BCD ADDITION AND SUBTRACTION

Exp No.: 7 **Name:** S Vishakan

Date: 07-10-2020 **Reg. No:** 18 5001 196

AIM:

To write assembly language programs to perform the following BCD arithmetic operations:

1. BCD Addition.
2. BCD Subtraction.

PROGRAM – 1: BCD ADDITION:

ALGORITHM:

1. Begin.

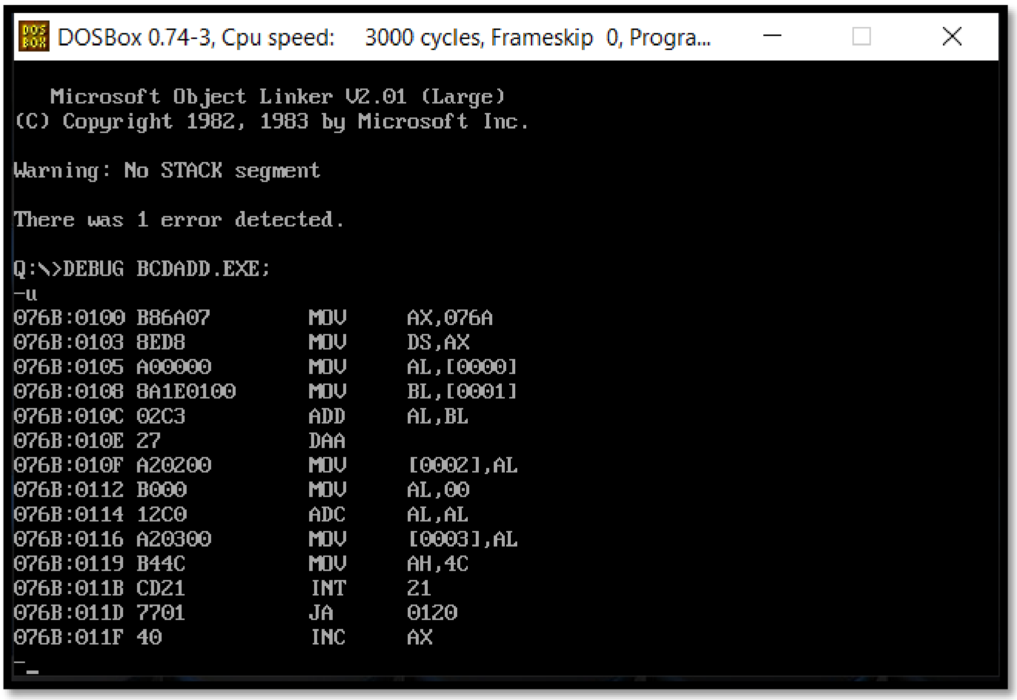
2. Declare the data segment.
3. Initialize data segment with the 2 BCD numbers and variables for storing their sum and carry.
4. Close the data segment.

5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Move the contents of the two numbers num1 and num2 to AL and BL register.
10. Add them and store the value in AL.
11. Move the contents of AL to sum.
12. Perform decimal adjust after addition on AL to get BCD result (HEX to BCD)
13. Check if the above adjustment produced a carry.
    a. If carry was produced, set the variable carry to 1.
    b. Else, continue.
14. Transfer the adjusted addition result to the variable sum.
15. Introduce an interrupt for safe exit. (INT 21h)
16. Close the code segment.

17. End.

PROGRAM	**COMMENTS**
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
num1 db 26h	Stores the first BCD number.
num2 db 99h	Stores the second BCD number.
res db ?	Variable to store the sum of the 2 numbers.
carry db ?	Variable to store the carry of the above sum.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from “data” to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov al, num1	Copy num1 to AL.
mov bl, num2	Copy num2 to BL.
mov cl, 00h	Clear CL register.
add al, bl	AL = AL + BL
daa	Adjust HEX result to BCD after subtraction.
jnc resume	If carry was not produced, jump to “resume”.
inc cl	Increment CL register by 1.
resume: mov res, al	Transfer AL contents to variable res.
mov carry, cl	Transfer CL contents to variable carry.
break: mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

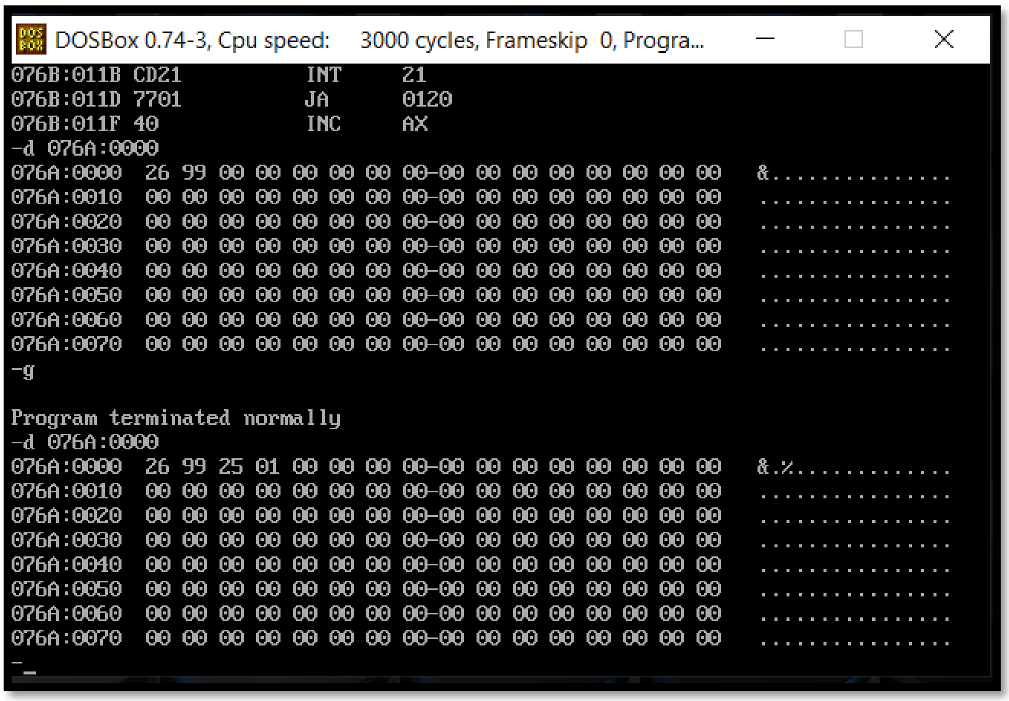
UNASSEMBLED CODE:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG BCDADD.EXE;
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00000        MOV     AL,[0000]
076B:0108 8A1E0100      MOV     BL,[0001]
076B:010C 02C3          ADD     AL,BL
076B:010E 27            DAA
076B:010F A20200        MOV     [0002],AL
076B:0112 B000          MOV     AL,00
076B:0114 12C0          ADC     AL,AL
076B:0116 A20300        MOV     [0003],AL
076B:0119 B44C          MOV     AH,4C
076B:011B CD21          INT     21
076B:011D 7701          JA      0120
076B:011F 40            INC     AX
-_
```

SAMPLE I/O SNAPSHOT:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
076B:011B CD21          INT     21
076B:011D 7701          JA      0120
076B:011F 40            INC     AX
-d 076A:0000
076A:0000  26 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00   &...............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076A:0000
076A:0000  26 99 25 01 00 00 00 00-00 00 00 00 00 00 00 00   &.%.............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

PROGRAM – 2: BCD SUBTRACTION:

ALGORITHM:

1. Begin.

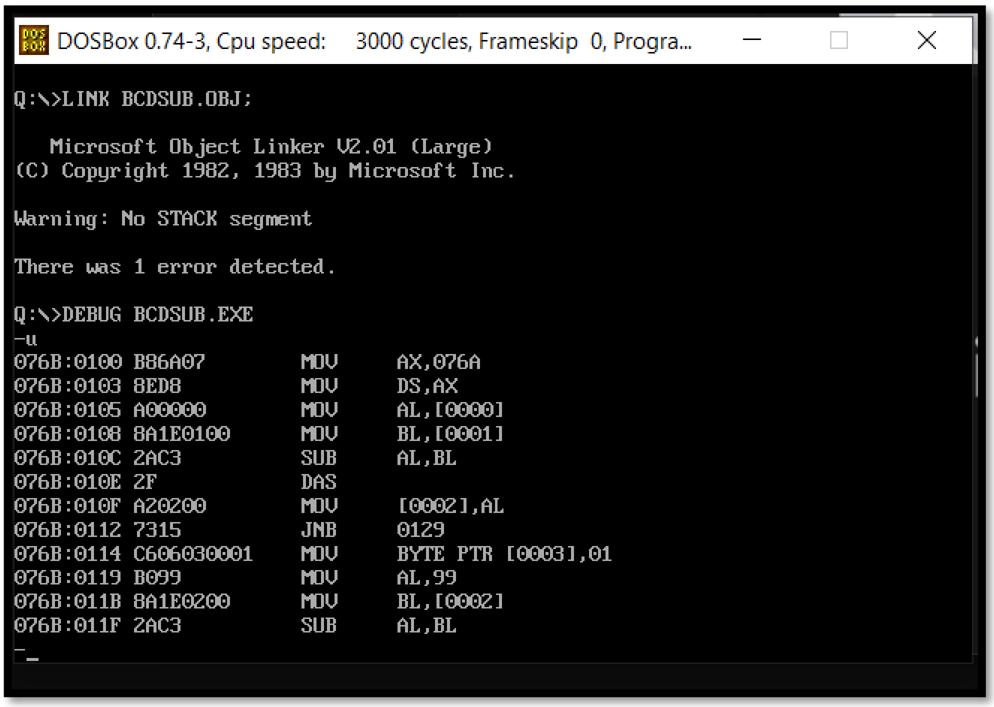
2. Declare the data segment.
3. Initialize data segment with the 2 BCD numbers and variables for storing their difference (diff) and sign.
4. Close the data segment.

5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Move the contents of the two numbers num1 and num2 to AL and BL register.
10. Subtract them and store the value in AL.
11. Transfer the contents of AL to diff.
12. If carry flag is set: (Performing 10’s complement)
    a. Set sign as 01h.
    b. Move the contents of diff to BL register.
    c. Move 99h to AL register.
    d. Subtract BL from AL and store the value in AL register.
    e. Move 01h to BL register.
    f. Add AL and BL.
    g. Perform decimal adjust on the addition in AL. (HEX to BCD).
    h. Transfer the contents of AL to diff.
13. Introduce an interrupt for safe exit. (INT 21h)
14. Close the code segment.

15. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
num1 db 26h	Stores the first BCD number.
num2 db 99h	Stores the second BCD number.
diff db ?	Variable to store the difference of the 2 numbers.
sign db ?	Variable to store the sign of the above difference.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from “data” to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov al, num1	Copy num1 to AL.
mov bl, num2	Copy num2 to BL.
sub al, bl	AL = AL – BL
das	Adjust HEX result to BCD after subtraction.
mov diff, al	Transfer AL contents to diff.
jnc break	If carry was not produced, jump to “break”.
mov sign, 01h	If carry was produced, set sign to 1.
mov al, 99h	Set AL = 99h to perform 9’s complement.
mov bl, diff	Transfer diff to BL.
sub al, bl	AL = 99h – BL (9’s complement)
mov bl, 01h	Set BL = 01h.
add al, bl	AL = AL + BL
daa	AL value is decimal adjusted after addition (HEX to BCD)
mov diff, al	Transfer AL contents to diff.
break: mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

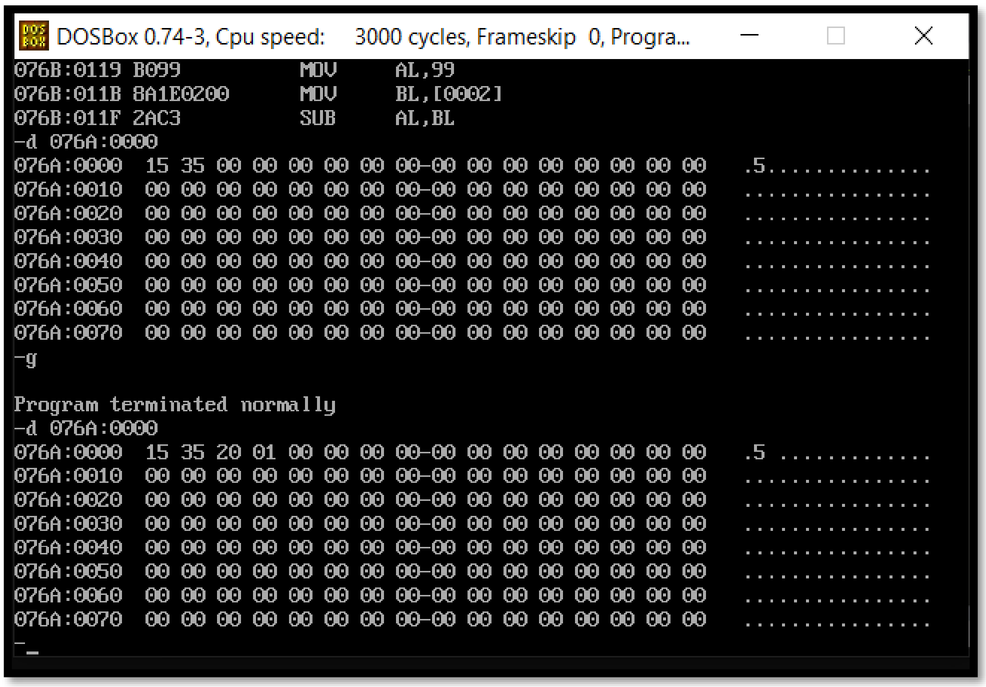
UNASSEMBLED CODE:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
Q:\>LINK BCDSUB.OBJ;

   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG BCDSUB.EXE
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00000        MOV     AL,[0000]
076B:0108 8A1E0100      MOV     BL,[0001]
076B:010C 2AC3          SUB     AL,BL
076B:010E 2F            DAS
076B:010F A20200        MOV     [0002],AL
076B:0112 7315          JNB     0129
076B:0114 C606030001    MOV     BYTE PTR [0003],01
076B:0119 B099          MOV     AL,99
076B:011B 8A1E0200      MOV     BL,[0002]
076B:011F 2AC3          SUB     AL,BL
-_
```

SAMPLE I/O SNAPSHOT:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
076B:0119 B099          MOV     AL,99
076B:011B 8A1E0200      MOV     BL,[0002]
076B:011F 2AC3          SUB     AL,BL
-d 076A:0000
076A:0000  15 35 00 00 00 00 00 00-00 00 00 00 00 00 00 00   .5..............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076A:0000
076A:0000  15 35 20 01 00 00 00 00-00 00 00 00 00 00 00 00   .5 .............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

<u>RESULT:</u>

The assembly level programs were written to perform the above specified BCD arithmetic operations and their output was verified.