

Microprocessadores e Microcontroladores

Programação Assembly

José Tarcísio Franco de Camargo

Pseudo-instruções

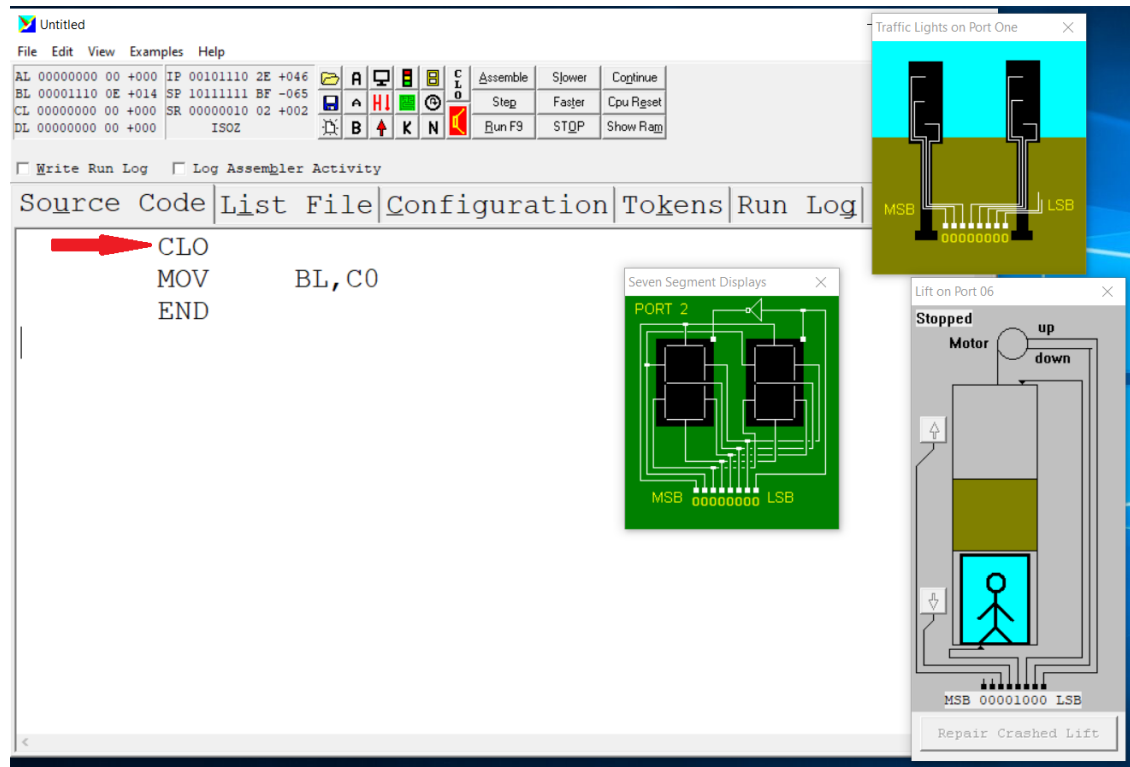
Pseudo-instruções

- Pseudo-instruções não são instruções assembly de fato.
- São comandos utilizados pelo montador (assembler) para realização de algumas tarefas.
- Alguns exemplos de pseudo-instruções são:
 - **CLO**
 - **ORG**
 - **DB**

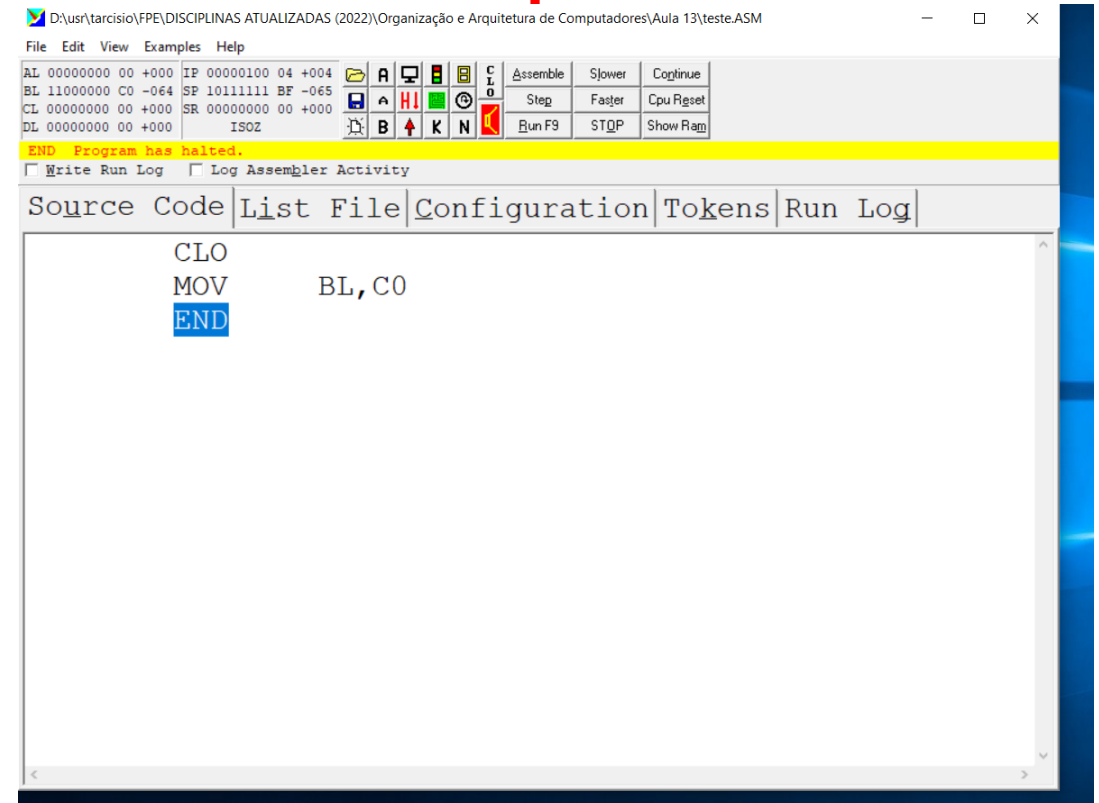
CLO

- Pseudo-instrução, normalmente usada no início de um programa, que fecha todas as janelas de simulação que estiverem abertas.
- Veja o exemplo a seguir.

Antes



Depois



ORG

- Pseudo-instrução que define um endereço de memória a partir do qual um determinado trecho de código será inserido.
- Esta pseudo-instrução pode ser usada várias vezes em um programa.

The screenshot displays the MASM6400 assembler interface. The main window shows the source code with two instances of the `ORG` pseudo-instruction, each marked with a blue arrow. The first instance is at address 30, and the second is at address 50. A secondary window titled "RAM Source Code View" is open, showing a memory dump. In this window, the lines corresponding to the `ORG` instructions are highlighted with blue boxes and blue arrows: line 30 for `MOV AL, 42 MOV BL, C0` and line 50 for `MOV CL, 10 MOV DL, 20`. The memory dump shows hexadecimal addresses on the left, followed by the instruction in hexadecimal, and then the instruction in ASCII/Source code. The `END` instruction is visible at the end of the program.

```
File Edit View Examples Help
AL 00000000 00 +000 IP 00000000 00 +000
BL 00000000 00 +000 SP 10111111 BF -065
CL 00000000 00 +000 SR 00000000 00 +000
DL 00000000 00 +000 ISOZ

Write Run Log Log Assembler Activity

Source Code | List File | Configuration | Tokens | Run Log |

ORG 30
MOV AL, 42
MOV BL, C0

ORG 50
MOV CL, 10
MOV DL, 20
END
```

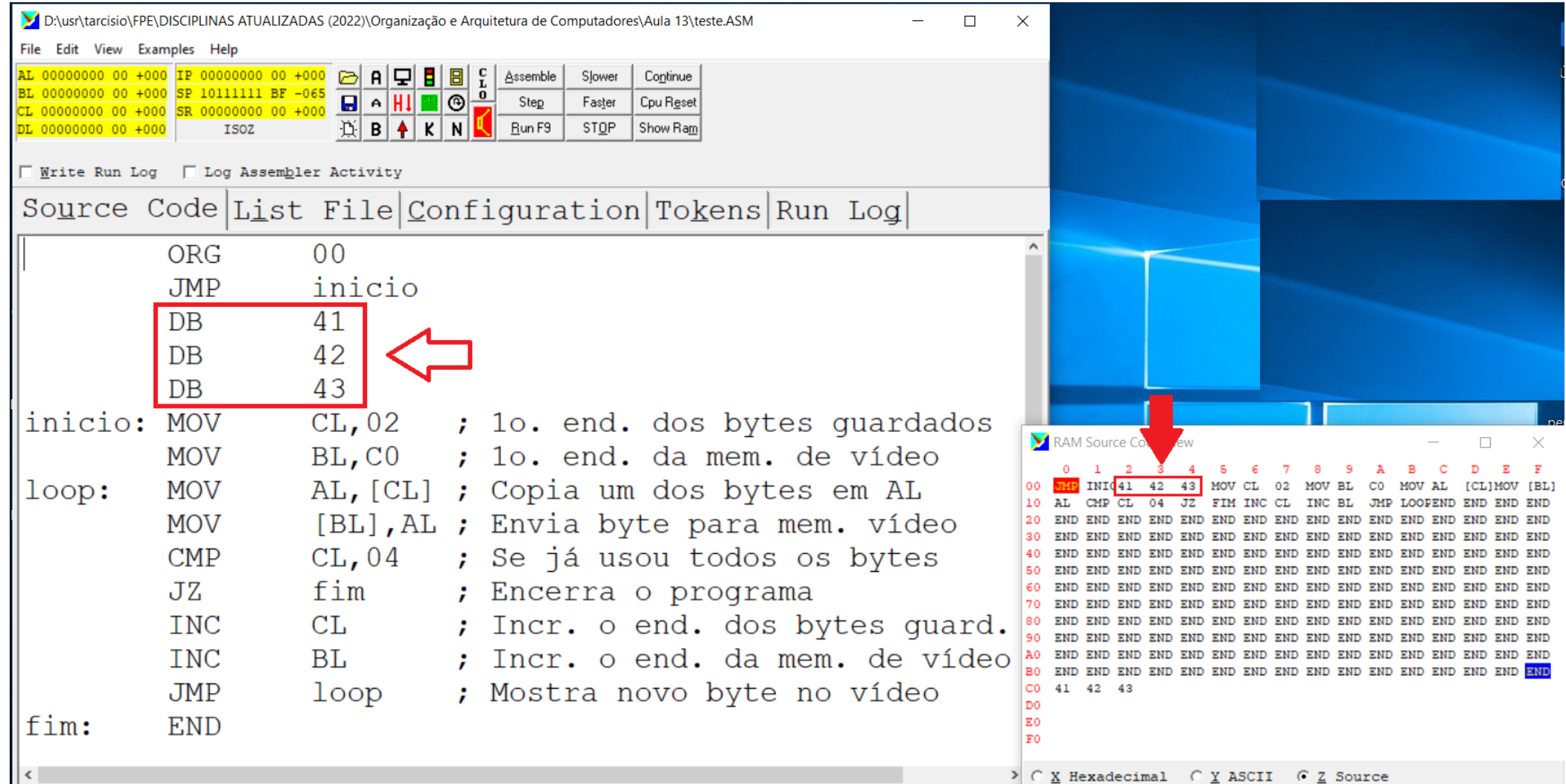
RAM Source Code View

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	MOV	AL	42	MOV	BL	C0	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	MOV	CL	10	MOV	DL	20	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0																
D0																
E0																
F0																

Hexadecimal ASCII Source

DB

- Pseudo-instrução que reserva um endereço de memória onde será guardado um byte qualquer.



The screenshot displays an x86 assembler window with the following components:

- File Path:** D:\usr\tarcisio\FPE\DISCIPLINAS ATUALIZADAS (2022)\Organização e Arquitetura de Computadores\Aula 13\teste.ASM
- Registers:** AL, BL, CL, DL (all 00000000 00 +000); IP (00000000 00 +000); SP (10111111 BF -065); SR (00000000 00 +000); ISOZ.
- Buttons:** Assemble, Slower, Continue, Step, Faster, Cpu Reset, Run F9, STQP, Show Ram.
- Tabs:** Source Code, List, File, Configuration, Tokens, Run Log.
- Source Code:**

```
ORG 00
JMP inicio
DB 41
DB 42
DB 43
inicio: MOV CL, 02 ; 1o. end. dos bytes guardados
MOV BL, C0 ; 1o. end. da mem. de vídeo
loop: MOV AL, [CL] ; Copia um dos bytes em AL
MOV [BL], AL ; Envia byte para mem. vídeo
CMP CL, 04 ; Se já usou todos os bytes
JZ fim ; Encerra o programa
INC CL ; Incr. o end. dos bytes guard.
INC BL ; Incr. o end. da mem. de vídeo
JMP loop ; Mostra novo byte no vídeo
fim: END
```
- RAM Source Code View:** A window showing the memory layout. The first three bytes (addresses 41, 42, 43) are highlighted in red and contain the values 41, 42, and 43 respectively. The rest of the memory is filled with 'END'.

Resultado da execução do programa anterior

Source Code

```
ORG 00
JMP inicio
DB 41
DB 42
DB 43
inicio: MOV CL,02 ; 1o. end. dos bytes guardados
MOV BL,C0 ; 1o. end. da mem. de vídeo
loop: MOV AL,[CL] ; Copia um dos bytes em AL
MOV [BL],AL ; Envia byte para mem. vídeo
CMP CL,04 ; Se já usou todos os bytes
JZ fim ; Encerra o programa
INC CL ; Incr. o end. dos bytes guard.
INC BL ; Incr. o end. da mem. de vídeo
JMP loop ; Mostra novo byte no vídeo
fim: END
```

END Program has halted.

VIDEO CO.FE

A	B	C

RAM Source Code View

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	JMP	INIC41	42	43	MOV	CL	02	MOV	BL	C0	MOV	AL	[CL]	MOV	[BL]	
10	AL	CMP	CL	04	JZ	FIM	INC	CL	INC	BL	JMP	LOOP	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0	41	42	43													
D0																
E0																
F0																

Hexadecimal ASCII Source