CODE CONVERSIONS
LECTURE 1

1. BCD to Binary Conversion

A BCD number between 0 and 99 is stored in an R/W memory location called the input buffer (UNBUF). WAP and a conversion subroutine (BSDBIN) to convert the BCD number into its equivalent binary number. Store the result in a memory location defined as Output Buffer(OUTBUF).

```
START:   LXI SP,STACK
         LXI H,INBUF
         LXI B,OUTBUF
         MOV A,M
         CALL BCDBIN
         STAX B
         HLT
BCDBIN:  PUSH B
         PUSH D
         MOV B,A
         ANI 0FH
         MOV C,A
         MOV A,B
         ANI F0H
         JZ BCD1
         RRC
         RRC
         RRC
         RRC
         MOV D,A
         XRA A
```

(CONT.)

```
        MVI E,OAH
SUM:    ADD E
        DCR D
        JNZ SUM
BCD1:   ADD C
        POP D
        POP B
        RET
```

- The main program initializes the stack pointer and two memory indexes. It brings BCD number into the accumulator and passes that parameter into subroutine.
- After returning from the subroutine ,the main program stores the binary equivalent in output buffer memory.
- Subroutine saves the content of BC and DE because these registers are used in the subroutine. The acc contents are not saved because that information is passed on to the subroutine.
- The conversion from BCD to binary is illustrated in subroutine 72 BCD converted to binary.

2. Binary to BCD Conversion

A binary group is stored in memory location BINBYT. Convert the number into BCD, and store each BCD as two unpacked BCD digits in the output buffer. To perform this task, WAP two subroutines: one to supply the powers of ten, and the other to perform the conversion.

```
START:      LXI SP,STACK
            LXI H,BINBYT
            MOV A,M
            CALL PWRTEN
            HLT
PWRTEN:     LXI H,OUTBUF
            MVI,64H
            CALL BINBCD
            MVI B,0AH
            CALL BINBCD
            MOV M,A
            RET
BINBCD:     MVI M,FFH
NXTBUF:     INR M
            SUB B
            JNC NXTBUF
            ADD B
            INX H                                   RET
```

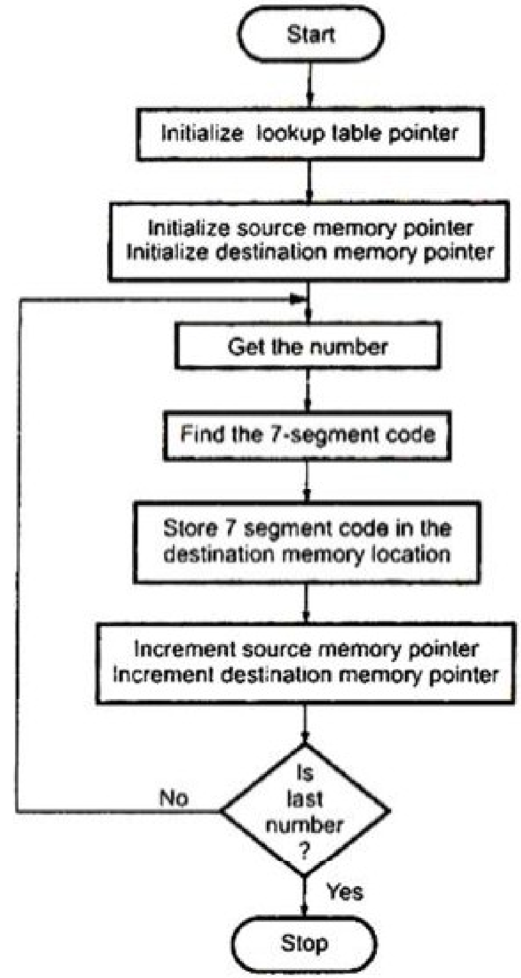
3. BCD to Seven Segment LED CODE CONVERSION

Problem Statement: A set of three packed BCD numbers are stored in memory location starting at XX50H. The seven segment codes of digits 0 to 9 for common cathode LED are stored in memory location starting at XX70H and output buffer is reserved at XX90H.
WAP & two subroutines called UNPAK and LEDCOD to unpack BCD numbers and select an appropriate seven segment code for each digit. The code should be stored in output buffer memory.

```
                LXI SP, 27FFH
                LXI H,XX50H
                MVI D, 03H
                CALL UNPAK
                HLT

UNPAK:     LXI B, BUFFER
NXTBCD:    MOV A,M
                ANI F0H
                RRC
                RRC
                RRC
                RRC
                CALL LEDCOD
                INX B
                MOV A,M
                ANI 0FH
                CALL LEDCOD
                INX B
                INX H
                DCR D
                JNZ NXTBCD
                RET
```

BCD to Seven Segment LED CODE CONVERSION (Cont.)

```
LEDCOD:  PUSH H
         LXI H, CODE
         ADD L
         MOV L, A
         MOV A, M
         STAX B
         POP H
         RET
CODE:    3F        ;Digit 0
         06        ; Digit 1
         5B        ; Digit 2
         4F        ; Digit 3
         66        ; Digit 4
         6D        ; Digit 5
         7D        ; Digit 6
         07        ; Digit 7
         7F        ; Digit 8
         6F        ; Digit 9
         00        ; Invalid Digit
```

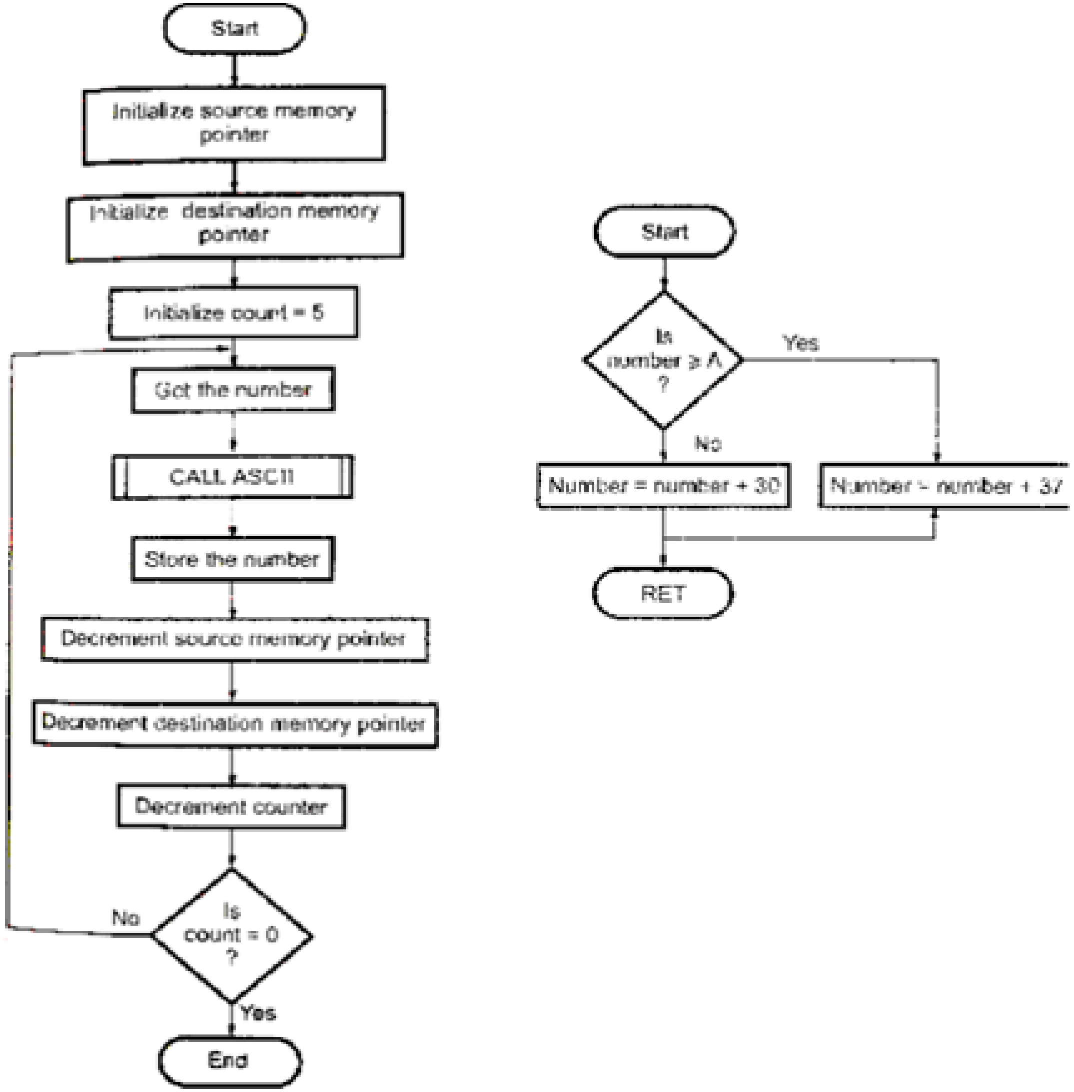


4. Binary to ASCII CODE CONVERSION

Problem statement: WAP to convert the content of 5 memory locations starting from 2000H into ASCII character. Place the result in five memory locations starting from 2200H.

```
     LXI SP, 27FFH
     LXI H, 2000H
     LXI D, 2200H
     MVI C, 05H
X:   MOV A,M
     CALL ASCII
     STAX D
     INX H
     INX D
     DCR C
     JNZ X
     HLT
```

Flowchart for binary to ASCII

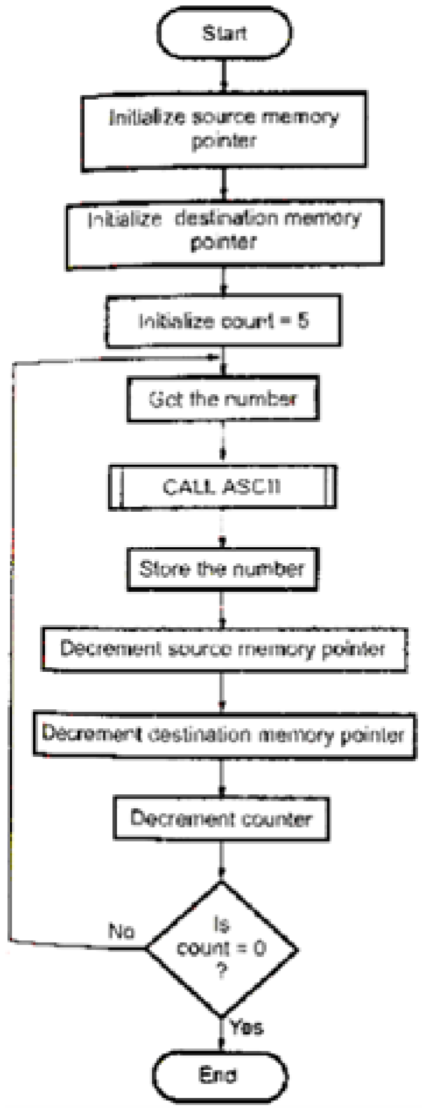


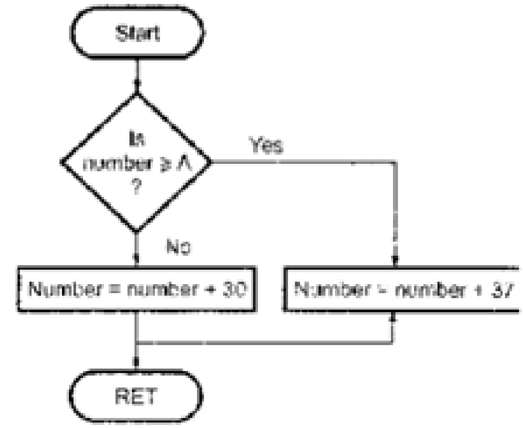
5. ASCII to Binary CODE CONVERSION

Problem statement: WAP to convert the content of 5 memory locations starting from 2000H into Binary code. Place the result in five memory locations starting from 2200H.

```
        LXI SP, 27FFH
        LXI H, 2000H
        LXI D, 2200H
        MVI C, 05H

X:      MOV A,M
        CALL ASCII
        STAX D
        INX H
        INX D
        DCR C
        JNZ X
        HLT

ASCII:CPI 3AH
        JNC Y
        SUI 37H
        JMP Z

Y:      SUI 30H
Z:      RET
```

Flowchart for ASCII to binary

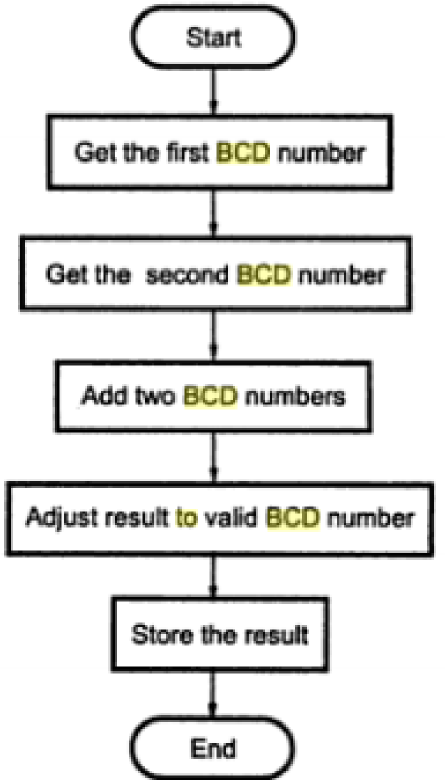




6. BCD ADDITION

Problem statement: Add two 2-digit BCD numbers in memory location 2200H and 2201H and store the result in memory location 2300H.

```
LXI H, 2200H
MOV A,M
INX H
ADD M
DAA
STA 2300H
HLT
```



Flow chart for BCD addition

Example of BCD ADDITION

Problem statement: Add two 4 digits BCD numbers in HL and DE register pairs and store the result in memory locations 2300H and 2301H. Ignore carry after 16bit.

```
MOV A, L
ADD E
DAA
STA 2300H
MOV A, H
ADC D
DAA
STA 2301H
HLT
```

7. BCD SUBTRACTION

SUBTRACTION OF TWO BCD NUMBERS

Problem statement: Subtract the BCD number stored in E register from the number stored in D register.

Process: (i) Find 100's complement of subtrahend
(ii) Add two numbers using BCD addition

```
MVI A, 99H
SUB E
INR A
ADD D
DAA
HLT
```

ADVANCED INSTRUCTIONS

1. **LHLD Address(16 bit)-** This instruction is used to load the contents of memory location given within the instruction into L register and the contents of memory location next to it will be stored in H register.

Example: LHLD 5000H- It will load the contents of memory location 5000H into L register and the contents of memory location 5001H will be stored in H register.

2. **SHLD Address(16 bit)-** This instruction will store the contents of L register into the memory address as specified within the instruction and store the contents of H register into memory location next to it.

Example: SHLD 5000H- This instruction will store the contents of L register into the memory address 5000 and store the contents of H register into memory location 5001.

3. **XCHG-** This instruction is used to exchange the contents of HL register pair with the contents of DE register pair.

4. **XTHL-** This instruction is used to exchange the contents of HL register pair with the contents of top of stack.

5. **SPHL-** This instruction is used to copy the contents of HL register pair into top of stack.

6. **PCHL-** This instruction is used to copy the contents of HL register pair into program counter.

7. **ADC R-** This instruction is used to add the contents of accumulator with the contents of specified register and carry and store the result in accumulator.

8. **ADC M-** This instruction is used to add the contents of accumulator with the contents of memory location as pointed by HL register pair and carry and store the result in accumulator.

9. **ACI Data-** This instruction is used to add the contents of accumulator with the immediate data given within the instruction and carry and store the result in accumulator.

10. **SBB R-** This instruction is used to subtract the contents of specified register from the contents of accumulator and carry and store the result in accumulator.

11. **SBB M-** This instruction is used to subtract the contents of memory location as pointed by HL register pair from the contents of accumulator and carry and store the result in accumulator.

12. **SBI data-** This instruction is used to subtract the contents of immediate data given within the instruction from the contents of accumulator and carry and store the result in accumulator.

8. MULTIPLICATION

A multiplicand is stored in memory location XX50H and multiplier is stored in location XX51H.WAP to transfer the two numbers from memory locations to the HL registers and store the product in the output buffer at XX90H. Write a subroutine to multiply two unsigned numbers placed in registers H and L. Return the result to HL pair.

```
                LXI SP,STACK
                LHLD XX50H
                XCHG
                CALL MLTPLY
                SHLD XX90H
                HLT
MLTPLY:         MOV A,D
                MVI D,00H
                LXI H,0000H
                MVI B,08H
 NXTBIT:        RAR
                JNC NOADD
                DAD D
NOADD:          XCHG
                DAD H
                XCHG
                DCR B
                JNZ NXTBIT                                    RET
```