CODE CONVERSION

Exp No: 4 Name: Srinithyee S K

Date: 22-09-20 Register Number: 185001166

AIM:

To write assembly language programs to perform the following code conversions.

1. BCD to Hexadecimal Code Conversion
2. Hexadecimal to BCD Code Conversion

PROGRAM 1: BCD TO HEXADECIMAL

ALGORITHM:

1. Begin.

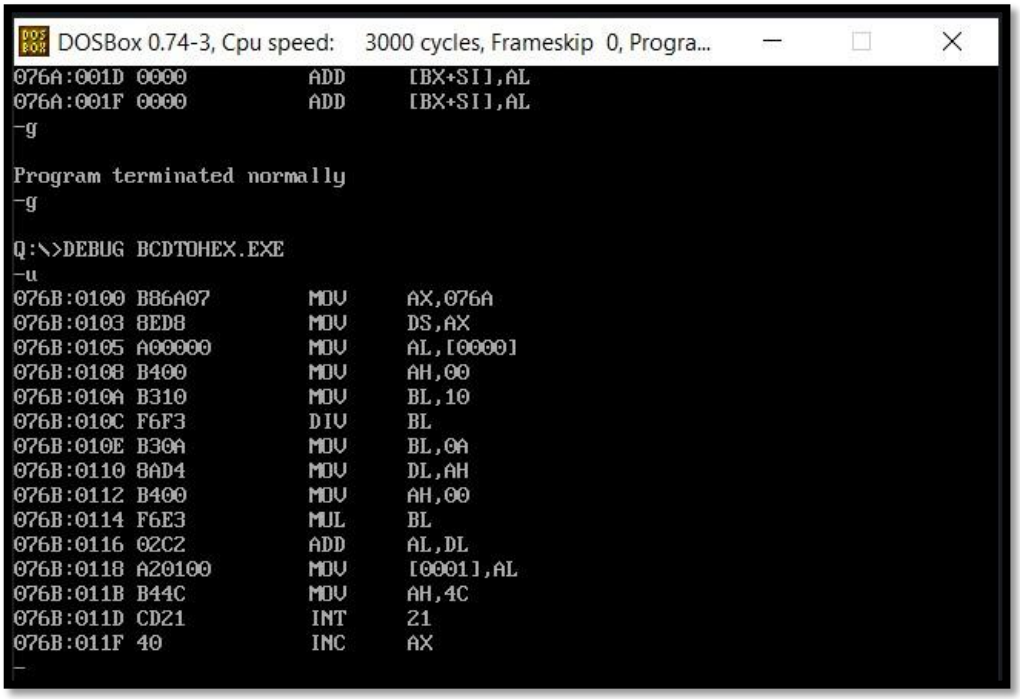
2. Declare the data segment.
3. Initialize the data segment with variables to hold the BCD and HEX values.
4. Close the data segment.

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. Clear AH register.
10. Load the BCD value to AL.
11. Load 10H to BL.
12. Divide the value at AL by BL.
13. Load the LSB at AH to DL.
14. Multiple AL by 10 and add it to value at DL.
15. Move the result at AL to HEX.
16. Introduce an interrupt for safe exit. (INT 21h)
17. Close the code segment.

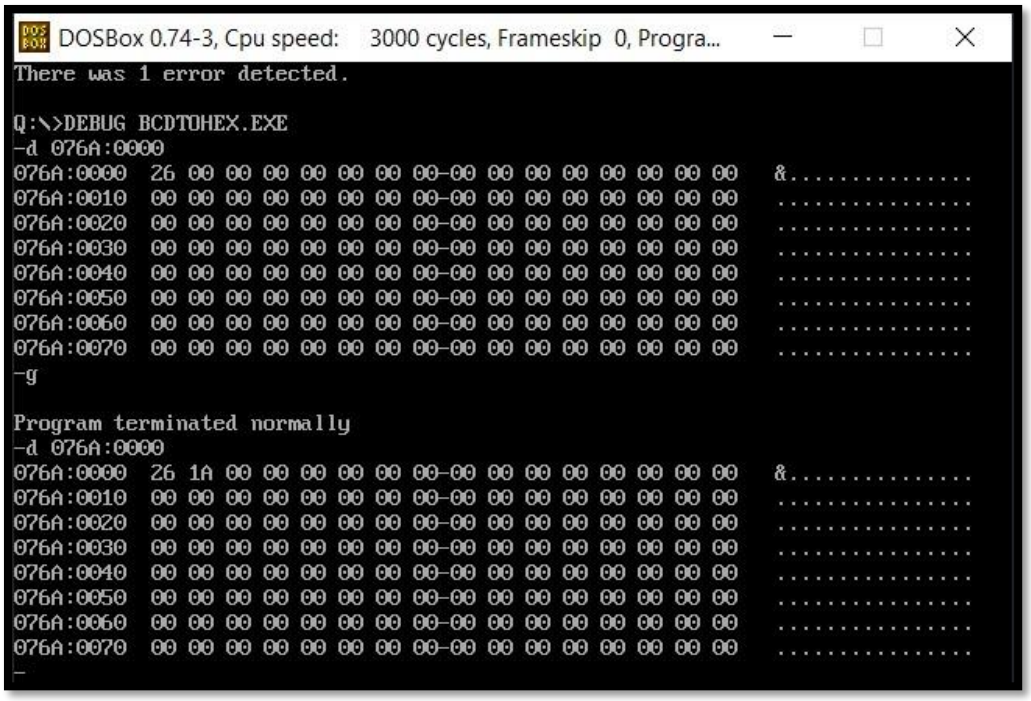
18. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
bcd db 026h	Stores the given BCD value.
hex db ?	Stores the required HEX value.
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, bcd	Transfer the given BCD byte to AL.
mov ah, 00h	Clear AH register.
mov bl, 10h	Transfer 16 to BL.
div bl	Divide AX by BL. (Quotient in AL, Remainder in AH)
mov bl, 0Ah	Transfer 10 to BL.
mov dl, ah	Copy the contents of AH to DL.
mov ah, 00h	Clear AH register.
mul bl	AX = AL * BL (Multiply MSB by 10)
add al, dl	AL = AL + DL (Add LSB to the hex result)
mov hex, al	Store the value in AL as the final HEX converted code.
mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:



SAMPLE I/O SNAPSHOT:



PROGRAM – 2: HEXADECIMAL TO BCD

ALGORITHM:

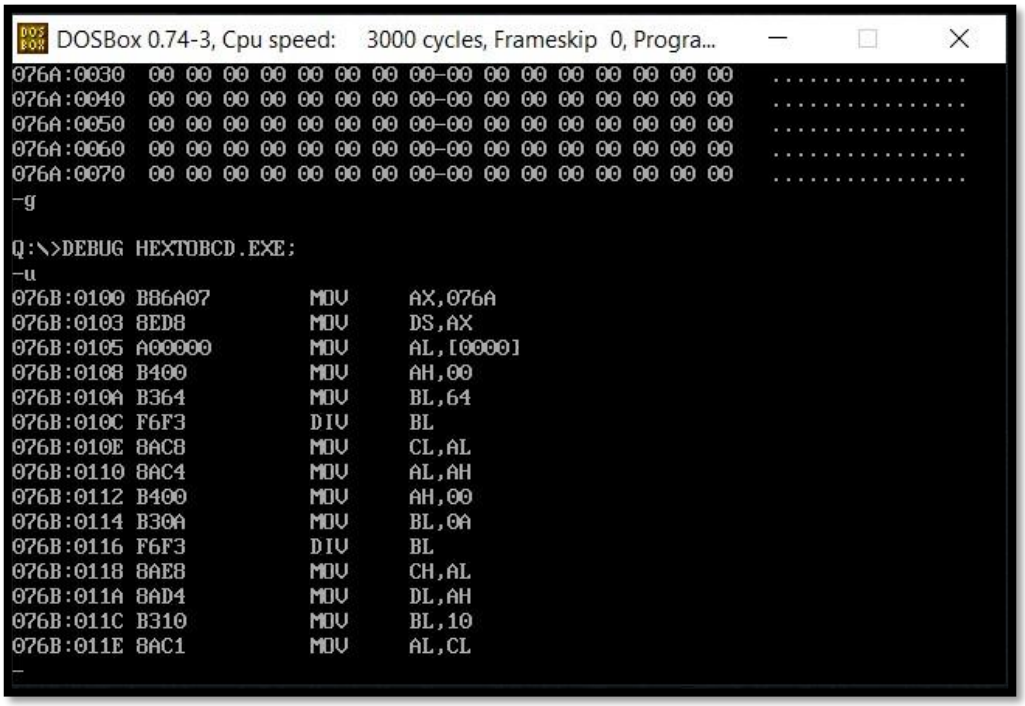
1. Begin.

2. Declare the data segment.
3. Initialize data segment with variables to hold BCD and HEX values.
4. Close the data segment.

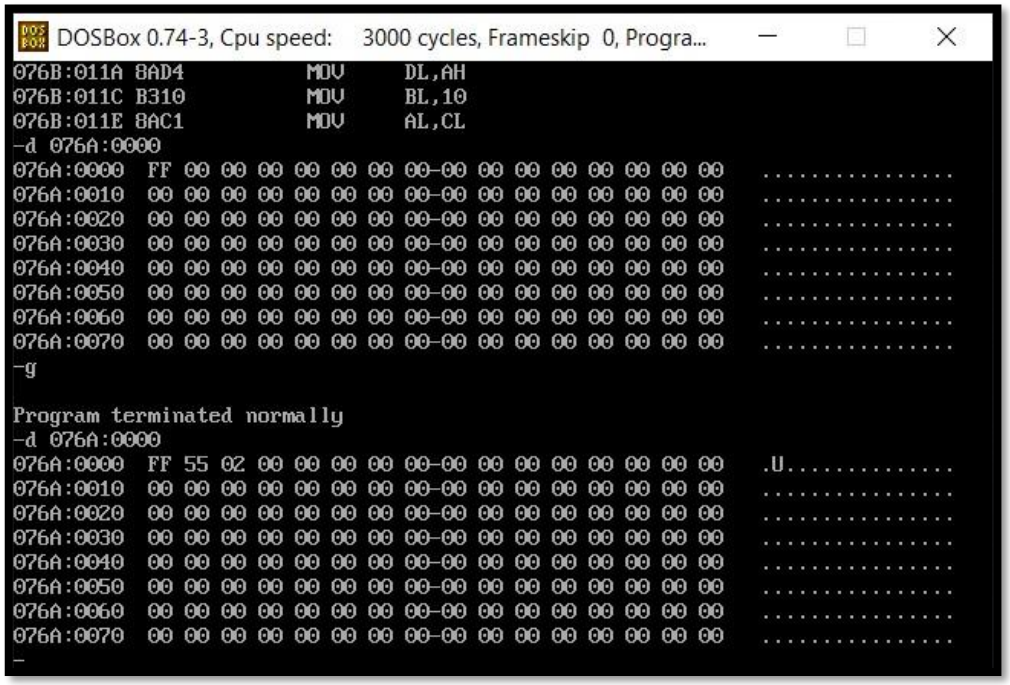
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Clear AH register.
10. Load the Hex value to AL.
11. Load 100(64H) to BL.
12. Divide the value at AX by BL.
13. Move the MSB at AL to CL.
14. Move the LSBs at AH to AL.
15. Clear AH register
16. Load the 10(0AH) to BL.
17. Dive the value at AX by BL.
18. Move the second bit of BCD to CH.
19. Move the LSB of BCD to DL.
20. Apply [CL]*100 + [CH]*10 + [DL] and store the result at AX.
21. Move the result at AX to BCD.
22. Introduce an interrupt for safe exit. (INT 21h)
23. Close the code segment.
24. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
hex db 0FFh	Stores the given HEX value.
bcd db ?	Stores the required BCD value.
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, hex	Transfer the given BCD byte to AL.
mov ah, 00h	Clear AH register.
mov bl, 64h	Transfer 100 to BL.
div bl	Divide AX by BL. (Quotient in AL, Remainder in AH)
mov cl, al	Transfer the quotient to CL register. (MSB of BCD)
mov al, ah	Transfer the remainder to AL register.
mov ah, 00h	Clear AH register.
mov bl, 0Ah	Transfer 10 to BL.
div bl	Divide AX by BL.
mov ch, al	Transfer the quotient to CH register. (2^{nd} MSB of BCD)
mov dl, ah	Transfer the remainder to DL register. (LSB of BCD)
mov bl, 10h	Transfer 16 to BL.
mov al, cl	Transfer the MSB of BCD to AL register.
mul bl	AX = AL * BL (Multiply MSB by 10)
add al, ch	AL = AL + CH (Add 2^{nd} MSB to the BCD result)
mul bl	AX = AL * BL (MSB * 100 + 2^{nd} MSB * 10)
add al, dl	AL = AL + DL (MSB * 100 + 2^{nd} MSB * 10 + LSB)
mov bcd, ax	Store the value in AX as the final BCD converted code.
mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:



SAMPLE I/O SNAPSHOT:



RESULT:

The assembly level programs were written to perform the above specified code conversions and the output was verified.