

Universidade Federal do Espírito Santo

Departamento de Engenharia

Engenharia de Computação

Laboratório 01

Utilização do DEBUG do DOSBox

Aluno(a): Arthur Coelho Estevão

Professora: Camilo Arturo Rodriguez Diaz

Vitória

2022

1 Introdução

Este Laboratório tem como objetivo principal conhecer sobre o software DosBox, aprendendo os seus comandos, treinando a utilização do programa DEBUG do DOS para verificar conteúdo de registros, posições de memória, assembler e desassembler programas e também executar programas com pontos de paradas (breakpoints).

2 Teoria

2.1 Chamada do Debug

O debug não é um recurso profissional para desenvolvimento de aplicações em Assembly. Entretanto, sua simplicidade de operação nos permite rapidamente testar alguns comandos(1).

Para a criação de um programa em assembler existem 2 opções: usar o TASM –Turbo Assembler da Borland, ou o DEBUGGER. Será utilizado o debug, uma vez que pode encontrá-lo em qualquer PC com o MS-DOS.

Debug pode apenas criar arquivos com a extensão .COM, e devido as características deste tipo de programa, eles não podem exceder os 64 Kb, e também devem iniciar no endereço de memória 0100H dentro do segmento específico. É importante observar isso, pois deste modo os programas .COM não são relocáveis.

Os principais comandos do programa debug são:

- A Montar instruções simbólicas em código de máquina
- D Mostrar o conteúdo de uma área da memória
- E Entrar dados na memória, iniciando num endereço específico
- G Rodar um programa executável na memória
- N Dar nome a um programa
- P Proceder, ou executar um conjunto de instruções relacionadas
- Q Sair do programa debug
- R Mostrar o conteúdo de um ou mais registradores
- T Executar passo a passo as instruções
- U Desmontar o código de máquina em instruções simbólicas
- W Gravar um programa em disco

2.2 Registradores

AX (acumulador)

Este é o registro do acumulador. Ele é usado em instruções aritméticas, lógicas e de transferência de dados.

BX (Base Register)

Este é o registro de base. O registro BX é um registro de endereço. Geralmente contém um ponteiro de dados usado para endereçamento baseado, indexado baseado ou indireto de registro.

CX (registro de contagem)

Este é o registro de contagem. Isso serve como um contador de loop. As construções de loop de programa são facilitadas por ele. O registrador de contagem também pode ser usado como um contador na manipulação de strings e nas instruções de deslocamento / rotação.

DX (registro de dados)

Este é o registro de dados. O registro de dados pode ser usado como um número de porta em operações de E / S. Também é usado na multiplicação e divisão.

SP (Stack Pointer)

Este é o registro do ponteiro da pilha apontando para a pilha do programa. É usado em conjunto com SS para acessar o segmento de pilha.

BP (Ponteiro de Base)

Este é um registro de ponteiro de base apontando para dados no segmento de pilha. Ao contrário do SP, podemos usar o BP para acessar dados nos outros segmentos.

SI (índice de origem)

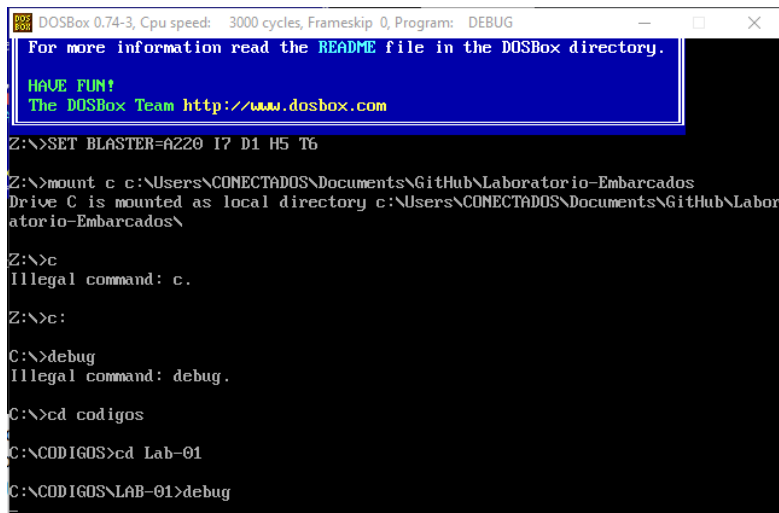
Este é o registro de índice de origem que é usado para apontar para locais de memória no segmento de dados endereçado pelo DS. Assim, quando incrementamos o conteúdo do SI, podemos acessar facilmente as localizações consecutivas da memória.

DI (Índice de Destino)

Este é o registro de índice de destino que executa a mesma função do SI. Existe uma classe de instruções chamadas operações de string, que usam DI para acessar os locais de memória endereçados pelo ES.

3 Resultados

Nesse começo foi usado o comando debug Figura 1.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
For more information read the README file in the DOSBox directory.
HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c c:\Users\CONECTADOS\Documents\GitHub\Laboratorio-Embarcados
Drive C is mounted as local directory c:\Users\CONECTADOS\Documents\GitHub\Laboratorio-Embarcados\

Z:\>c
Illegal command: c.

Z:\>c:

C:\>debug
Illegal command: debug.

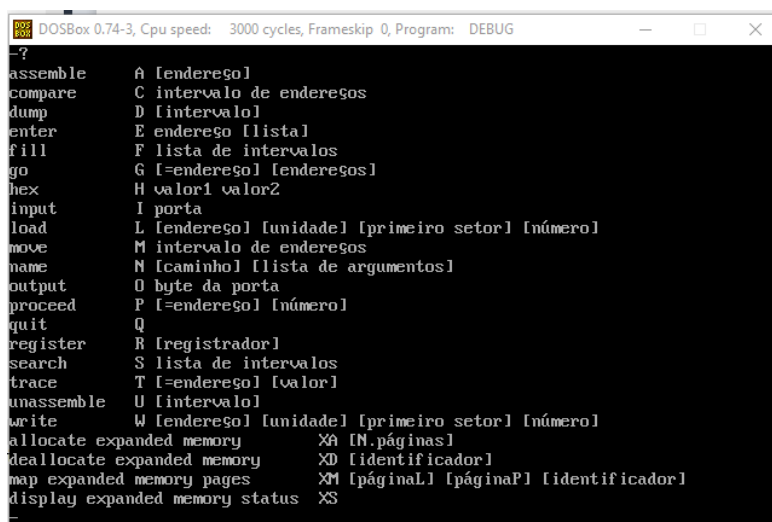
C:\>cd codigos

C:\CODIGOS>cd Lab-01

C:\CODIGOS\LAB-01>debug
```

Figura 1: Execução do debug

Após o DEBUG estar ativo ao digitar “?” é listado um resumo dos principais comandos Figura 2. Sendo a primeira coluna o nome da instrução, as letras na segunda coluna referente ao comando e a última coluna sendo os parâmetros do comando. Algo interessante aqui é que os comandos podem ser escritos usando letras maiúsculas e minúsculas, porém, assumiremos as instruções em maiúsculo e os parâmetros em minúsculo.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
?
assemble      A [endereço]
compare       C intervalo de endereços
dump          D [intervalo]
enter         E endereço [lista]
fill         F lista de intervalos
go            G [=endereço] [endereços]
hex          H valor1 valor2
input        I porta
load         L [endereço] [unidade] [primeiro setor] [número]
move        M intervalo de endereços
name        N [caminho] [lista de argumentos]
output      O byte da porta
proceed     P [=endereço] [número]
quit        Q
register     R [registrador]
search      S lista de intervalos
trace       T [=endereço] [valor]
unassemble  U [intervalo]
write       W [endereço] [unidade] [primeiro setor] [número]
allocate expanded memory  XA [N.páginas]
deallocate expanded memory XD [identificador]
map expanded memory pages XM [páginaL] [páginaP] [identificador]
display expanded memory status XS
```

Figura 2: Lista comandos debug

3.1 Verificação de um conjunto de posições de memória

Com o comando "R" podemos listar todos os registradores associados ao nosso processador Figura 3, bem como o valor em cada registrador. Vemos que os registradores são acompanhados da letra X. A arquitetura do 8086 é de 16 bits. É dividido em parte alta e baixa com 8 bits cada, contudo caso um registrador terminar com L significa dizer que seus bits são o menos significativos caso contrario terminado com H são os mais significativos e "AX" é o registrador completo.

```
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0100 NU UP EI PL NZ NA PO NC
075F:0100 0000 ADD [BX+SI],AL DS:0000=CD
```

Figura 3: Comando R

Cada um dos registradores tem um propósito específico, podendo ser averiguado na Subseção 2.2.

Caso queira-se modificar o valor de algum registrador basta utilizar o comando R juntamente com o nome do registrador, que em seguida mostrara o valor atual do registrador para posteriormente solicitar o novo valor Figura 4.

```
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0100 NU UP EI PL NZ NA PO NC
075F:0100 0000 ADD [BX+SI],AL DS:0000=CD
-RBX
BX 0000
:100
-R
AX=0000 BX=0100 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0100 NU UP EI PL NZ NA PO NC
075F:0100 0000 ADD [BX+SI],AL DS:0100=00
```

Figura 4: Alteração do valor do registrador BX

Na Figura 4 vemos que na primeira execução do comando R, no registrador "BX", temos o valor, em hexadecimal, 0000. Ao digitar "RBX" foi inserido um novo valor, no caso 0100, que pode se visto na segunda execução do comando "R" onde todos os registradores foram listados e "BX" com seu novo valor.

Outro comando "RF" mostra as flags que indicam o estado do processador. Com este comando é possível mudar seus valores também porem alguns são somente para leitura.

3.2 Verificando e modificando o conteúdos dos registradores

O comando "D" mostra uma parte dos segmentos de dados. Como demonstrado na Figura 5, onde foi colocado um número D100 que leva para posição selecionada no data segment da memória. O valor após o comando D pode ser em hexadecimal ou o nome do registro do segmento.

```

-D100
075F:0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
075F:0110 00 00 00 00 00 00 00 00 00 00 34 00 4E 07 .....4.N.
075F:0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
075F:0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
075F:0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
075F:0150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
075F:0160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
075F:0170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figura 5: Comando D

Uma adaptação deste comando é a de poder lista um trecho específico da memória por meio de "D< *posição de memória* >:< *ponto inicial* >,< *ponto de parada* >" Figura 6, caso o ponto de parada não seja especificado o comando lista 128 bytes.

```

-D100:100,110
0100:0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0110 00 .....
-D100:100
0100:0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0100:0170 4D 18 01 12 00 00 00 00 00 00 00 00 00 00 M.....

```

Figura 6: Comando D com especificação de trecho

O comando também pode ser utilizado para um segmento de dados de um registrador específico, basta basta ao invés de digitar a posição da memória especificar o nome do registrador ao qual deseja analisar, por exemplo "DA", referente ao registrador "A".

3.3 Verificar/Alterar o conteúdo de uma posição de memória

Com o comando "D" é possível visualizar os dados em data segment, mas com o comando "E" eu consigo alterar o seu conteúdo. A forma geral deste comando é "E< *segmento* >:< *offset início* >< *enter* >".

Na Figura 7 foi listado o segment register "C" com o comando "DC", para posteriormente com a utilização do comando "E", alterar os seus primeiros 4 bytes.

```

-DC
075F:0000                                     03 04 8A 03
075F:0010 A3 01 17 03 A3 01 92 01-01 01 01 00 02 FF FF FF .....
075F:0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
075F:0030 00 00 14 00 18 00 5F 07-FF FF FF FF 00 00 00 00 .....
075F:0040 05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
075F:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 00 20 20 20 .!.....
075F:0060 20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20 .....
075F:0070 20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00 .....
075F:0080 00 0D 00 00 00 00 00 00-00 00 00 00 .....
-EC
075F:000C 03.1 04.2 8A.3 03.4
-DC
075F:0000                                     01 02 03 04
075F:0010 A3 01 17 03 A3 01 92 01-01 01 01 00 02 FF FF FF .....
075F:0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
075F:0030 00 00 14 00 18 00 5F 07-FF FF FF FF 00 00 00 00 .....
075F:0040 05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
075F:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 00 20 20 20 .!.....
075F:0060 20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20 .....
075F:0070 20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00 .....
075F:0080 00 0D 00 00 00 00 00 00-00 00 00 00 .....

```

Figura 7: Alteração de segmento de dados do registrador

3.4 Assemblar um programa

O comando "A"(assembly) insere instruções a partir de onde o "IP"está apontando, com isso, os comandos digitados na Figura () são escritos direto na pilha de instruções a serem executadas, para conferir se os comandos foram inseridos corretamente é possível usar o comando de desassemblar "U". Nele será listado todos os comandos digitados pelo modo assembler, executado por meio do formato "U< segmento >:< offset início >,< offset fim >".

```

-A
075F:0100 MOV AX, 40
075F:0103 MOV BX, 30
075F:0106 MOV CX, 20
075F:0109 ADD AX, BX
075F:010B ADD AX, CX
075F:010D
-U
075F:0100 B84000      MOV     AX,0040
075F:0103 BB3000      MOV     BX,0030
075F:0106 B92000      MOV     CX,0020
075F:0109 01D8        ADD     AX,BX
075F:010B 01C8        ADD     AX,CX
075F:010D 0000        ADD     [BX+SI],AL
075F:010F 0000        ADD     [BX+SI],AL
075F:0111 0000        ADD     [BX+SI],AL
075F:0113 0000        ADD     [BX+SI],AL
075F:0115 0000        ADD     [BX+SI],AL
075F:0117 0000        ADD     [BX+SI],AL
075F:0119 0000        ADD     [BX+SI],AL
075F:011B 0034        ADD     [SI],DH
075F:011D 004E07      ADD     [BP+07],CL

```

Figura 8: Assembly e Desassemble

3.5 Execução linha por linha

Com o comando "T"é possível fazer o debug de instrução por instrução sendo executada no código, a partir da Figura 8 foram executadas as linhas chamando o comando "R" para mostrar o valor dos registradores e posteriormente o comando "T"para executar a instrução no topo da pilha.Podemos na Figura 9 que o valor do registrador "AX"passou de 0000 para 0040, re-

ferente à instrução "MOV AX,0040", e que "BX"e "CX"passaram para o valor de 0030 e 0020 respectivamente, contudo uma das ultimas instruções alterou o valor de "AX"para a soma de "AX"com "BX"equivalente a 0070. Uma informação interessante é que esse comando T mostra a posição do data segment, o "IP"da próxima instrução e o comando que foi executado, lembrando sempre de garantir que o par CS:IP esteja apontando para a primeira instrução do programa.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-R
AX=0000 BX=0100 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0100  NU UP EI PL NZ NA PO NC
075F:0100 B84000      MOV     AX,0040
-T
AX=0040 BX=0100 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0103  NU UP EI PL NZ NA PO NC
075F:0103 BB3000      MOV     BX,0030
-T
AX=0040 BX=0030 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0106  NU UP EI PL NZ NA PO NC
075F:0106 B92000      MOV     CX,0020
-T
AX=0040 BX=0030 CX=0020 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0109  NU UP EI PL NZ NA PO NC
075F:0109 01D8      ADD     AX,BX
-T
AX=0070 BX=0030 CX=0020 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=010B  NU UP EI PL NZ NA PO NC
075F:010B 01C8      ADD     AX,CX

```

Figura 9: Execução de Linha

3.6 Execução até encontrar um terminador

Uma outra forma de executar um programa é por meio do comando "G". Com o comando "G", ao assembler, podemos definir um breakpoint com a flag "INT 3", para sinalizar uma parada do programa Figura 10 ou executar instruções dentro de um intervalo 11, caso seja definido a flag "INT 3" e colocar um intervalo maior que o local da flag, ira finalizar no local ao qual a flag foi instanciada Figura 10.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-A
075F:010D MOV BX,1000
075F:0110 MOV CX,2000
075F:0113 ADD AX,BX
075F:0115 ADD AX,CX
075F:0117
-G=010D,0113
AX=0090 BX=1000 CX=2000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=0113  NU UP EI PL NZ NA PE NC
075F:0113 01D8      ADD     AX,BX

```

Figura 10: Execução de comandos por flag

```

-A
075F:0117 MOV CX,1000
075F:011A MOV BX,2000
075F:011D INT 3
075F:011E ADD AX,CX
075F:0120
-G=117,120

AX=0090 BX=2000 CX=1000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=075F ES=075F SS=075F CS=075F IP=011D  NU UP EI PL NZ NA PE NC
075F:011D CC          INT     3
-

```

Figura 11: Execução de comandos por intervalo

3.7 Criação de Programa Somador

Primeiramente foram somados os dois menos significativos dos números de 32 bits dos registradores da posição 1000 e 2000. Foi movido o resultado para o menos significativo da posição 3000 e armazenamos o carry no registrador cx, depois foi somado os mais significativos junto com o carry e o resultado foi movido para a posição 3000, como demonstrado no código da Figura 12.

```

-A
075F:0120 MOV AX,[1000]; somando bits menos significativos
075F:0123 MOV BX,[2000]; somando bits menos significativos BX
075F:0127 MOV CX,0; limpa o registrador
075F:012A ADD AX,BX; soma os valores
075F:012C ADC CX,0; guarda o carry no CX
075F:012F MOV [3000],AX; move o resultado dos menos significativos para 3000 K K
075F:0132 MOV AX,[1002]; somando mais significativos
075F:0135 MOV BX,[2002]; somando mais significativos BX
075F:0139 ADD AX,BX; soma AX com BX
075F:013B ADD AX,CX; soma AX com o carry
075F:013D MOV [3002],AX; move resultado dos mais significativos para 3002
075F:0140
-

```

Figura 12: Código Somador

A execução do código pode ser analisada através da Figura ??, onde após execução é possível analisar por meio das Figuras 13 e 14 o segmento de dados na posição de memória 3000 antes e após a execução do código.

```

-D3000:3000,3060
3000:3000 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
3000:3010 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
3000:3020 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
3000:3030 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
3000:3040 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
3000:3050 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
3000:3060 00
-

```

Figura 13: Segmento de memória antes da execução

```

D3000
075F:3000 8B DE F6 F6 00 00 00 00-00 00 00 00 D8 6C 36 6C .....161

```

Figura 14: Segmento de memória depois da execução

3.8 Outros comandos

3.8.1 Comparação (C)

Compara dois blocos de memória. Se não houver diferenças, o debug simplesmente exibirá outro prompt (-). A Figura ?? demonstra uma situação ao qual não são diferenças.

```

C:\>debug
-C 140 148 340
075F:0140 7E 00 075F:0340
075F:0141 18 00 075F:0341
075F:0142 18 00 075F:0342
075F:0143 18 00 075F:0343
075F:0144 7E FE 075F:0344
075F:0145 3C C6 075F:0345
075F:0146 18 06 075F:0346
075F:0147 00 0C 075F:0347
075F:0148 00 18 075F:0348

```

Figura 15: comando C

Os bytes nas localizações 140 a 148 estão sendo comparados aos de 340 (a 348, implícito); os bytes são exibidos lado a lado para aqueles que são diferentes (com suas localizações exatas, incluindo o segmento, em cada lado deles).

3.8.2 Iniciar bloco de memória (F)

Este comando também pode ser usado para limpar grandes áreas da memória, bem como preencher áreas menores com uma frase de repetição contínua ou byte único, Figura ??.

```

-f 100 12f 'BUFFER'
-d 100 12f
075F:0100 42 55 46 46 45 52 42 55-46 46 45 52 42 55 46 46 BUFFERBUFFERBU
075F:0110 45 52 42 55 46 46 45 52-42 55 46 46 45 52 42 55 ERBUFFERBUFFERBU
075F:0120 46 46 45 52 42 55 46 46-45 52 42 55 46 46 45 52 FFERBUFFERBUFFER
-

```

Figura 16: comando F

3.8.3 Somar e subtrair números (H)

Uma calculadora hexadecimal muito simples (apenas adicionar e subtrair). Basta inserir os valores que ele retorna o resultado da soma e da subtração.

```
-h 103 100
0203 0003
```

Figura 17: comando H

3.8.4 Mover blocos de memória (M)

Este copia todos os bytes de dentro do intervalo especificado para um novo endereço, Figura 18.

```
-M 7C00 7CFF 600
```

Figura 18: comando M

Copia todos os bytes entre o deslocamento 7C00 e 7CFF (inclusive) para o deslocamento 0600 e seguintes.

```
-D7C00
075F:7C00 38 6C 6C 38 00 7C 00 00-00 00 00 00 00 30 30 8118.1.....00
075F:7C10 00 30 30 60 C6 C6 7C 00-00 00 00 00 7C 82 B2 AA .00'.1.....1...
075F:7C20 B2 AA AA 82 7C 00 00 00-00 00 00 00 FE 06 06 ....1.....
075F:7C30 06 00 00 00 00 00 60 E0-63 66 6C 18 30 6E C3 06 .....cf1.0n..
075F:7C40 0C 1F 00 00 60 E0 63 66-6C 18 36 6E DA 3F 06 06 ....cf1.6n.?..
075F:7C50 00 00 00 00 18 18 00 18-18 3C 3C 3C 18 00 00 00 .....<<<....
075F:7C60 00 00 00 00 00 00 36 6C-D8 6C 36 00 00 00 00 .....61.16.....
075F:7C70 00 00 00 00 D8 6C 36 6C-D8 00 00 11 44 11 44 11 .....161....D.D.
-D600
075F:0600 18 18 00 38 18 18 18 18-3C 00 00 00 00 06 06 ...8....<.....
075F:0610 00 0E 06 06 06 06 66 66-3C 00 00 00 E0 60 60 66 .....ff<.... f
075F:0620 6C 78 6C 66 E6 00 00 00-00 00 38 18 18 18 18 18 lxf.....8.....
075F:0630 18 18 3C 00 00 00 00 00-00 00 00 EC FE D6 D6 D6 ..<.....
075F:0640 D6 00 00 00 00 00 00 00-00 DC 66 66 66 66 66 00 .....ffffff.
075F:0650 00 00 00 00 00 00 00 7C-C6 C6 C6 C6 7C 00 00 00 .....1.....
075F:0660 00 00 00 00 00 DC 66 66-66 66 7C 60 F0 00 00 00 .....ffffl ....
075F:0670 00 00 00 76 CC CC CC CC-7C 0C 1E 00 00 00 00 00 ...o....1.....
```

Figura 19: Antes de Mover

```
-M 7C00 7CFF 600
-D600
075F:0600 38 6C 6C 38 00 7C 00 00-00 00 00 00 00 30 30
075F:0610 00 30 30 60 C6 C6 7C 00-00 00 00 00 7C 82 B2 AA
075F:0620 B2 AA AA 82 7C 00 00 00-00 00 00 00 FE 06 06
075F:0630 06 00 00 00 00 00 60 E0-63 66 6C 18 30 6E C3 06
075F:0640 0C 1F 00 00 60 E0 63 66-6C 18 36 6E DA 3F 06 06
075F:0650 00 00 00 00 18 18 00 18-18 3C 3C 3C 18 00 00 00
075F:0660 00 00 00 00 00 00 36 6C-D8 6C 36 00 00 00 00 00
075F:0670 00 00 00 00 D8 6C 36 6C-D8 00 00 11 44 11 44 11
```

Figura 20: Depois de Mover

4 Referências

http://www2.unemat.br/rhycardo/download/tutorial_debug.pdf

<https://montcs.bloomu.edu/Information/LowLevel/DOS-Debug.html>

<http://www.eng.uerj.br/raul/micro/DEBUG>

<http://venugopal417.blogspot.com/2012/09/8086-program-to-add-2-32-bit-numbers.html>