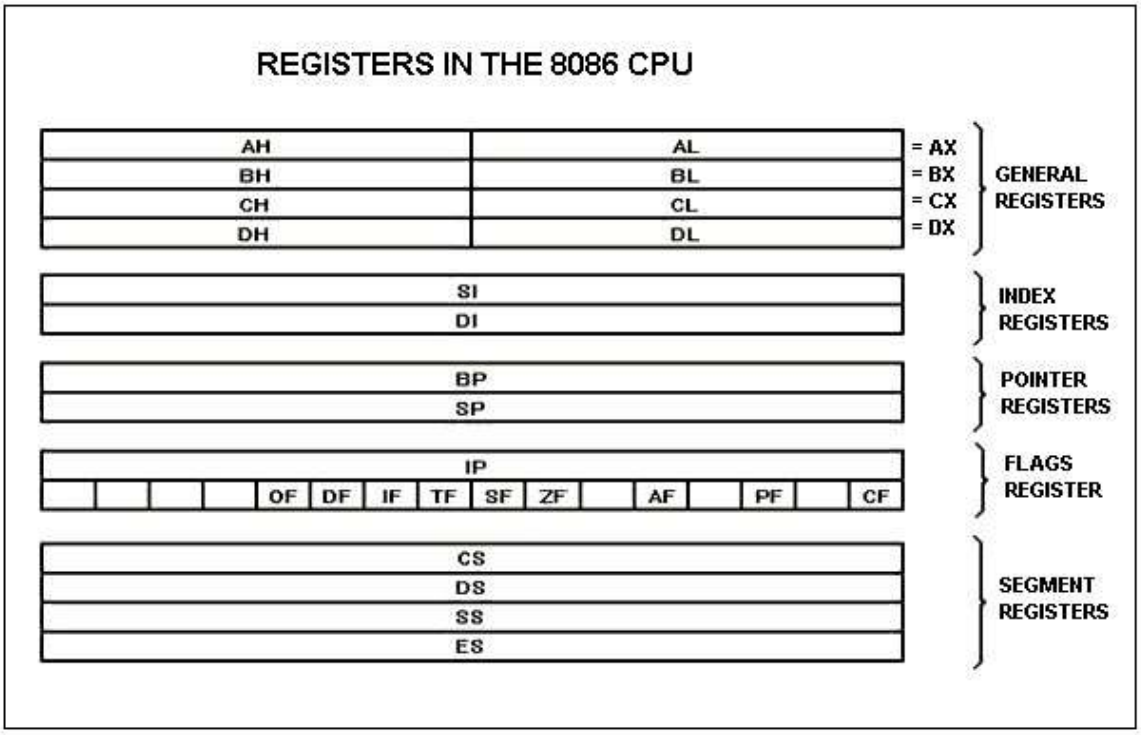
Experiment 03

Learning Objective: Student should be able to Convert HEX to BCD and BCD to HEX using stack in ALP.

Tools: TASM/MASM

Theory:

Software Architecture / Register Set/ Programmer's model of Intel 8086 Microprocessor:



GENERAL PURPOSE REGISTERS

8086 CPU has 8 general purpose registers, each register has its own name:

AX - the accumulator register (divided into AH / AL):

1. Generates shortest machine code
2. Arithmetic, logic and data transfer
3. One number must be in AL or AX
4. Multiplication & Division
5. Input & Output

BX - the base address register (divided into BH / BL).

CX - the count register (divided into CH / CL):

1. Iterative code segments using the LOOP instruction
2. Repetitive operations on strings with the REP command
3. Count (in CL) of bits to shift and rotate

DX - the data register (divided into DH / DL):

1. DX:AX concatenated into 32-bit register for some MUL and DIV operations
2. Specifying ports in some IN and OUT operations

SI - source index register:

1. Can be used for pointer addressing of data
2. Used as source in some string processing instructions
3. Offset address relative to DS

DI - destination index register:

1. Can be used for pointer addressing of data
2. Used as destination in some string processing instructions
3. Offset address relative to ES

BP - base pointer:

1. Primarily used to access parameters passed via the stack
2. Offset address relative to SS

SP - stack pointer:

1. Always points to top item on the stack
2. Offset address relative to SS
3. Always points to word (byte at even address)
4. An empty stack will had SP = FFFEh

SEGMENT REGISTERS

CS - points at the segment containing the current program.

DS - generally points at segment where variables are defined.

ES - extra segment register, it's up to a coder to define its usage.

SS - points at the segment containing the stack.

Flag Register of 8086:

- A flag is a flip-flop which indicates some condition produced by the execution of an instruction or controls certain operations of the EU.
- The Flag Register is a special register associated with the ALU.
- A 16-bit flag register in the EU contains nine active flags.
- Fig. shows the location of the nine flags in the flag register

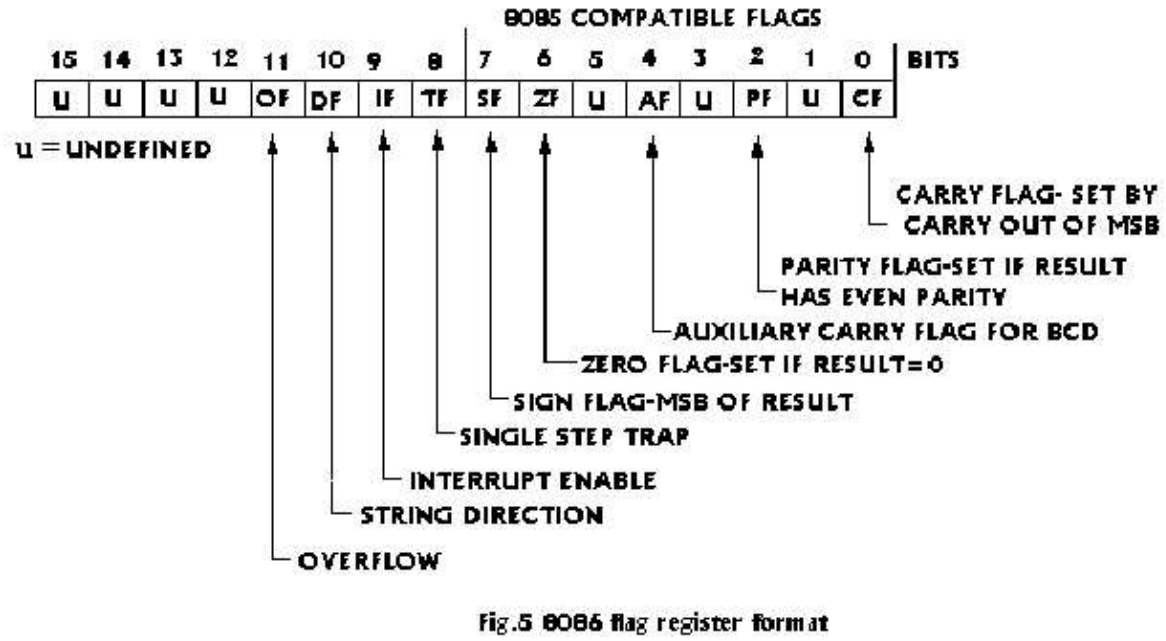


Fig.5 8086 flag register format

Fig. 1.4 Flag Register structure

Flags is a 16-bit register containing 9 1-bit flags:

- Overflow Flag (OF) - set if the result is too large positive number, or is too small negative number to fit into destination operand.
- Direction Flag (DF) - if set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented.
- Interrupt-enable Flag (IF) - setting this bit enables maskable interrupts.

- Single-step Flag (TF) - if set then single-step interrupt will occur after the next instruction.
- Sign Flag (SF) - set if the most significant bit of the result is set.
- Zero Flag (ZF) - set if the result is zero.
- Auxiliary carry Flag (AF) - set if there was a carry from or borrow to bits 0-3 in the AL register.
- Parity Flag (PF) - set if parity (the number of "1" bits) in the low-order byte of the result is even.
- Carry Flag (CF) - set if there was a carry from or borrow to the most significant bit during last result calculation.

Procedure to Convert 4 digit Hex number to its equivalent BCD number.

We have 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 number is greater than 10,000 we add it to 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 number 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 number obtained in AX with 10 decimal. 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 with contains the 5th bit if present and register AX. Display the result.

Algorithm:

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 number 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 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 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.

Application: Conversion of HEX to BCD and BCD to HEX.

Design:

```
.model small
.stack
.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

.startup
        jmp main
endp
end
```

Output:

```
    MENU
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 1
Enter 4 Digit Hex Number: AAAF
Equivalent BCD Number: 44970
```

```
    MENU
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 2
Enter 5 Digit BCD Number: 18970
Equivalent Hex Number: 4A1A
```

```
    MENU
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 3


Program successfully executed !
Press any key to continue.
```

Result and Discussion:

1. We have learned about PUSH and POP instructions.
2. We have also learned to use stack for converting HEX to BCD and vice versa.

Learning Outcomes: The student should have the ability to

LO1: Draw and explain the format of PUSH and POP instructions.
LO2: Explain the concept of Number systems.
LO3: Apply stack instructions to convert HEX to BCD and BCD to HEX.

Course Outcomes: Upon completion of the course students will be able to make use of instructions of 8086 to build assembly and Mixed language programs.

Conclusion: We have learned about the PUSH and POP instruction. A program was implemented which uses stack to convert HEX to BCD.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				