

## Section #02

### (Debug Commands)

**This section is bound to discuss the following elements:**

- ❖ Revision of hexadecimal and binary numbering systems.
- ❖ What is the "main memory" and its addresses?
- ❖ What are "CPU Registers"?
- ❖ Debug:
  - What is debug, advantages, and disadvantages.
  - Debug commands.
  - How to use debug to write, save, load, and show machine code for programs written in assembly language.

#### 1) Hexadecimal and binary numbering systems:

- The hexadecimal (hex) numbering system is used in programmable controllers because a word of data consists of 16 data bits, or two 8-bit bytes.
- The hexadecimal system is a base 16 system, with A to F used to represent decimal numbers 10 to 15 (Table 2.1).
- The hexadecimal numbering system allows the status of many binary bits to be represented in a small space, such as on a computer screen.

Hexadecimal	Binary	Decimal
0 . . . . .	0000 . . . . .	0
1 . . . . .	0001 . . . . .	1
2 . . . . .	0010 . . . . .	2
3 . . . . .	0011 . . . . .	3
4 . . . . .	0100 . . . . .	4
5 . . . . .	0101 . . . . .	5
6 . . . . .	0110 . . . . .	6
7 . . . . .	0111 . . . . .	7
8 . . . . .	1000 . . . . .	8
9 . . . . .	1001 . . . . .	9
A . . . . .	1010 . . . . .	10
B . . . . .	1011 . . . . .	11
C . . . . .	1100 . . . . .	12
D . . . . .	1101 . . . . .	13
E . . . . .	1110 . . . . .	14
F . . . . .	1111 . . . . .	15

*Table 2.1. Hexadecimal and binary numbering systems.*

- Hexadecimal numbers can easily be converted to binary numbers. Conversion is accomplished by writing the 4-bit binary equivalent of the hex digit for each position, as illustrated in Figure 2.1.

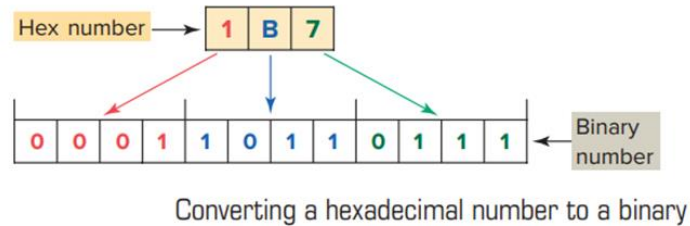


Figure 2.1 Converting a hexadecimal number to binary.

- Example of adding two binary numbers:

$$\begin{array}{r}
 0111 \\
 00111 \quad 7 \\
 10101 \quad 21 \\
 \hline
 11100 = 28
 \end{array}$$

- Example of adding two hexadecimal numbers:

$$\begin{array}{r}
 \phantom{+} \phantom{000} 1 \phantom{00} 1 \phantom{000} 1 \\
 9 \text{ C } 3 \text{ 7 } 2 \text{ 8 } 6 \text{ 5} \\
 + 1 \text{ 3 } 9 \text{ 5 } \text{ E } 8 \text{ 4 } \text{ B} \\
 \hline
 \text{A F C D } 1 \text{ 0 B } 0
 \end{array}$$

## 2) What is the "main memory" and its addresses?

- The main memory in a computer is called Random Access Memory (RAM).
- This is the part of the computer that stores operating system software, software applications and other information for the central processing unit (CPU) to have fast and direct access when needed to perform tasks.
- It is called "random access" because the CPU can go directly to any section of main memory and does not have to go about the process in a sequential order.

- Real memory addressing:
  - Memory is divided into segments as shown in Figure 2.2.
  - Each segment in memory is 64 KB.
  - Each segment number in debug is four hexadecimal numbers.
  - Each offset address in debug is four hexadecimal numbers.
  - Actual address = segment address + offset address as shown in Figure 2.3
  - The CPU deals with these segments through segment registers.

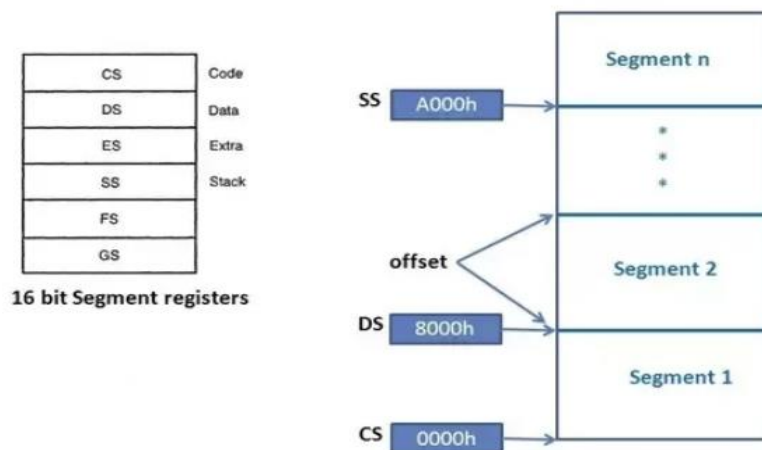


Figure 2.2 Real memory addressing.



Figure 2.3 Memory address as shown in debug.

### 3) What are "CPU Registers"?

- Register is one of a small set of data-holding places that are part of the computer processor.
- It is used to store information temporarily in the CPU, and it is a quickly accessible location available to a computer's processor.

- Registers usually consist of a small amount of fast storage (16-bits).
- As shown in Table 2.2, general-purpose registers can be accessed as 16-bit or 8-bit registers; all other registers can be accessed only as the full 16 bits.
- The data always fill the low 8-bits of a register then if still needed use some or all the high 8-bits.

Category	Bits	Register
General	<b>16</b>	AX, BX, CX, DX
	<b>8</b>	AH, AL, BH, BL, CH, CL, DH, DL
Pointer	<b>16</b>	SP (Stack Pointer), BP (Base Pointer).
Index	<b>16</b>	SI (Source Index), DI (Destination Index)
Segment	<b>16</b>	CS (Code segment), DS (Data Segment) SS (Stack Segment), ES (Extra Segment).
Instruction	<b>16</b>	IP (Instruction Pointer).
Flag	<b>16</b>	FR (Flag Register).

*Table 2.2 Processor Registers*

#### 4) What is debug, advantages, and disadvantages.

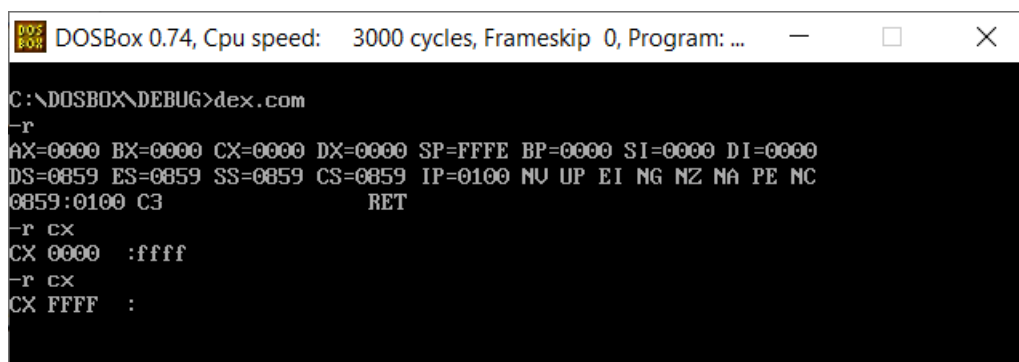
- Debug is a DOS command that works directly with memory and processor registers.
- used for programming in assembly language.
- Only files ending in ".com" can be created.
- The size cannot be larger than 64Kb.
- You can see all debug commands by typing? in DOS.
- All numbers are in hexadecimal.
- can use upper and lower case.
- Ctrl-c stops any command.

## 5) Debug Commands:

MS-DOS Debug Commands		
assemble	A	[address]
compare	C	range address
dump	D	[range]
enter	E	address [list]
fill	F	range list
go	G	[=address] [addresses]
hex	H	value1 value2
input	I	port
load	L	[address] [drive] [firstsector] [number]
move	M	range address
name	N	[pathname] [arglist]
output	O	port byte
proceed	P	[=address] [number]
quit	Q	
register	R	[register]
search	S	range list
trace	T	[=address] [number]
unassemble	U	[range]
write	W	[address] [drive] [firstsector] [number]

### 1. R (Register)

- The **R** command displays and modifies the register contents.
- *Syntax is → R registerName*
- **R CX** → only displays the contents of the CX register, followed by:
- You can enter the hexadecimal number to change the content of the cx register.
- Pad the left with zero if the hex is less than four digits.
- You cannot insert a digit less than 0000 and larger than FFFF.



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0100 NV UP EI NG NZ NA PE NC
0859:0100 C3          RET
-r cx
CX 0000 :ffff
-r cx
CX FFFF :
```

## 2. F (Fill)

- Filling block of memory with data.
- *Syntax is  $\rightarrow F \text{ StartAddress EndAddress Data}$*
- F 0100 010f ff  $\rightarrow$  filling 16 bytes of memory with ff
- F 0100 012f ff  $\rightarrow$  filling  $3 \times 16 = 48$  bytes of memory with ff
- F 0100 01ff ff  $\rightarrow$  filling  $16 \times 16 = 256$  bytes of memory with ff

## 3. D (Dumb)

- Display the contents of an area of memory.
- *Syntax is  $\rightarrow D \text{ StartAddress EndAddress}$*
- *Syntax is  $\rightarrow D \text{ StartAddress}$*
- *Syntax is  $\rightarrow D \text{ StartAddress length}$*

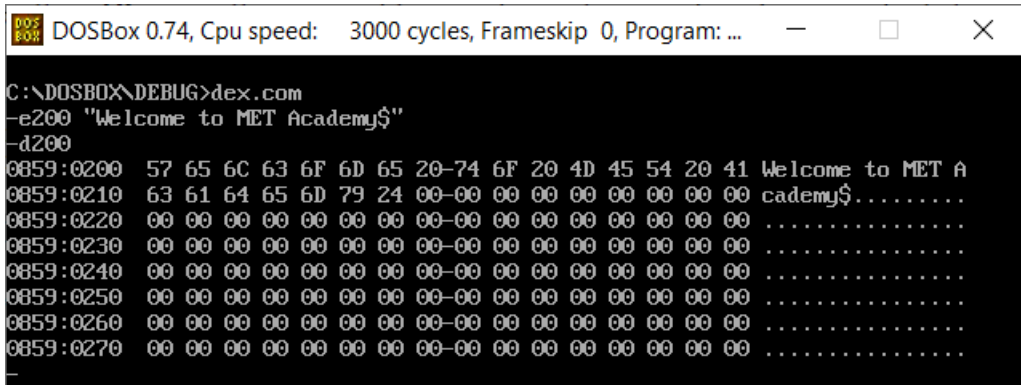
```
DOS
BOX DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-f 0100 010f ff
-d 0100 010f
0059:0100  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
-
```

```
DOS
BOX DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-f 0100 012f ff
-d 0100 012f
0059:0100  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0110  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0120  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
-
```

```
DOS
BOX DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-f 0100 01ff ff
-d 0100 L100
0059:0100  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0110  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0120  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0130  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0140  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0150  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0160  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0170  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:0190  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:01A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:01B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:01C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:01D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:01E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0059:01F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
-
```

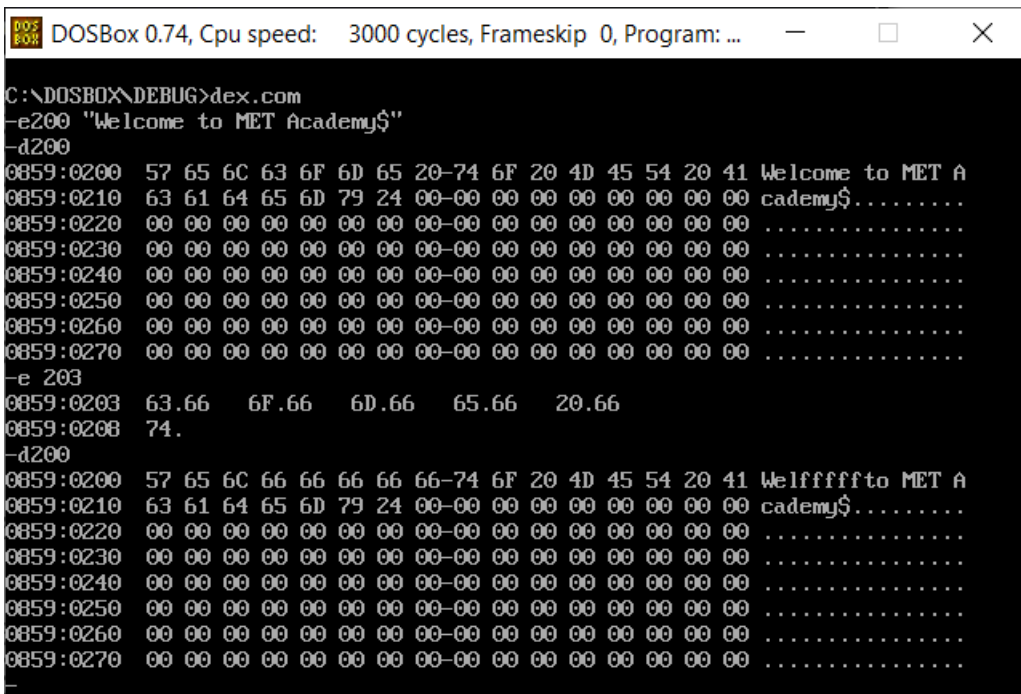
## 4. E (Enter)

- The E command is used to enter data directly into memory locations.
- Single(') or double("") quote marks are acceptable for entering ASCII data.
- *Syntax is → EAddress "ASCII data\$"*



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-e200 "Welcome to MET Academy$"
-d200
0859:0200 57 65 6C 63 6F 6D 65 20-74 6F 20 4D 45 54 20 41 Welcome to MET A
0859:0210 63 61 64 65 6D 79 24 00-00 00 00 00 00 00 00 00 cademy$.
0859:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

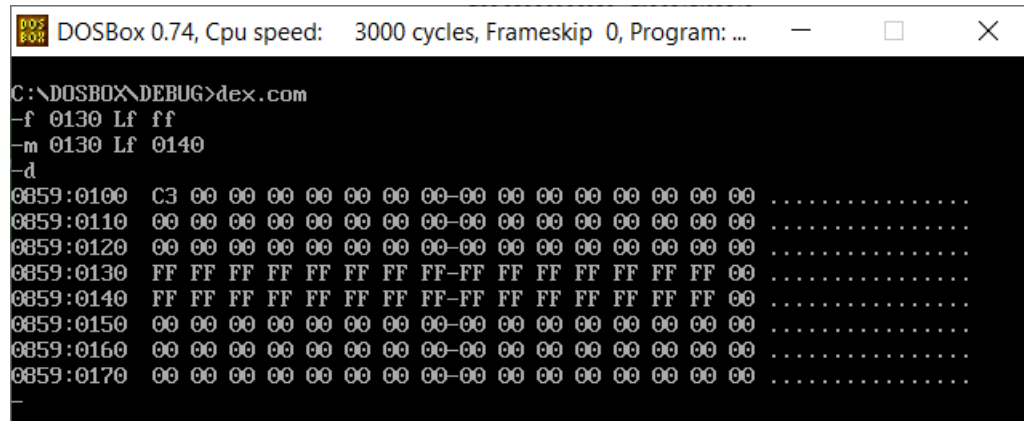
- You can alter data with E command.



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-e200 "Welcome to MET Academy$"
-d200
0859:0200 57 65 6C 63 6F 6D 65 20-74 6F 20 4D 45 54 20 41 Welcome to MET A
0859:0210 63 61 64 65 6D 79 24 00-00 00 00 00 00 00 00 00 cademy$.
0859:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
-e 203
0859:0203 63.66 6F.66 6D.66 65.66 20.66
0859:0208 74.
-d200
0859:0200 57 65 6C 66 66 66 66 66-74 6F 20 4D 45 54 20 41 Welffffffto MET A
0859:0210 63 61 64 65 6D 79 24 00-00 00 00 00 00 00 00 00 cademy$.
0859:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0859:0270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

## 5. M (Move)

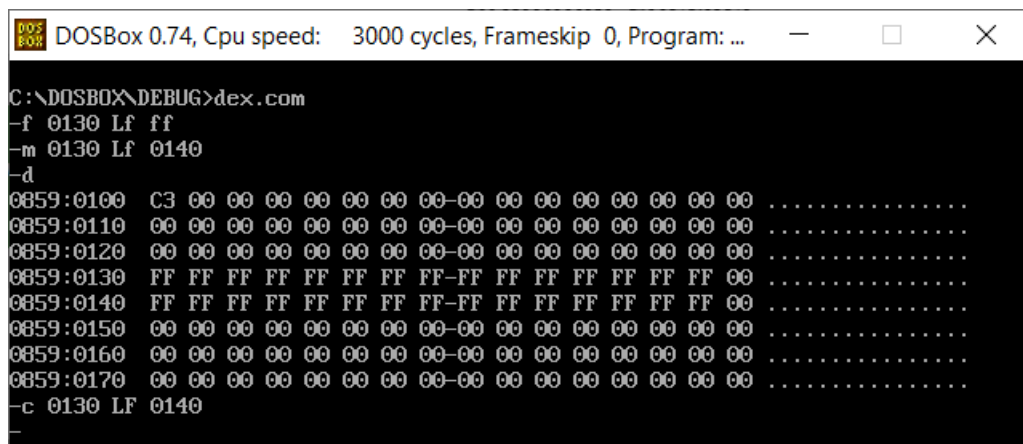
- The M command is used to move or copy data from one location to another.
- *Syntax is* → **M [StartA EndA DestinationA]**.
- M 0130 Lf 0140 → Copy data in address (0130 : 013f) to (0140:014f)



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-f 0130 Lf ff
-m 0130 Lf 0140
-d
0059:0100  C3 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0130  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF 00 .....
0059:0140  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF 00 .....
0059:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
```

## 6. C Compare

- The C command is used to check two areas of memory and display bytes that contain different data.
- If two area are identical, debug replay with –
- *Syntax is* → **C [StartA EndA StartCompareA]**.
- C 0130 L5 0140 → This return – (There is no difference).



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-f 0130 Lf ff
-m 0130 Lf 0140
-d
0059:0100  C3 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0130  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF 00 .....
0059:0140  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF 00 .....
0059:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0059:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-c 0130 LF 0140
-
```



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-f 0130 Lf ff
-f 0140 Lf aa
-c 0130 Lf 0140
0059:0130 FF AA 0059:0140
0059:0131 FF AA 0059:0141
0059:0132 FF AA 0059:0142
0059:0133 FF AA 0059:0143
0059:0134 FF AA 0059:0144
0059:0135 FF AA 0059:0145
0059:0136 FF AA 0059:0146
0059:0137 FF AA 0059:0147
0059:0138 FF AA 0059:0148
0059:0139 FF AA 0059:0149
0059:013A FF AA 0059:014A
0059:013B FF AA 0059:014B
0059:013C FF AA 0059:014C
0059:013D FF AA 0059:014D
0059:013E FF AA 0059:014E
```

## 7. S (Search)

- The S command is used to search a block of data for a specific value.
- *Syntax is*  $\rightarrow S[StartA\ EndA\ Value]$
- S 0130 Lf ff  $\rightarrow$  returns with an address that contains ff value.

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-f 0130 Lf ff
-s 0130 Lf ff
0059:0130
0059:0131
0059:0132
0059:0133
0059:0134
0059:0135
0059:0136
0059:0137
0059:0138
0059:0139
0059:013A
0059:013B
0059:013C
0059:013D
```

## 8. H (Hex)

- The H command is used to add and subtract two hexadecimal numbers.
- Show sum first, then difference.

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\DOSBOX\DEBUG>dex.com
-h aaa 531
0FDB 0579
-h fff 3
1002 0FFC
-h dbf ace
188D 02F1
-h 4 fffc
00010000 0008
-h 100 123
0223 FFDD
-h 7fff 8000
FFFF FFFF
-
```

6) How to use debug to write, save, load, and show machine code for programs written in assembly language. (Section No.3)

**Scan the QR code to download debug64-bit V1.11.**

