CS-322 Lab 1

Name: P.V. Sriram

Roll No.: 1801CS37

Q1) 8085 simulator program for Hexadecimal to Decimal number conversion

```
# ORG F000
        LXI H,8000 // Initialize memory pointer
        MVI D,00   // Clear D- reg for Most significant Byte
        XRA A      // Clear Accumulator
        MOV C,M    // Get HEX data

LOOP:   ADI 01     // Count the number one by one
        DAA // Adjust for BCD count
        JNC SKIP   // Jump to SKIP
        INR D      // Increase D

SKIP:   DCR C      // Decrease C register
        JNZ LOOP   // Jump to LOOP
        MOV L,A    // Load the Least Significant Byte
        MOV H,D    // Load the Most Significant Byte
        SHLD 8050  // Store the BCD
        HLT  // Terminate the program

# ORG 8000
# DB 34
```

Input: 34H (Addr – 8000H)

Output: 52 (Addr – 8050H)

Q2) 8085 simulator program for Decimal to Hexadecimal number conversion

```
# ORG 2000H
          LDA 201F   // Load value from 201f to accumulator
          MOV B,A    // Move value from A to B
          ANI 0F     // Bitwise AND of A with 0F
          MOV C,A    // Move A to C
          MOV A,B    // Move B to A
          ANI F0     // Bitwise and of A to F0
          JZ SKIPMULTIPLY
          RRC // Shift right
          RRC
          RRC
          RRC
          MOV D,A    // Move A to D
          XRA A      // Bitwise XOR with A
          MVI E,0A   // Store a value 0A in E

SUM:      ADD D      // Add D to A
          DCR E      // Decrement E  by 1
          JNZ SUM

SKIPMULTIPLY:        ADD C       // Add C to A
          STA 2020   // Store value from A to 2020
          HLT
```

# ORG 201F

# DB 72H

Input: 72 (Addr - 201F)

Output: 48H (Addr - 2020H)

Q3) 8085 simulator program to find the sum of Odd numbers in an input array

```
        LDA 2200
        MOV C,A    // Initialize counter
        LXI H,2201 // Initialize pointer
        MVI E,00   // Sum low = 0
        MOV D,E    // Sum high = 0

BACK:   MOV A,M    // Get the number
        ANI 01H    // Mask Bit 1 to Bit7
        JZ SKIP    // Don't add if number is even
        MOV A,E    // Get the lower byte of sum
        ADD M      // Sum = sum + data
        MOV E,A    // Store result in E register
        JNC SKIP
        INR D      // Add carry to MSB of SUM

SKIP:   INX H      // Increment pointer
        DCR C      // Decrement counter
        JNZ BACK   // Check if counter 0 repeat
```

```
            MOV A,E
            STA 2300   // Store lower byte
            MOV A,D
            STA 2301   // Store higher byte
            HLT  // Terminate program execution
# ORG 2200
# DB 04H, 9AH, 52H, 89H, 3FH
```

Input:

Val	Addr
04H	**(2200H)**
9AH	**(2201H)**
52H	**(2202H)**
89H	**(2203H)**
3FH	**(2204H)**

Output:

C8H (Addr - 2300H)

Q4) 8085 simulator program to find the sum of Even numbers in an input array

```
            LDA 2200   // [A] <- 2200H
            MOV C,A    // Initialize counter
            MVI B,00   // sum = 0
            LXI H,2201 // Initialize pointer
```

```
BACK:   MOV A,M  // Get the number
        ANI 01H  // Mask Bit l to Bit7
        JNZ SKIP // Don't add if number is ODD
        MOV A,B  // Get the sum
        ADD M    // SUM = SUM + data
        MOV B,A  // Store result in B register

SKIP:   INX H    // increment pointer
        DCR C    // Decrement counter
        JNZ BACK // if counter 0 repeat
        MOV A,B  // Store result in A register
        STA 2210 // store sum
        HLT  // Terminate program execution

# ORG 2200
# DB 04H, 20H, 15H, 13H, 22H
```

Input:

Val	**Addr**
04H	**(2200H)**
20H	**(2201H)**
15H	**(2202H)**
13H	**(2203H)**
22H	**(2204H)**

Output:

42H (Addr - 2210H)

Q5) 8085 simulator program to sort the input array in ascending order, in place

```
LXI H,4200
          MOV C,M // MOVE FROM M TO C
          DCR C // DECREMENT IN C

REPEAT:          MOV D,C //MOVE FROM C TO D
          LXI H,4201

LOOP:     MOV A,M //MOVE FROM M TO A
          INX H //POINT TO NEXT LOCATION
          CMP M //COMPARE M WITH A
          JC SKIP //IF JC IS TRUE GO TO SKIP ELSE CONTINUE
          MOV B,M //MOVE M TO B
          MOV M,A // MOVE A TO M
          DCX H //POINT TO POINT LOCATION
          MOV M,B //MOVE B TO M
          INX H //POINT TO NEXT LOCATION

SKIP:     DCR D // DECREMENT IN D
          JNZ LOOP //JUMPS TO LOOP IF ZEROFLAG=0
          DCR C //DECREMENT IN D
          JNZ REPEAT //JUMPS TO REPEAT IF ZEROFLAG=0
          HLT
```

```
# ORG 4200

# DB 06,06,05,04,01,02,03
```

Input:

Val	Addr
06H	**(4200H)**
06H	**(4201H)**
05H	**(4202H)**
04H	**(4203H)**
01H	**(4204H)**
02H	**(4205H)**
03H	**(4206H)**

Output:

Val	Addr
01H	**(4201H)**
02H	**(4202H)**
03H	**(4203H)**
04H	**(4204H)**
05H	**(4205H)**
06H	**(4206H)**

Q6) 8085 simulator program for BCD to Hexadecimal number conversion

```
        LXI H,4150
```

```
          MOV A,M  // Initialize memory pointer
          ADD A    // MSD * 2
          MOV B,A  // Store MSD * 2
          ADD A    // MSD * 4
          ADD A    // MSD * 8
          ADD B    // MSD * 10
          INX H    // Point to LSD
          ADD M    // Add to form HEX
          INX H
          MOV M,A
          HLT

# ORG 4150
# DB 02H, 09H
```

Input:

02H (Addr - 4150) : MSB

09H (Addr - 4151) : LSB // Input is 29

Output:

1DH (Addr - 4152)

Q7) 8085 simulator program for Hexadecimal to BCD number conversion

```
          LXI H,4150// Initialize memory pointer
          MVI D,00  // Clear D-reg for most significant byte
```

```
        XRA A       // Clear Accumulator
        MOV C,M     // Get HEX input

LOOP2:      ADI 01      // Increment Accumulator by 1
        DAA // Adjust for BCD count
        JNC LOOP1       // Jump if not carry
        INR D       // Increment D

LOOP1:      DCR C       // Decrement C
        JNZ LOOP2
        STA 4151    // Store value of A into 4151
        MOV A,D     // Move D to A
        STA 4152    // Store value of A into 4152
        HLT  // Terminate

# ORG 4150
# DB FF
```

Input: FF (Addr - 4150H)

Output:

55H (Addr - 4151) - LSB

02H (Addr - 4152) – MSB

Q8) 8085 simulator program to Transfer contents to overlapping memory blocks

```
         MVI C,FF    // Initialize counter"
         LXI H,3005 // Initialize source memory pointer 3l4FH
         LXI D,3008 // Initialize destination memory pointer

BACK:    MOV A,M   // Get byte from source memory block
         STAX D      // Store byte in the destination memory block
         DCX H        // Decrement source memory pointer
         DCX D        // Decrement destination memory pointer
         DCR C        // Decrement counter
         JNZ BACK   // If counter 0 repeat
         HLT  // Stop execution

# ORG 3000
# DB 01H, 02H, 03H, 04H, 05H, 06H
```

Input:

<u>Val   Addr</u>

01H (3000H)

02H (3001H)

03H (3002H)

04H (3003H)

05H (3004H)

06H (3005H)

Output:

<u>Val   Addr</u>

01H (3003H)

02H (3004H)

03H (3005H)

04H (3006H)

05H (3007H)

06H (3008H)

Q9) 8085 program for Masking of lower and higher nibbles of 8-bit number

```
LDA 2050   // A <- M [2050]
MOV B,A    // B <- A
ANI 0F     // A <- A (AND) 0F
STA 3050   // M [3050] <- A
MOV A,B    // A <- B
ANI F0     // A <- A (AND) 0F
RLC  // rotate content of A left by 1 bit without carry
RLC  // rotate content of A left by 1 bit without carry
RLC  // rotate content of A left by 1 bit without carry
RLC  // rotate content of A left by 1 bit without carry
STA 3051   // M[3051] <- A
HLT  // END
```

```
# ORG 2050
# DB 64H
```

Input: 64H (Addr - 2050H)

Output:

04H (Addr - 3050H)

06H (Addr - 3051H)

Q10) 8085 simulator program to find the Sum of squares of first N natural numbers

```
        LDA 201B   // Load value from 201B to A
        MOV B,A    // Move value of A to B
        ADD B      // Add B to A
        INR A      // Increment A
        MOV D,A    // Move from A to D
        MOV A,B    // Move value of B to A
        INR A      // Increment A
        MOV C,A    // Mover from A to C
        MVI A,00   // Initialize A to 0

LOOP1:      ADD B      // Add B to A in loop
        DCR C      // Decrement C until 0
        JNZ LOOP1// Loop until zero flag
        MOV E,A    // Move A to E
        MVI A,00   // Initialize A to 0

LOOP2:      ADD E      // Add E to A in loop
        DCR D      // Decrement D until 0
        JNZ LOOP2// Loop until zero flag
        MVI C,06   // C value for diivision
        MVI B,00   // Initialize B value to 0

LOOP3:      INR B        // Increment B by 1
        SUB C        // Subtraact C from A until 0
        JNZ LOOP3// Loop until zero flag
        MOV A,B   // Move B to A
        STA 201C   // Push value of A to 201C
        HLT  // Terminate
# ORG 201B
# DB 4
```

Input: 04 (Addr - 201BH)

Output: 1EH (Addr - 201C)