Programs

- **A table contains ten 8-bit data starting at 8050H. Write an 8085 program to store the sum of odd numbers at 8060H and sum of even numbers at 8070H.**

```
        LXI H,8050H     ;source table
        MVI D,00H       ;register to store sum of odd numbers
        MVI E,00H       ;register to store sum of even numbers
        MVI C,0AH       ;counter
UP:     MOV A,M
        RRC             ;rotate right to check whether LSB=0 or 1(to check even or odd)
        JC ODDSUM       ;if LSB=1(i.e. odd) goto ODDSUM
        RLC             ;otherwise number is even, calculate sum of even number
        ADD E
        MOV E,A
        JMP PASS
ODDSUM:RLC
        ADD D
        MOV D,A
PASS:   INX H
        DCR C
        JNZ UP
        MOV A,D
        STA 8060H               ;store sum of odd numbers at 8060H
        MOV A,E
        STA 8070H               ;store sum of odd numbers at 8070H
        HLT
```

Programs

- **Transfer sixteen data with even parity from location 5270H to 5280H else transfer the data by clearing bit D7 and setting bit D2.**

```
        LXI H,5270H     ;source table
        LXI D, 5280H    ;destination table
        MVI C,10H       ;counter(for 16 numbers)
UP:     MOV A,M
        ADI 00H         ;only arithmetic and logical operation changes the flag
        JPE PASS        ;if data has even parity goto PASS
        ANI 7FH         ;clear bit D7
        ORI 04H         ;set bit D2
PASS:   STAX D
        INX H
        INX D
        DCR C           ;decrease the value of counter
        JNZ UP          ;goto UP until value of C is zero
        HLT
```

Programs

- **Write a program for 8085 to swap bit D3 and D6 of ten numbers stored in memory at 9650H if any number is greater than 70H and less than A0H. Otherwise set D3 and reset D6 of the number stored.**

```
        LXI H,9650H     ;source table
        MVI C,0AH       ;counter
UP:     MOV A,M
        CPI 71H         ;check of number is greater or equal to 71H
        JC PASS         ;if no goto PASS
        CPI A0H         ;check of number is less than A0H
        JNC PASS        ;if no goto PASS
        ANI 48H         ;mask D3 and D6
        JZ PASS1        ;if yes goto PASS1(no need to swap, both bits are zero)
        CPI 48H         ;check if result is 48H
        JZ PASS1        ;if yes goto PASS1(no need to swap, both bits are one)
        MOV A,M
        XRI 48H         ;toggle bit D3 and D6
        MOV M,A
        JMP PASS1
PASS:   ORI 08H         ;set D3 bit     0000 1000
        ANI BFH         ;reset D6 bit   1011 1111
        MOV M,A
PASS1:  INX H
        DCR C
        JNZ UP
        HLT
```

	D7	D6	D5	D4	D3	D2	D1	D0
ANDing	0	1	0	0	1	0	0	0
	0	D6	0	0	D3	0	0	0
IF ANS IS	0	0	0	0	0	0	0	0
IF ANS IS	0	1	0	0	1	0	0	0
IF ANS IS	0	0	0	0	1	0	0	0
IF ANS IS	0	1	0	0	0	0	0	0

If both bits are 0 or 1, no need to swap

If D6=0 and D3=1 OR D6=1 and D3=0; swap the bit by complementing them

Programs

- **Write a program in 8085 to transfer ten 8-bit numbers from one table to another if sum of higher nibble and lower nibble is less than 10H else store 00H in another table.**

```
        LXI H,2050H         ;source table
        LXI B,2060H         ;destination table
        MVI E,0AH ;counter
UP:     MOV A,M
        ANI F0H             ;masking higher nibble
        RRC
        RRC
        RRC
        RRC
        MOV D,A             ; D<- higher nibble
        MOV A,M
        ANI 0FH             ;masking lower nibble A<- lower nibble
        ADD D
        CPI 10H             ;check if sum is less than 10H
        JNC PASS            ;if no goto PASS
        MOV A,M
        JMP DOWN
PASS:   MVI A,00H
DOWN:   STAX B
        INX H
        INX B
        DCR E
        JNZ UP
        HLT
```

`

2050: 24H

```
24H:    0010 0100
ANDing  0000 1111
        0000 0100 == 04H

24H:    0010 0100
ANDing  1111 0000
        0010 0000 == 20H
        0001 0000
        0000 1000
        0000 0100
        0000 0010 == 02H
```

Programs

- **Ten data are stored in memory location starting at 8345H. Write a program to convert BCD number to binary number and store the result in the second table in the memory location starting at 8365H. Make subroutine for conversion process**

```
        LXI H,8345H     ;source table
        LXI B,8365H     ;destination table
        MVI E,0AH       ;counter
UP1:    MOV A,M
        ANI F0H         ;to separate higher nibble from a number
        RRC
        RRC
        RRC
        RRC
        MOV D,A
        MVI A,  00H     ;clear accumulator
UP:     ADI 0AH
        DCR D
        JNZ UP
        MOV D,A
        MOV A,M
        ANI 0FH         ;to separate lower nibble from a number
        ADD D
        STAX B
        INX H
        INX B
        DCR E
        JNZ UP1
        HLT
```

Branching Instructions

- **Call and Return Instructions**
  - These instructions are used to call a subroutine and return from that after the successful execution of that subroutine.
  - Call and return instructions are analogous to function call and return in C/C++
  - These instructions change the program sequence to the location of a subroutine to accomplish a certain task; and to return to the calling program after its execution
  - Two types of Call and Return Instructions
    - Unconditional Call and Return
    - Conditional Call and Return
  - **Unconditional Call and Return**
  - Instruction: **CALL** 16- bit address

    …

    …

    RET

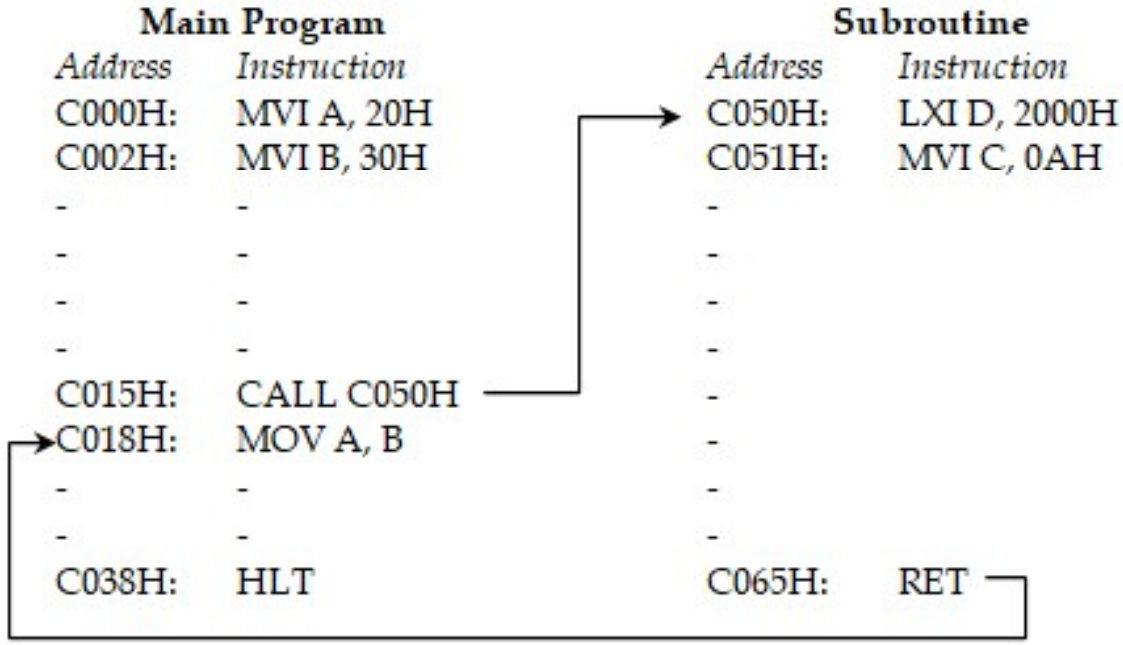
Branching Instructions

- **Call and Return Instructions**
  - **Unconditional Call and Return**
  - Instruction: **CALL** 16- bit address

...

...

RET



Branching Instructions

- **Call and Return Instructions**
  - **Unconditional Call and Return**
  - How **CALL** and **RET** works?
  - For **CALL**
    - Saves the content of PC as a return address into the stack.
    - Loads the PC with new subroutine address.
  - For **RET**
    - Pops the return address from stack.
    - Loads that return address into the PC.

Programs

- **Ten data are stored in memory location starting at 8345H. Write a program to convert BCD number to binary number and store the result in the second table in the memory location starting at 8365H. Make subroutine for conversion process**

```
        LXI B,8365H     ;destination table
        MVI E,0AH       ;counter
UP1:    MOV A,M
        CALL CONV       ;call subroutine
        STAX B
        INX H
        INX B
        DCR E
        JNZ UP1
        HLT
```

```
CONV:   ANI F0H ;to separate higher nibble from a number
        RRC
        RRC
        RRC
        RRC
        MOV D,A
        MVI A, 00H      ;clear accumulator
UP:     ADI 0AH
        DCR D
        JNZ UP
        MOV D,A
        MOV A,M
        ANI 0FH ;to separate lower nibble from a number
        ADD D
        RET
```

Programs

- **Ten data are stored in memory location starting at 8345H. Write a program to convert binary number to BCD number and store the result in the second table in the memory location starting at 8445H.**

```
        LXI H,8345H     ;source table
        LXI B,8445H     ;destination table
UP:     MVI D,00H       ; register to count the no. of subtraction done
        MOV A,M
        CPI 0AH         ;check if the number is less than 0AH
        JC DOWN         ;if yes goto DOWN(no need to convert)
UP1:    SUI 0AH
        INR D
        CPI 0AH
        JC PASS         ;repeat subtraction process until result is less than 0AH
        JMP UP1
PASS:   MOV E,A
        MOV A,D
        RLC
        RLC
        RLC
        RLC
        ADD E
DOWN:   STAX B
        INX H
        INX B
        MOV A,L
        CPI 4FH         ;check value of L with 4FH(for 10 data)
        JNZ UP
        HLT
```

Branching Instructions

- **Call and Return Instructions**
  - **Conditional Call and Return**

Instructions	Comments
CC 16-bit address	Call on carry(CY=1)
CNC 16-bit address	Call on not carry(CY=0)
CZ 16-bit address	Call on zero(Z=1)
CNZ 16-bit address	Call on not zero (Z=0)
CPE 16-bit address	Call on parity even (P=1)
CPO 16-bit address	Call on parity odd (P=0)
CP 16-bit address	Call on plus (S=0)
CM 16-bit address	Call on Minus (S=1)

Instructions	Comments
RC 16-bit address	Return on carry(CY=1)
RNC 16-bit address	Return on not carry(CY=0)
RZ 16-bit address	Return on zero(Z=1)
RNZ 16-bit address	Return on not zero (Z=0)
RPE 16-bit address	Return on parity even (P=1)
RPO 16-bit address	Return on parity odd (P=0)
RP 16-bit address	Return on plus (S=0)
RM 16-bit address	Return on Minus (S=1)

Branching Instructions

- **Restart Instructions**
  - They are one byte call instructions.
  - Whenever a RST N instruction is written, it saves the returning address (content of PC) into the stack and jumps to a pre-specified location

Opcode/Operand	*Restart Address (H)*
RST 0	0000
RST 1	0008
RST 2	0010
RST 3	0018
RST 4	0020
RST 5	0028
RST 6	0030
RST 7	0038

Machine Control Instructions

- These instructions are used to control the mechanism of the microprocessor while executing a program. There are only two instructions **HLT** and **NOP** in this group.
  - **HLT** terminates the program
  - **NOP** does no operation

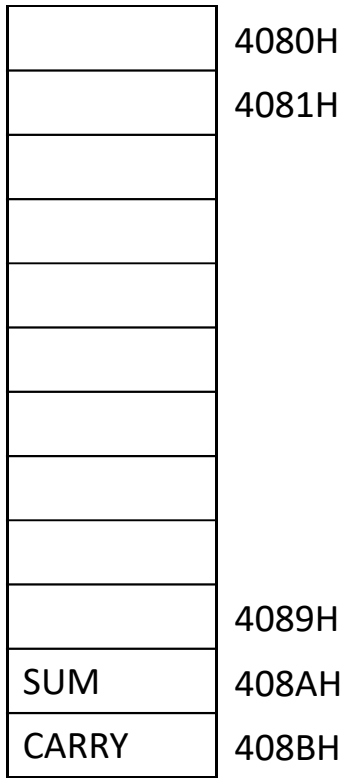
Miscellaneous Instructions

- **DAA(Decimal Adjust Accumulator) Instruction**
  - Used for BCD addition
  - Should be used immediately after add instruction
  - Working:
    - If lower nibble value of greater than 9 or AC is set, it add 6 to the lower nibble
    - If higher nibble value of greater than 9 or CY is set, it add 6 to the higher nibble

Programs

- **Write an 8085 program to add ten BCD numbers stored in the consecutive memory locations starting from 4080H and store the 16-bit result at end of the table.**

```
        LXI H,4080H     ;source table
        MVI C,0AH       ;counter
        MVI D,00H       ;sum register
        MVI E,00H       ;carry register
UP:     MOV A,M
        ADD D           ;A<-A+D
        DAA
        MOV D,A
        JNC PASS ;goto PASS if carry is not generated
        INR E           ;otherwise increase carry register by 1
PASS:   INX H
        DCR C
        JNZ UP
        MOV M, D
        INX H
        MOV M, E
        HLT
```



Miscellaneous Instructions

- **CMC:** Complement Carry (CY <- CY')
- **STC**: Set Carry (CY = 1)
- **EI:** Enable Interrupt
- **DI:** Disable Interrupt
- **SIM:** Set Interrupt Mask
- **RIM:** Read Interrupt Mask

Programs

- **A set of three readings is stored in memory starting at 2050H. Write a program for 8085 to sort the reading in ascending order.**

```
START:  LXI H,2050H     ;source table
        MVI D,00H       ;indicator(D not equal to 00H means sorting process to repeat)
        MVI C,02H       ;counter(for 5 readings)
UP:     MOV A,M
        INX H
        CMP M           ;compare bytes
        JC PASS         ;if A<second byte, do not exchange
        MOV B,M
        MOV M,A
        DCX H
        MOV M,B
        INX H
        MVI D,01H
PASS:   DCR C
        JNZ UP
        MOV A,D
        RRC
        JC START
        HLT
```

Original Value	
2050H	03
2051H	02
2052H	01

First Pass	
2050H	02
2051H	01
2052H	03

Second Pass	
2050H	01
2051H	02
2052H	03

Third Pass	
2050H	01
2051H	02
2052H	03