

*Assembly Language  
Programming and  
Organization of the  
IBM PC*

*By Yu and Marut*

**Manual solution**

**Ch. 1 to ch. 10**

Written by:

*Eng. Amer Mohammed Al-khsabah  
VIPER-VIRUS@LIVE.COM*

## Contents

Preface .....	1
Ch.1 .....	2
Ch.2 .....	7
Ch.3 .....	14
Ch.4 .....	17
Ch.5 .....	32
Ch.6 .....	37
Ch.7 .....	58
Ch.8 .....	84
Ch.9 .....	93
Ch.10 .....	110
Appendix A (How to run a program).....	112
Appendix B ( Some useful procedure).....	117

## Chapter One

### Microcomputer Systems

(1)

a)

- 111111100010001b
- 010101011111111b
- 0001000111011101b

b)

- 0
- 1
- 1
- 1

---

(2)

- a) 1101  
b) 0101
- 

(3)

- a) RAM  
b) ROM  
c) RAM  
d) ROM
-

## Chapter One

(4)

a) *the microprocessor* :

brain of the computer , it controls the computer by executing programs stored in memory.

b) *the buses*:

connect the different component to make processor communicate with memory and I/O circuits.

---

(5)

a) *EU (Execution Unit )* :

execute instructions – because it contains the ALU.

b) *BIU (Bus Interface Unit )* :

facilitates communication between the EU and the memory or I/O circuits , and useful for speed-up the processor by instruction prefetch .

---

(6)

a) *IP* : contains the address of the next instruction to be executed by EU .

b) *ALU* : perform arithmetic & logic operation .

---

## **Chapter One**

(7)

a) *I/O ports* : transfer points between the CPU & I/O devices

b) because they have addresses connected to the bus system known as I/O addresses and they can only be used in input or output instruction , this allows the CPU to distinguish between them .

---

(8)

4-Bytes

---

(9)

a)

- Fetch an instruction from memory.
- Decode the instruction to determine the operation.

b)

- Perform the operation if needed .
  - Store the results in memory.
- 

(10)

a) *Advantages of H.L.L:*

1. closer to natural language , so it's easier to read and understand the program .
2. assembly contains more statement than an equivalent HLL ,so It's need more time to code the assembly language program.

## Chapter One

3. each computer has its own unique assembly language , so It's limited to one machine ,but HLL can be executed on any machine has the compiler of that language .

b) *primary advantage of assembly language:*

It's so close to machine language ,so it's produce a faster shorter machine language program .

And some operation (read /write) can be done on specific Memory location and I/O ports, and that may be impossible In HLL

---

## **Chapter Two**

(1)

<u>Binary</u>	<u>Decimal</u>	<u>Hex</u>
1001	9	9
1010	10	A
1101	13	D
1100	12	C
1110	14	E
1011	11	B

---

(2)

- a) 14d                    d: decimal
  - b) 2397d
  - c) 1130d
  - d) 1027628d
- 

(3)

- a) 1100001b            b: binary
  - b) 1001110011b
  - c) 399h                    h: hexadecimal
  - d) 17E8h
- 

(4)

- a) 4Bh
- b) 95AEh
- c) 101000101100b
- d) 1011001101001101b

## Chapter Two

---

(5)

- a) 111100b
  - b) 10110110110b
  - c) C9CDFh
  - d) 1FACABh
- 

(6)

- a) 101b
  - b) 1001010b
  - c) 25041h
  - d) D00DFh
- 

(7)

- a)  $234d = 0000000011101010b = \text{00Eah}$
- b)  $16d = 0000000010000b$   
 $-16d = 2\text{'s comp. (16)}$   
 $= 111111111110000b = \text{FFF0h}$
- c)  $31634d = 0111101110010010b = \text{7B92h}$
- d)  $32216d = 011110111011000b$   
 $-32216d = 2\text{'s comp.(32216)}$   
 $= 1000001000101000b = \text{8228h}$

Note: 2's comp. = 1's comp + 1

1's comp. = NOT

## Chapter Two

---

(8)

Take 2's comp. for negative number ,then Add the two numbers.,if no.1> no.2 : positive result stay as it.

If no.1< no.2 :negative take 2's comp.

a) 2's comp.(-10010111b) = 111111101101001b

$$10110100b + 111111101101001b = 0000000000011101b$$

Positive no. = **11101b**

b) 2's comp.(-11110111b) = 111111100001001b

$$10001011b + 111111100001001b = 111111110010100b$$

Negative no. = 2's comp.( 111111110010100)

$$= \textcolor{red}{-1101100b}$$

c) 2's comp.(-12ABh) = ED55h

$$\text{FE0Fh} + \text{ED55h} = \text{EB64h}$$

Positive no. = **EB64h**

d) 2's comp.(-B3EAh) = 4C16h

$$\text{1ABCh} + \text{4C16h} = \text{66D2h}$$

Negative no. = 2's comp.(66D2h)

$$= \textcolor{red}{-992Eh}$$

*Note : we can judge on no. from subtraction equation*

*(e.g: 1ABCh - B3EAh = ( negative number) : because*

*1ABC < B3EAh ,so we must take 2's comp.*

## Chapter Two

(9)

Unsigned :convert the number as it is.

Signed :if msb=0 convert the number as it is.

If msb=1 take 2's comp. then convert.

	<u>Unsigned</u>	<u>signed</u>
a) 7FFEh	32766d	32766d
b) 8543h	34115d	-31421d
c) FEh	254d	-2d
d) 7Fh	127d	127d

Note :msb (most significant bit) last bit on the left.  
i.e :bit-15 in word OR bit-7 in byte.

---

(10)

120d = 01111000b      8-bit

0000000001111000b      16-bit

To make it negative take 2's comp.

-120d = 10001000b      8-bit

111111110001000b      16-bit

a) 10001000b

b) 111111110001000b

---

## Chapter Two

(11)

	<b>16-bit</b>	<b>8-bit</b>
a) 32767d	111111111111111b	X :too big
b) -40000d	X :too big	X :too big
c) 65536d	X :too big	X :too big
d) 257d	0000000100000001b	X :too big
e) -128	0000000100000000b	10000000b

Note : Range of unsigned integer :

0 – 255                      for byte 8-bit  
0 – 65535                    for word 16-bit

Range of signed numbers :

-128 – 127                for byte 8-bit  
-32768 – 32767            for word 16-bit  
(msb represente the sign)

(12)

If msb =0 positive else negative

- a) 1010010010010100b    Negative
- b) 78E3h = 0111100011010011b    Positive
- c) CB33h = 1100101100110011b    Negative
- d) 807Fh = 1000000001111111b    Negative
- e) 9AC4h = 1001101011000100b    Negative

## Chapter Two

(13)

<u>Address</u>	<u>Content(hex)</u>	<u>Data</u>
0	24	\$
1	31	1
2	32	2
3	2E	.
4	37	7
5	35	5

---

(14)

41	74	74	61	63	6B	20	61	74	20	44	61
A	t	t	a	c	k		a	t		D	a
77	6E										
w	n										

**“Attack at Dawn”**

---

(15)

$$A = 41h \quad a = 61h$$

The difference is 20h

Add 20h to convert from upper to lower case.

---

## **Chapter Two**

(16)

“0” = 30h              numerical value = 0h

“1” = 31h              numerical value = 1h

The difference is 30h

**SUB 30h** to get the numerical value.

---

(17)

Just a technique:

0 1 2 3 4 5 6 7 8 9 A B C D E F .....->

5(c) B23CDh

+ 17912h

D+2= go from D to the right 2 places = F

C+1= go from C to the right 1 place = D

3+9= go from 3 to the right 9 places = C

2+7= go from 2 to the right 7 places = 9

B+1= go from B to the right 1 place = C

= **C9CDFh**

---

## Chapter Three

(1)

8086 & 80286 have 16-bit but 80286 has this advances:

- Two mode of operation.
  - More addressable memory.
  - Virtual memory in protected mode.
- 

(2)

<u>Registers</u>	<u>M.L</u>
Inside CPU	in RAM
Perform operation	store data
Known as name	known by Address
14-reg 16-bit	1MB 8-bit

Note : these information for 8086.

---

(3)

**AX** : multiplication & division operation.

**BX** : serves as an address register.

**CX** : serves as a loop counter.

**DX** : used in I/O operation.

---

## Chapter Three

(4)

**0A51:CD90h = seg:offset**

$$\text{Physical address} = \text{seg} \times 10h + \text{offset}$$

$$= 0A51 \times 10h + CD90h$$

$$= 0A510 + CD90h$$

$$= 172A0h$$

(5)

**Physical Address = 4A37Bh**

a) **seg. = 40FFh**

$$\text{Physical address} = \text{seg} \times 10h + \text{offset}$$

$$\text{Offset} = \text{physical address} - \text{seg} \times 10h$$

$$= 4A37B - 40FF \times 10h$$

$$= 4A37B - 40FF0$$

$$= 938Bh$$

**40FF:938Bh**

b) **offset = 123Bh**

$$\text{Physical address} = \text{seg} \times 10 + \text{offset}$$

$$\text{Seg} \times 10h = \text{physical address} - \text{offset}$$

$$\text{Seg} \times 10h = 4A37B - 123B$$

$$\text{Seg} \times 10h = 49140$$

$$\text{Seg} = 49140/10h$$

$$= 4914h$$

**4914: 123Bh**

## Chapter Three

(6)

Paragraph = 16 byte

So any address accept division on 16d /10h call paragraph boundary.

(i.e : any address ending in 0)

Ex:

15230 : P.B                    P.B :Paragraph Boundary

66DA0: P.B

3ACA1: not P.B

.....  
(7)

The compatibility of PC clones with the IBM PC depends on how well their BIOS routines match thoses of the IBM PC.

.....  
(8)

That depends on the size of RAM used.

For 8086 1MB is used 10-segment to loading and running applications , these 10-segment give us 640 KB of memory.

If we use 512-KB it has only 8-segment

## Chapter Four

(1)

- a) legal
  - b) legal
  - c) illegal ; contains a blank (space)
  - d) legal
  - e) legal
  - f) illegal ; contains an illegal character ‘
  - g) illegal ; contains an illegal character =
- 

(2)

- a) 246 ; legal – decimal
  - b) 246h ; legal – hexa
  - c) 1001 ; legal – decimal
  - d) 1,101 ; illegal – nondigit character ,
  - e) 2A3h ; legal – hexa
  - f) FFFEh ; illegal – must begin with digit
  - g) 0Ah ; legal – hexa
  - h) Bh ; illegal – must begin with digit
  - i) 1110b ; legal – binary
- 

(3)

- a) A DW 52
- b) WORD1 DW ?
- c) B DB 25h
- d) C1 DB ?
- e) illegal , number too big can't fit in word
- f) ARRAY1 DW 1,2,3,4,5

## Chapter Four

g) BELL EQU 07h  
 h) MSG EQU ‘THIS IS A MESSAGE\$’

---

(4)

<b>Address</b>	<b>content</b>	<b>beginning of var.</b>
0000h	7	A
0001h	BCh	B
0002h	1Ah	
0003h	‘H’	C
0004h	‘E’	
0005h	‘L’	
0006h	‘L’	
0007h	‘O’	

a) var. A : 0000h

var. B : 0001h

var. C : 0003h

b) content of byte 0002h is : 1Ah

c) content of byte 0004h is : ‘E’

d) offset address of character ‘O’ is : 0007h

---

## Chapter Four

(5)

- a) legal
  - b) illegal ; constant to segment
  - c) illegal ; segment to segment
  - d) legal
  - e) illegal ; memory to memory
  - f) illgal ; can't store in constant
  - g) illegal ; memory to memory operation
  - h) illegal ; out of range can't fit in 8-bit
  - i) illegal ; momory to memory / different size
- 

(6)

- a) MOV AX,B  
SUB AX,A  
MOV A,AX
- b) MOV AX,A  
INC AX  
NEG AX  
MOV A,AX
- c) MOV AX,A  
ADD AX,B  
MOV C,AX

## **Chapter Four**

d)    MOV     AX,B  
      ADD     AX,B  
      ADD     AX,B  
      ADD     AX,7  
      MOV     B,AX

e)    MOV     AX,B  
      SUB     AX,A  
      DEC     AX  
      MOV     A,AX

---

(7)

a)    MOV     AH,1  
      INT     21H  
      MOV     DL,AL  
      MOV     AH,2  
      INT     21H

Note : The cursor will move automatically to next position  
after reading.

b)    MOV     AH,1  
      INT     21H  
      ADD     AL,20H ;convert to lower case  
      MOV     DL,AL  
      MOV     AH,2  
      INT     21H

## **Chapter Four**

---

### Programming Exercises

(8)

```
.MODEL SMALL
.DATA
MSG    DB  0AH,0DH,'THE SUM OF '
C1     DB  ?, ' AND '
C2     DB  ?, ' IS '
SUM    DB  ?, '$'

.CODE
Main PROC

MOV    AX,@DATA      ; initialize DS
MOV    DS,AX

MOV    AH,2            ; display '?'
MOV    DL,'?'
INT    21H

MOV    AH,1
INT    21H            ; read 1st digit
MOV    C1,AL           ; store it in memory

INT    21H            ; read 2nd digit
MOV    C2,AL           ; store it in memory
```

## Chapter Four

```

ADD    AL,C1      ; add the two numbers
SUB    AL,30H     ; *
MOV    SUM,AL     ; store result in memory

LEA    DX,MSG     ; get the start address
MOV    AH,9
INT    21H        ; print the MSG until $

MOV    AH,4CH     ;return to DOS
INT    21H

Main   ENDP
END Main

```

\* when we add the two numbers , suppose we enter 2 and 7  
 Their ASCII code are 32h and 37h respectively, after  
 adding AL will contains 69h , now we have to SUB 30h to  
 get the ASCII code of 9 and it is 39h.

## **Chapter Four**

(9)

```

.MODEL SMALL
.DATA
MSG DB 0AH,0DH,"ENTER THREE INITIALS: $"
C1    DB      ?,0AH,0DH
C2    DB      ?,0AH,0DH
C3    DB      ?,'$'
.CODE
MAIN PROC
MOV    AX,@DATA      ;initialze DS
MOV    DS,AX

MOV    AH,9           ; display MSG
LEA    DX,MSG
INT    21H

MOV    AH,1           ; read & store 1st char.
INT    21H
MOV    C1,AL

INT    21H           ; read & store 2nd char.
MOV    C2,AL

INT    21H           ; read & store 3rd char.
MOV    C3,AL

```

## Chapter Four

```
MOV    AH,2          ; display enter
MOV    DL,0AH
INT    21H
MOV    DL,0DH
INT    21H

MOV    AH,9          ; display message from c1 until $
LEA    DX,C1
INT    21H

MOV    AH,4CH         ; return to DOS
INT    21H

MAIN  ENDP
END   MAIN
```

-----  
-----

## **Chapter Four**

(10)

```

.MODEL SMALL
.DATA
MSG    DB  0AH,0DH,'ENTER A HEX DIGIT: $'
MSG2   DB  0AH,0DH,'IN DECIMAL IT IS  1'
C1     DB  ?,'$'

.CODE
MAIN  PROC
MOV    AX,@DATA      ;initialize DS
MOV    DS,AX

MOV    AH,9            ; display 1st MSG
LEA    DX,MSG
INT    21H

MOV    AH,1            ; read hex digit (A – F)
INT    21H

SUB    AL,11H          ;*
;*

MOV    C1,AL           ;store it

MOV    AH,9            ;display 2nd MSG until $
LEA    DX,MSG2
INT    21H

```

## Chapter Four

```
MOV     AH,4CH          ; return to DOS  
INT     21H
```

```
MAIN    ENDP  
END     MAIN
```

\* ASCII for A is 41h  
ASCII for 0 is 30h

Difference is 11h

And for B,C,D,E,F are the same

Knowing that we include '1' in 2<sup>nd</sup> MSG



## Chapter Four

(11)

```
.MODEL SMALL

.DATA
STARS DB 0AH,0DH,'*****','$'

.CODE
MAIN PROC

MOV AX,@DATA           ;initialize DS
MOV DS,AX

MOV AH,9                ; prepare to display
LEA DX,STARS
INT 21H                 ; display 10 times *
INT 21H
```

## Chapter Four

```
MOV AH,4CH ;return to DOS  
INT 21H
```

```
MAIN ENDP  
END MAIN
```

\* it will be easy after study loop statement ch.6 to replace  
10 statement by the code :

```
MOV CX,10  
L: INT 21H  
LOOP L
```

---

---

## **Chapter Four**

(12)

.MODEL SMALL

.DATA

STARS DB ,\*\*\*\*\*',0AH,0DH,'\$'

MID DB '\*\*\* '

C1 DB ?

C2 DB ?

C3 DB ?,\*\*\*',0AH,0DH,'\$'

.CODE

MAIN PROC

MOV AX,@DATA ; initialize DS

MOV DS,AX

MOV AH,1 ;read & store 3

initials

INT 21H

MOV C1,AL

INT 21H

MOV C2,AL

INT 21H

MOV C3,AL

MOV AH,2 ; CR to current line

MOV DL,0DH

INT 21H

## Chapter Four

```

MOV AH,9          ;( display stars from the
                  ; beginning of line that will
                  ; cause to erase the 3 initial
                  ; and write over them)

LEA DX,STARS
INT 21H
INT 21H
INT 21H
INT 21H
INT 21H
INT 21H
LEA DX,MID      ;( display the middle
                  ; line with initials)
INT 21H

LEA DX,STARS      ;continue displaying stars
INT 21H
INT 21H
INT 21H
INT 21H
INT 21H

MOV AH,4CH        ; return to DOS
INT 21H
MAIN ENDP
END MAIN

```

## Chapter Four

Sample execution for question 12:

```
C:\ Administrator: Amer VIPER-VIRUS@LIVE.COM
C:\TEST>CH4Q12
*****
*****
*****
*****
*****
*** AHU ***
*****
*****
*****
*****
*****
C:\TEST>
```

## Chapter Five

(1)

Assuming that the flags are initially = 0

\*

Content of	CF	SF	ZF	PF	OF
a) AX= 8000h	0	1	0	1	1
b) AL= 02h	1	0	0	0	0
c) AL= FFh	0	1	0	1	0
d) AL= 81h	1	1	0	1	0
e) AX= 712Ah BX= 1ABCh	0	0	0	0	0
f) AL= 7Fh	1	0	0	0	1
g) AX= 8000h	1	1	0	1	1
h) AX= FFFFh	1	1	0	1	0

\* to know how we find flags value see Q2,Q3 &amp; Q4

(2)

cyout 0 1 cyinto

a)

$$\begin{array}{r}
 \text{AX=} \quad 01\text{xx} \text{XXXX XXXX XXXX} \\
 \text{BX=} \quad 01\text{xx} \text{XXXX XXXX XXXX} \\
 \hline
 \text{10}\text{xx} \text{XXXX XXXX XXXX}
 \end{array} +$$

x: could be 0 or 1 (i.e any number)

01

ex:

$$\begin{array}{l}
 \text{AX=} 7FA0h = 0111 \text{ 1111 1010 0000} \\
 \text{BX=} 600Dh = 0110 \text{ 0000 0000 1101} \\
 \hline
 \text{DFADh} = 1101 \text{ 1111 1010 1101}
 \end{array}$$

+

Cyinto = 1 ,cyout = 0

Cyinto XOR cyout = 1 XOR 0 = 1 = OF (signed overflow)

## Chapter Five

b)

$$\begin{array}{r}
 \text{cyout} \quad 1 \quad 0 \quad \text{cyinto} \\
 \uparrow \quad \uparrow \\
 \text{AX= } 10\text{xx } \text{XXXX } \text{XXXX } \text{XXXX} \\
 \underline{\text{BX= } 10\text{xx } \text{XXXX } \text{XXXX } \text{XXXX}} \quad + \\
 \text{100xx } \text{XXXX } \text{XXXX } \text{XXXX}
 \end{array}$$

Ex:

$$\begin{array}{r}
 \text{cyout} \quad 1 \quad 0 \quad \text{cyinto} \\
 \text{AX=} 9\text{DE4h} = 1000 \ 1101 \ 1110 \ 0100 \\
 \underline{\text{BX=} \ B216h = 1011 \ 0010 \ 0001 \ 0110} \quad + \\
 = 10011 \ 1111 \ 1111 \ 1010
 \end{array}$$

$$\text{Cyinto} = 0, \text{cyout} = 1$$

$\text{Cyinto XOR cyout} = 0 \text{ XOR } 1 = 1 = \text{OF}$  (signed overflow)

## Chapter Five

(3)

We'll show solution of one and the rest are the same

a.

$$\begin{array}{r}
 & & 0 & 1 \\
 512\text{Ch} & = & 0101 & 0001 & 0010 & 1100 \\
 4185\text{h} & = & 0100 & 0001 & 1000 & 0101 \\
 \hline
 & & 92\text{B1h} & = & 1001 & 0010 & 1011 & 0001
 \end{array}$$

**Method 1:**

By taking XOR between  $\text{cy}_{\text{into}}$  and  $\text{cy}_{\text{out}}$

$\text{Cy}_{\text{into}} = 1$  ,  $\text{cy}_{\text{out}} = 0$  see Q2.a

$1 \text{ XOR } 0 = 1 \quad \text{OF} = 1 \text{ (signed)}$

$\text{Cy}_{\text{out}} = \text{CF} = 0 \quad \text{(unsigned)}$

**Method 2 :**

First number is positive

Second number is positive

After addition the result expected is positive , but the result is negative ,so there are error occurred

The error indicate by  $\text{OF} = 1$

---

	<b>Content of AX</b>	<b>signed OF</b>	<b>unsigned CF</b>
a)	92B1h	1	0
b)	18DDh	0	1
c)	BC97h	0	1
d)	E132h	1	0
e)	74FFh	0	0

## Chapter Five

(4)

We'll show solution of one and the rest are the same

c.

$$\begin{array}{r}
 \text{borrow 1} \quad 0 \text{ borrow} \\
 \text{into msb} \quad \quad \quad \text{into bit-14} \\
 19\text{BCh} = 0001 \ 1001 \ 1011 \ 1100 \\
 81\text{FEh} = 1000 \ 0001 \ 1111 \ 1110 \\
 \hline
 97\text{BEh} = 10001 \ 0111 \ 1011 \ 1110
 \end{array}$$

Method 1:

By taking XOR between borrow<sub>into msb</sub> and borrow<sub>into bit-14</sub>  
 borrow<sub>into msb</sub> XOR borrow<sub>into bit-14</sub>

$$1 \text{ XOR } 0 = 1 \quad \text{OF} = 1 \quad (\text{signed})$$

$$\text{borrow}_{\text{into msb}} = \text{CF} = 1 \quad (\text{unsigned})$$

Method 2:

First number is positive

Second number is negative

But second number is greater than first number.

And  $1^{\text{st}} - (-2^{\text{nd}}) = 1^{\text{st}} + 2^{\text{nd}}$  = positive.

The result is negative so an error occurred  $\text{OF} = 1$

---

## Chapter Five

---

<b>Content of AX</b>	<b>signed OF</b>	<b>unsigned CF</b>
a) 07BDh	0	0
b) 6878h	1	0
c) 97BEh	1	1
d) 01F3h	0	1
e) 1A22h	1	0

## **Chapter Six**

(1)

a)      CMP      AX,0  
           JGE      END\_IF  
           MOV      BX,-1

---

END\_IF:

b)      CMP      AL,0  
           JNL      ELSE\_  
           MOV      AH,0FFh  
           JMP      END\_IF  
ELSE\_:    MOV      AH,0  
END\_IF:

OR

CMP      AL,0  
JL        THEN\_  
MOV      AH,0  
JMP      END\_IF  
THEN\_:    MOV      AH,0FFh  
END\_IF:

---

c)      CMP      DL,'A'  
JL        END\_IF  
CMP      DL,'Z'  
JG        END\_IF  
MOV      AH,2      ; display DL  
INT      21H  
END\_IF:

## **Chapter Six**

d)      CMP      AX,BX  
           JGE      END\_IF  
           CMP      BX,CX  
           JGE      ELSE\_  
           MOV      AX,0               ;then  
           JMP      END\_IF

---

ELSE\_:    MOV      BX,0  
 END\_IF:

e)      CMP      AX,BX  
           JL       THEN\_  
           CMP      BX,CX  
           JL       THEN\_  
           MOV      DX,1               ;else  
           JMP      END\_IF

---

THEN\_:    MOV      DX,0  
 END\_IF:

f)      CMP      AX,BX  
           JNL      ELSE\_  
           MOV      AX,0  
           JMP      END\_IF

---

ELSE\_:    CMP      BX,CX  
           JNL      ELSE\_2

## Chapter Six

MOV BX,0  
JMP END\_IF

ELSE\_2: MOV CX,0  
END\_IF:

OR

CMP AX,BX  
JL THEN\_  
CMP BX,CX  
JL THEN\_2  
MOV CX,0  
JMP END\_IF

THEN\_: MOV AX,0  
JMP END\_IF  
THEN\_2:MOV BX,0

END\_IF:

---

.....

## **Chapter Six**

(2)

```
MOV    AH,1      ;read a character
INT    21H
```

```
CMP    AL,'A'   ;AL has the letter in.
JE     EXE_CR
CMP    AL,'B'
JE     EXE_LF
;any letter else
MOV    AH,4Ch    ;return to DOS
INT    21H
JMP    END_CASE
```

```
EXE_CR: MOV    AH,2
        MOV    DL,0Dh
        INT    21H
        JMP    END_CASE
```

```
EXE_LF: MOV    AH,2
        MOV    DL,0Ah
        INT    21H
```

END\_CASE:

---

## Chapter Six

(3)

- a) first we find how many loops needed:  
(last term – first term )/ difference  
 $(148 - 1)/3 = 49$  loops

```
MOV    CX,49
MOV    AX,1      ;first term
MOV    BX,1
```

```
L1:   ADD    BX,3  ;add the diff. between each term
       ADD    AX,BX
```

LOOP L1

---

## Chapter Six

- b) (first term – last term)/ difference  
 $(100 - 5)/5 = 19$  loops

```
MOV    CX,19
MOV    AX,100
MOV    BX,100
```

```
L1:      SUB    BX,5
          ADD    AX,BX
```

LOOP L1

---

(4)

a) MOV CX,50
 MOV DX,1
 MOV AX,1
 L1:ADD AX,4

LOOP L1

---

## Chapter Six

b)

MOV	AH,1	
INT	21H	
MOV	AH,2	
MOV	DL,0AH	:enter
INT	21H	
MOV	DL,0DH	
INT	21H	
MOV	DL,AL	
MOV	CX,80	
<hr/>		
DISPLAY:	INT	21H
	LOOP	DISPLAY

c)

MOV	CX,5	
MOV	AH,7	;*
L1:	INT	21H
	LOOP	L1
<hr/>		
	MOV	DL,'X'
	MOV	CX,5
	MOV	AH,2
L2:	INT	21H
	LOOP	L2

\* function 7 of int 21h it's job is:

Input character without showing it at the screen and the cursor still in the first position

The ASCII of letter after reading in AL

## Chapter Six

(5)

```
MOV AX,0
;CX dividend    BX divisor      AX quotient
while_:
        CMP CX,BX
        JL END_WHILE
        INC AX
        SUB CX,BX
        JMP WHILE_
END_WHILE:
```

; you can test the program by DEBUG program

---

(6)

```
XOR CX,CX
; CX product      BX,AX  positive numbers
L1: ADD CX,AX
    DEC BX
    JNZ L1
```

;enter values of AX,BX and test the code by DEBUG

---

## **Chapter Six**

(7)  
a)

```

        MOV    AH,1
        MOV    CX,80
L1:   INT    21H
        CMP    AL,20H      ; blank = 20h
        LOOPE  L1

```

; always enter blank if any char. else go out

---

b)

```

        MOV    AH,1
        MOV    CX,80
L1:   INT    21H
        CMP    AL,0DH
        LOOPNE L1

```

; compare will set the flag ZF if equal 0DH and that affect on LOOPNE .

.....  
.....

## Chapter Six

### Programming Exercises

(8)

.MODEL SMALL

.CODE

MAIN PROC

```
    MOV AH,2
    MOV DL,'?'
    INT 21H           ;display ‘?’
```

```
    MOV AH,1
    INT 21H ; read 1st char. & put it in BL
    MOV BL,AL
    INT 21H ;read 2nd char. in AL
```

-----

```
    CMP BL,AL
    JG SWITCH ;if not ordered
    JMP DISPLAY
```

SWITCH: XCHG AL,BL

DISPLAY:

```
    MOV AH,2
    MOV DL, 0AH      ;enter
    INT 21H
    MOV DL,BL
    INT 21H
    MOV DL,AL
```

## Chapter Six

INT 21H

OUT\_ :           MOV AH,4CH  
                  INT 21H

MAIN ENDP  
END MAIN



## **Chapter Six**

(9)

.MODEL SMALL

.CODE

MAIN PROC

;BH counter (10 char per line)

MOV BH,0

MOV AH,2

MOV CX,7FH ; 127 char.

MOV DL,80H ;1<sup>ST</sup> letter

MOV BL,80H

CONTINUE: MOV DL,BL  
 INT 21H  
 DEC CX  
 INC DL  
 MOV BL,DL  
 MOV DL,20H ;blank space  
 INT 21H  
 INC BH  
 CMP BH,10  
 JE COUNT  
 CMP CX,0  
 JNE CONTINUE ;is letter finished?  
 JMP out\_

COUNT: MOV BH,0 ;start again counter  
 MOV DL,0AH ;enter  
 INT 21H  
 MOV DL,0DH  
 INT 21H

## Chapter Six

Out\_ :                   JMP CONTINUE  
                          MOV AH,4CH  
                          INT 21H

MAIN ENDP  
END MAIN

---

## Sample execution for Q9

```
C:\TEST>CH6Q9
C:\TEST>
```

## Chapter Six

(10)

TITLE Q9:WRITTEN BY AMER AL-KHSABA (V.V)

.MODEL SMALL

.STACK 64

;-----

.DATA

CR EQU 0DH

LF EQU 0AH

M1 DB CR,LF,'ENTER A HEXA DIGIT :','\$'

M2 DB CR,LF,'IN DECIMAL IT IS :'

C1 DB ?,'\$'

M3 DB CR,LF,'DO YOU WANT TO DO IT AGAIN ?'

DB 'Y/N','\$'

M4 DB CR,LF,'ILLEGAL CHARACTER - ENTER '

DB '0..9OR A..F :','\$'

M5 DB CR,LF,'IN DECIMAL IT IS :1'

C2 DB ?,'\$'

;-----

.CODE

MAIN PROC

MOV AX,@DATA ;Initialize DS

MOV DS,AX

BEGIN: MOV AH,9

LEA DX,M1

INT 21H

;-----

NEW: MOV AH,1 ;read hexa digit

INT 21H

;-----

## **Chapter Six**

```

CMP AL,'0'           ;dectictive for errors
JL ILLEGAL
CMP AL,'9'
JG ILLEGAL
;-----
MOV C1,AL
MOV AH,9
LEA DX,M2
INT 21H
;-----
MSG:    MOV AH,9   ; do you want to do it again
        LEA DX,M3
        INT 21H
;-----
MOV AH,1           ; yes or no
INT 21H
CMP AL,'Y'
JE BEGIN
CMP AL,'y'
JE BEGIN
JMP ENDD ;any char. else go out
;-----
ILLEGAL: CMP AL,'A'      ; dectictive for letter error
          JL ILLEGAL2
          CMP AL,'F'
          JG ILLEGAL2
          SUB AL,11H
          MOV C2,AL
;-----
          MOV AH,9

```

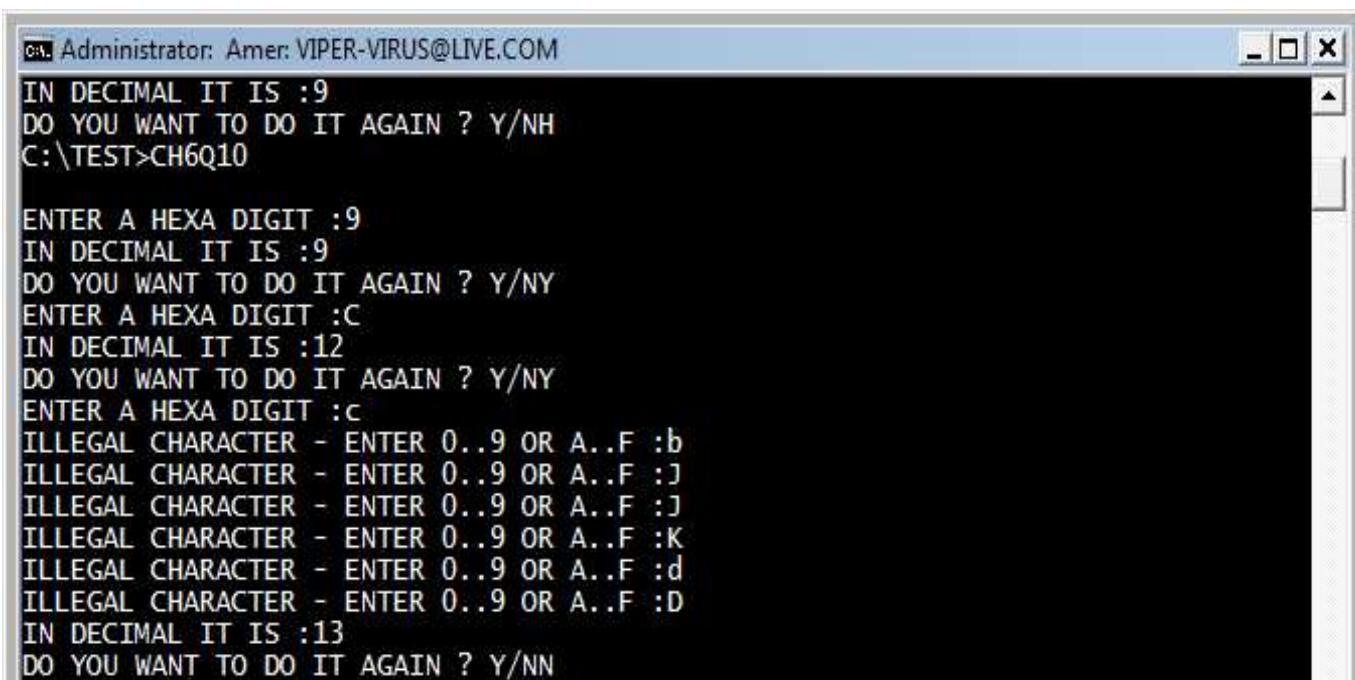
## Chapter Six

```

        LEA DX,M5
        INT 21H
        ;-----
        JMP MSG
        ;-----
ILLEGAL2: MOV AH,9
        LEA DX,M4
        INT 21H
        JMP NEW
ENDD:   MOV AH,4CH
        INT 21H

MAIN ENDP
END MAIN

```



The screenshot shows a Windows Command Prompt window with the title bar "Administrator: Amer: VIPER-VIRUS@LIVE.COM". The window contains the following text:

```

C:\ Administrator: Amer: VIPER-VIRUS@LIVE.COM
IN DECIMAL IT IS :9
DO YOU WANT TO DO IT AGAIN ? Y/NH
C:\TEST>CH6Q10

ENTER A HEXA DIGIT :9
IN DECIMAL IT IS :9
DO YOU WANT TO DO IT AGAIN ? Y/NY
ENTER A HEXA DIGIT :C
IN DECIMAL IT IS :12
DO YOU WANT TO DO IT AGAIN ? Y/NY
ENTER A HEXA DIGIT :C
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :b
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :J
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :J
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :K
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :d
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :D
IN DECIMAL IT IS :13
DO YOU WANT TO DO IT AGAIN ? Y/NN

```

## Chapter Six

(11)

Same Q10 but we put error counter in illegal lable.

TITLE Q9:WRITTEN BY AMER AL-KHSABA (V.V)

.MODEL SMALL

.STACK 64

-----

.DATA

CR EQU 0DH

LF EQU 0AH

M1 DB CR,LF,'ENTER A HEXA DIGIT :','\$'

M2 DB CR,LF,'IN DECIMAL IT IS :'

C1 DB ?,'\$'

M3 DB CR,LF,'DO YOU WANT TO DO IT AGAIN ?'

DB 'Y/N','\$'

M4 DB CR,LF,'ILLEGAL CHARACTER – ENTER ‘

DB ‘0..9 OR A..F :','\$'

M5 DB CR,LF,'IN DECIMAL IT IS :1'

C2 DB ?,'\$'

M6 DB CR,LF,'ERROR THREE TIMES ENTERD’

DB ‘ILLEGAL CHAR !','\$'

-----

.CODE

MAIN PROC

MOV AX,@DATA ;INITIALIZE DS

MOV DS,AX

-----

BEGIN: MOV BL,0 ;counter for error

MOV AH,9

LEA DX,M1

## Chapter Six

```

INT 21H
;-----
NEW:    MOV AH,1      ;read hexa digit
        INT 21H
;-----
        CMP AL,'0'    ;dectective for errors
        JL ILLEGAL
        CMP AL,'9'
        JG ILLEGAL
;-----
        MOV C1,AL
        MOV AH,9
        LEA DX,M2
        INT 21H
;-----
MSG:    MOV AH,9
        LEA DX,M3
        INT 21H
;-----
        MOV AH,1      ;do u want to do it again
        INT 21H
        CMP AL,'Y'
        JE BEGIN
        CMP AL,'y'
        JE BEGIN
        JMP ENDD
;-----
ILLEGAL: CMP AL,'A'
        JL ILLEGAL2
        CMP AL,'F'

```

## **Chapter Six**

JG ILLEGAL2  
SUB AL,11H

```

MOV C2,AL
;-----
MOV AH,9
LEA DX,M5
INT 21H
;-----
JMP MSG
;-----
ILLEGAL2: MOV AH,9
          LEA DX,M4
          INT 21H
; counter for 3-times illegal input
          INC BL
          CMP BL,3
          JE ENDDD
          JMP NEW
ENDDD:   MOV AH,9
          LEA DX,M6
          INT 21H
ENDD:    MOV AH,4CH
          INT 21H
MAIN ENDP
END MAIN

```

## Chapter Six

Sample execution for Q11

```
C:\TEST>ch6q11

ENTER A HEXA DIGIT :A
IN DECIMAL IT IS :10
DO YOU WANT TO DO IT AGAIN ? Y/Ny
ENTER A HEXA DIGIT :a
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :[
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :?
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :
ERROR THREE TIMES ENTERD ILLEGAL CHAR !
C:\TEST>ch6q11

ENTER A HEXA DIGIT :9
IN DECIMAL IT IS :9
DO YOU WANT TO DO IT AGAIN ? Y/Ny
ENTER A HEXA DIGIT :c
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :e
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :r
ILLEGAL CHARACTER - ENTER 0..9 OR A..F :
ERROR THREE TIMES ENTERD ILLEGAL CHAR !
C:\TEST>
```

## Chapter Seven

(1)

- a) 10001011b
  - b) 11111001b
  - c) 10100110b
  - d) 10100001b
- 

(2)

- a) AND AX,AAAAh
  - b) OR BL,81h
  - c) XOR DX,8000h
  - d) method 1 : NOT WORD1
  - method 2 : XOR WORD1,FFFFh
- 

(3)

- a) TEST AX,FFFFh
  - b) TEST BX,0001h
  - c) TEST DX,8000h
  - d) TEST DX,8000h
  - e) TEST BL,FFh
-

## Chapter Seven

(4)

Assuming AL = 11001011b = CBh      then

- a) AL = 10010110b = 96h
  - b) AL = 01100101b = 65h
  - c) AL = 00101111b = 2Fh
  - d) AL = 01111001b = 79h
  - e) AL = 11110010b = F2h
  - f) AL = 10010111b = 97h
  - g) AL = 11111001b = F9h
- 

(5)

a) SHL B5,1

b) MOV CL,3  
SHL AL,CL

c) MOV AX,32142  
MOV CL,2  
SHR AX,CL

d) MOV BX,-2145  
MOV CL,4  
SAR BX,CL ;SAR to keep the sign

---

## Chapter Seven

(6)

- a) OR AL,30h
  - b) OR DL,20h
- 

(7)

- a) MOV DL,BL  
MOV CL,3  
SHL BL,CL ; BL= 8BL  
SHL DL,1 ; DL= 2BL  
ADD BL,DL ; BL= 8BL+2BL =10BL
  
- b) MOV CL,3  
MOV AH,0 ;empty AH  
ROR AX,CL ; remainder in AH

## Chapter Seven

### Programming Exercises

(8)

.MODEL SMALL

.DATA

M1 DB 'TYPE A CHARACTER :','\$'

M2 DB 0AH,0DH,'THE ASCII CODE OF '

C1 DB ?,' IN BINARY IS :','\$'

M3 DB 0AH,0DH,'THE NUMBER OF 1 BIT IS '

C2 DB ?,'\$'

.CODE

MAIN PROC

MOV AX,@DATA ;Initialize DS

MOV DS,AX

MOV AH,9 ;prompt the user

LEA DX,M1

INT 21H

-----

MOV AH,1 ;read character

INT 21H

MOV BL,AL

MOV C1,AL

;store character

MOV AH,9

;display results

```
LEA DX,M2
INT 21H
```

## Chapter Seven

```
MOV BH,0 ;counter for one's
MOV CX,8
MOV AH,2
```

```
L1: SHL BL,1 ;display content of BL
```

```
JC L2
MOV DL,'0'
INT 21H
JMP L4
```

```
L2: MOV DL,'1'
```

```
INT 21H
INC BH
```

```
L4: LOOP L1
```

```
ADD BH,30H ;convert to char.
MOV C2,BH
MOV AH,9 ;display number of one's
LEA DX,M3
INT 21H
```

```
;-----
```

```
MOV AH,4CH ;return to DOS
INT 21H
```

```
MAIN ENDP
END MAIN
```

## Chapter Seven

Sample execution for Q8

```
C:\TEST>CH7Q8
TYPE A CHARACTER :b
THE ASCII CODE OF b IN BINARY IS:01100010
THE NUMBER OF 1 BIT IS 3
C:\TEST>
C:\TEST>CH7Q8
TYPE A CHARACTER :B
THE ASCII CODE OF B IN BINARY IS:01000010
THE NUMBER OF 1 BIT IS 2
C:\TEST>
C:\TEST>CH7Q8
TYPE A CHARACTER :K
THE ASCII CODE OF K IN BINARY IS:01001011
THE NUMBER OF 1 BIT IS 4
C:\TEST>
```

## **Chapter Seven**

(9)

```

.MODEL SMALL
.DATA
M1 DB 0AH,0DH,'TYPE A CHARACTER :','$'
M2 DB 0AH,0DH,'THE ASCII CODE OF '
C1 DB ?,'IN HEXA IS ','$'

.CODE
MAIN PROC
MOV AX,@DATA           ;intialize DS
MOV DS,AX
;-----
BEGIN:    MOV AH,9      ;prompt user
          LEA DX,M1
          INT 21h

          MOV AH,1      ;read char.
          INT 21H

          CMP AL,0DH    ;if CR exit
          JE OUT_

          MOV C1,AL     ;store char.
          MOV BL,AL     ;take a copy of char

          MOV AH,9      ;display 2nd MSG
          LEA DX,M2

```

## INT 21H

;-----

**Chapter Seven**

```
MOV CL,4
SHR C1,CL      ;prapare for display 1st half
;* note below
ADD C1,30H      ;convert to char.
MOV DL,C1
JMP EXE1
```

continue: AND BL,0FH ;convert 2<sup>nd</sup> half to char.  
 CMP BL,9 ;if >9 mean A,B,C.....hex ch.  
 JG ERROR\_

```
ADD BL,30H      ;convert to char.
MOV DL,BL
JMP EXE2
```

EXE1: MOV AH,2 ;1<sup>st</sup> half displayed  
 INT 21H  
 JMP continue

EXE2: MOV AH,2
 INT 21H ;2<sup>nd</sup> half displayed

;-----  
 JMP BEGIN ;ask if you want to do it again

ERROR\_: ADD BL,37H ;convert to A,B,C.... hexa ch.  
 MOV DL,BL
 MOV AH,2 ;display it

INT 21H

## Chapter Seven

JMP BEGIN ;ask if you want to do it again

OUT\_ : MOV AH,4CH ;return to DOS  
INT 21H

MAIN ENDP  
END MAIN

\* note : in first half we doesn't need to compare if no. in C1 >9 because it will not be .  
Great no. could reach is 7 in “z”  
But in second half all hexa digit could be shown  
(Ex: z = 7A o = 6F )

sample execution for Q9

```

C:\TEST>ch7q9
TYPE A CHARACTER :a
THE ASCII CODE OF a IN HEXA IS 61
TYPE A CHARACTER :A
THE ASCII CODE OF A IN HEXA IS 41
TYPE A CHARACTER :z
THE ASCII CODE OF z IN HEXA IS 7A
TYPE A CHARACTER :Z
THE ASCII CODE OF Z IN HEXA IS 5A
TYPE A CHARACTER :
C:\TEST>

```

## **Chapter Seven**

(10)

```

.MODEL SMALL
.DATA
M1 DB 0AH,0DH,'TYPE A HEXA NUMBER (0 - FFFF) : ','$'
M2 DB 0AH,0DH,'IN BINARY IT IS : ','$'
M3 DB 0AH,0DH,'ILLEGAL HEXA DIGIT, TRY AGAIN : ','$'

.CODE
MAIN PROC
    MOV AX,@DATA      ;intialize DS
    MOV DS,AX

    MOV AH,9          ; prompts user
    LEA DX,M1
    INT 21H

    START :XOR BX,BX
            MOV CL,4
            MOV AH,1
            INT 21H      ;read

    WHILE_ :CMP AL,0DH
              JE END_WHILE ;exit if CR
              CMP AL,'0'   ;detect for error
              JL ERR
              CMP AL, '9'
              JG LETTER
              AND AL,0FH    ;get numerical value
              JMP SHIFT

```

## Chapter Seven

```
LETTER:    CMP AL,'F' ;if not in range (A...B) error  
          JG ERR  
          CMP AL,'A'  
          JL ERR  
          SUB AL,37H ;get numerical value
```

```
SHIFT:    SHL BX,CL ; ready to enter in register  
          OR BL,AL  
          INT 21H  
          JMP WHILE_ ;continue reading hexa
```

```
END WHILE:MOV AH,9 ; if finish reading
              LEA DX,M2 ;display result
              INT 21H
```

```
        MOV CX,16      ;display content of BX
        MOV AH,2
SHOW:   SHL BX,1
        JC ONE
```

```
MOV DL,'0'  
INT 21H  
JMP LOOP1
```

ONE:           MOV DL,'1'  
                INT 21H

LOOP1:      LOOP SHOW                ;do it 16 times  
                  JMP OUT

## **Chapter Seven**

```

ERR:      MOV AH,9
          LEA DX,M3      ;display error MSG

          INT 21H
          JMP START      ;and read again

OUT_:     MOV AH,4CH
          INT 21H

MAIN ENDP
END MAIN

```

sample execution for Q10

```

C:\ Administrator: Amer V.V
C:\TEST>ch7q10
TYPE A HEXA NUMBER (0 - FFFF): 12b
ILLEGAL HEXA DIGIT, TRY AGAIN :123
IN BINARY IT IS : 0000000100100011
C:\TEST>
C:\TEST>ch7q10
TYPE A HEXA NUMBER (0 - FFFF): 1A
IN BINARY IT IS : 0000000000011010
C:\TEST>
C:\TEST>ch7q10
TYPE A HEXA NUMBER (0 - FFFF): FE1C
IN BINARY IT IS : 1111111000011100
C:\TEST>

```

## *Chapter Seven*

(11)

.MODEL SMALL

.DATA

M1 DB 0AH,0DH,'TYPE A BINARY DIGIT ,UP TO 16-BIT: ','\$'  
M2 DB 0AH,0DH,'IN HEXA IT IS :','\$'

.CODE

MAIN PROC

MOV AX,@DATA ;initialize DS  
MOV DS,AX

MOV AH,9 ;prompt user

LEA DX,M1  
INT 21H

-----

MOV BX,0 ;prepare for read binary  
MOV AH,1  
MOV CX,16

L1:

INT 21H  
CMP AL,0DH ;finish enter  
JE OUT\_

AND AL,01H ;convert to numerical 0,1  
SHL BX,1 ;store it in BX  
OR BL,AL  
LOOP L1

## **Chapter Seven**

OUT\_ :      MOV AH,9  
               LEA DX,M2  
               INT 21H

L2:            MOV DH,4                 ;show 4-digit hexa  
               MOV CL,4  
               MOV DL,BH  
               SHR DL,CL  
               MOV AH,2

              CMP DL,9                 ;detect the digit  
               JG LETTER  
               ADD DL,30H  
               INT 21H  
               JMP L3

LETTER:

              ADD DL,37H  
               INT 21H  
               L3:          ROL BX,CL         ;get second hexa digit

              DEC DH  
               CMP DH,0  
               JNE L2                 ;if 4-digit hexa displayed exit

## Chapter Seven

MOV AH,4CH

;return to DOS

INT 21H

MAIN ENDP

END MAIN

sample execution for Q11

The screenshot shows a Windows Command Prompt window titled "Administrator: Amer V.V". The window contains the following text:

```
C:\TEST>CH7Q11
TYPE A BINARY DIGIT ,UP TO 16-BIT: 11101101
IN HEXA IT IS :00ED
C:\TEST>CH7Q11
TYPE A BINARY DIGIT ,UP TO 16-BIT: 1010000010110
IN HEXA IT IS :1416
C:\TEST>CH7Q11
TYPE A BINARY DIGIT ,UP TO 16-BIT: 00001111
IN HEXA IT IS :000F
C:\TEST>CH7Q11
TYPE A BINARY DIGIT ,UP TO 16-BIT: 1111
IN HEXA IT IS :000F
C:\TEST>
```

## Chapter Seven

(12)

.MODEL SMALL

.DATA

B1 DB ?

B2 DB ?

M1 DB 0AH,0DH,'TYPE A BINARY NUMBER , UP TO 8 DIGITS :','\$'

M2 DB 0AH,0DH,'THE BINARY SUM IS ','\$'

;-----

.CODE

MAIN PROC

MOV AX,@DATA ;initialize DS  
MOV DS,AX

MOV AH,9 ;prompt user

LEA DX,M1

INT 21H

MOV BL,0

MOV CX,8

MOV AH,1

L1: INT 21H

CMP AL,0DH

JE OUT

SUB AL,30H ;or use AND AL,01H

SHL BL,1

OR BL,AL

LOOP L1

## **Chapter Seven**

OUT: MOV B1,BL ;1st no. read and store

MOV AH,9 ;prompt user for 2nd no.

INT 21H

MOV BL,0 ;read 2nd no.

MOV CX,8

MOV AH,1

L2: INT 21H

CMP AL,0DH

JE OUT2

SUB AL,30H

SHL BL,1

OR BL,AL

LOOP L2

OUT2: MOV B2,BL ;2nd no. read and store

MOV AH,9 ;show result

LEA DX,M2

INT 21H

ADD BL,B1 ;BL has 2nd no. (last one read)

MOV AH,2

MOV CX,8

L3: SHL BL,1 ;display binary

JC ONE

MOV DL,'0'

INT 21H

JMP continue

ONE: MOV DL,'1'

## Chapter Seven

```
INT 21H  
continue: LOOP L3  
;-----
```

```
MOV AH,4CH      ;return to DOS  
INT 21H  
MAIN ENDP  
    END MAIN
```

sample execution for Q12

```
C:\> Administrator: Amer V.V  
C:\TEST>ch7q12  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :11001010  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :10011100  
THE BINARY SUM IS 01100110  
C:\TEST>  
C:\TEST>ch7q12  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :0100  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :0101  
THE BINARY SUM IS 00001001  
C:\TEST>ch7q12  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :11111111  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :11111111  
THE BINARY SUM IS 11111110  
C:\TEST>ch7q12  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :01  
TYPE A BINARY NUMBER , UP TO 8 DIGITS :11111110  
THE BINARY SUM IS 11111111  
C:\TEST>
```

## **Chapter Seven**

(13)

In this question we use procedures technique to make the solution easier and readable

You'll know more about procedure in next chapter (ch.8)

.MODEL SMALL

.DATA

M1 DB 0AH,'TYPE A HEXA NUMBER 0 -FFFF :\$'

M2 DB 0AH,'THE SUM IN HEXA IS \$'

COUNTER DB 4

NUM DW ?

.CODE

MAIN PROC

MOV AX,@DATA ;initialize DS

MOV DS,AX

MOV AH,9 ;prompt user

LEA DX,M1

INT 21H

CALL READ

MOV NUM,BX ;store 1st num.

MOV AH,9 ;prompt user for 2nd number

LEA DX,M1

INT 21H

CALL READ ;no need to store 2nd num.

;because it will still in BX

## **Chapter Seven**

;<-----

```
MOV AH,9      ;display result msg
LEA DX,M2
INT 21H
```

```
ADD BX,NUM
;BX=num1(NUM)+num2(BX)
JC SHOWCY      ;if there carry
MOV AH,2      ;no carry
MOV DL,'0'
INT 21H
JMP NEXT
```

```
SHOWCY:MOV AH,2
      MOV DL,'1'
      INT 21H
```

NEXT: CALL SHOW

```
MOV AH,4CH      ;return to DOS
INT 21H
```

MAIN ENDP ;end of main program

## Chapter Seven

```

;-----;
READ PROC      ;for read hexa digit
    XOR BX,BX
    MOV CL,4
    MOV AH,1
    INT 21H
WHILE_: CMP AL,0DH
        JE END_W
        CMP AL,'9' ;detect for letter hexa
        JG LETTER
        AND AL,0FH
        JMP SHIFT

LETTER: SUB AL,37H
SHIFT:   SHL BX,CL
          OR BL,AL
          INT 21H
          JMP WHILE_

END_W: RET
READ ENDP
;-----;
SHOW PROC      ;to display result of addition
    MOV CL,4
START: MOV DL,BH
        SHR DL,CL
        CMP DL,9
        JG LETTER1
        ADD DL,30H
        JMP SHOW1

```

## Chapter Seven

LETTER1:ADD DL,37H

```
SHOW1: MOV AH,2  
        INT 21H  
        ROL BX,CL  
        DEC COUNTER  
        CMP COUNTER,0  
        JNE START
```

RET

SHOW ENDP

-----

END MAIN

\* NOTE: taking a look on the result you will find it consist of 5-hex-digit ,last digit on the left is the carry .

So adding 4-hexa- digit will produce :

- No carry (last digit on the left = 0)
- carry (last digit on the left =1)

## Chapter Seven

sample execution for Q13

```
C:\ Administrator: Amer:VIPER-VIRUS@LIVE.COM
C:\TEST>CH7Q13

TYPE A HEXA NUMBER 0 -FFFF :21AB
TYPE A HEXA NUMBER 0 -FFFF :FE03
THE SUM IN HEXA IS 11FAE
C:\TEST>CH7Q13

TYPE A HEXA NUMBER 0 -FFFF :10
TYPE A HEXA NUMBER 0 -FFFF :E5
THE SUM IN HEXA IS 000F5
C:\TEST>CH7Q13

TYPE A HEXA NUMBER 0 -FFFF :E100F0
TYPE A HEXA NUMBER 0 -FFFF :1
THE SUM IN HEXA IS 00F09
C:\TEST>CH7Q13

TYPE A HEXA NUMBER 0 -FFFF :C100
TYPE A HEXA NUMBER 0 -FFFF :A000
THE SUM IN HEXA IS 16100
C:\TEST>
```

## Chapter Seven

(14)

Procedures are used

```
.MODEL SMALL
.DATA
M1 DB 0AH,'ENTER A DECIMAL DIGIT STRING :$'
M2 DB 0AH,'THE SUM OF THE DIGIT IN HEX IS $'
COUNTER DB 4

.CODE
MAIN PROC
    MOV AX,@DATA      ;initialize DS
    MOV DS,AX
    MOV AH,9    ;prompt user
    LEA DX,M1
    INT 21H
    CALL READ
    MOV AH,9
    LEA DX,M2  ;display result msg
    INT 21H
    CALL SHOW

    MOV AH,4CH      ;return to DOS
    INT 21H

MAIN ENDP
```

## **Chapter Seven**

```

;-----
READ PROC ;read decimal digit and add them
    XOR BX,BX
WHILE_: MOV AH,1
    INT 21H
    CMP AL,0DH
    JE END_W
    AND AL,0FH
    CBW      ;Convert Byte to Word
;since we cann't add BX,AL we use CBW
    ADD BX,AX
    JMP WHILE_

```

```

END_W:
    RET
READ ENDP
;-----

```

```

SHOW PROC
    MOV CL,4
START: MOV DL,BH
    SHR DL,CL
    CMP DL,9
    JG LETTER1
    ADD DL,30H
    JMP SHOW1

```

LETTER1:ADD DL,37H

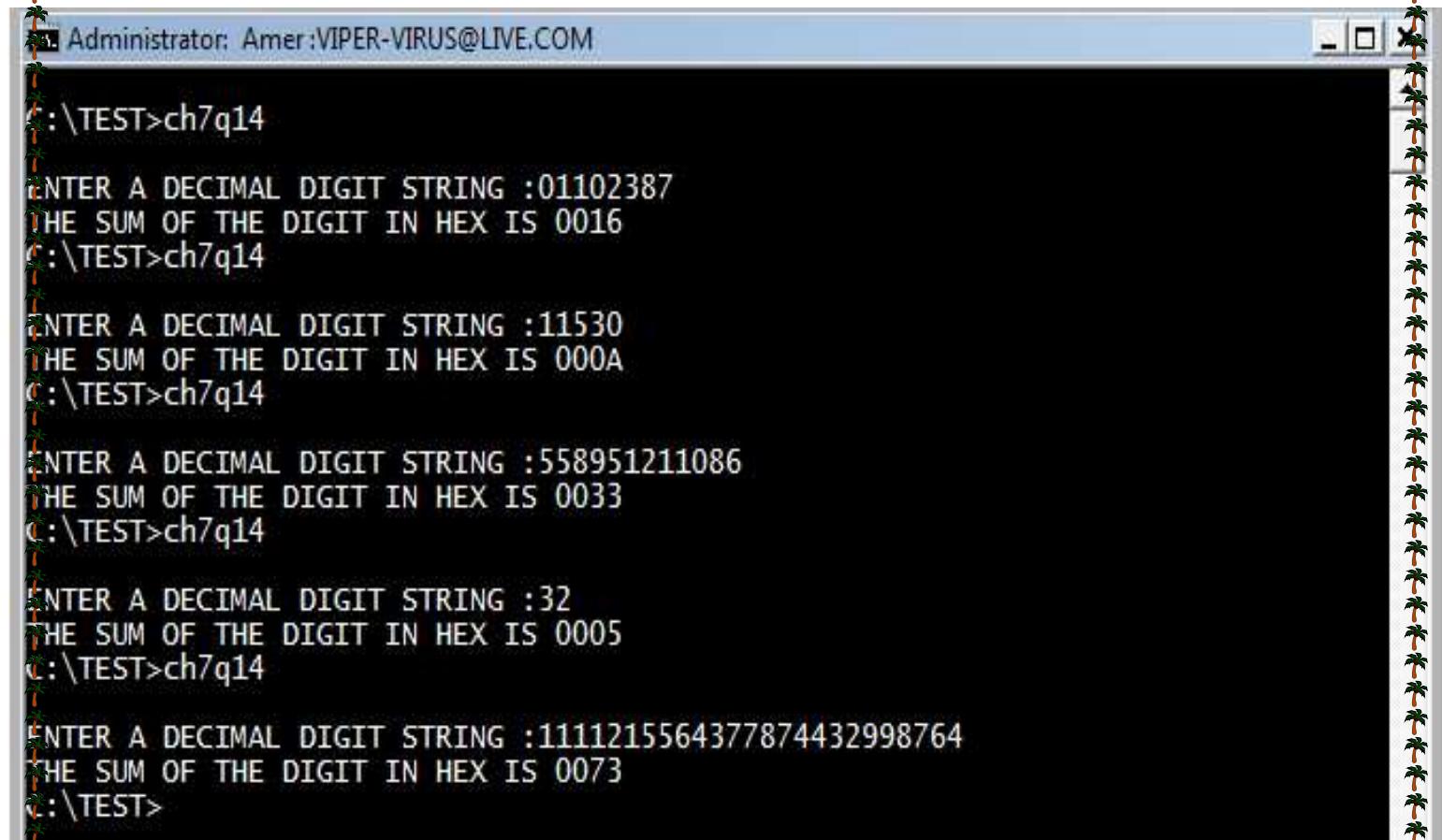
SHOW1: MOV AH,2

## INT 21H

Chapter Seven

```
ROL BX,CL  
DEC COUNTER  
CMP COUNTER,0  
JNE START  
RET  
SHOW ENDP  
  
END MAIN
```

sample execution for Q14



```
C:\> Administrator: Amer:VIPER-VIRUS@LIVE.COM  
C:\TEST>ch7q14  
ENTER A DECIMAL DIGIT STRING :01102387  
THE SUM OF THE DIGIT IN HEX IS 0016  
C:\TEST>ch7q14  
ENTER A DECIMAL DIGIT STRING :11530  
THE SUM OF THE DIGIT IN HEX IS 000A  
C:\TEST>ch7q14  
ENTER A DECIMAL DIGIT STRING :558951211086  
THE SUM OF THE DIGIT IN HEX IS 0033  
C:\TEST>ch7q14  
ENTER A DECIMAL DIGIT STRING :32  
THE SUM OF THE DIGIT IN HEX IS 0005  
C:\TEST>ch7q14  
ENTER A DECIMAL DIGIT STRING :1111215564377874432998764  
THE SUM OF THE DIGIT IN HEX IS 0073  
C:\TEST>
```

## Chapter Eight

(1)

- a) SP = **0100h**
  - b) 100h byte for stack mean  $100h/2$  word  
□ 80h word
- 

(2)

AX = **9ABCh**  
BX = **9ABCh**  
CX = **5678h**  
SP = **00FEh**

---

(3)

When SP= 0 and decrement its become SP= FFFEh.  
Program will not produce error ,but CS area is mix with SS area ,that might produce error in result in big program.

---

(4)

IP = **0300h**  
SP = **0108h**  
Word on top of stack is **0203h**

---

## Chapter Eight

(5)

a) IP = 012Ah  
SP = 0202h

b) IP = 012Ah  
SP = 0206h

Note : RET POP\_VALUE

Mean the stack get more size

Original value of SP = 202h + POP\_VALUE(4) = 206h

.....

(6)

a) POP AX  
PUSH AX

b) POP AX  
POP CX  
PUSH CX  
PUSH AX

c) POP AX  
POP BX  
PUSH AX  
PUSH BX

.....

## **Chapter Eight**

(7)

a) SAVE\_REGS PROC

```

POP AX ;save return address
PUSH BX ;save regs in the stack
PUSH CX
PUSH DX
PUSH SI
PUSH DI
PUSH BP
PUSH DS
PUSH AX ;top of stack is return address

```

RET

SAVE\_REGS ENDP

---

b) RESTORE\_REGS PROC

```

POP AX ; save return address
POP DS ; restore regs from the stack
POP BP
POP DI
POP SI
POP DX
POP CX
POP BX
PUSH AX ; top of stack is return address

```

RET

RESTORE\_REGS ENDP

## Chapter Eight

### Programming Exercises

(10)

a) Note : to make the range between 1 to 32767 we just ignore msb (sign bit /bit-15) by enter 15 bit.

```

INBINARY      PROC
    XOR  BX,BX
    Mov   CX,15
    MOV  AH,1
    INT  21H
WHILE_: CMP  AL,0DH
        JE   END_WHILE
        AND  AL,01H
        SHL  BX,1
        OR   BL,AL
        INT  21H
        LOOP WHILE_
END_WHILE:
RET
INBINARY      ENDP

```

---

## **Chapter Eight**

b)

by following the algorithm:

RANDOM PROC

;input in AX : start with any number in the range  
;output in AX (Random number)

PUSH BX ;save regs

PUSH DX

SHL AX,1 ; shift left once

MOV DX,AX ;get copy

SHL DX,1 ;make bit-14 be bit-15

XOR AX,DX ;xor bit-14 with bit-15

TEST DX,8000h ;result of XOR bit-14 with bit-15

JZ NOT\_ONE

OR AX,1 ;replace bit-0 with result (1)

JMP CL\_B15 ;clear bit-15

NOT\_ONE: AND AX,0FFE0H ;replace bit-0 with (0)

CL\_B15: AND AX,7FFFH

POP DX ;restore regs

POP BX

RET

---

RANDOM ENDP

## **Chapter Eight**

c)

OUTBIN PROC

;input AX

;output on screen

;we copy number in BX to make sure not changing the

;number because of output function that depend on AX

                MOV BX,AX ;get copy

                MOV CX,16

start:         ROL BX,1

                JC ONE

                MOV AH,2

                MOV DL,'0'

                INT 21H

                JMP L1

ONE:         MOV AH,2

                MOV DL,'1'

                INT 21H

L1:         LOOP start

;put the random number in AX to be new number to  
generate another random number

                MOV AX,BX

RET

OUTBIN ENDP

---

## **Chapter Eight**

The full program looks like :

```

.MODEL SMALL
.stack 64
.DATA
Counter    DB    100    ;number to be generated
Counter2   DB    4      ;4 number on same line

.CODE
Main    PROC
    MOV AX,@DATA
    MOV DS,AX
;
;----- Begin: L10:
    CALL INBINARY
    MOV AX,BX
    MOV D,4
    POP AX
    CALL RANDOM
    CALL OUTBIN
    PUSH AX
;
;----- MOV AH,2
;----- MOV DL,20H ;blank space
;----- INT 21H

```

;---

## **Chapter Eight**

```

DEC counter2
CMP counter2,0 ;is there 4-number on line
    JE NEW_LINE
    DEC Counter
    CMP Counter,0
    JNE L10

JMP OUT_

NEW_LINE:   MOV AH,2
            MOV DL,0AH ;enter LF
            INT 21H
            JMP Begin

OUT_:      MOV AH,4CH
            INT 21H

;-----


Main ENDP
;see Appendix C for include pseudo-ops
;Test is folder's name in c:\ contain procedures
INCLUDE C:\TEST\RANDOM.ASM
INCLUDE C:\TEST\INBINARY.ASM
INCLUDE C:\TEST\OUTBIN.ASM

END Main

```

## **Chapter Eight**

sample execution for Q10 (ran. Num.  
gen.)

```
C:\TEST>nt
011101110010000 0011000101100001 0010011101000111 0101001110010011
0110100101101010 0111011101111101 0011001100001111 0010101000100011
0111110011001011 0000101010111011 001111110011010 0000000101011101
0000011111001110 0001000010100100 0110001111011000 0100100011010001
0011001011100110 0010111001010101 011001011111111 0101110000000011
0100100000001010 0011000000111100 0010000010001001 0100001100110111
0000101010110010 001111110101100 0000000111101001 0000010001110110
0001100100110100 0101011010111000 0111011110010000 0011000101100001
0010011101000111 0101001110010011 0110100101101010 0111011101111101
0011001100001111 0010101000100011 0111110011001011 0000101010111011
001111110011010 00000000101011101 0000011111001110 0001000010100100
0110001111011000 0100100011010001 00110010111100110 0010111001010101
0110010111111111 0101110000000011 0100100000001010 0011000000111100
0010000010001001 0100001100110111 0000101010110010 001111110101100
00000000111101001 0000010001110110 0001100100110100 010101101011100
0111011110010000 00110000101100001 0010011101000111 0101001110010011
0110100101101010 0111011101111101 0011001100001111 0010101000100011
0111110011001011 0000101010111011 0011111110011010 0000000101011101
00000011111001110 0001000010100100 0110001111101100 0100100011010001
00110010111100110 0010111001010101 011001011111111 0101110000000011
0100100000001010 0011000000111100 0010000010001001 0100001100110111
0000101010110010 001111110101100 0000000111101001 0000010001110110
0001100100110100 0101011010111000 0111011110010000 0011000101100001
0010011101000111 0101001110010011 0110100101101010 0111011101111101
0011001100001111 0010101000100011 0111110011001011 0000101010111011
001111110011010 00000000101011101 0000011111001110 0001000010100100
0110001111011000 0100100011010001 00110010111100110 0010111001010101
0110010111111111 0101110000000011 0100100000001010 0011000000111100
0010000010001001 0100001100110111 0000101010110010 001111110101100
00000000111101001 0000010001110110 0001100100110100 010101101011100
0111011110010000 00110000101100001 0010011101000111 0101001110010011
0110100101101010 0111011101111101 0011001100001111 0010101000100011
0111110011001011 0000101010111011 0011111110011010 0000000101011101
00000011111001110 00000101010111011 0011111110011010 0000000101011101
```

## **Chapter Nine**

(1)

DX	AX	CF/OF
a) 0000	0018	0
b) 000F	F000	1
c) FFFF	FFFB	0
d) 0000	8000	1
e) Illegal : source must be register or M.L		

---

(2)

	AX	CF/OF
a)	0AB0	1
b)	FAB0	1
c)	00AB	0
d)	FFF6	0

---

(3)

	DX	AX
a)	0001	0003
b)	000E	0FFF
c)	FFFF	FFFF
d) divide overflow		

---

## Chapter Nine

(4)

	AH	AL
a)	01	04
b)	FF	02
c)	0E	0F
d)	divide overflow	

---

(5)

	DX
a)	0000
b)	FFFF
c)	0000

---

(6)

	AX
a)	FFF0
b)	00F5
c)	FF80

---

## **Chapter Nine**

(7)

a) MOV AX,5 ; AX=5  
 IMUL A ;AX=5\*A  
 SUB AX,7 ;AX=5\*A-7  
 MOV A,AX ;A=5\*A-7

---

b) MOV AX,A ;AX=A  
 SUB AX,B ;AX=A-B  
 ADD B,10 ;B=B+10  
 IMUL B ;AX=(A-B)\*(B+10)  
 MOV B,AX ;B=(A-B)\*(B+10)

---

c) MOV AX,9 ;AX=9  
 IMUL A ;AX=9\*A  
 MOV BX,6 ;BX=6  
 SUB BX,AX ;BX=6-9\*A  
 MOV A, BX ;A=6-9\*A

---

## **Chapter Nine**

d) MOV AX,A ;AX=A  
 IMUL AX ;AX=A<sup>2</sup>  
 MOV A,AX ;A=A<sup>2</sup>  
 MOV AX,B ;AX=B  
 IMUL AX ;AX=B<sup>2</sup>  
 MOV BX,AX ;BX=B<sup>2</sup>  
 ADD BX,A ;BX=B<sup>2</sup>+A<sup>2</sup>  
 MOV AX,C ;AX=C  
 IMUL AX ;AX=C<sup>2</sup>

CMP BX,AX ;A<sup>2</sup>+B<sup>2</sup>=? C<sup>2</sup>  
 JE SET\_CF  
 CLC ;CLEAR CF  
 JMP END\_IF  
 SET\_CF: STC ;SET CF

END\_IF:

-----  
 -----

## Chapter Nine

### Programming Exercises

(8)

We check for overflow in 2 steps:

1. after line 42 we check for overflow if DX not zero

CMP DX,0

JNE @NOT\_DIGIT

2. after line 44 we check for overflow if there carry

JC @NOT\_DIGIT

You can see the procedure in Appendix C “INDEC2”

## Chapter Nine

(9)

```
.MODEL SMALL
```

```
.STACK 64
```

```
.DATA
```

```
M1 DB 0AH,0DH,'THE TIME IS $'
```

```
.CODE
```

```
MAIN PROC
```

```
    MOV AX,@DATA           ;intialize DS
    MOV DS,AX
```

```
    CALL INDEC
```

```
    MOV DX,0    ;make sure no thing in DX
```

```
    MOV BX,3600          ;60min*60sec=1hr
```

```
    DIV BX
```

```
    PUSH AX
```

```
    PUSH DX
```

```
;-----
```

```
    MOV AH,9             ;display result
```

```
    LEA DX,M1
```

```
    INT 21H
```

```
    POP DX
```

```
    POP AX
```

```
;AX has hours
```

```
    CALL OUTDEC
```

```
    PUSH DX
```

## Chapter Nine

```

MOV AH,2
MOV DL,':
INT 21H

;-----;

POP AX
MOV BX,60
MOV DX,0
DIV BX
CALL OUTDEC ;AX has mins
PUSH DX

;-----;

MOV AH,2
MOV DL,':
INT 21H
POP AX
CALL OUTDEC ;AX has sec remainder
;-----;
MOV AH,4CH ;return to DOS
INT 21H

MAIN ENDP

```

```

INCLUDE C:\TEST\OUTDEC.ASM
INCLUDE C:\TEST\INDEC.ASM
END MAIN

```

You can get procedures code that included above from Appendix C ,Test :is name for folder in c:\ that contain procedures

## Chapter Nine

sample execution for Q9

```
C:\TEST>ch9q9
?3730
THE TIME IS 1:2:10
C:\TEST>ch9q9
?60
THE TIME IS 0:1:0
C:\TEST>ch9q9
?3600
THE TIME IS 1:0:0
C:\TEST>ch9q9
?65
THE TIME IS 0:1:5
C:\TEST>ch9q9
?4300
THE TIME IS 1:11:40
```

## **Chapter Nine**

(10)

Note: Half = 50 cents

Quarter = 25 cents

Dime = 10 cents

Nickle = 5 cents

Penny = 1 cents

```
.MODEL SMALL
```

```
.DATA
```

```
START DB 0AH,0DH
```

```
HLF DB ?, 'Half '
```

```
QUA DB ?, 'Quarter '
```

```
DIM DB ?, 'Dimes '
```

```
NIC DB ?, 'Nickels '
```

```
PEN DB ?, 'Pennies ','$'
```

```
.CODE
```

```
MAIN PROC
```

```
    MOV AX,@DATA
```

```
    MOV DS,AX
```

```
;
```

```
    CALL INDEC ;AX has cents
```

```
    MOV BL,50 ;half DollaR
```

```
    DIV BL
```

```
    ADD AL,30h ;CONVERT TO DIGIT
```

```
    MOV HLF,AL
```

## **Chapter Nine**

```

;-----  

MOV AL,AH  

MOV AH,0  

    MOV BL,25 ;Quarter  

    DIV BL  

    ADD AL,30h  

    MOV QUA,AL  

;-----  

MOV AL,AH  

MOV AH,0  

    MOV BL,10 ;DIME  

    DIV BL  

    ADD AL,30H  

    MOV DIM,AL  

;-----  

MOV AL,AH  

MOV AH,0  

    MOV BL,5 ;nickle  

    DIV BL  

    ADD AL,30h  

    MOV NIC,AL  

;-----  

ADD AH,30h  

MOV PEN,AH ;remainder is pennies  

;-----  

MOV AH,9 ;display result  

LEA DX,START  

INT 21H

```

## Chapter Nine

```
;-----  
MOV AH,4CH  
INT 21H  
  
MAIN ENDP  
INCLUDE C:\TEST\INDEC.ASM  
  
END MAIN
```

sample execution for Q10

```
C:\TEST>ch9q10  
?99  
1 Half 1 Quarter 2 Dimes 0 Nickels 4 Pennies  
C:\TEST>ch9q10  
?25  
0 Half 1 Quarter 0 Dimes 0 Nickels 0 Pennies  
C:\TEST>ch9q10  
?4  
0 Half 0 Quarter 0 Dimes 0 Nickels 4 Pennies  
C:\TEST>ch9q10  
?100  
2 Half 0 Quarter 0 Dimes 0 Nickels 0 Pennies  
C:\TEST>ch9q10  
?135  
2 Half 1 Quarter 1 Dimes 0 Nickels 0 Pennies  
C:\TEST>ch9q10  
?45  
0 Half 1 Quarter 2 Dimes 0 Nickels 0 Pennies  
C:\TEST>ch9q10  
?15  
0 Half 0 Quarter 1 Dimes 1 Nickels 0 Pennies
```

## **Chapter Nine**

(11)

```

.MODEL SMALL
.DATA
M1 DB 0AH,0DH,'enter M : $'
M2 DB 0AH,0DH,'enter N : $'
M3 DB 0AH,0DH,'RESULT IS    $'

.CODE
MAIN PROC

MOV AX,@DATA
MOV DS,AX
    MOV AH,9
    LEA DX,M1
    INT 21H
    ;-----
    CALL INDEC      ;enter M
    MOV BX,AX        ; M in BX
    MOV AH,9
    LEA DX,M2
    INT 21H
    CALL INDEC;      read N in AX
    PUSH AX
    ;-----
    MOV AH,9
    LEA DX,M3
    INT 21H
    MOV AH,2          ;print ". "
    MOV DL,'!'

```

## **Chapter Nine**

```

INT 21H
;-----

POP AX
XCHG AX,BX      ;AX is M BX is N

WHILE_:    MOV CX,10
           MOV DX,0
           IMUL CX      ;AX=AX*10
           DIV BX

           CALL OUTDEC

           MOV AX,DX      ;Replace M by R
           CMP DX,0
           JNE WHILE_

;-----

;-----MOV AH,4CH
INT 21H

MAIN ENDP
INCLUDE C:\TEST\INDEC.ASM
INCLUDE C:\TEST\OUTDEC.ASM

END MAIN

```

## Chapter Nine

Note: don't use numbers produce infinit result

e.g : 5/6 , 88/90 ...

sample execution for Q11

```
Administrator: Amer Al-khsabah
C:\TEST>ch9q11

enter M : ?3
enter N : ?4
RESULT IS      .75
C:\TEST>ch9q11

enter M : ?44
enter N : ?50
RESULT IS      .88
C:\TEST>ch9q11

enter M : ?122
enter N : ?125
RESULT IS      .976
C:\TEST>
```

## **Chapter Nine**

(18)

```

.MODEL SMALL
.DATA
M1 DB 0AH,0DH,'ENTER M $'
M2 DB 0AH,0DH,'ENTER N $'
M3 DB 0AH,0DH,'GCD IS $'

.CODE
MAIN PROC
MOV AX,@DATA
MOV DS,AX
;-----
    MOV AH,9
    LEA DX,M1
    INT 21H
    CALL INDEC      ;read M
    PUSH AX          ;save M
    MOV AH,9
    LEA DX,M2
    INT 21H
    CALL INDEC      ;read N
    PUSH AX          ;save N
;-----
    POP BX
    POP AX
L1: MOV DX,0
    DIV BX      ;M/N
    CMP DX,0
    JE GCD_FOUND ;BX IS N

```

## Chapter Nine

```
MOV AX,BX      ;replace M by N
MOV BX,DX      ;replace N by R
JMP L1

GCD_FOUND:
    MOV AH,9
    LEA DX,M3
    INT 21H
    MOV AX,BX
    CALL OUTDEC

;-----
MOV AH,4CH
INT 21H
MAIN ENDP

INCLUDE C:\TEST\OUTDEC.ASM
INCLUDE C:\TEST\INDEC.ASM
END MAIN
```

## Chapter Nine

sample execution for Q12

```
Administrator: MR.AMER.K@HOTMAIL.COM
ENTER M    ?65
ENTER N    ?4
GCD IS  1
C:\TEST>CH9Q12

ENTER M    ?99
ENTER N    ?3
GCD IS  3
C:\TEST>CH9Q12

ENTER M    ?33
ENTER N    ?5
GCD IS  1
C:\TEST>CH9Q12

ENTER M    ?762
ENTER N    ?23
GCD IS  1
C:\TEST>CH9Q12

ENTER M    ?675
ENTER N    ?3
GCD IS  3
C:\TEST>
```

## Chapter Ten

( 1 )

- |            |                  |
|------------|------------------|
| a) legal   | DI=1500h         |
| b) legal   | DI=0200h         |
| c) legal   | AX=0650h         |
| d) legal   | BX=0E00h         |
| e) legal   | BX=2000h         |
| f) illegal | memory-to-memory |
| g) illegal | []must be 16-bit |
| h) illegal | different size   |
| i) legal   | AX=0300h         |
- 

(2)

- |            |                |
|------------|----------------|
| a) legal   | AH=01h         |
| b) legal   | AX=0504        |
| c) legal   | AX='BA'        |
| d) illegal | different size |
| e) legal   | AH='A'         |
- 

(3)

- a)
- ```

MOV  BP,SP
MOV  [BP],0
MOV  [BP+2],0

```

## Chapter Ten

b)

.DATA

ST\_ARR DW 5 DUP(?)

.CODE

MOV BX,0

MOV CX,5

MOV BP,SP

L1: MOV AX,WORD PTR [BP]

MOV WORD PTR ST\_ARR[BX] , AX

ADD BP,2

ADD BX,2

LOOP L1

## Appendix A

# How To Run A Program

Creating and running a program are done in four simple steps:

### **Step # 1 :**

- Use a text editor or word processor to create a source file.
- After that you should save the program with a name and extension \*.ASM
- You must know where you save the file in which folder in which drive (i.e : know the path of the file)

### **Step # 2:**

- Use an assembler (MASM or TASM) to create a machine language ... (i.e : create the object file with extension \*.OBJ)
- You can do this by using the command (if we suppose work on TASM) **tasm** followed by the name of the source file with its extension... knowing that you are already in path of file (ex: **tasm page1.ASM**)
- After doing this assembler do its work and display some info. Like copy right and most important thing error messages and they are located.
- By checking no errors you can continue else you must go back to source file and fix the error.

## Appendix A

### **Step # 3:**

- use the link program to link one or more object files to create a Run file with extension \*.EXE
- you can do this by using command ... *tlink* with name of file , no need to include the extension here... (ex: *tlink page1*)

### **Step # 4:**

- here the executable file is created
- you can run it by typing only the name of file ... (ex: *page1*)

here we'll see partical example explain the four steps.....

## Appendix A

### Example showing run program in DOS

#### **Step # 1:**

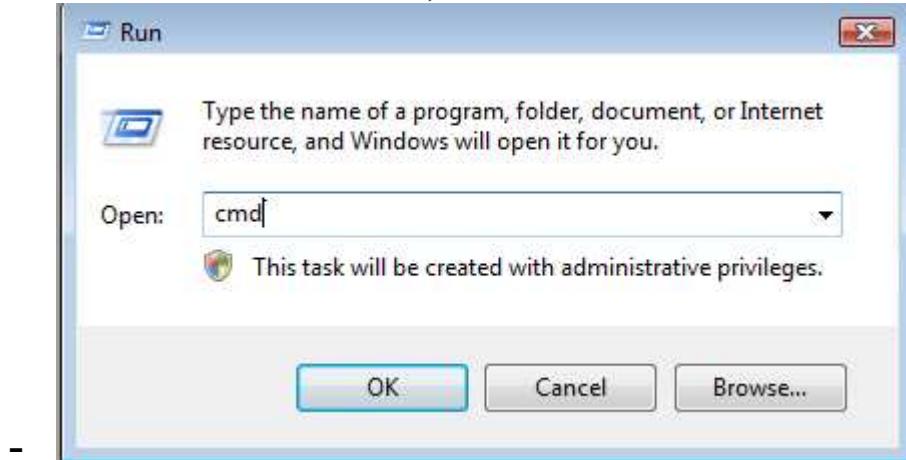
Write the code of program by using *notepad* editor

Save the file with name student.ASM in derive *C:* inside folder its name *test*

(the file save in path *c:\test\student.asm*)

#### **Step # 2 :**

- Open command prompt (you can open it by typing *cmd* in run window)



## Appendix A

- go back to drive c:\  
by use the command *cd\*

```
C:\ Administrator: viprus
C:\Users\Amer>cd\
C:\>-
```

- now enter to folder that has the source file  
by use the command *cd test*

```
C:\ Administrator: viprus
C:\Users\Amer>cd\
C:\>cd test
C:\TEST>-
```

- the file student is visual to us , we can assemble it by  
*tasm student.ASM*

```
C:\TEST>tasm student.ASM
Turbo Assembler Version 4.0 Copyright (c) 1988, 1993 Borland International

Assembling file: student.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 404k
```

Error msg  
No error

If no error you can continue by link

## Appendix A

### **Step # 3:**

Use *tlink* command

```
C:\TEST>tlink student
Turbo Link Version 7.00 Copyright (c) 1987, 1994 Borland International
```

### **Step # 4:**

Now you can run the program by typing only its name or with the extesion \*.exe

```
C:\TEST>student
C:\TEST>student.exe
```

After this if you open the folder test in c: drive you will see 4-files ,

Student.ASM  
Student.OBJ  
Student.MAP  
Student.EXE

source file  
object file produced by tasm  
contain segment addresses  
run program (you can run it by direct click on it )

## Appendix B

# Some Useful Procedures

***Input in Decimal***

INDEC PROC

;READ NUMBER IN RANGE -32768 TO 32767

;input :none

;output :AX =binary equivalent of number

    PUSH BX

    PUSH CX

    PUSH DX

;print prompt

@BEGIN:

    MOV AH,2

    MOV DL,'?'

    INT 21H ;print '?'

;total =0

    XOR BX,BX ;BX hold total

;negative =false

    XOR CX,CX ;CX hold sign

;read char.

    MOV AH,1

    INT 21H

;case char. of

    CMP AL,'-' ;minus sign

    JE @MINUS ;yes, set sign

    CMP AL,'+' ;plus sign

    JE @PLUS ;yes, get another char.

    JMP @REPEAT2 ;start processing char.

## Appendix B

@MINUS: MOV CX,1

@PLUS: INT 21H

;end case

@REPEAT2:

;if char. is between '0' and '9'

CMP AL,'0' ;char >='0'?

JNGE @NOT\_DIGIT ;illegal char.

CMP AL,'9' ;char<='9' ?

JNLE @NOT\_DIGIT

;then convert char to digit

AND AX,000FH

PUSH AX

;total =total \*10 +digit

MOV AX,10

MUL BX

POP BX

ADD BX,AX

;read char

MOV AH,1

INT 21H

CMP AL,0DH ;CR

JNE @REPEAT2

;until CR

MOV AX,BX

;if negative

OR CX,CX

JE @EXIT

NEG AX

;end if

@EXIT: POP DX

POP CX

## Appendix B

POP BX

RET

;here if illegal char entered

@NOT\_DIGIT:

MOV AH,2

MOV DL,0DH

INT 21H

MOV DL,0AH

INT 21H

JMP @BEGIN

INDEC ENDP

## Appendix B

***Input in Decimal  
With overflow check***

```

INDEC2 PROC
;READ NUMBER IN RANGE -32768 TO 32767
;input :none
;output :AX =binary equivalent of number
    PUSH BX
    PUSH CX
    PUSH DX

;print prompt
@BEGIN:
    MOV AH,2
    MOV DL,'?'
    INT 21H      ;print '?'
;total =0
    XOR BX,BX    ;BX hold total
;negative =false
    XOR CX,CX    ;CX hold sign
;read char.
    MOV AH,1
    INT 21H
;case char. of
    CMP AL,'-'
    JE @MINUS   ;minus sign
    CMP AL,'+'
    JE @PLUS    ;plus sign
    JMP @REPEAT2 ;start processing char.

@MINUS: MOV CX,1

```

## Appendix B

```

@PLUS: INT 21H
;end case
@REPEAT2:
;if char. is between '0' and '9'
    CMP AL,'0'      ;char >='0'?
    JNGE @NOT_DIGIT ;illegal char.
    CMP AL,'9'      ;char<='9' ?
    JNLE @NOT_DIGIT
;then convert char to digit
    AND AX,000FH
    PUSH AX
;total =total *10 +digit
    MOV AX,10
    MUL BX
;-----
    CMP DX,0
    JNE @NOT_DIGIT
;-----
    POP BX
    ADD BX,AX
;-----
    JC @NOT_DIGIT
;-----
;read char
    MOV AH,1
    INT 21H
    CMP AL,0DH ;CR
    JNE @REPEAT2
;until CR
    MOV AX,BX
;if negative
    OR CX,CX

```

## Appendix B

JE @EXIT

NEG AX

;end if

@EXIT: POP DX

POP CX

POP BX

RET

;here if illegal char entered

@NOT\_DIGIT:

MOV AH,2

MOV DL,0DH

INT 21H

MOV DL,0AH

INT 21H

JMP @BEGIN

INDEC2 ENDP

## Appendix B

### *Output in Decimal*

OUTDEC PROC

```
PUSH AX
PUSH BX
PUSH CX
PUSH DX
OR AX,AX
JGE @END_IF1
```

```
PUSH AX
MOV DL,'-'
MOV AH,2
INT 21H
```

```
POP AX
NEG AX
```

@END\_IF1:

```
XOR CX,CX
MOV BX,10D
```

@REPEAT1:

```
XOR DX,DX
DIV BX
PUSH DX
INC CX
```

## Appendix B

```
OR AX,AX  
JNE @REPEAT1  
  
MOV AH,2  
@PRINT_LOOP:  
    POP DX  
    OR DL,30H  
    INT 21H  
  
LOOP @PRINT_LOOP  
    POP DX  
    POP CX  
    POP BX  
    POP AX  
RET  
  
OUTDEC ENDP
```

## Appendix B



INBINARY PROC

```
XOR     BX,BX  
MOV     AH,1  
INT    21H
```

WHILE\_:

```
CMP AL,0DH  
JE END_WHILE  
AND AL,0FH  
SHL BX,1  
OR  BL,AL  
INT 21H  
JMP WHILE_
```

END\_WHILE:

RET

INBINARY ENDP

## Appendix B

*Output in Binary*

OUTBINARY PROC

;input AX

MOV BX,AX

MOV CX,16

L0: ROL BX,1

JC ONE

MOV AH,2

MOV DL,'0'

INT 21H

JMP L1

ONE:

MOV AH,2

MOV DL,'1'

INT 21H

L1: LOOP L0

RET

OUTBINARY ENDP

## Appendix B



```
READ_STR PROC NEAR
    PUSH DI          ;di=string to store the data
    PUSH AX          ;cx=# of char read
    XOR BX,BX
    MOV AH,1
    WHILE1:
        INT 21H
        CMP AL,0DH
        JE END_WHILE1
        STOSB
        INC BX
        JMP WHILE1
    END_WHILE1:POP AX
                POP DI
                RET
    READ_STR ENDP
```

## Appendix B



```
DISPLAY_STR PROC NEAR
    PUSH AX    ;SI = BEGIN OF STRING
    PUSH SI    ;INPUT CX=# OF CHAR TO DISPLAY
    MOV AH,2
    MOV DL,0AH
    INT 21H
L1: LODSB
    MOV DL,AL
    INT 21H
    LOOP L1
    POP SI
    POP AX
    RET
DISPLAY_STR ENDP
```

# The END

Thanks for using this copy  
Best Wishes

Written by:

Engineer Amer Mohammed Al-khsabah  
[Eng.Amer.K@hotmail.com](mailto:Eng.Amer.K@hotmail.com)

Date : Sun 25/10/2009