debug (compilação linha por linha com o comando `t` e visualização da memória com o comando `d`).



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
AX=6B44 BX=8A77 CX=004B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=076E ES=075A SS=076F CS=076A IP=0043 OV UP EI PL NZ AC PE CY
076A:0043 3E DS:
076A:0044 A30230 MOV [3002],AX DS:3002=74C0
-t
AX=6B44 BX=8A77 CX=004B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=076E ES=075A SS=076F CS=076A IP=0047 OV UP EI PL NZ AC PE CY
076A:0047 B44C MOV AH,4C
-dds:3000,3003
076E:3000 FF 21 44 68 .!Dh
```

