AACS3064
Computer System Architecture

5. Machine Execution: Using the Debug Program

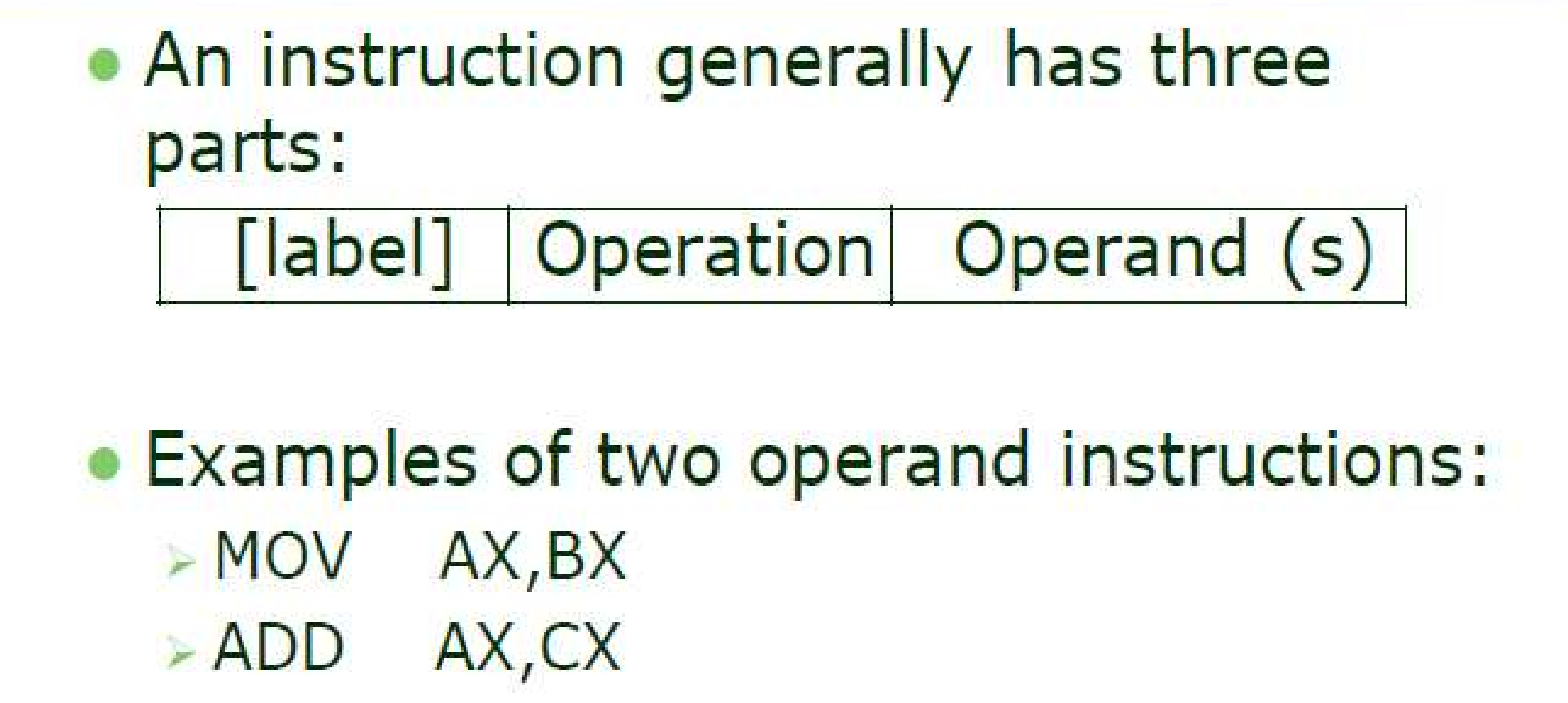
Chapter Overview

1) Instruction format
2) Machine language instructions
3) Instruction execution
4) Introduction to Debug program

1. Instruction format

1. Instruction format

Instruction Format



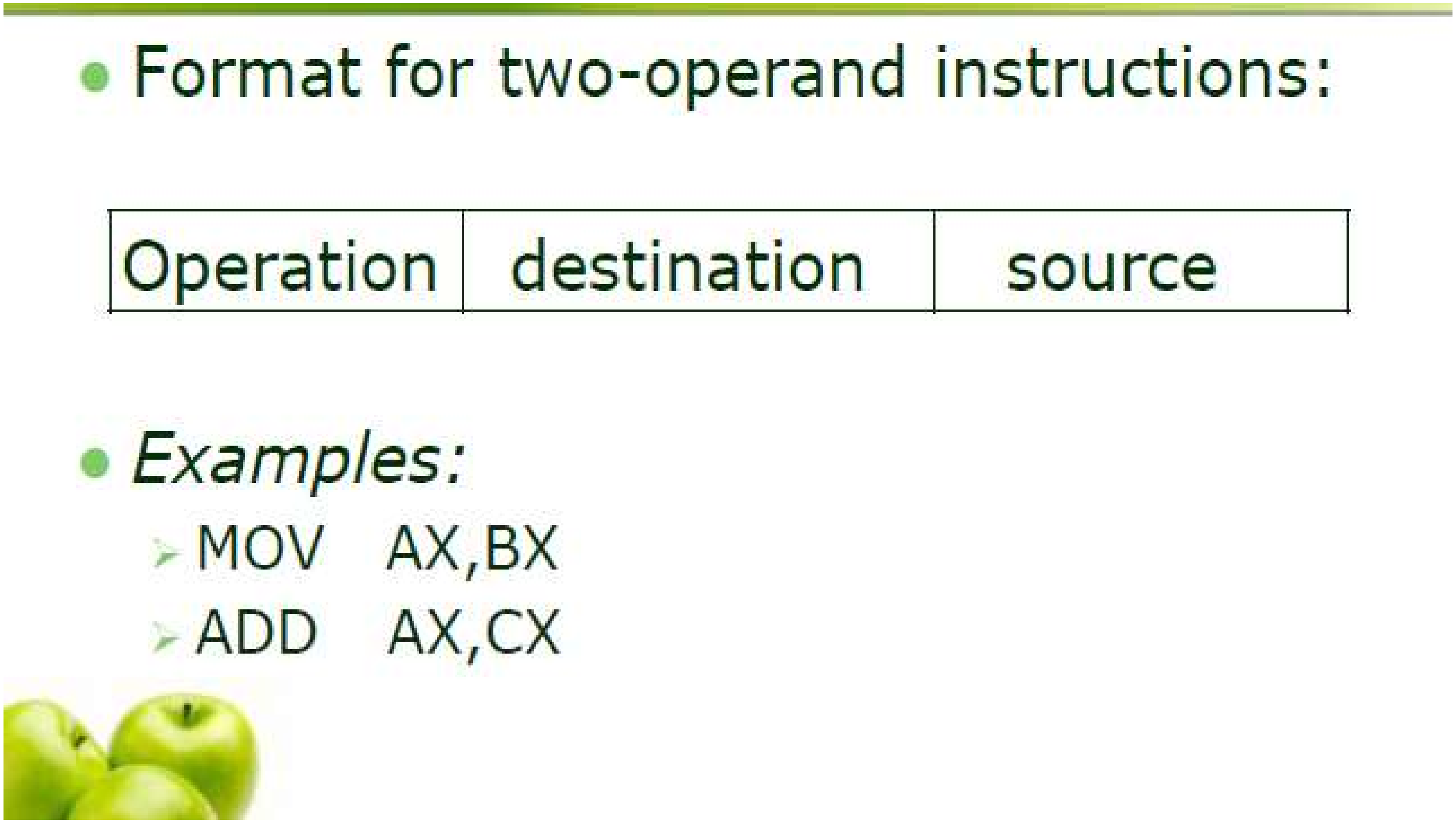
- An instruction generally has three parts:

[label]	Operation	Operand (s)

- Examples of two operand instructions:
  - MOV AX,BX
  - ADD AX,CX

1. Instruction format (Continued)

- **Two-operand Instruction**



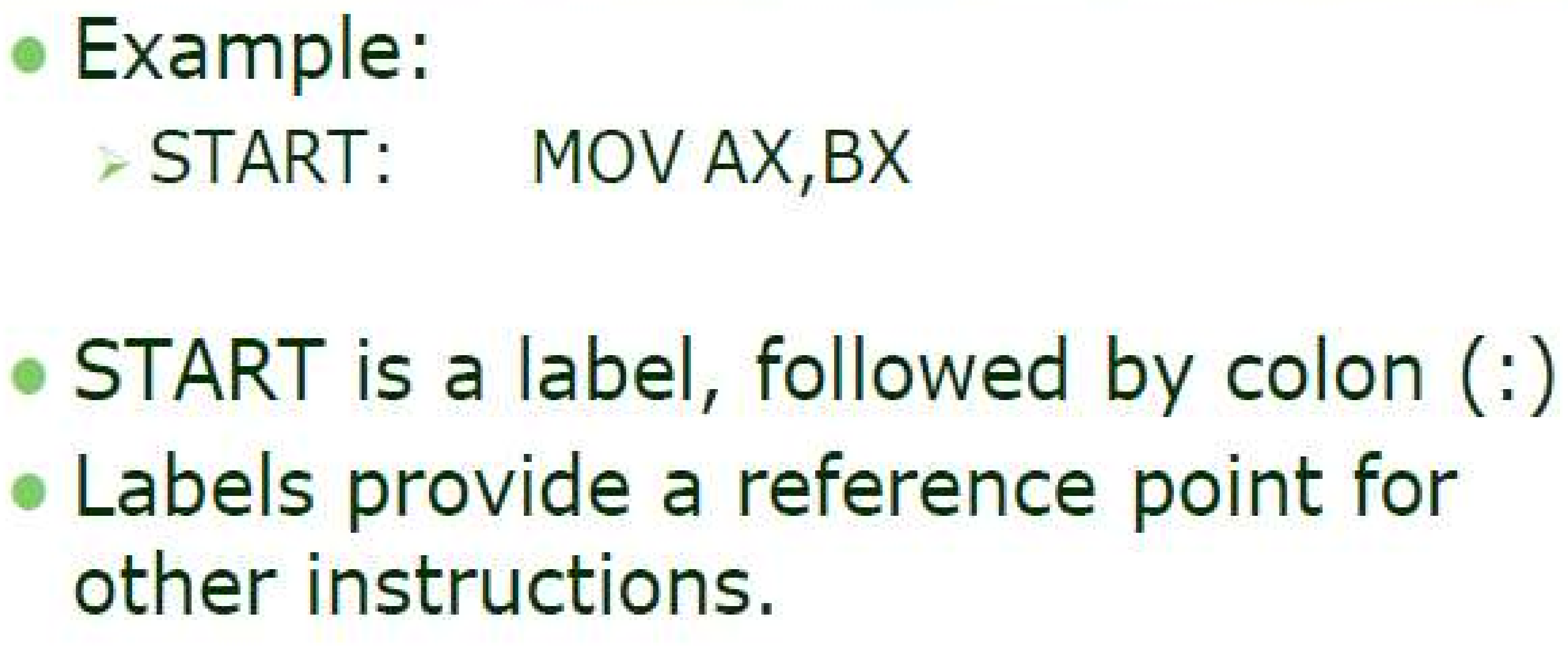
- Format for two-operand instructions:

Operation	destination	source

- *Examples:*
  - MOV AX,BX
  - ADD AX,CX

1. Instruction format (Continued)

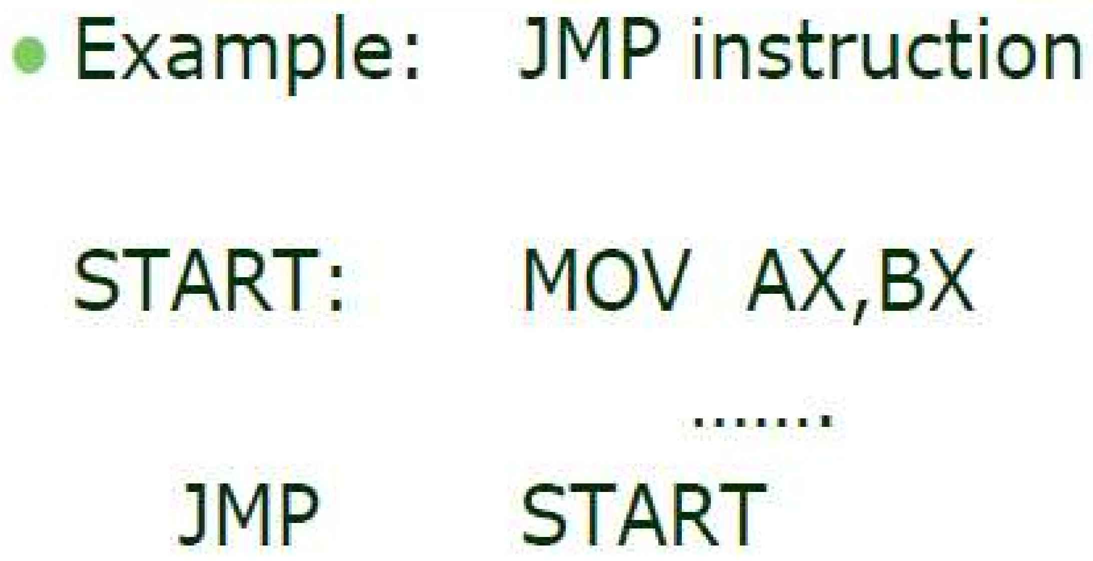
- Instruction with label



- Example:
  - START: MOV AX,BX
- START is a label, followed by colon (:)
- Labels provide a reference point for other instructions.

1. Instruction format (Continued)

- **Single-Operand Instruction**

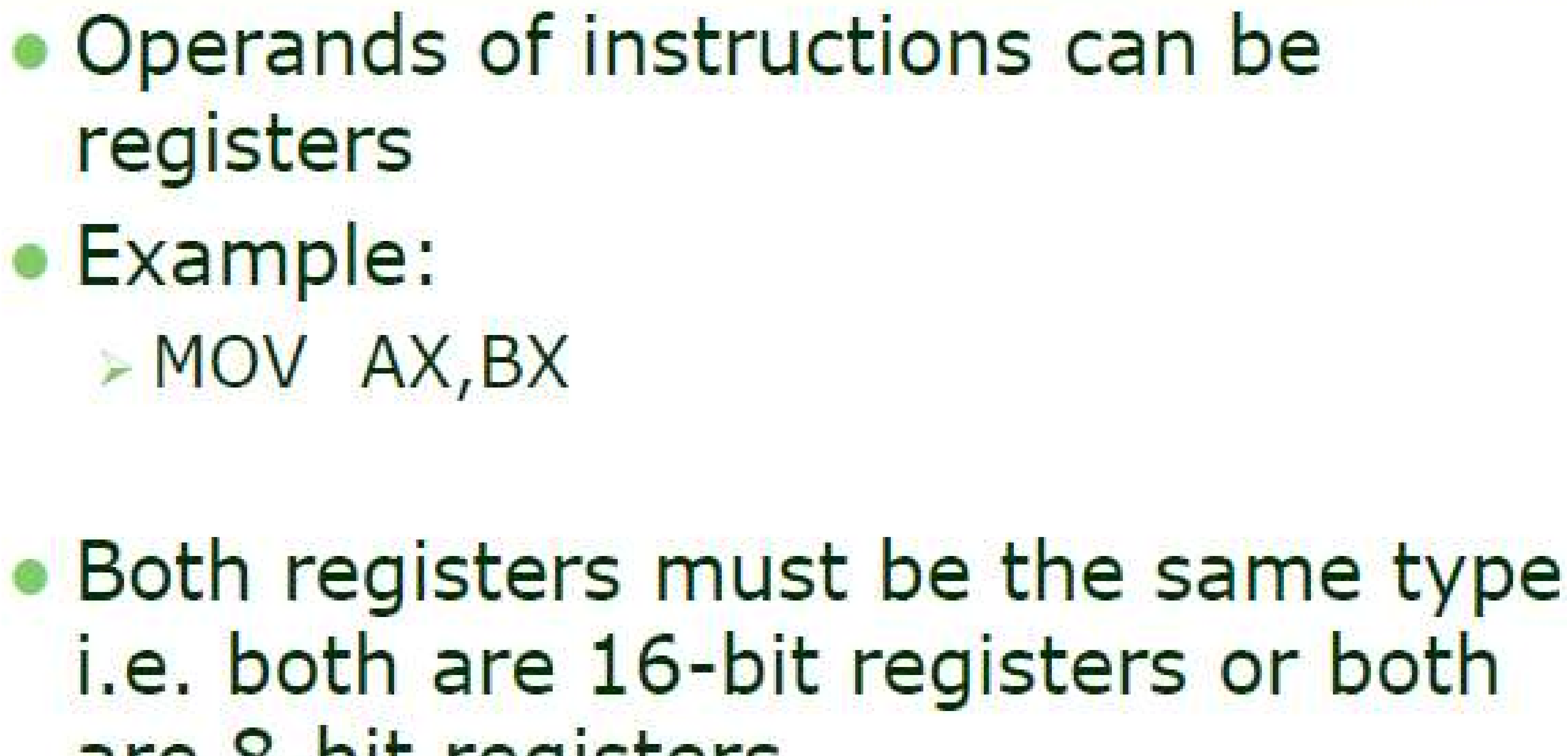


- Example: JMP instruction

```
START:     MOV  AX,BX
                 .......
    JMP     START
```

1. Instruction format (Continued)

- **Registers as Operand**

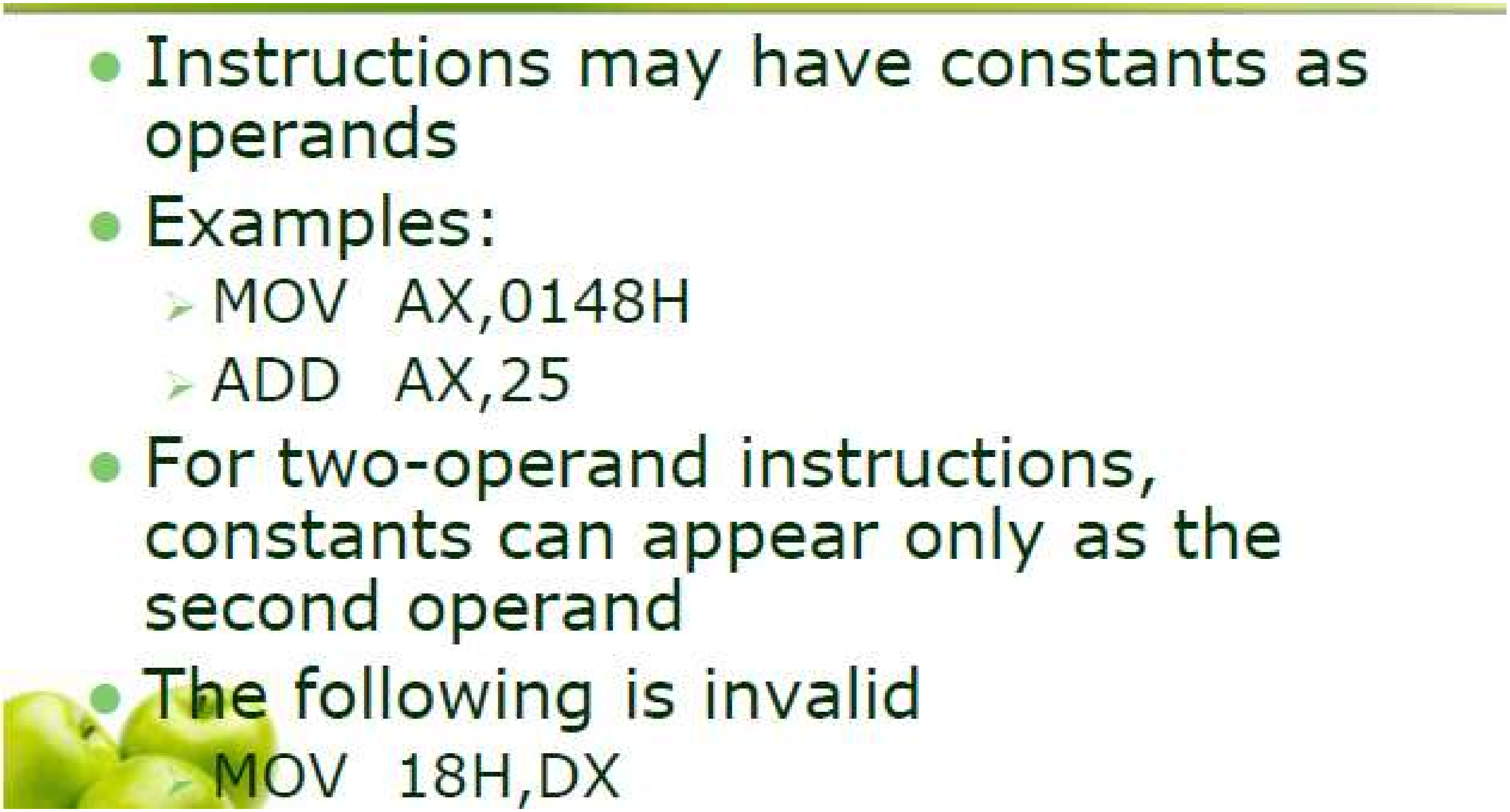


- Operands of instructions can be registers
- Example:
  - MOV AX,BX

- Both registers must be the same type i.e. both are 16-bit registers or both are 8-bit registers.

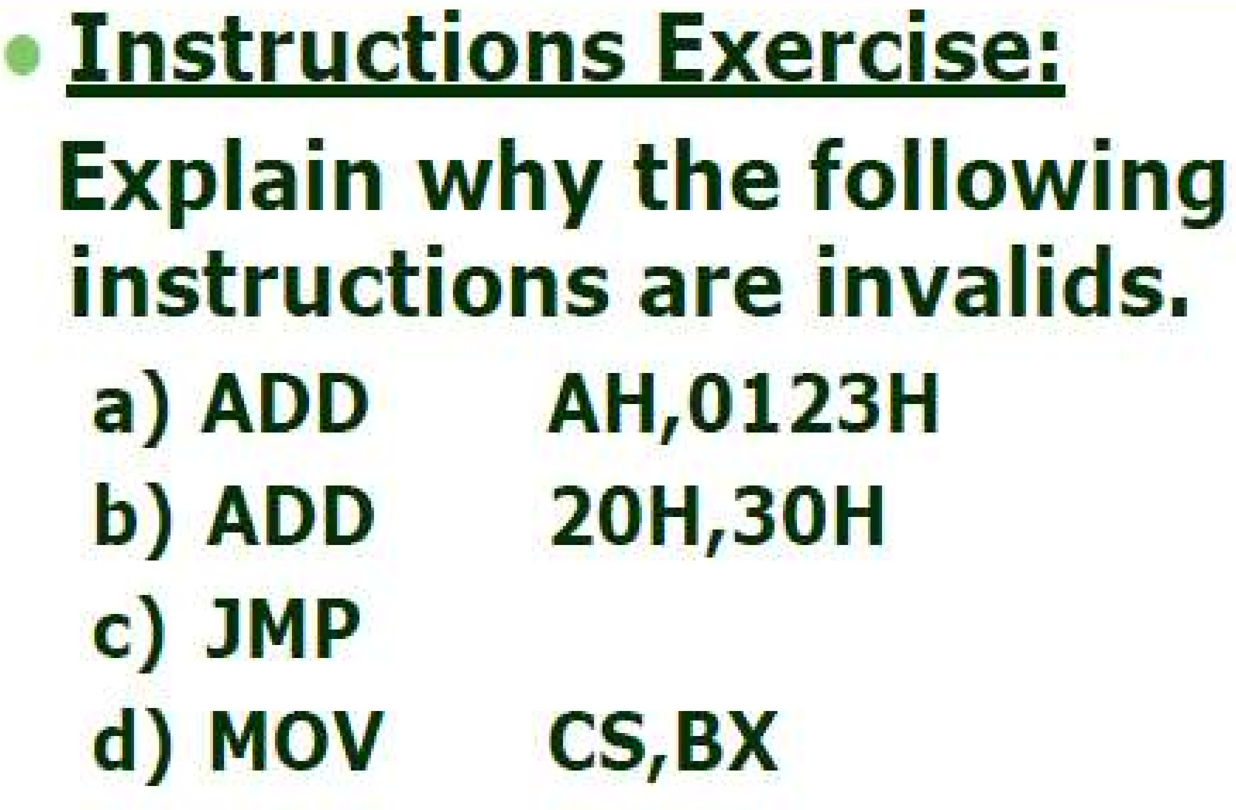
1. Instruction format (Continued)

- **Constants as Operand**



- Instructions may have constants as operands
- Examples:
  - MOV AX,0148H
  - ADD AX,25
- For two-operand instructions, constants can appear only as the second operand
- The following is invalid
  - MOV 18H,DX

1. Instruction format (Continued)



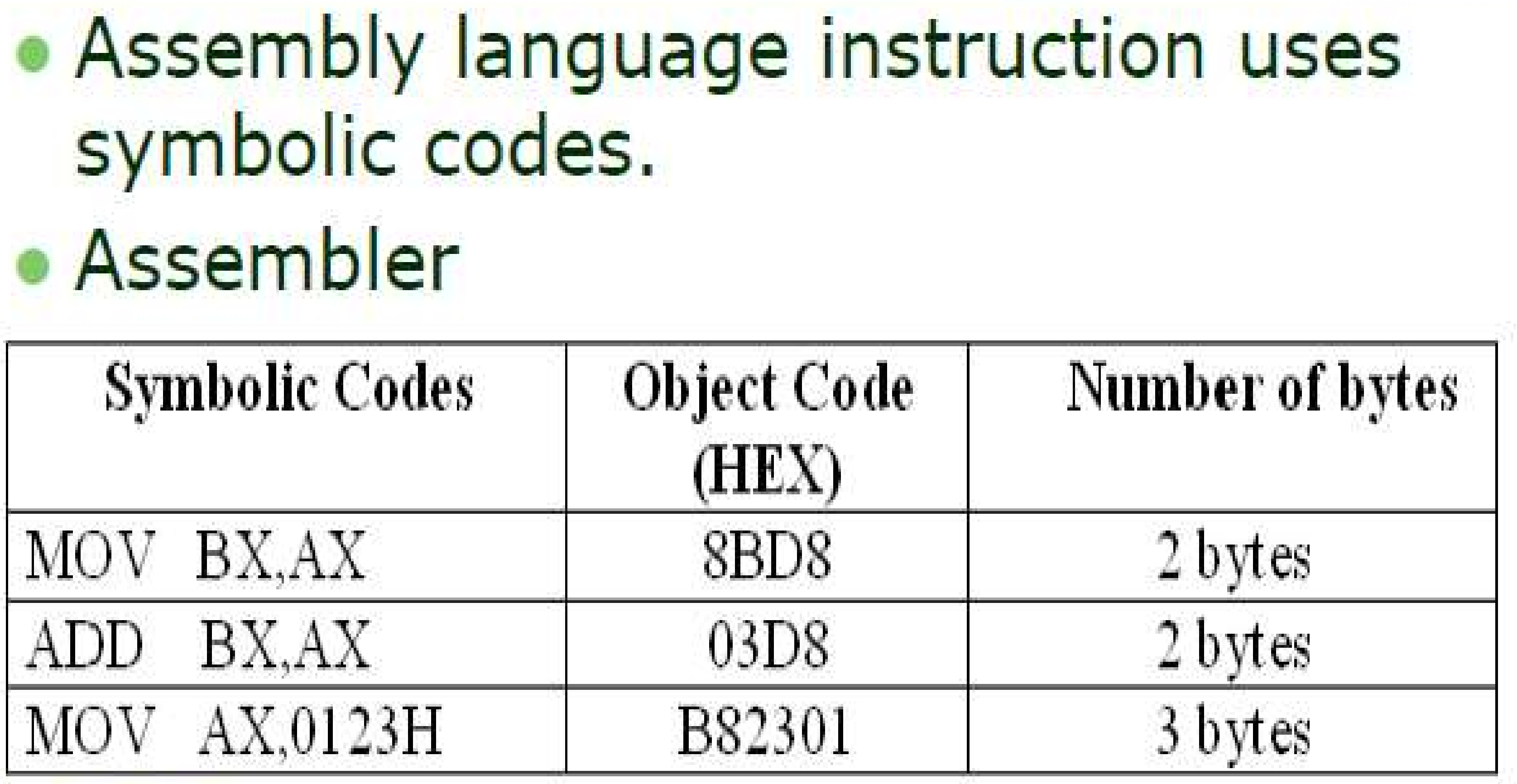
- **Instructions Exercise:**

Explain why the following instructions are invalids.

```
a) ADD    AH,0123H
b) ADD    20H,30H
c) JMP
d) MOV    CS,BX
```

2. Machine language instructions

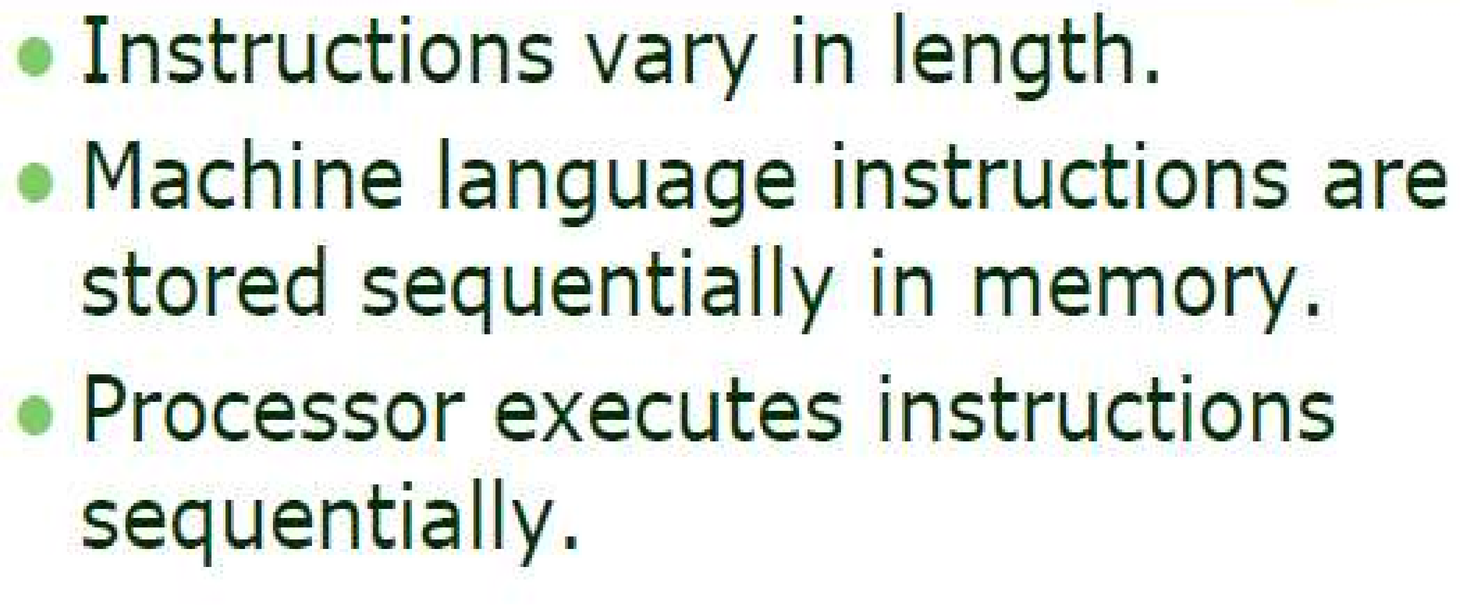
2. Machine language instructions



- Assembly language instruction uses symbolic codes.
- Assembler

Symbolic Codes	Object Code (HEX)	Number of bytes
MOV BX,AX	8BD8	2 bytes
ADD BX,AX	03D8	2 bytes
MOV AX,0123H	B82301	3 bytes

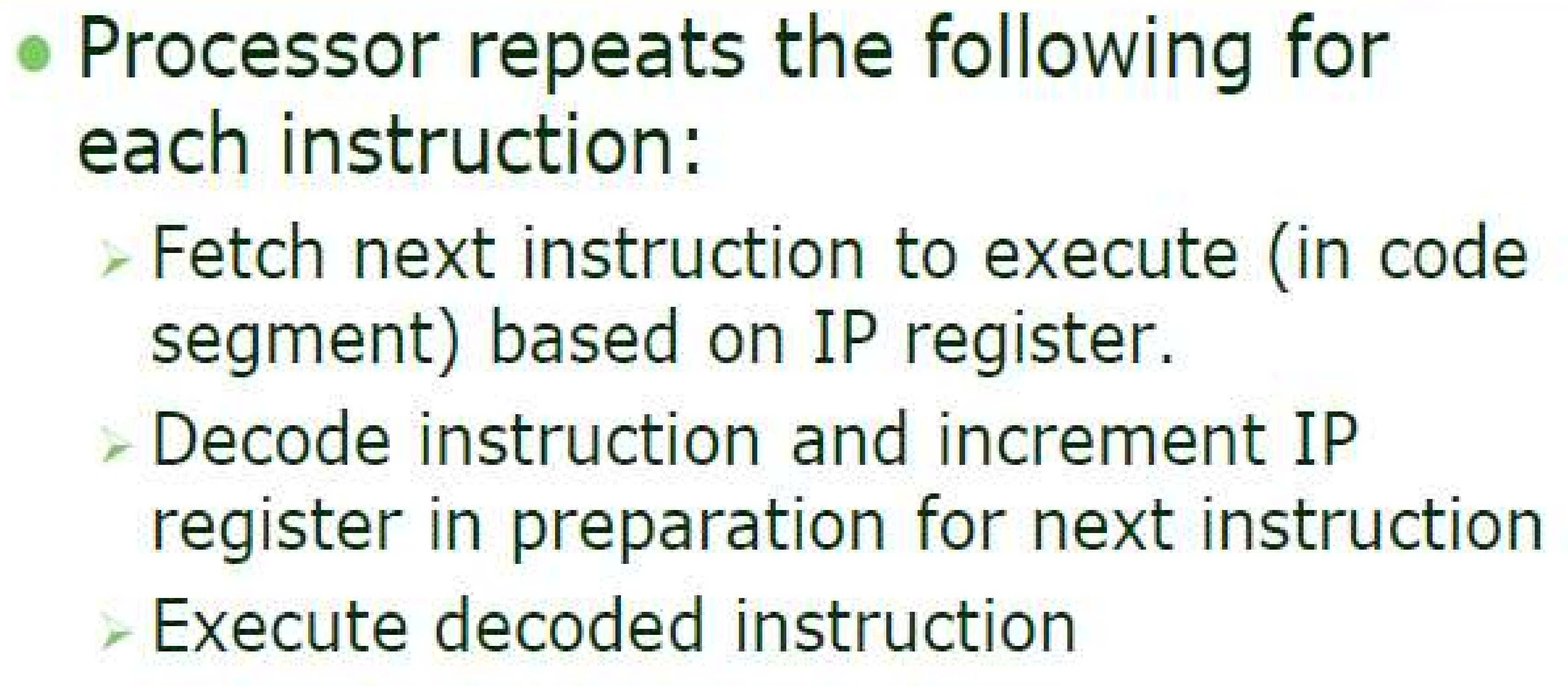
2. Machine language instructions (Continued)



- Instructions vary in length.
- Machine language instructions are stored sequentially in memory.
- Processor executes instructions sequentially.

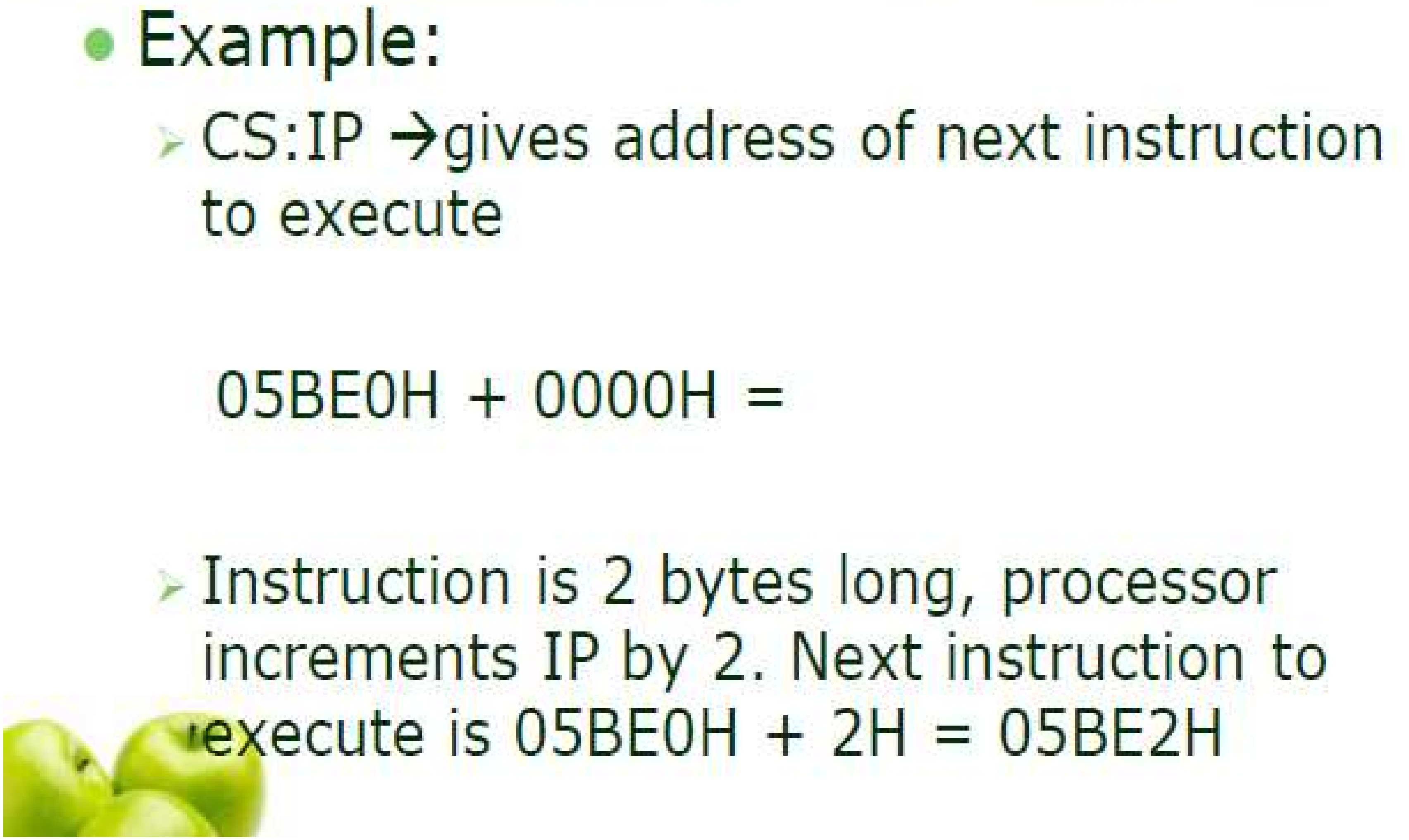
3. Instruction execution

3. Instruction execution



- Processor repeats the following for each instruction:
  - Fetch next instruction to execute (in code segment) based on IP register.
  - Decode instruction and increment IP register in preparation for next instruction
  - Execute decoded instruction

3. Instruction execution (Continued)

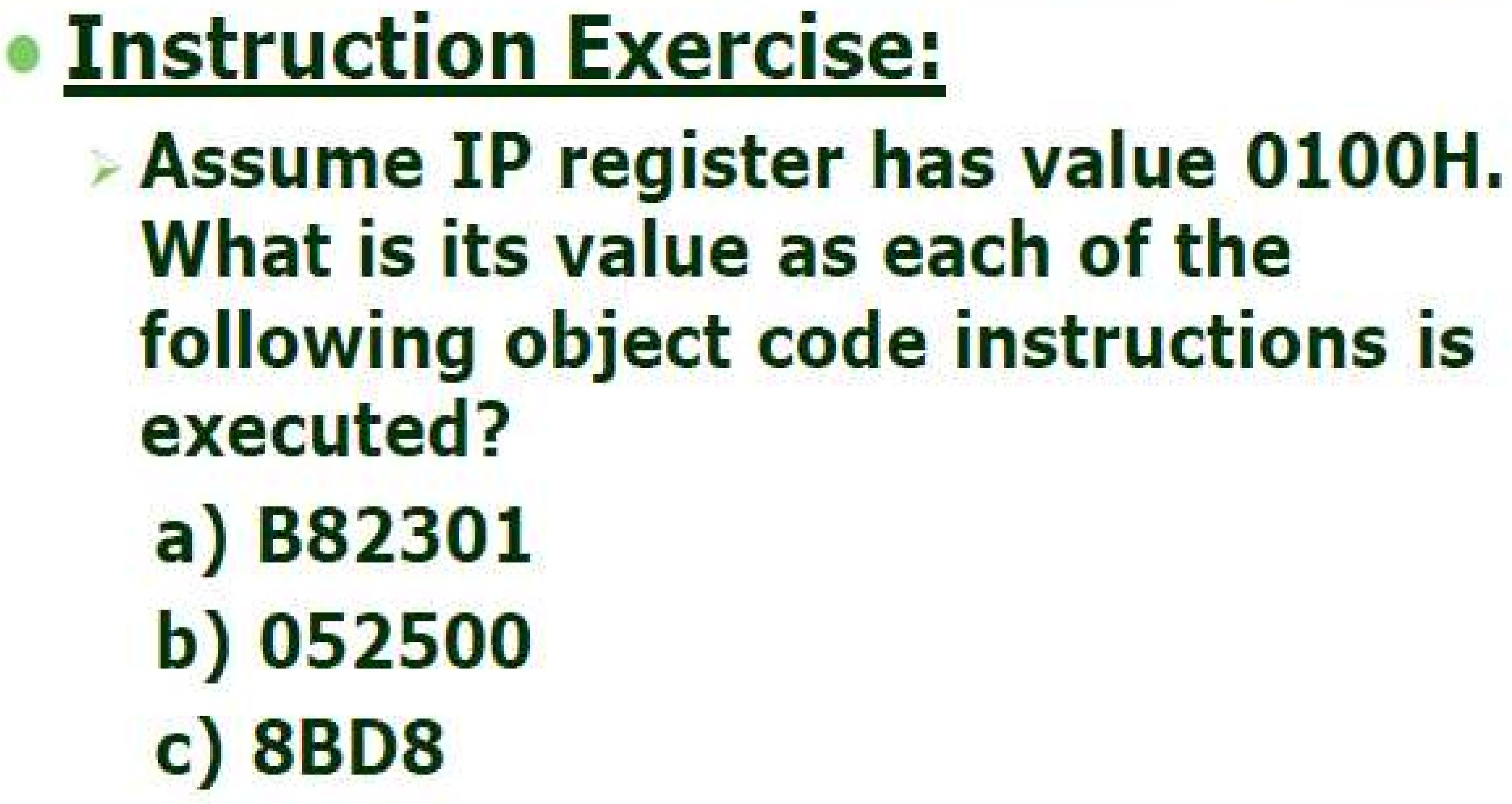


- Example:
  - CS:IP →gives address of next instruction to execute

    05BE0H + 0000H =

  - Instruction is 2 bytes long, processor increments IP by 2. Next instruction to execute is 05BE0H + 2H = 05BE2H

3. Instruction execution (Continued)



- **Instruction Exercise:**
  - **Assume IP register has value 0100H. What is its value as each of the following object code instructions is executed?**

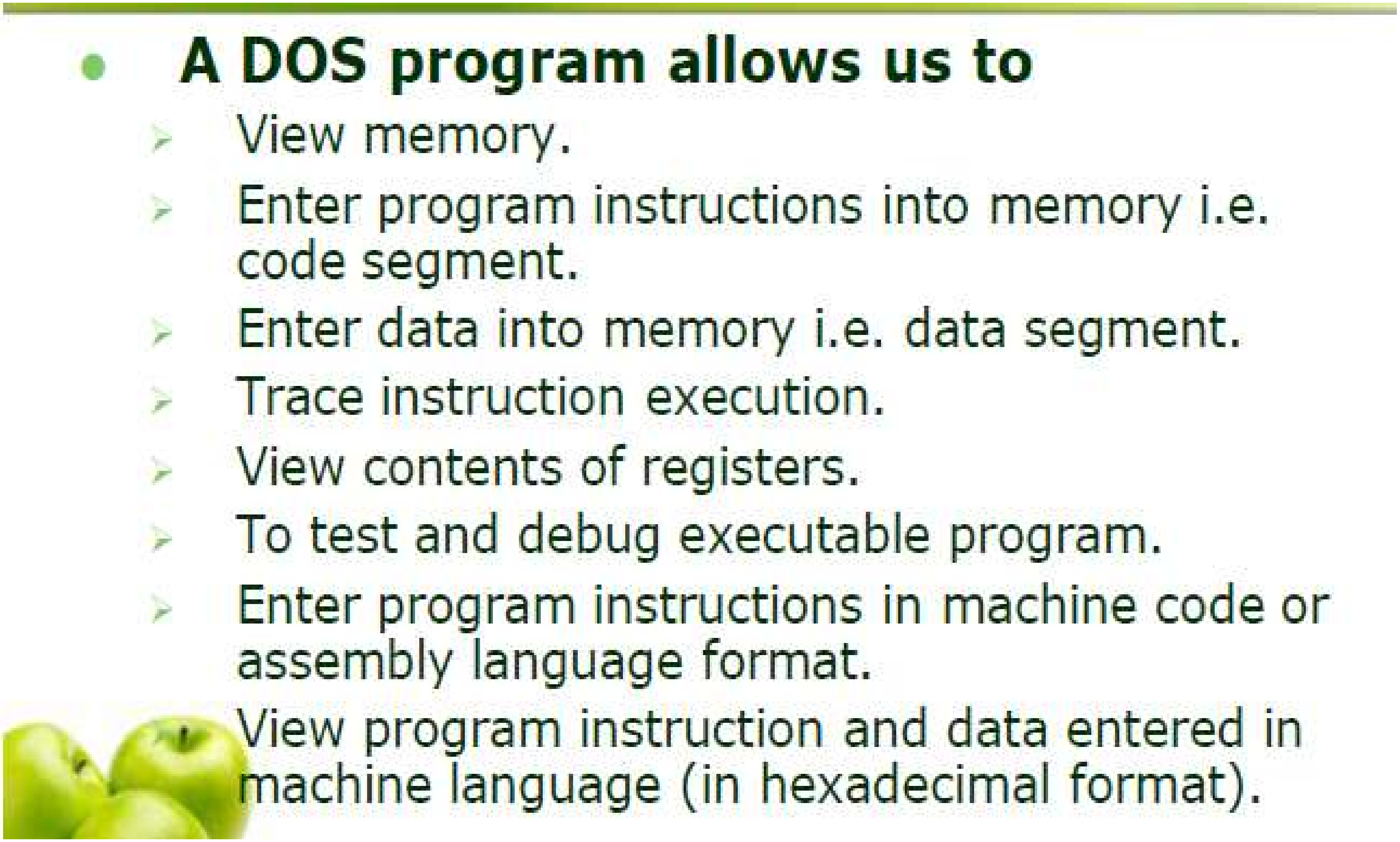
    a) B82301

    b) 052500

    c) 8BD8

4. Introduction to Debug program

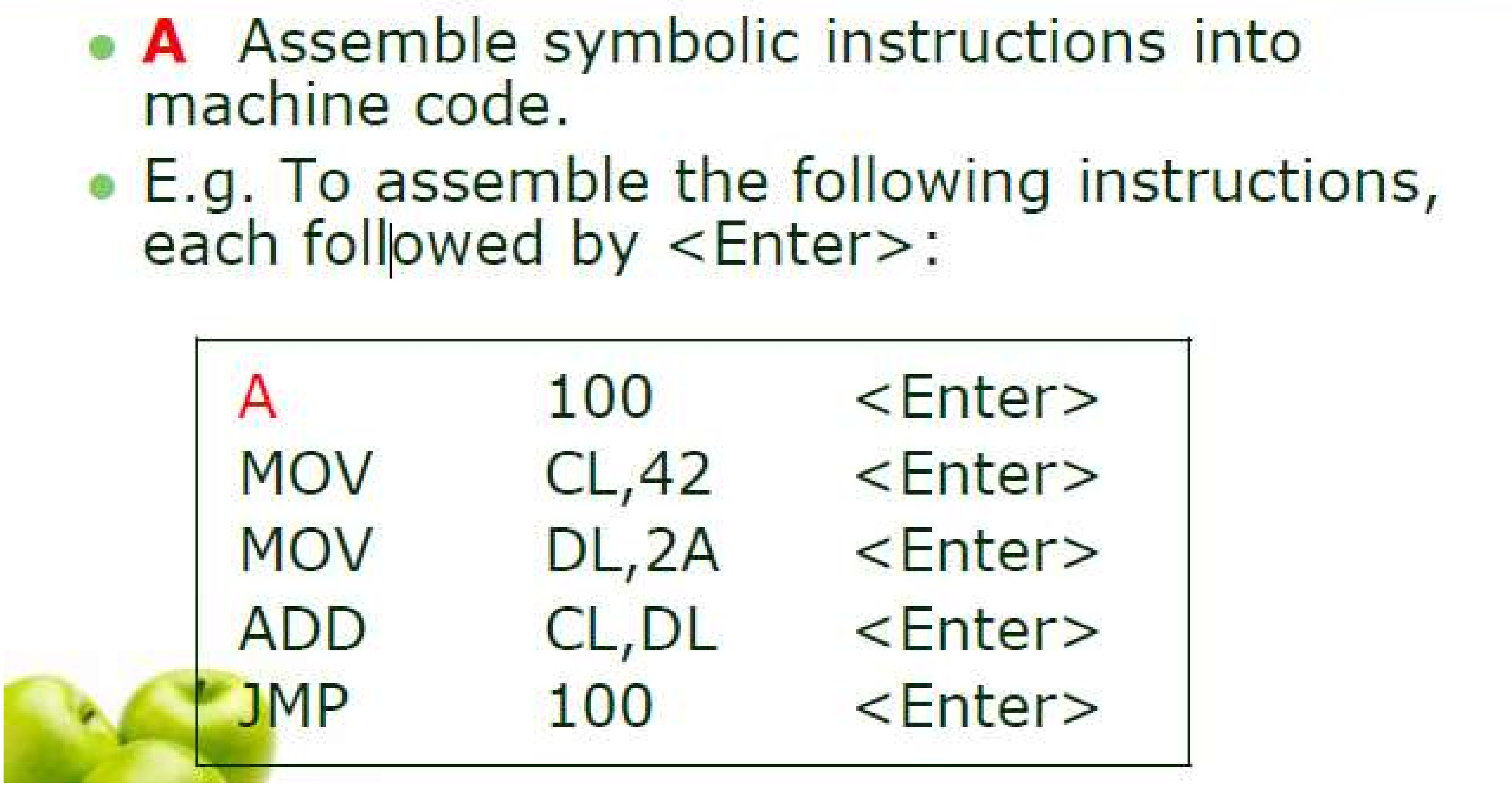
4. Introduction to Debug program



- **A DOS program allows us to**
  - View memory.
  - Enter program instructions into memory i.e. code segment.
  - Enter data into memory i.e. data segment.
  - Trace instruction execution.
  - View contents of registers.
  - To test and debug executable program.
  - Enter program instructions in machine code or assembly language format.
  - View program instruction and data entered in machine language (in hexadecimal format).

4. Introduction to Debug program (Continued)

Sets of commands in DEBUG program

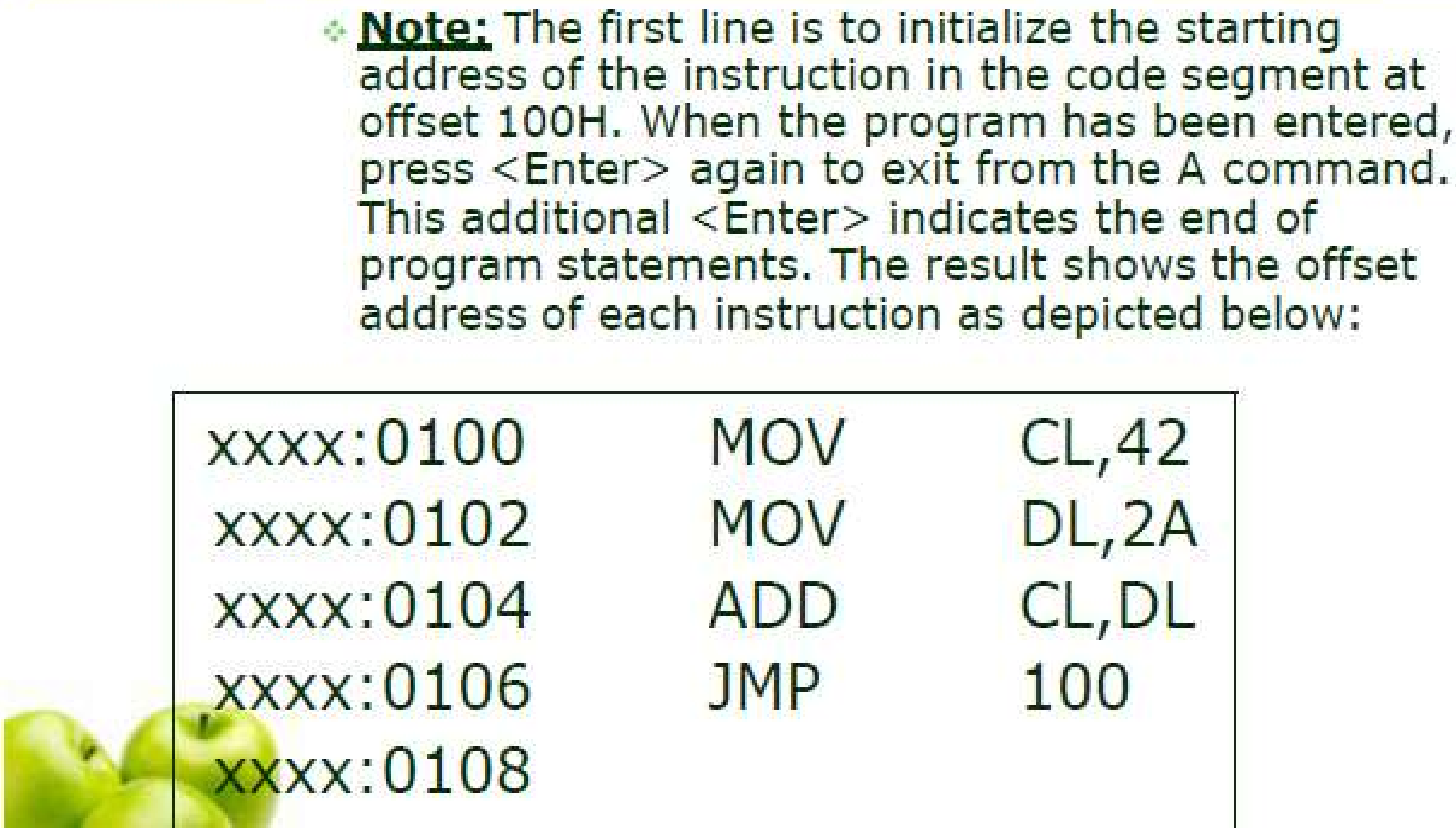


- **A** Assemble symbolic instructions into machine code.
- E.g. To assemble the following instructions, each followed by <Enter>:

```
A       100     <Enter>
MOV     CL,42   <Enter>
MOV     DL,2A   <Enter>
ADD     CL,DL   <Enter>
JMP     100     <Enter>
```

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**

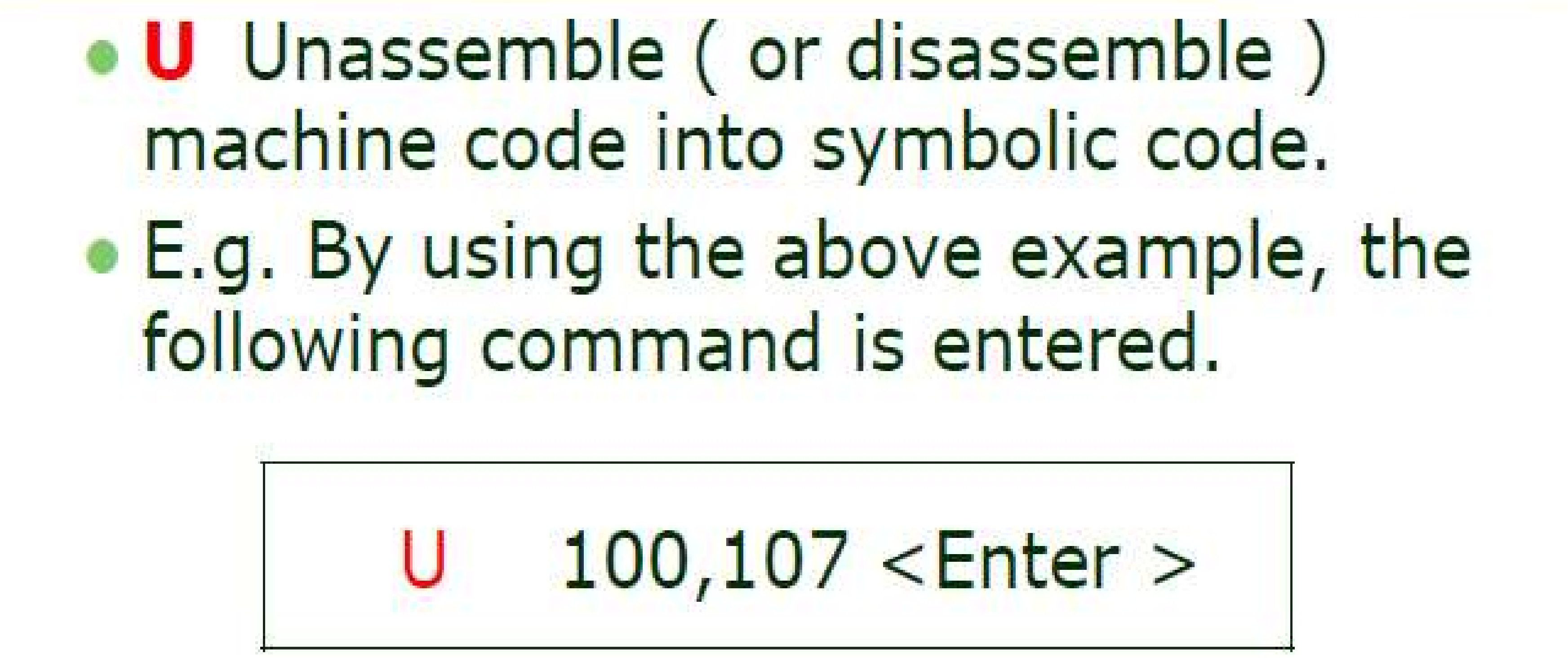


Note: The first line is to initialize the starting address of the instruction in the code segment at offset 100H. When the program has been entered, press <Enter> again to exit from the A command. This additional <Enter> indicates the end of program statements. The result shows the offset address of each instruction as depicted below:

```
xxxx:0100     MOV     CL,42
xxxx:0102     MOV     DL,2A
xxxx:0104     ADD     CL,DL
xxxx:0106     JMP     100
xxxx:0108
```

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**

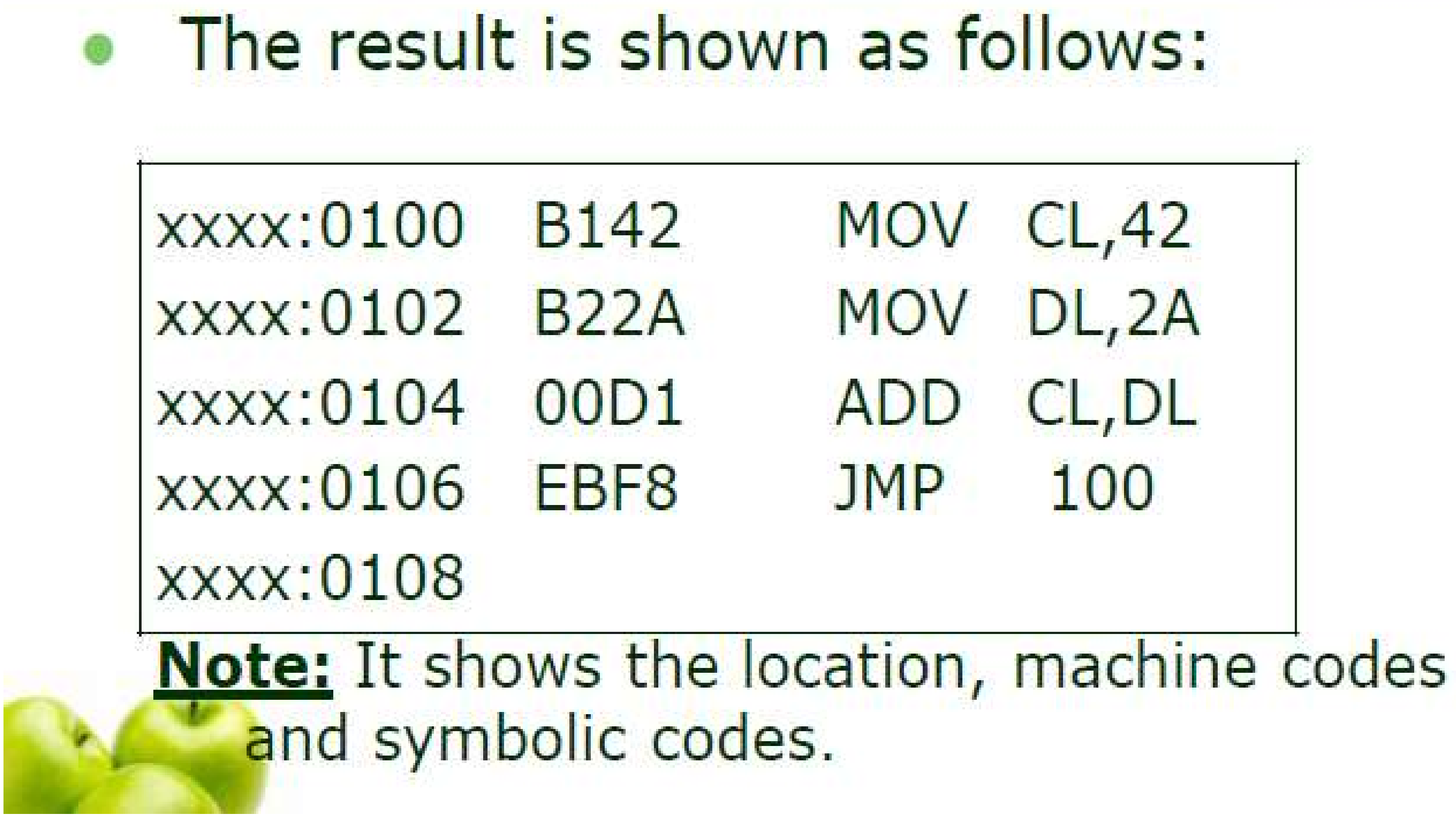


- **U** Unassemble (or disassemble) machine code into symbolic code.
- E.g. By using the above example, the following command is entered.

```
U    100,107 <Enter >
```

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**



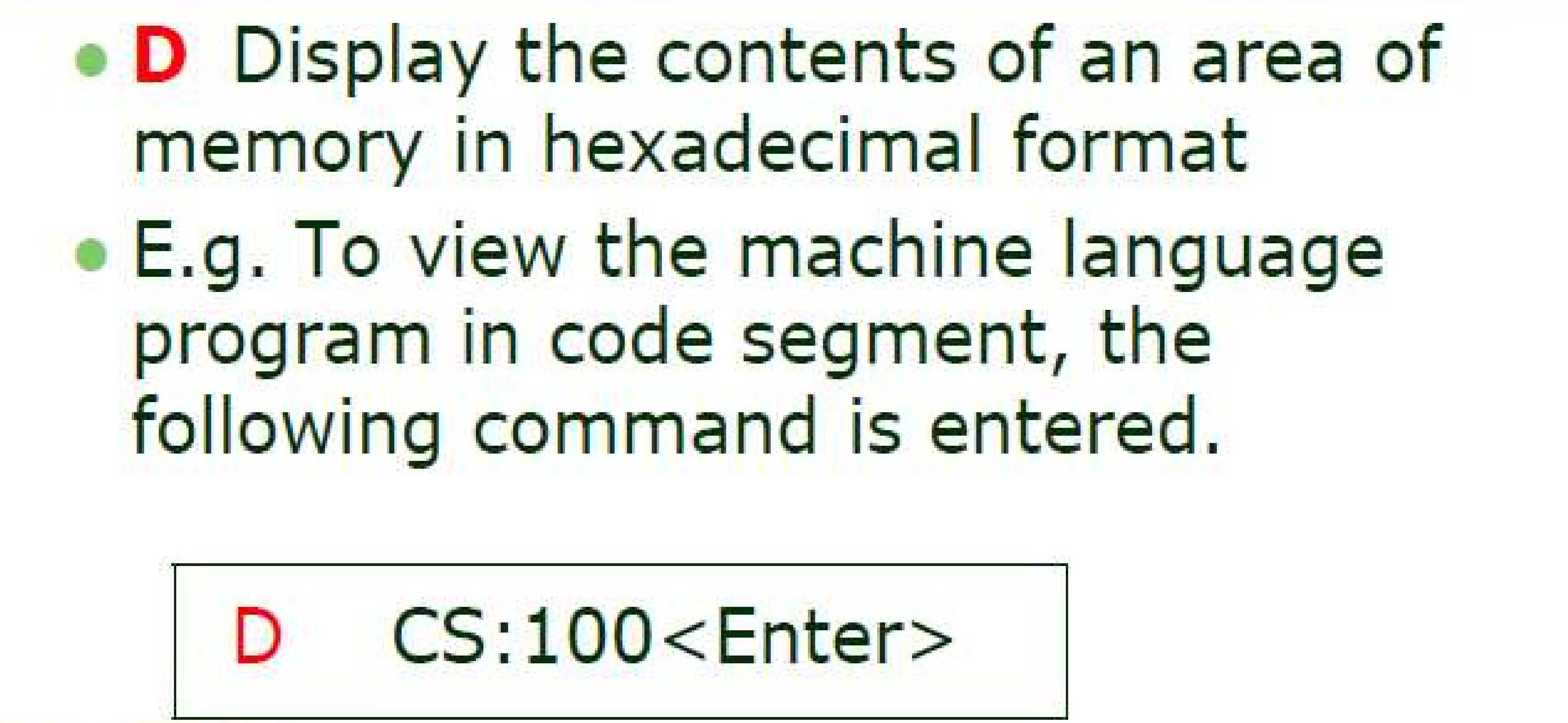
- The result is shown as follows:

```
xxxx:0100   B142      MOV   CL,42
xxxx:0102   B22A      MOV   DL,2A
xxxx:0104   00D1      ADD   CL,DL
xxxx:0106   EBF8      JMP    100
xxxx:0108
```

Note: It shows the location, machine codes and symbolic codes.

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**

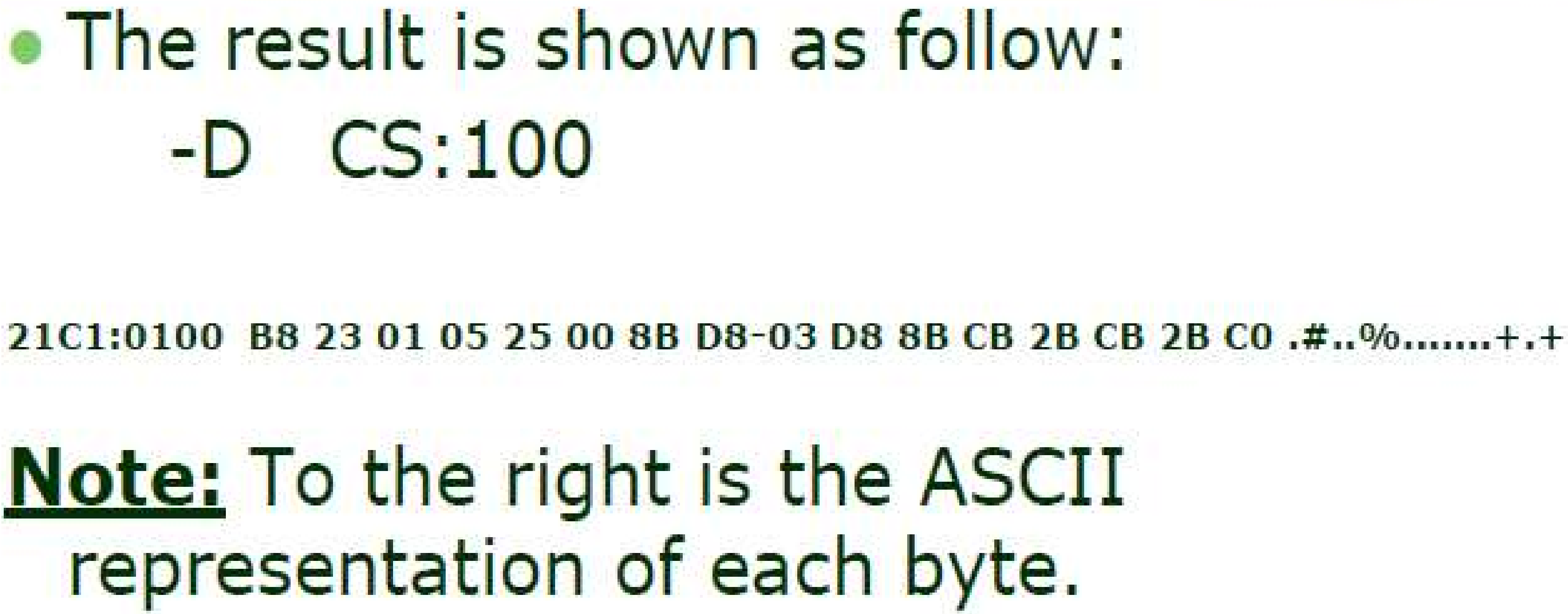


- **D** Display the contents of an area of memory in hexadecimal format
- E.g. To view the machine language program in code segment, the following command is entered.

```
D    CS:100<Enter>
```

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**



- The result is shown as follow:

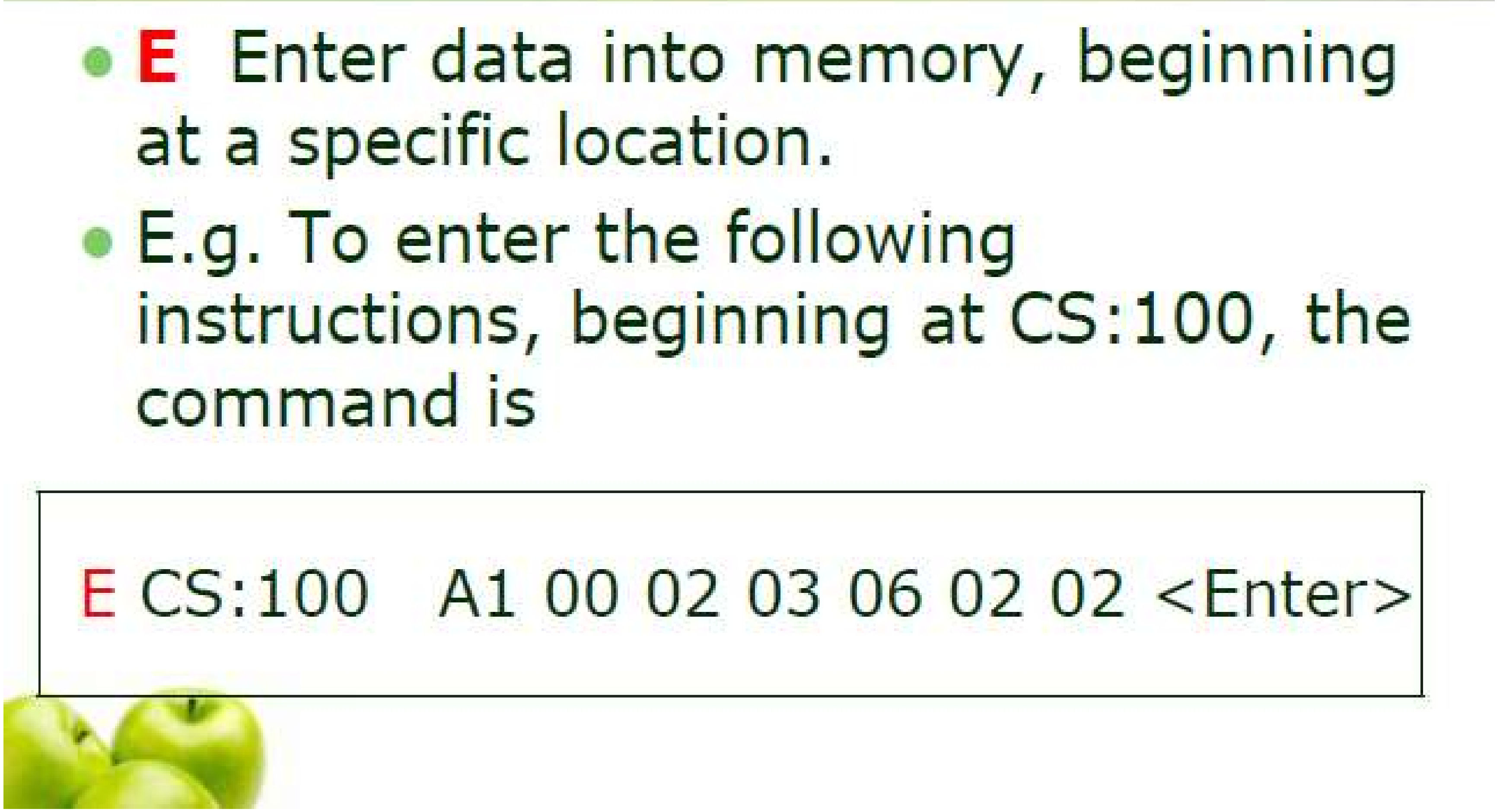
```
-D   CS:100
```

```
21C1:0100  B8 23 01 05 25 00 8B D8-03 D8 8B CB 2B CB 2B C0 .#..%.......+.+
```

Note: To the right is the ASCII representation of each byte.

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**

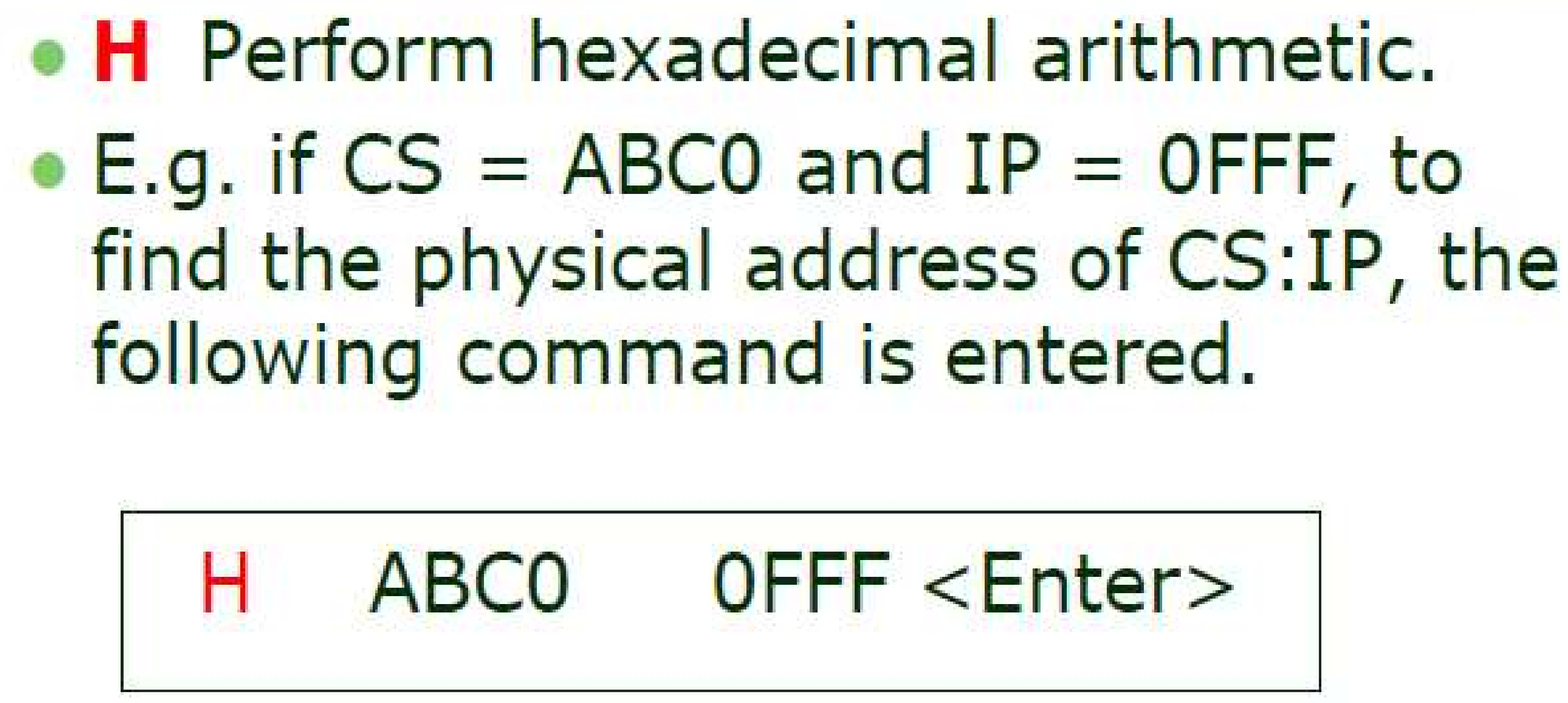


- E Enter data into memory, beginning at a specific location.
- E.g. To enter the following instructions, beginning at CS:100, the command is

```
E CS:100   A1 00 02 03 06 02 02 <Enter>
```

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**

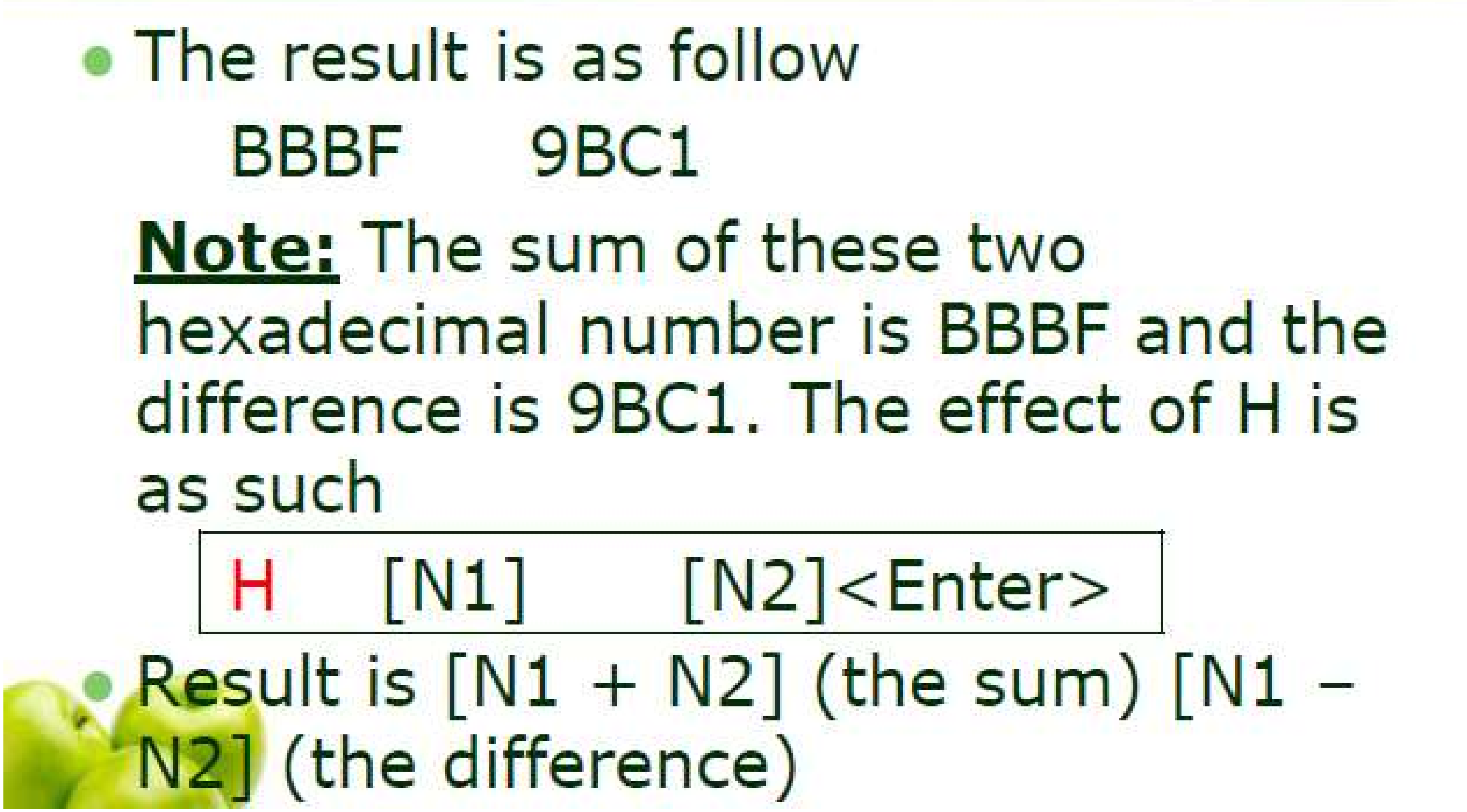


- **H** Perform hexadecimal arithmetic.
- E.g. if CS = ABC0 and IP = 0FFF, to find the physical address of CS:IP, the following command is entered.

```
H    ABC0    0FFF <Enter>
```

4. Introduction to Debug program (Continued)

- **Sets of commands in DEBUG program**



- The result is as follow

  BBBF 9BC1

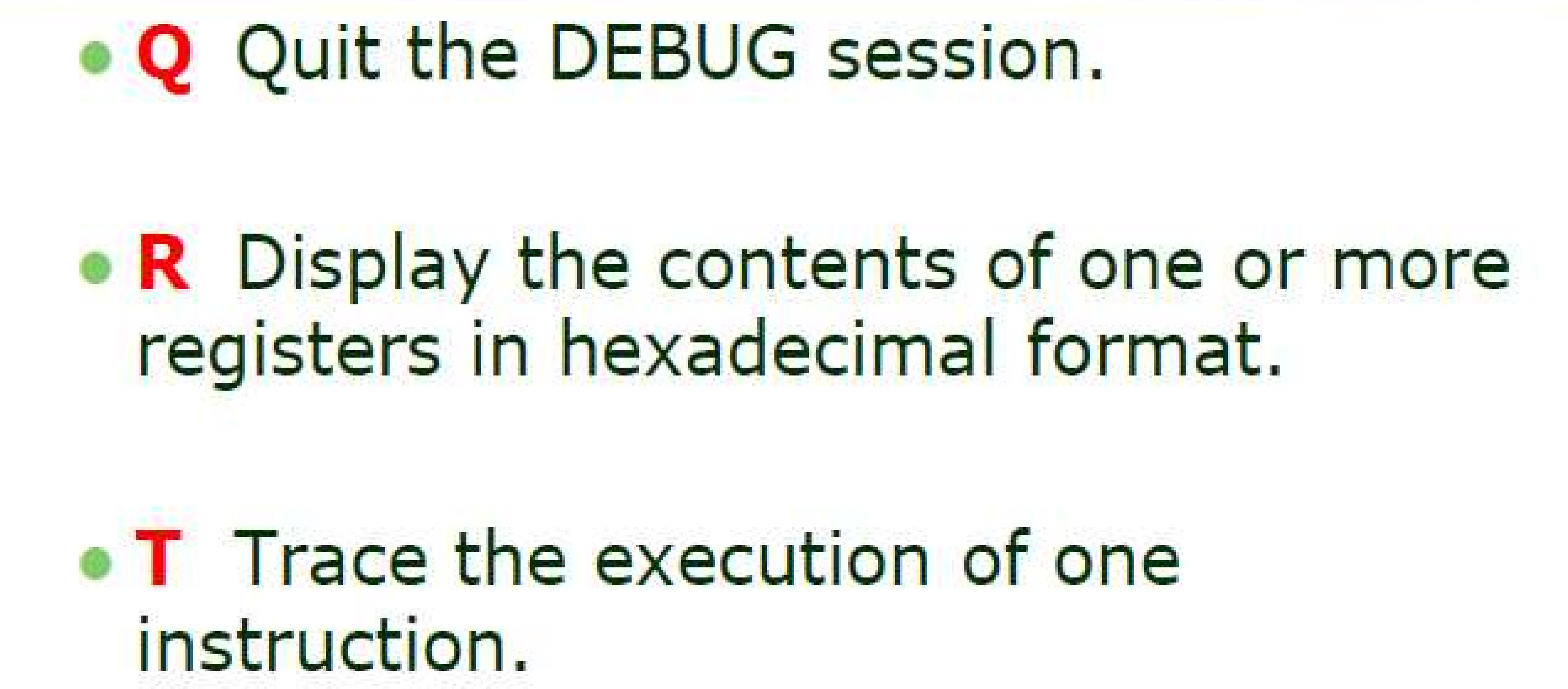
  Note: The sum of these two hexadecimal number is BBBF and the difference is 9BC1. The effect of H is as such

```
H    [N1]     [N2]<Enter>
```

- Result is [N1 + N2] (the sum) [N1 – N2] (the difference)

4. Introduction to Debug program (Continued)

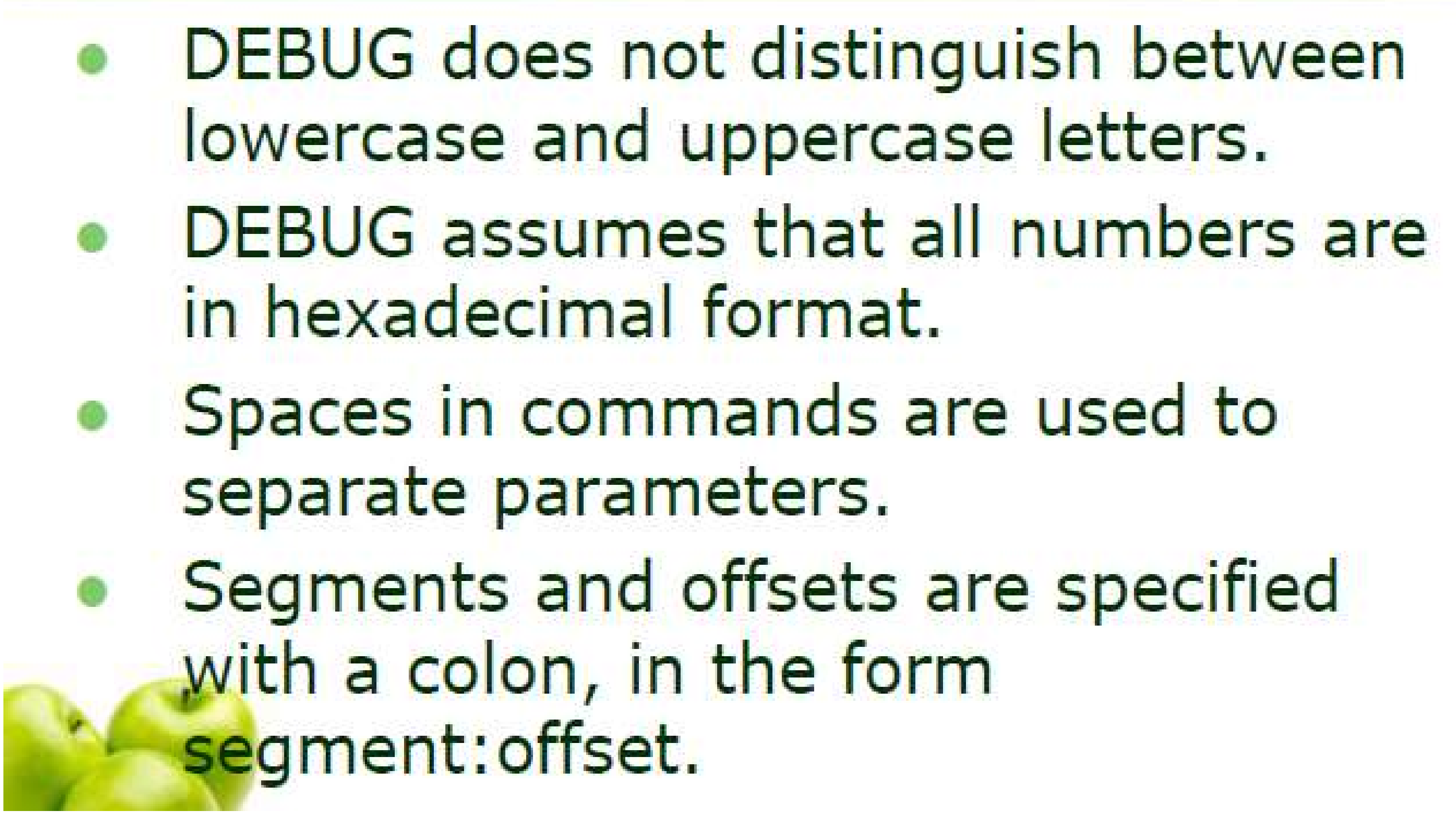
- **Sets of commands in DEBUG program**



- **Q** Quit the DEBUG session.
- **R** Display the contents of one or more registers in hexadecimal format.
- **T** Trace the execution of one instruction.

4. Introduction to Debug program (Continued)

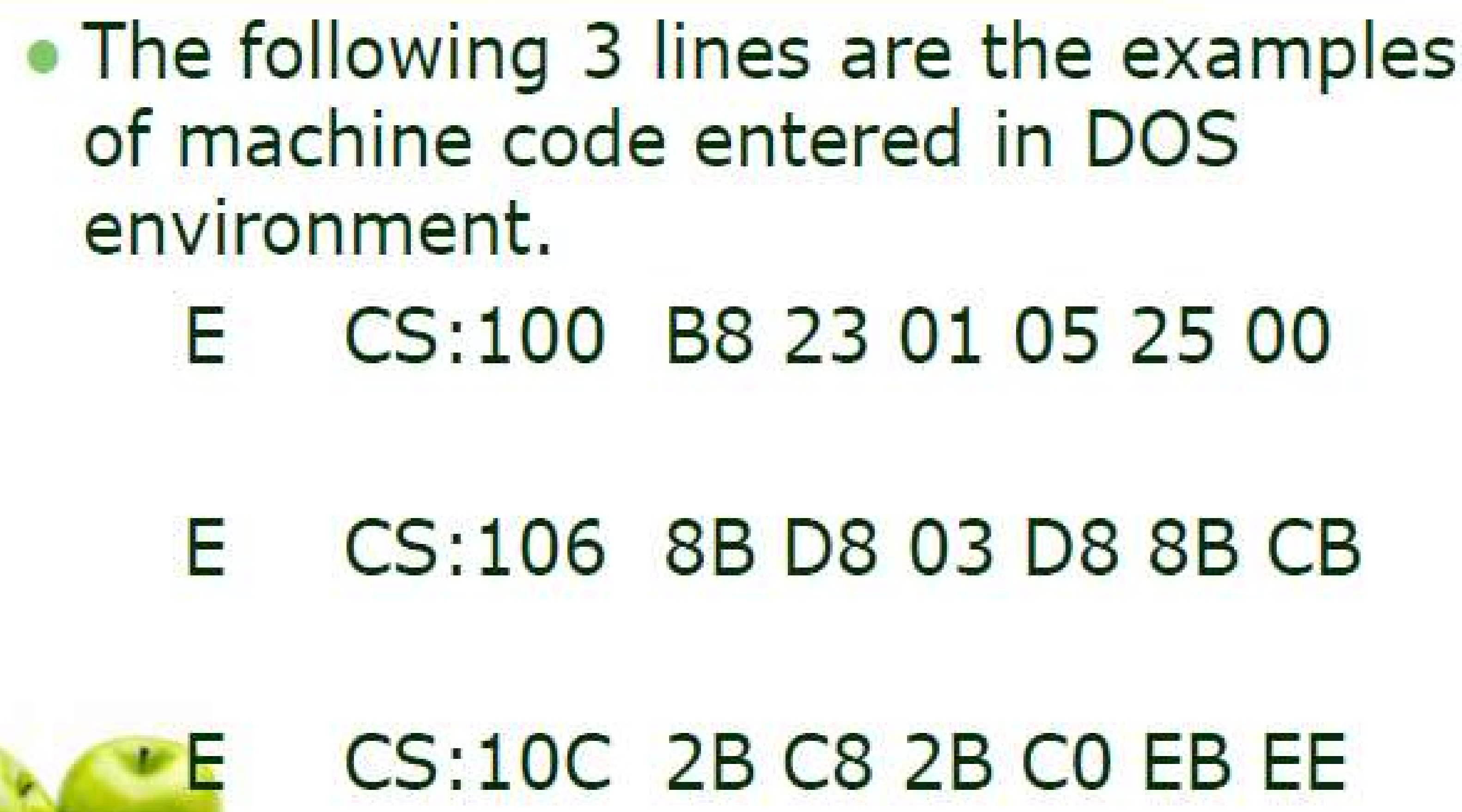
- **Rules of DEBUG commands**



- DEBUG does not distinguish between lowercase and uppercase letters.
- DEBUG assumes that all numbers are in hexadecimal format.
- Spaces in commands are used to separate parameters.
- Segments and offsets are specified with a colon, in the form segment:offset.

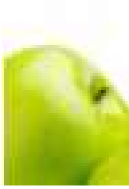
4. Introduction to Debug program (Continued)

- **Entering Program Instructions**



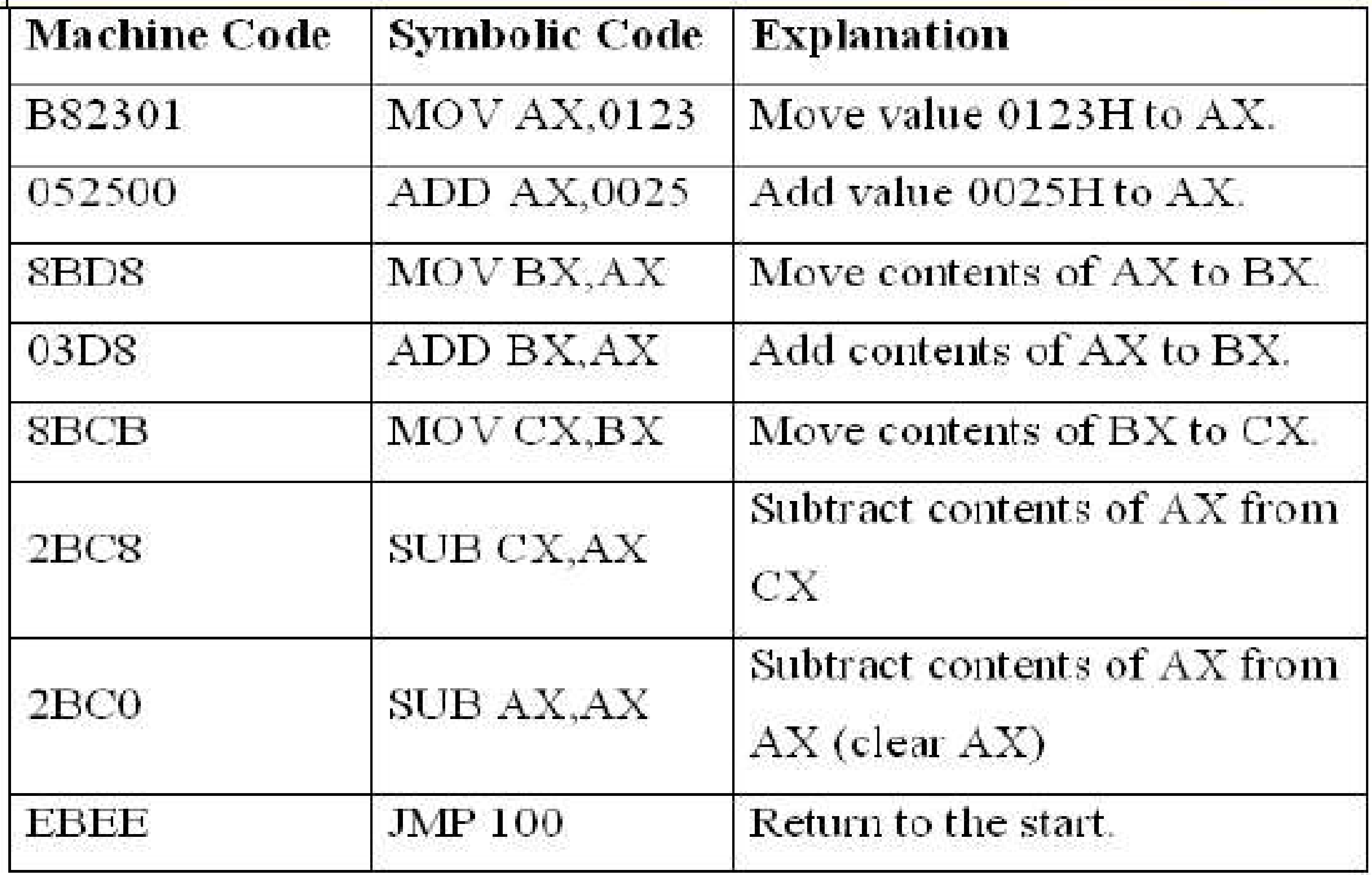
- The following 3 lines are the examples of machine code entered in DOS environment.

```
E    CS:100  B8 23 01 05 25 00

E    CS:106  8B D8 03 D8 8B CB

E    CS:10C  2B C8 2B C0 EB EE
```



4. Introduction to Debug program (Continued)

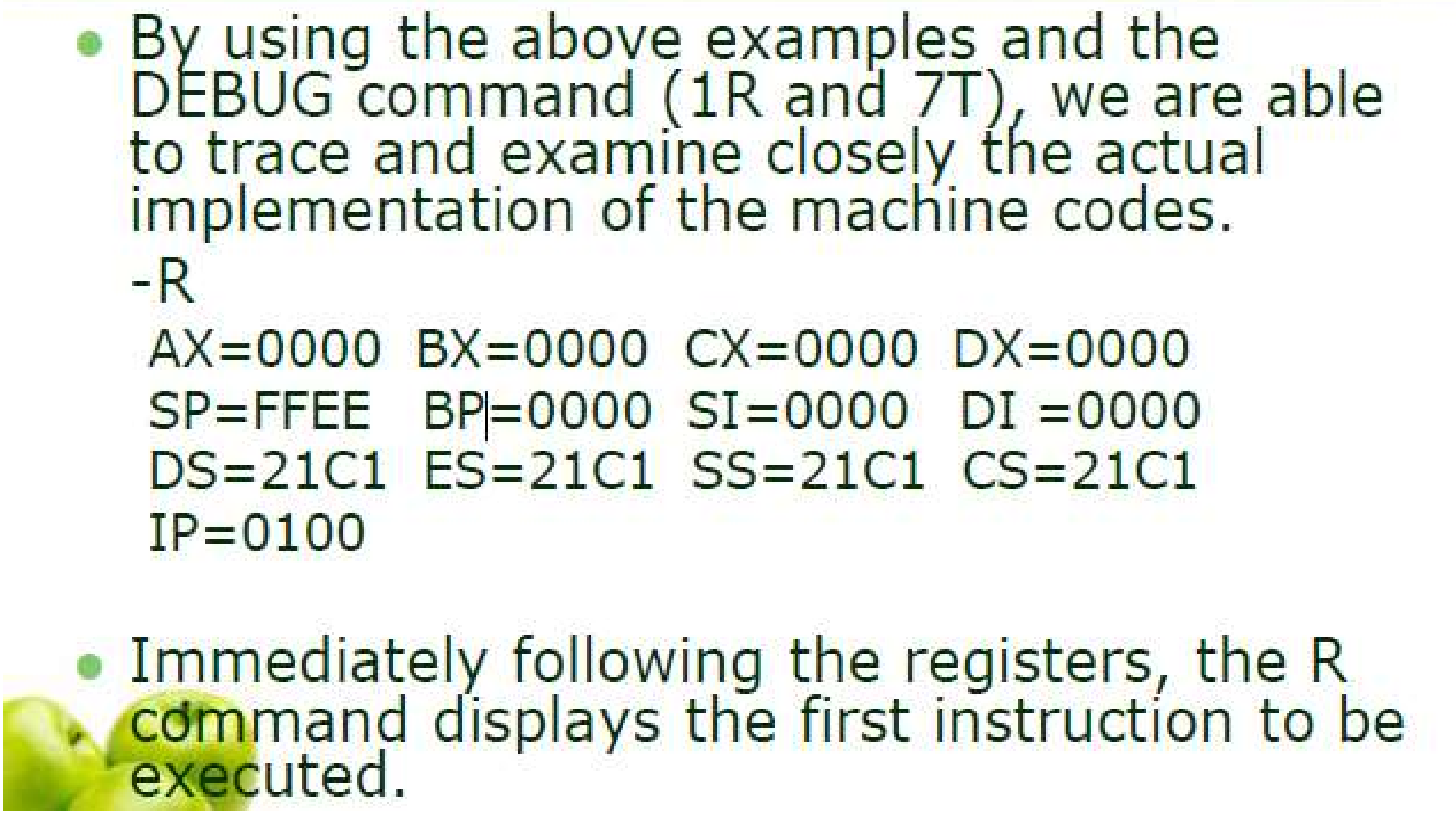
- **Entering Program Instructions**



Machine Code	Symbolic Code	Explanation
B82301	MOV AX,0123	Move value 0123H to AX.
052500	ADD AX,0025	Add value 0025H to AX.
8BD8	MOV BX,AX	Move contents of AX to BX.
03D8	ADD BX,AX	Add contents of AX to BX.
8BCB	MOV CX,BX	Move contents of BX to CX.
2BC8	SUB CX,AX	Subtract contents of AX from CX
2BC0	SUB AX,AX	Subtract contents of AX from AX (clear AX)
EBEE	JMP 100	Return to the start.

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



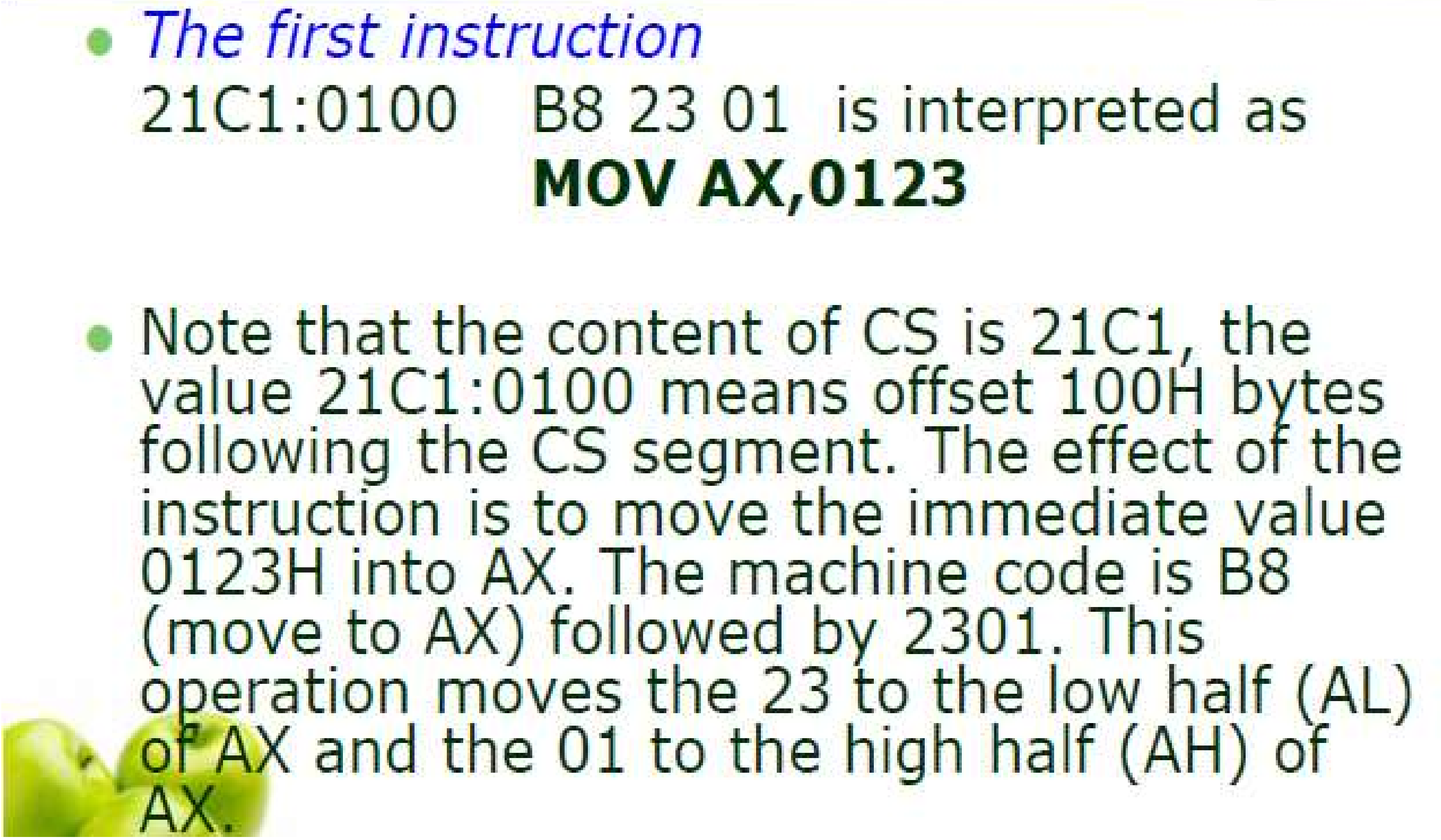
- By using the above examples and the DEBUG command (1R and 7T), we are able to trace and examine closely the actual implementation of the machine codes.

```
-R
 AX=0000  BX=0000  CX=0000  DX=0000
 SP=FFEE  BP=0000  SI=0000  DI =0000
 DS=21C1  ES=21C1  SS=21C1  CS=21C1
 IP=0100
```

- Immediately following the registers, the R command displays the first instruction to be executed.

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



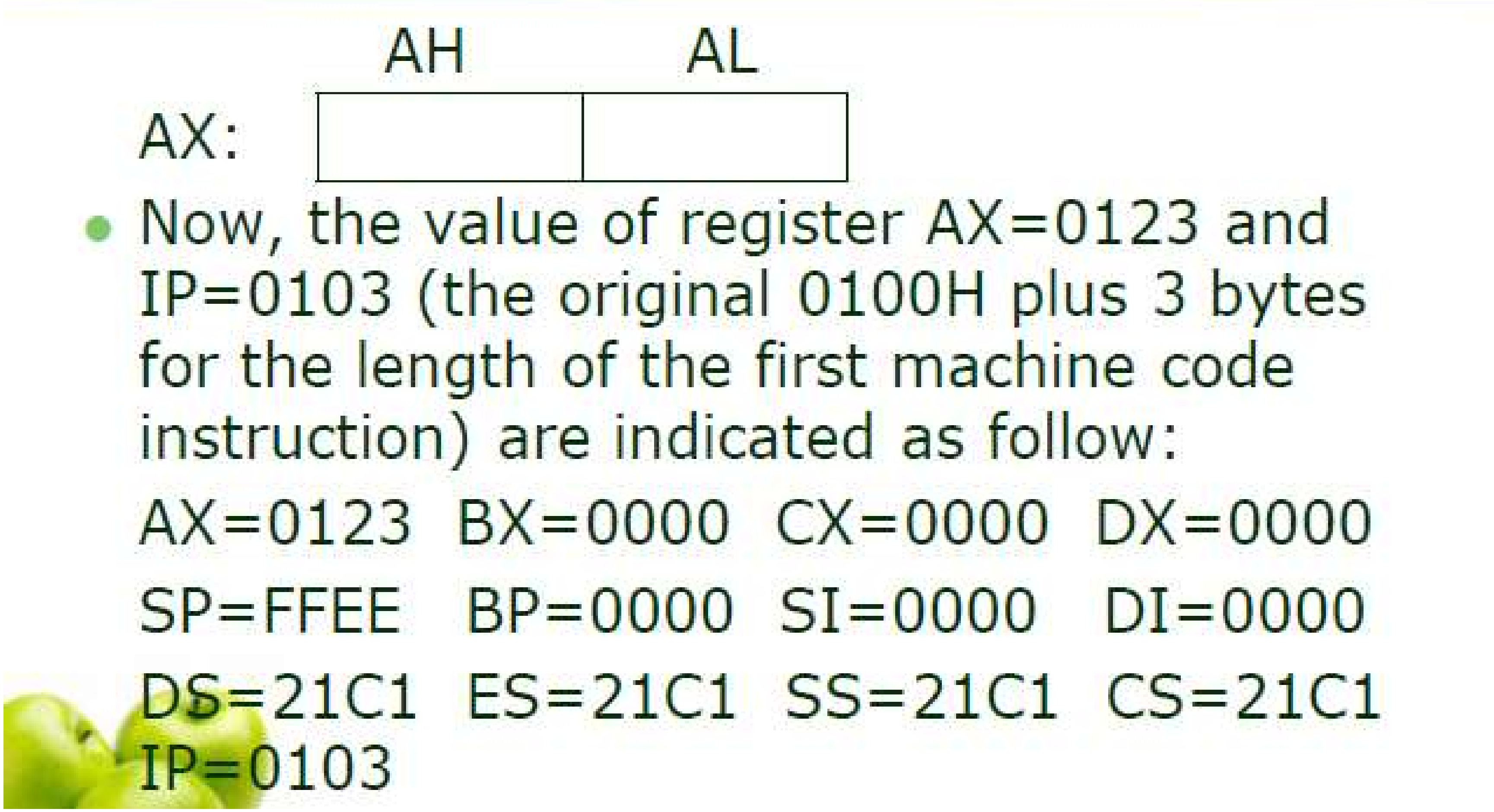
- *The first instruction*

  21C1:0100 B8 23 01 is interpreted as
  MOV AX,0123

- Note that the content of CS is 21C1, the value 21C1:0100 means offset 100H bytes following the CS segment. The effect of the instruction is to move the immediate value 0123H into AX. The machine code is B8 (move to AX) followed by 2301. This operation moves the 23 to the low half (AL) of AX and the 01 to the high half (AH) of AX.

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**

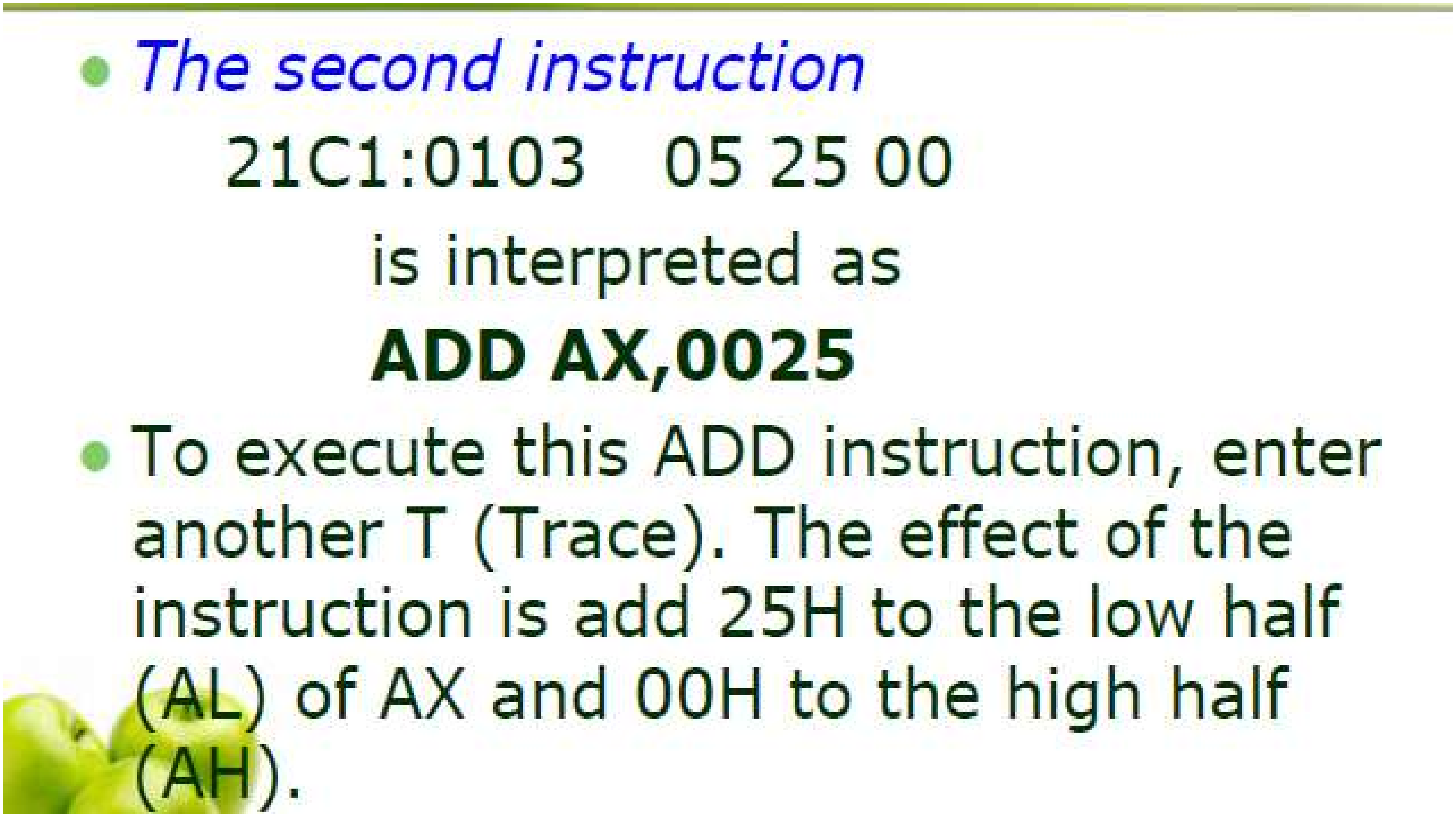


- Now, the value of register AX=0123 and IP=0103 (the original 0100H plus 3 bytes for the length of the first machine code instruction) are indicated as follow:

```
AX=0123  BX=0000  CX=0000  DX=0000
SP=FFEE  BP=0000  SI=0000  DI=0000
DS=21C1  ES=21C1  SS=21C1  CS=21C1
IP=0103
```

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



- *The second instruction*

```
21C1:0103   05 25 00
```

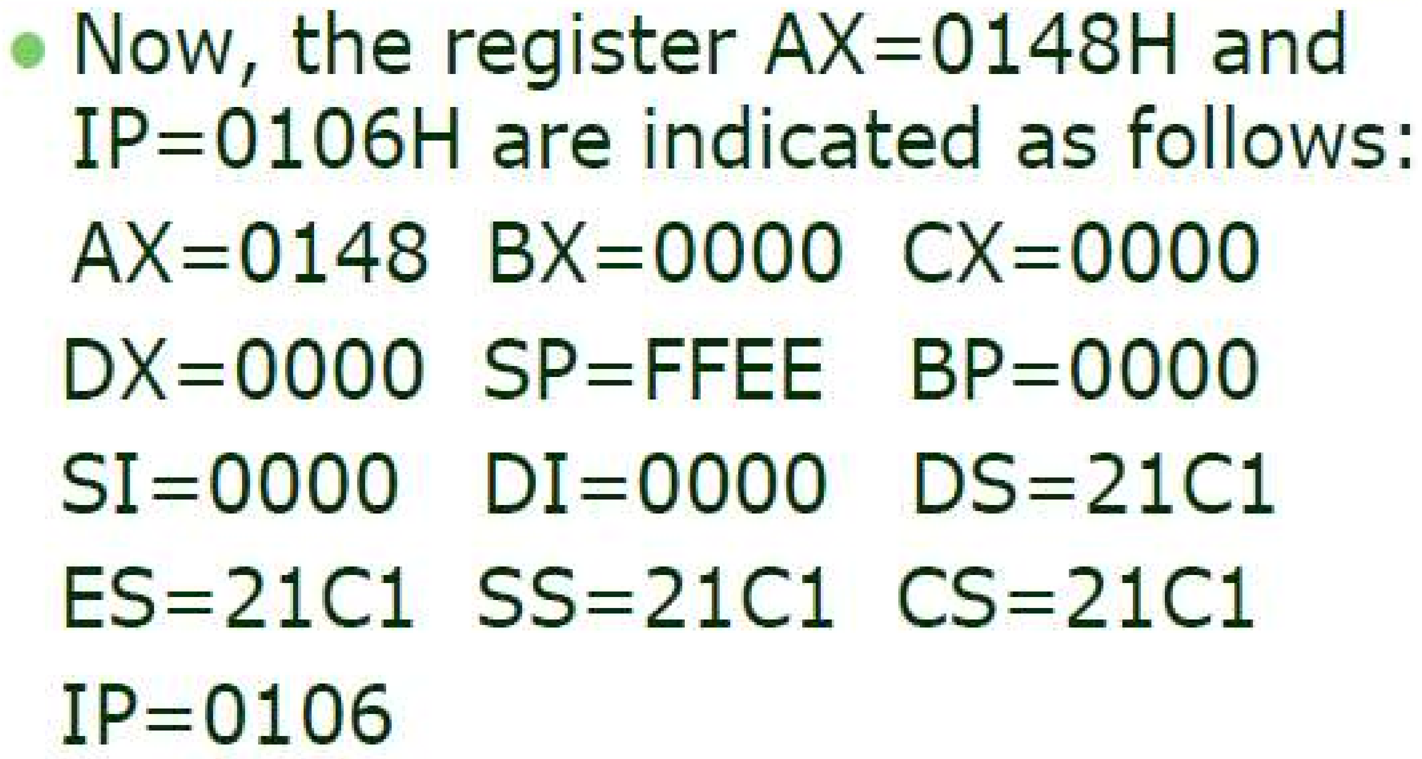
is interpreted as

ADD AX,0025

- To execute this ADD instruction, enter another T (Trace). The effect of the instruction is add 25H to the low half (AL) of AX and 00H to the high half (AH).

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**

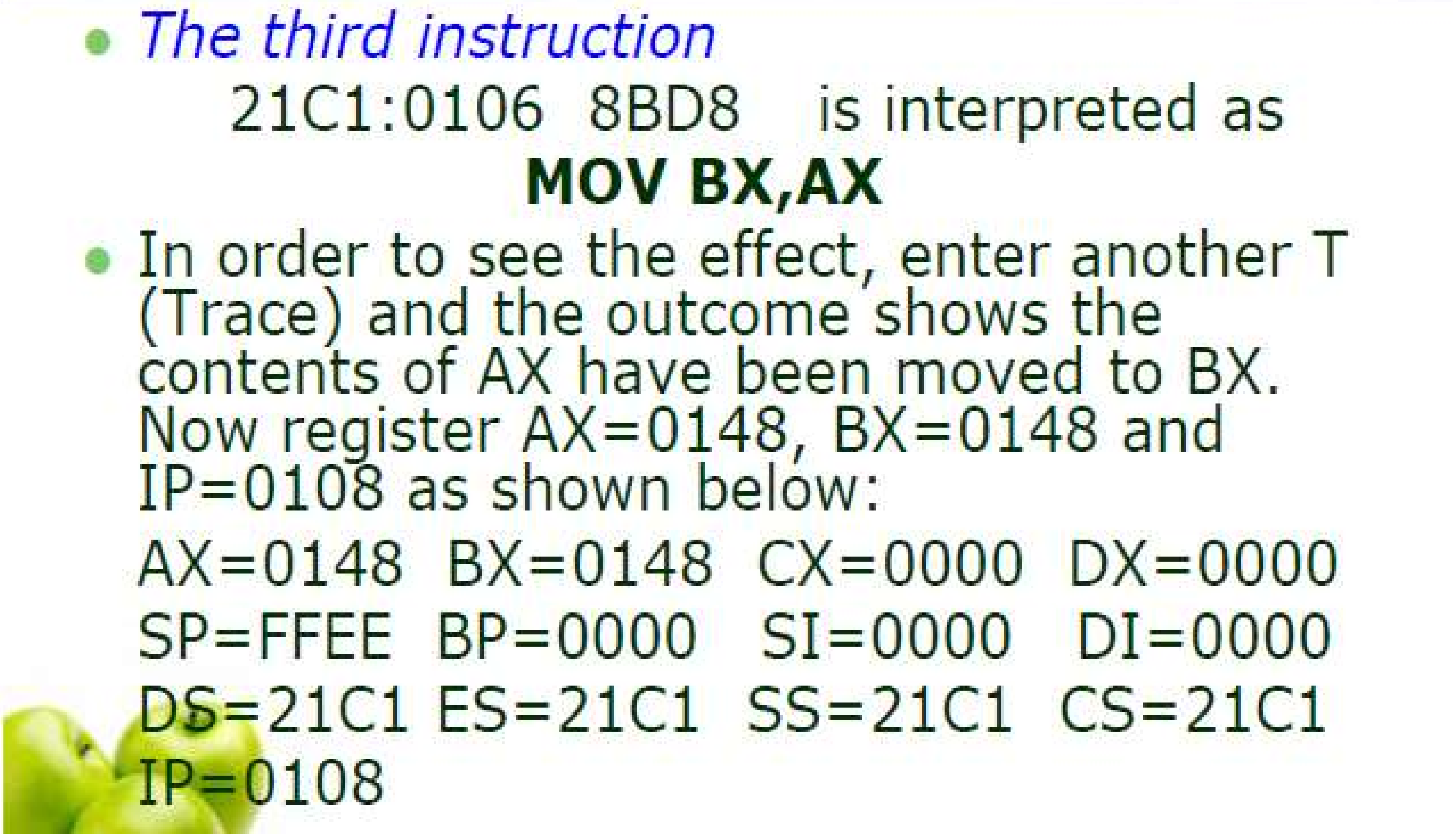


- Now, the register AX=0148H and IP=0106H are indicated as follows:

```
AX=0148  BX=0000  CX=0000
DX=0000  SP=FFEE  BP=0000
SI=0000  DI=0000  DS=21C1
ES=21C1  SS=21C1  CS=21C1
IP=0106
```

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



- *The third instruction*

  21C1:0106 8BD8   is interpreted as

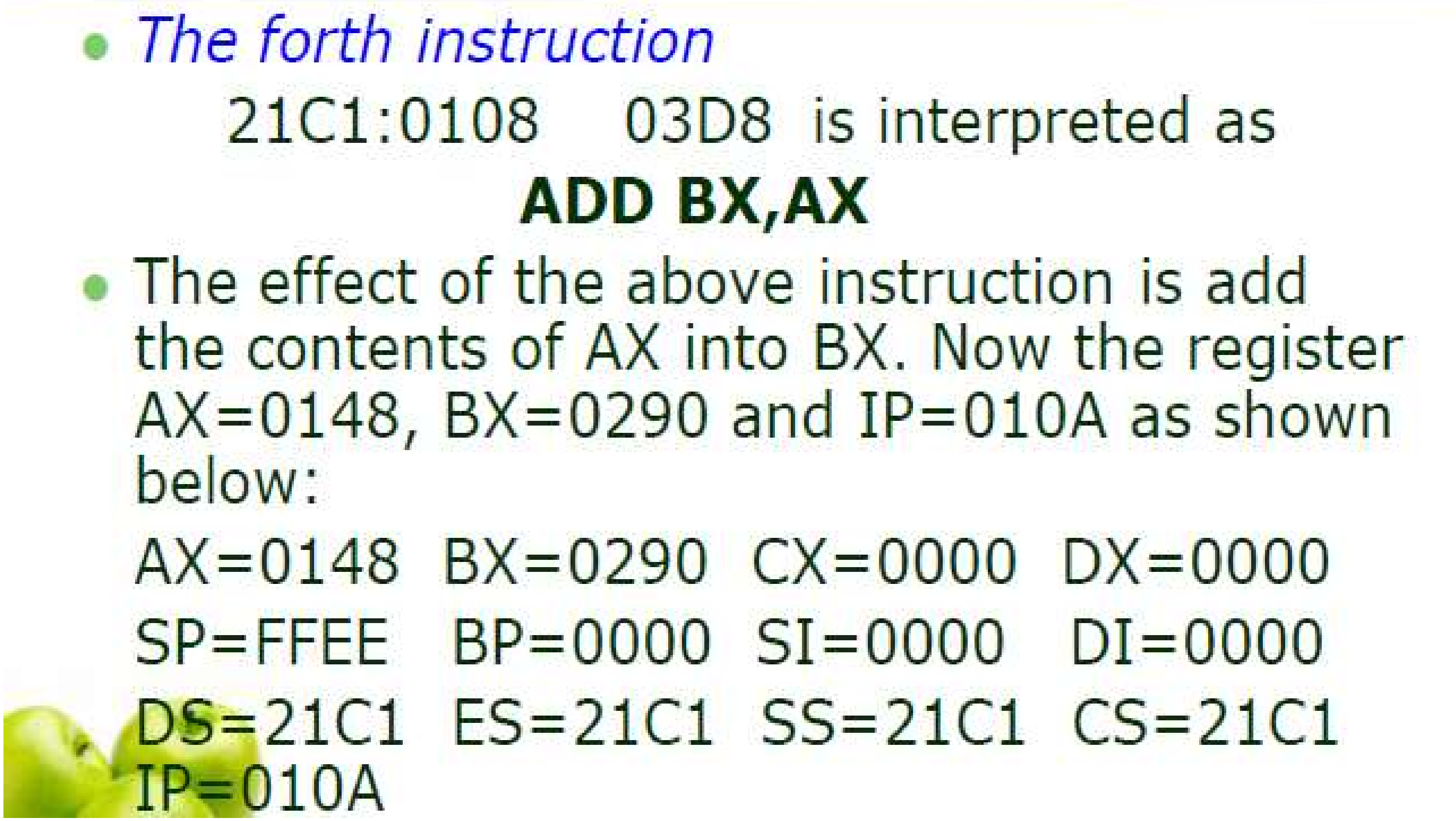
  MOV BX,AX

- In order to see the effect, enter another T (Trace) and the outcome shows the contents of AX have been moved to BX. Now register AX=0148, BX=0148 and IP=0108 as shown below:

```
AX=0148  BX=0148  CX=0000  DX=0000
SP=FFEE  BP=0000   SI=0000   DI=0000
DS=21C1 ES=21C1  SS=21C1  CS=21C1
IP=0108
```

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



- *The forth instruction*

  21C1:0108   03D8 is interpreted as

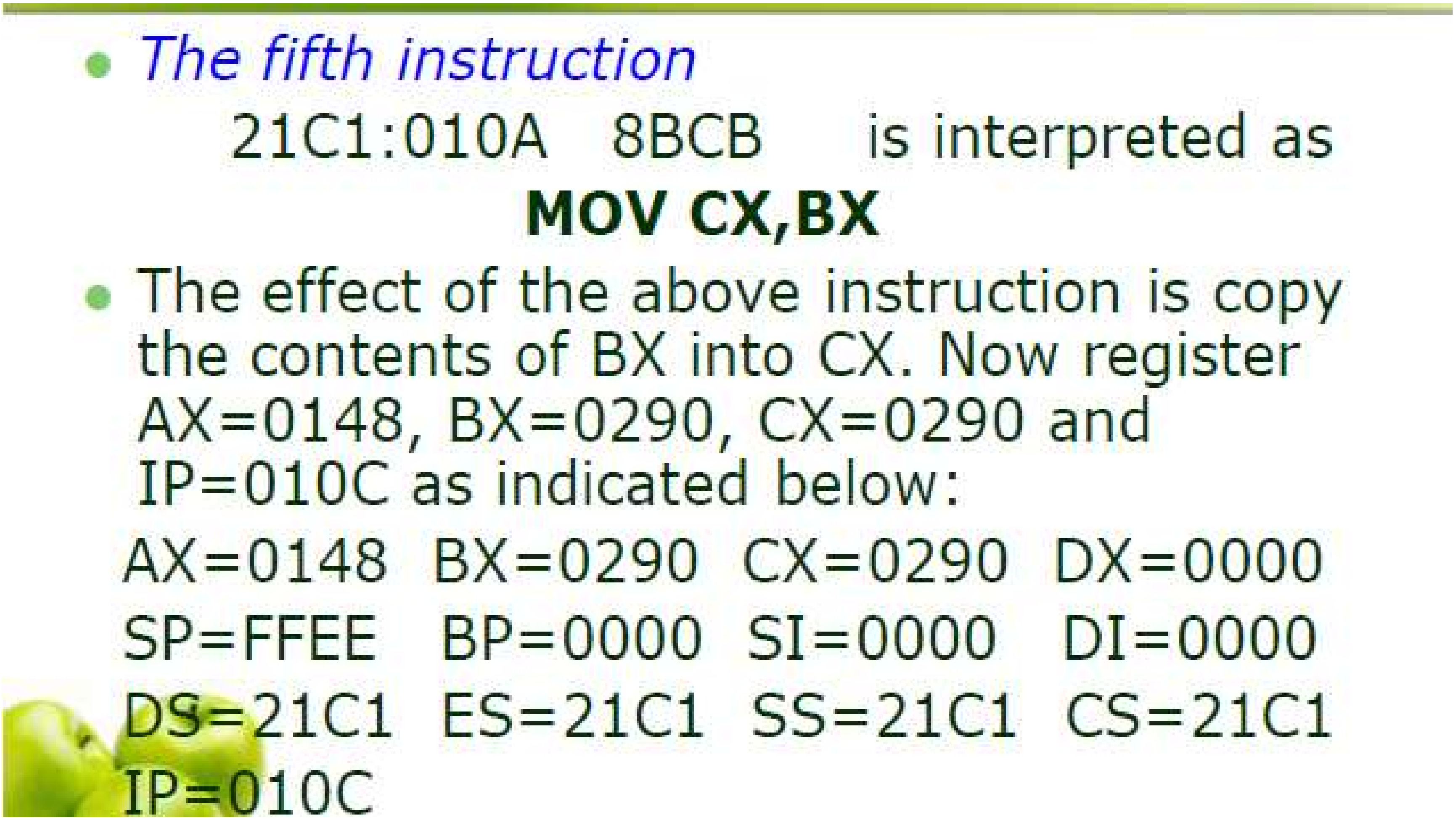
  ADD BX,AX

- The effect of the above instruction is add the contents of AX into BX. Now the register AX=0148, BX=0290 and IP=010A as shown below:

```
AX=0148  BX=0290  CX=0000  DX=0000
SP=FFEE   BP=0000  SI=0000   DI=0000
DS=21C1  ES=21C1  SS=21C1  CS=21C1
IP=010A
```

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



- *The fifth instruction*

  21C1:010A   8BCB   is interpreted as

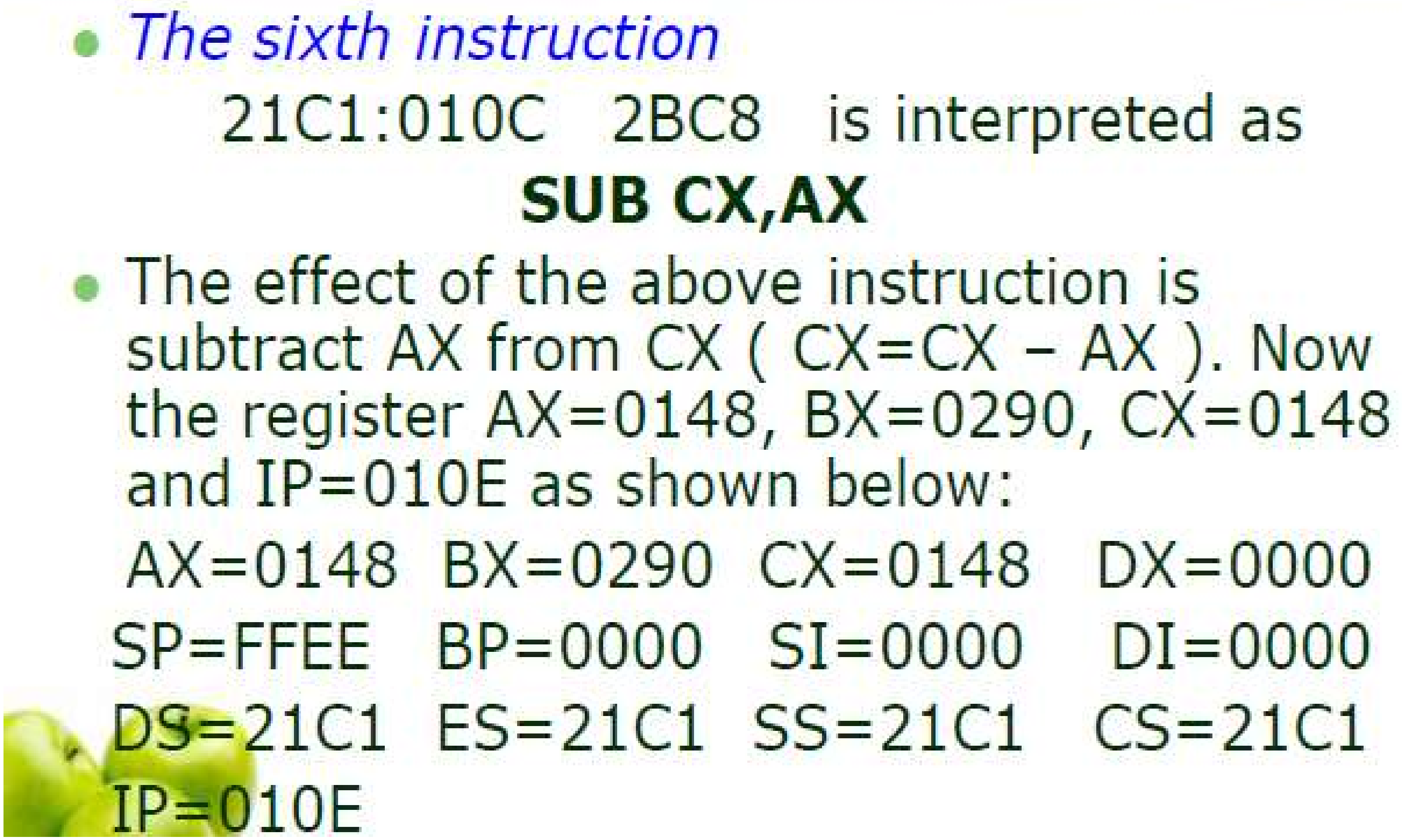
  MOV CX,BX

- The effect of the above instruction is copy the contents of BX into CX. Now register AX=0148, BX=0290, CX=0290 and IP=010C as indicated below:

```
AX=0148  BX=0290  CX=0290  DX=0000
SP=FFEE  BP=0000  SI=0000  DI=0000
DS=21C1  ES=21C1  SS=21C1  CS=21C1
IP=010C
```

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



- *The sixth instruction*

  21C1:010C 2BC8 is interpreted as

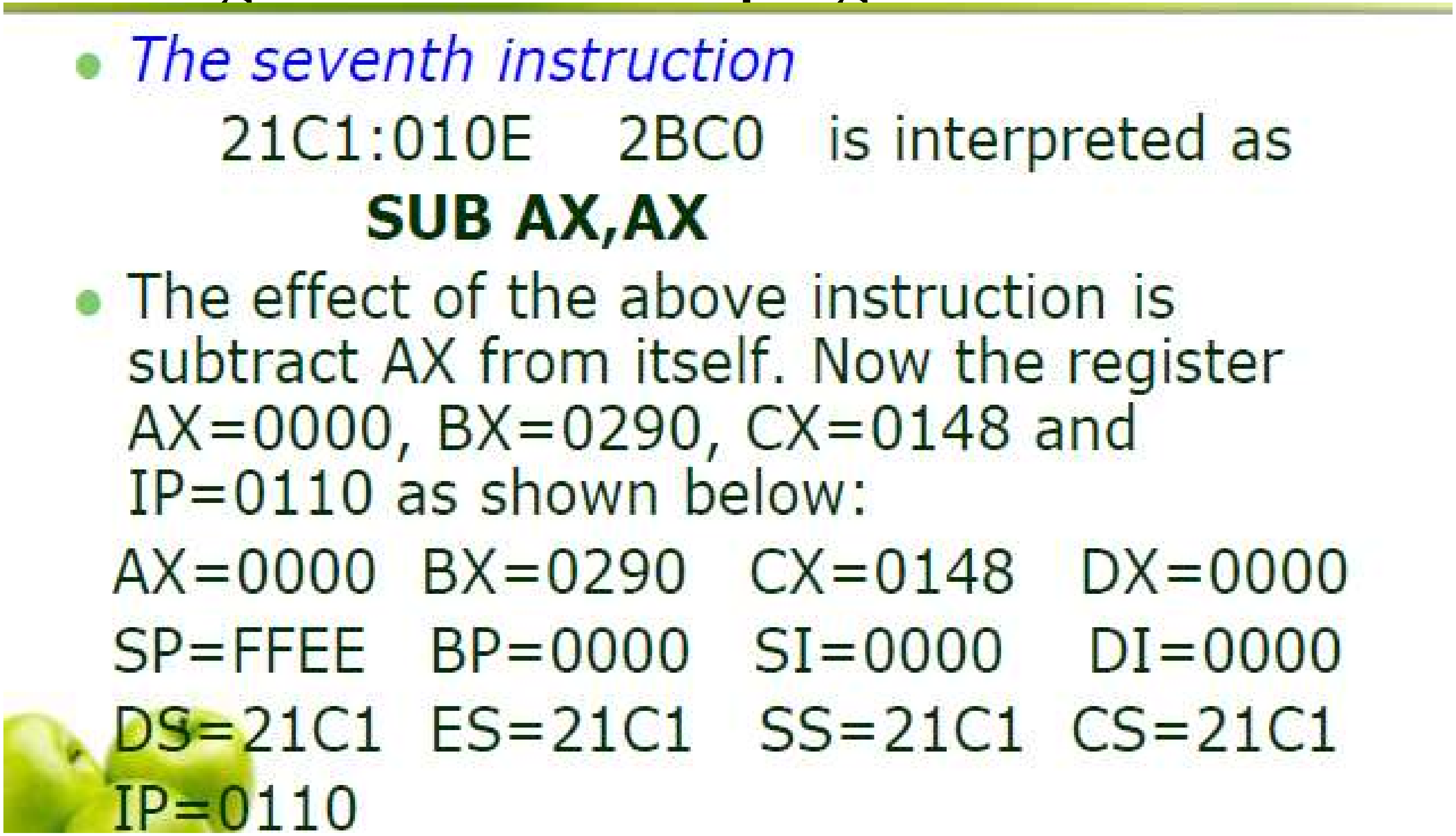
  SUB CX,AX

- The effect of the above instruction is subtract AX from CX (CX=CX – AX). Now the register AX=0148, BX=0290, CX=0148 and IP=010E as shown below:

```
 AX=0148  BX=0290  CX=0148   DX=0000
SP=FFEE   BP=0000   SI=0000   DI=0000
DS=21C1  ES=21C1  SS=21C1   CS=21C1
IP=010E
```

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



- *The seventh instruction*

  21C1:010E 2BC0 is interpreted as

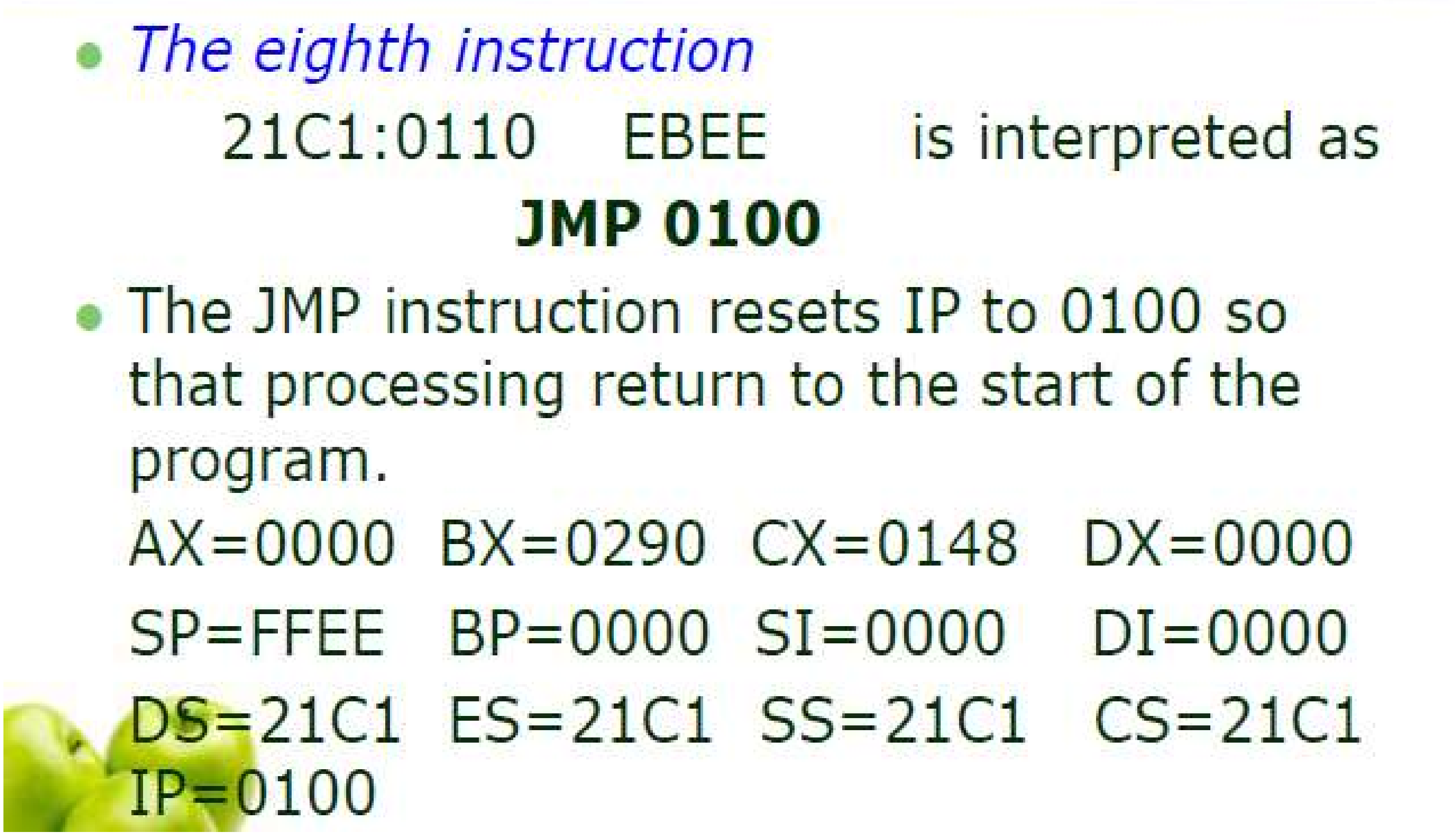
  SUB AX,AX

- The effect of the above instruction is subtract AX from itself. Now the register AX=0000, BX=0290, CX=0148 and IP=0110 as shown below:

```
AX=0000  BX=0290   CX=0148   DX=0000
SP=FFEE   BP=0000   SI=0000    DI=0000
DS=21C1  ES=21C1   SS=21C1  CS=21C1
IP=0110
```

4. Introduction to Debug program (Continued)

- **Tracing the execution of program**



- *The eighth instruction*

```
21C1:0110    EBEE      is interpreted as
```

JMP 0100

- The JMP instruction resets IP to 0100 so that processing return to the start of the program.

```
AX=0000  BX=0290  CX=0148   DX=0000
SP=FFEE   BP=0000  SI=0000   DI=0000
DS=21C1  ES=21C1  SS=21C1   CS=21C1
IP=0100
```

Chapter Review

1) Instruction format
2) Machine language instructions
3) Instruction execution
4) Introduction to Debug program