Terna Engineering College
Computer Engineering Department

Program: Sem V

Course: Microprocessor Lab

Faculty: ARATHI BOYANAPALLI

LAB Manual

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No. 2

A.1 Aim:
Write assembly language program to perform Hex to BCD code conversion.

A.2 Prerequisite:
Knowledge about Hexadecimal, BCD numbers and instruction set of 8086

A.3 Outcome:
After successful completion of this experiment students will be able to

1. Use appropriate instructions to program microprocessors to perform various tasks.
2. Develop the program in assembly/ mixed language for Intel 8086 processor.
3. Demonstrate the execution and debugging of assembly/ mixed language programs.

A.4 Theory

- Hexadecimal Number System:

The hexadecimal numeral system, often shortened to "hex", is a numeral system made up of 16 symbols (base 16). The standard numeral system is called decimal (base 10) and uses ten symbols: 0,1,2,3,4,5,6,7,8,9. Hexadecimal uses the decimal numbers and six extra symbols. There are no numerical symbols that represent values greater than ten, so letters taken from the English alphabet are used, specifically A, B, C, D, E and F. Hexadecimal A = decimal 10, and hexadecimal F = decimal 15. Being a Base-16 system, the hexadecimal numbering system therefore uses 16 (sixteen) different digits with a combination of numbers from 0 through to 15. In other words, there are 16 possible digit symbols.

- Binary-Coded Decimal (BCD)

A binary-coded decimal (BCD) is a type of binary representation for decimal values where each digit is represented by a fixed number of binary bits, usually between four and eight.The norm is four bits, which effectively represent decimal values 0 to 9. This writing format system is used because there is no limit to the size of a number. Four bits can simply be added as another decimal digit, versus real binary representation, which is limited to the usual powers of two, such as 16, 32 or 64 bits.
The following are the 4-bit binary representation of decimal values:
0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001

- Example of Hexadecimal to BCD:

To convert from HEX to BCD, you have to first convert the HEX to Decimal, and then convert the Decimal digits to BCD digits, by converting each Decimal digit to 4 binary digits.

Example: convert Hex 1A2B3C to BCD.

Convert HEX 1A2B3C to Decimal and convert the decimal to BCD.
Hex to decimal of 1A2B3C = 1715004. We have to take this as 01715004 (add a 0 to the start) to get an even number of digits. Now convert each digit to binary.
Decimal to BCD:- 01715004 = 0000 0001 0111 0001 0101 0000 0000 0100.
So Hex to BCD of 1A2B3C:- 0000 0001 0111 0001 0101 0000 0000 0100 BCD.

A.4 Algorithm

For a 4 digit Hex number whose equivalent binary number is to be found i.e. FFFF H. Initially we compare FFFF H with decimal 10000 (2710 H in Hex). If the number is greater than 10,000 we add it to the DH register. Also, we subtract decimal 10,000 from FFFF H, each time comparison is made. Then we compare the number obtained in AX by 1000 decimal. Each time we subtract 1000 decimal from AX and add 1000 decimal to BX. Then we compare numbers obtained in AX by 100 decimals. Each time we subtract 100 decimal from AX and add 100 decimal to BX to obtain BCD equivalent. Then we compare numbers obtained in AX with 10 decimals. Each time we subtract 10 decimal from AX and we add 10 decimal to BX. Finally we add the result in BX with remainder in AX. The final result is present in register DH which contains the 5th bit if present and register AX.

Algorithm For Hex to BCD Conversion -

Step I : Initialize the data segment.

Step II : Initialize BX = 0000 H and DH = 00H.

Step III : Load the number in AX.

Step IV : Compare numbers with 10000 decimal. If below goto step VII else goto step V.

Step V : Subtract 10,000 decimal from AX and add 1 decimal to DH

Step VI : Jump to step IV.

Step VII : Compare the number in AX with 1000, if below goto step X else goto step VIII.

Step VIII : Subtract 1000 decimal from AX and add 1000 decimal to BX.

Step IX : Jump to step VII.

Step X : Compare the number in AX with 100 decimal if below goto step XIII.

Step XI : Subtract 100 decimal from AX and add 100 decimal to BX.

Step XII : Jump to step X.

Step XIII : Compare the number in AX with 10. If below goto step XVI.

Step XIV : Subtract 10 decimal from AX and add 10 decimal to BX.

Step XV : Jump to step XIII.

Step XVI : Add remainder in AX with result in BX.

Step XVII : Display the result in DH and BX.

Step XVIII : Stop.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the ERP or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no ERP access available)

Roll No. : 50	Name: Amey Thakur
Class: TE-Comps B	Batch: B3
Date of Experiment: 03/08/2020	Date of Submission: 03/08/2020
Grade:	

B.1 Software Code written by student:

(Paste your code completed during the 2 hours of practical in the lab here)

Refer B.2

B.2 Input and Output:

➔ **#Program Code for Hex to BCD & BCD to Hex**

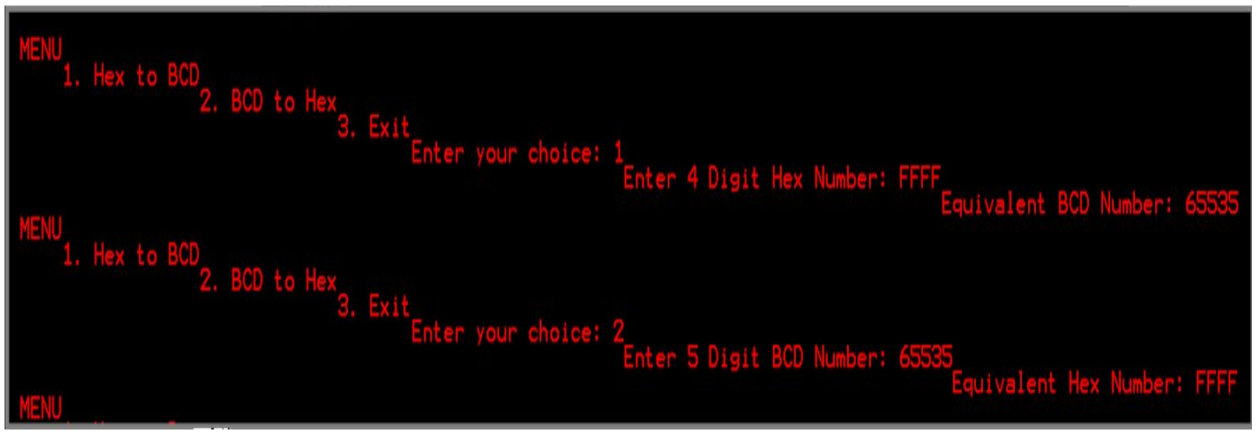
Input -

```
.model small
.data
    menu db 10d,13d," MENU"
       db 10d,"1. Hex to BCD"
       db 10d,"2. BCD to Hex"
       db 10d,"3. Exit"
       db 10d,"Enter your choice: $"
    m1 db 10d,"Enter 4 Digit Hex Number: $"
    m2 db 10d,"Equivalent BCD Number: $"
    num dw 0000h
    arr db 5 dup(0)
    count db 00h

    m3 db 10d,"Enter 5 Digit BCD Number: $"
    m4 db 10d,"Equivalent Hex Number: $"
    num1 dw 10000D
    num2 dw 10d
    num3 dw ?
```

```
.code
        mov ax,@data
        mov ds,ax

main:   lea dx,menu
        mov ah,09H
        int 21h
        mov ah,01h
        int 21h
        cmp al,'1'
        je case1
        cmp al,'2'
        je case2
        jmp exit

exit:   mov ah,4Ch
        int 21h
case1:  call hextobcd
        jmp main
case2:  call bcdtohex
        jmp main

hextobcd proc
        lea dx,m1
        mov ah,09h
        int 21h
        mov ch,04h

loop1:  mov ah,01h
        int 21h
        cmp al,39h
        jbe skip1
        sub al,07h

skip1:  sub al,30h
        mov ah,00h
        add num,ax
        rol num,04
        dec ch
        jnz loop1
        rol num,04
        mov ax,num
        mov dx,0000h
        mov bx,000Ah
        lea si,arr
```

```
loop2:  div bx
        mov [si],dx
        inc si
        inc count
        mov dx,0000h
        cmp ax,0000h
        jnz loop2

        lea dx,m2
        mov ah,09h
        int 21h
        dec si
        mov ch,05h

loop3:  mov dl,[si]
        cmp dl,09h
        jbe skip2
        add dl,07h

skip2:  add dl,30h
        mov ah,02h
        int 21h
        dec si
        dec ch
        jnz loop3
        ret
endp
bcdtohex proc
        lea dx,m3
        mov ah,09h
        int 21h
        mov bx,0000h
        lea si,count
        mov dl,05h
        mov [si],dl

loop4:  mov ah,01h
        int 21h
        cmp al,39h
        jbe skip3
        sub al,07h

skip3:  sub al,30h
        mov ah,00h
        mul num1
        add bx,ax
        mov dx,0000h
        mov ax,num1
        div num2
        mov num1,ax
        dec count
        jnz loop4
        lea dx,m4
        mov ah,09h
        int 21h
        mov ch,04h

loop5:  rol bx,04h
        mov num3,bx
        and bx,000Fh
        cmp bl,09h
        jbe skip4
        add bl,07h

skip4:  add bl,30h
        mov dl,bl
        mov ah,02h
        int 21h
        mov bx,num3
        dec ch
        jnz loop5
        ret
endp
end
```

Output -



```
MENU
    1. Hex to BCD
                 2. BCD to Hex
                              3. Exit
                                     Enter your choice: 1
                                                        Enter 4 Digit Hex Number: FFFF
                                                                                      Equivalent BCD Number: 65535
MENU
    1. Hex to BCD
                 2. BCD to Hex
                              3. Exit
                                     Enter your choice: 2
                                                        Enter 5 Digit BCD Number: 65535
                                                                                       Equivalent Hex Number: FFFF
MENU
```

B.3 Observations and learning:

(Students are expected to comment on the output obtained with clear observations and learning for each task/ sub part assigned)

- 8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of a powerful instruction set, which provides operations like multiplication and division easily.
- It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for systems having multiple processors and Minimum mode is suitable for systems having a single processor.

B.4 Conclusion:

*(**Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)***

We successfully learned assembly language programs to perform Hex to BCD code conversion.

B.5 Question of Curiosity

Q1. List out and explain BCD and ASCII Arithmetic instruction
Ans:

➔ BCD Arithmetic instruction

1. BCD addition
2. BCD subtraction
3. BCD multiplication
4. BCD division
5. BCD negate

1. BCD addition -
    - BCD adder A 4-bit binary adder that is capable of adding two 4-bit words having a BCD (binary-coded decimal) format. The result of the addition is a BCD-format 4-bit output word, representing the decimal sum of the addend and augend, and a carry that is generated if this sum exceeds a decimal value of 9.

2. BCD subtraction -
    - At first the decimal equivalent of the given Binary Coded Decimal (BCD) codes are found out.
    - Then the 9's complement of the subtrahend is done and then that result is added to the number from which the subtraction is to be done.
    - If there is any carry bit then the carry bit may be added to the result of the subtraction.

3. BCD multiplication -
    - A BCD-digit multiplier produces a two-BCD digit product from two input BCD digits.

4. BCD division -
    - BCD division is easily achievable by repeatedly adding the divisor to itself and counting the iterations required for the sum to equal the dividend. The resulting count yields the quotient, while the difference between the sum and dividend provides the remainder

➔ ASCII Arithmetic instruction
1. AAA (ASCII Adjust after Addition)
2. AAD (ASCII Adjust before Division)
3. AAM (ASCII Adjust after Multiplication)
4. AAS (ASCII Adjust after Subtraction)

1. AAA Instruction -
- Adjust the sum of two unpacked BCD values to create an unpacked BCD result.
- The AL register is the implied source and destination operand for this instruction.
- The AAA instruction is only useful when it follows an ADD instruction that adds (binary addition) two unpacked BCD values.
- The AAA instruction then adjusts the contents of AL register to contain the correct 1-digit unpacked BCD result.
- If the addition produces a decimal carry, the AH register is incremented by 1, and the CF and AF flags are set.

2. AAD Instruction -
- Unlike all other adjustment instructions, the AAD instruction appears before a division.
- The AAD instruction requires that the AX register contain a two-digit unpacked BCD number (not ASCII) before executing.
- After adjusting the AX register with AAD, it is divided by an unpacked BCD number to generate a single-digit result in AL with any remainder in AH.
- Example:

```
MOV BL, 9H
MOV AX, 702H
AAD
DIV BL
```

- The above example shows how 72 unpacked BCD is divided by 9 to produce a quotient of 8. The 0702H loaded into the AX register is adjusted by the AAD instruction to 0048H.

3. AAM Instruction -

- Adjust the result of the multiplication of two unpacked BCD values to create a pair of unpacked BCD values.
- The AX register is the implied source and destination operand for this instruction.
- The AAM instruction is only useful when it follows an MUL instruction that multiplies (binary multiplication) two unpacked BCD values and stores a word result in AX registers.

4. AAS Instruction -

- Adjust the result of the subtraction of two unpacked BCD values to create a unpacked BCD result.
- The AL register is the implied source and destination for these instructions.
- The AAS instruction is only useful when it follows a SUB instruction that subtracts (binary subtraction) one unpacked BCD valued from another and stores a byte result in the AL register.
- If the subtraction produces a decimal carry, the AH register is decremented by 1, and the CF and AF flags are set.
- If no decimal carry occurred, the CF and AF flags are cleared, and the AH register is unchanged.

Q2. Write a assembly language program in 8086 to add two 16 bit hexadecimal numbers

Ans: Input -

```
data segment
  a dw 0202h
  b dw 0408h
  c dw ?
data ends

code segment
assume cs:code,ds:data
  start:
  mov ax,data
  mov ds,ax
  mov ax,a
  mov bx,b
  add ax,bx
  mov c,ax
  int 3
  code ends
end start
```