Unit 3

Introduction to Microprocessor

==========

Program Level – 1

Store 8-bit data in memory

```
1. Program 1:
2. MVI A, 52H : "Store 32H in the accumulator"
3. STA 4000H  : "Copy accumulator contents at address 4000H"
4. HLT        : "Terminate program execution"

1. Program 2:
2. LXI H  : "Load HL with 4000H"
3. MVI M  : "Store 32H in memory location pointed by HL register pair (4000H)"
4. HLT    : "Terminate program execution"
```

`Note:` The result of both programs will be the same. In program 1 direct addressing instruction is used, whereas in program 2 indirect addressing instruction is used.

Exchange the contents of memory locations

`Statement:` Exchange the contents of memory locations 2000H and 4000H.

```
1. Program 1:
2. LDA 2000H   : "Get the contents of memory location 2000H into accumulator"
3.
4. MOV B, A    : "Save the contents into B register"
5. LDA 4000H   : "Get the contents of memory location 4000Hinto accumulator"
6.
7. STA 2000H   : "Store the contents of accumulator at address 2000H"
8. MOV A, B    : "Get the saved contents back into A register"
9. STA 4000H   : "Store the contents of accumulator at address 4000H"
10.
```

```
1. Program 2:
2. LXI H 2000H : "Initialize HL register pair as a pointer to memory location 2000H."
3. LXI D 4000H : "Initialize DE register pair as a pointer to memory location 4000H."
4. MOV B, M    : "Get the contents of memory location 2000H into B register."
5. LDAX D      : "Get the contents of memory location 4000H into A register."
6. MOV M, A    : "Store the contents of A register into memory location 2000H."
7. MOV A, B    : "Copy the contents of B register into accumulator."
8. STAX D      : "Store the contents of A register into memory location 4000H."
9. HLT         : "Terminate program execution."
```

`Note:` In Program 1, direct addressing instructions are used, whereas in Program 2, indirect addressing instructions are used.

Add two 8-bit numbers

`Statement:` Add the contents of memory locations 4000H and 4001H and place the result in memory location 4002H.

```
1. Sample problem
2. (4000H) = 14H
3. (4001H) = 89H
4. Result     = 14H + 89H = 9DH
5.
6. Source program
7. LXI H 4000H : "HL points 4000H"
8. MOV A, M    : "Get first operand"
9. INX H       : "HL points 4001H"
10.ADD M       : "Add second operand"
```

```
11.INX H        : "HL points 4002H"
12.MOV M, A     : "Store result at 4002H"
13.HLT          : "Terminate program execution"
```

Subtract two 8-bit numbers

`Statement:` Subtract the contents of memory location 4001H from the memory location 2000H and place the result in memory location 4002H.

```
1. Program -: Subtract two 8-bit numbers
2. Sample problem:
3. (4000H) = 51H
4. (4001H) = 19H
5. Result    = 51H - 19H = 38H
6.
7. Source program:
8. LXI H, 4000H : "HL points 4000H"
9. MOV A, M     : "Get first operand"
10.INX H        : "HL points 4001H"
11.SUB M        : "Subtract second operand"
12.INX H        : "HL points 4002H"
13.MOV M, A     : "Store result at 4002H"
14.HLT          : "Terminate program execution"
```

Add two 16-bit numbers

`Statement:` Add the 16-bit number in memory locations 4000H and 4001H to the 16-bit number in memory locations 4002H and 4003H. The most significant eight bits of the two numbers to be added are in memory locations 4001H and 4003H. Store the result in memory locations 4004H and 4005H with the most significant byte in memory location 4005H.

```
Sample problem:
(4000H) = 15H
(4001H) = 1CH
(4002H) = B7H
(4003H) = 5AH
Result = 1C15 + 5AB7H = 76CCH
(4004H) = CCH
(4005H) = 76H

Source Program 1:
LHLD 4000H  : "Get first I6-bit number in HL"
XCHG        : "Save first I6-bit number in DE"
LHLD 4002H  : "Get second I6-bit number in HL"
MOV A, E    : "Get lower byte of the first number"
ADD L       : "Add lower byte of the second number"
MOV L, A    : "Store result in L register"
MOV A, D    : "Get higher byte of the first number"
ADC H       : "Add higher byte of the second number with CARRY"
MOV H, A    : "Store result in H register"
SHLD 4004H  : "Store I6-bit result in memory locations 4004H and 4005H"
HLT         : "Terminate program execution"
```

```
Source program 2:
LHLD 4000H : Get first I6-bit number
XCHG : Save first I6-bit number in DE
LHLD 4002H : Get second I6-bit number in HL
DAD D : Add DE and HL
SHLD 4004H : Store I6-bit result in memory locations 4004H and 4005H.
HLT : Terminate program execution
```

`NOTE:` In program 1, eight bit addition instructions are used (ADD and ADC) and addition is performed in two steps. First lower byte addition using ADD instruction and then higher byte addition using ADC instruction.In program 2, 16-bit addition instruction (DAD) is used.

Add contents of two memory locations

Statement: Add the contents of memory locations 40001H and 4001H and place the result in the memory locations 4002Hand 4003H.

```
1. Sample problem:
2. (4000H) = 7FH
3. (400lH)  = 89H
4.          Result    = 7FH + 89H = lO8H
5. (4002H) = 08H
6. (4003H) = 0lH
7. Source program:
8. LXI H, 4000H  : "HL Points 4000H"
9. MOV A, M      : "Get first operand"
10.INX H         : "HL Points 4001H"
11.ADD M         : "Add second operand"
12.INX H         : "HL Points 4002H"
13.MOV M, A      : "Store the lower byte of result at 4002H"
14.MVIA, 00      : "Initialize higher byte result with 00H"
15.ADC A         : "Add carry in the high byte result"
16.INX H         : "HL Points 4003H"
17.MOV M, A      : "Store the higher byte of result at 4003H"
18.HLT           : "Terminate program execution"
```

Subtract two 16-bit numbers

Statement: Subtract the 16-bit number in memory locations 4002H and 4003H from the 16-bit number in memory locations 4000H and 4001H. The most significant eight bits of the two numbers are in memory locations 4001H and 4003H. Store the result in memory locations 4004H and 4005H with the most significant byte in memory location 4005H.

```
1. Sample problem:
2. (4000H) = 19H
3. (400IH) = 6AH
4. (4004H) = I5H (4003H) = 5CH
5. Result    = 6A19H – 5C15H = OE04H
6. (4004H) = 04H
7. (4005H) = OEH
8. Source program:
9. LHLD 4000H  : "Get first 16-bit number in HL"
10.XCHG        : "Save first 16-bit number in DE"
11.LHLD 4002H  : "Get second 16-bit number in HL"
12.MOV A, E    : "Get lower byte of the first number"
13.SUB L       : "Subtract lower byte of the second number"
14.MOV L, A    : "Store the result in L register"
15.MOV A, D    : "Get higher byte of the first number"
16.SBB H       : "Subtract higher byte of second number with borrow"
17.MOV H, A    : "Store 16-bit result in memory locations 4004H and 4005H"
```

```
SHLD 4004H  : "Store 16-bit result in memory locations 4004H and 4005H"
HLT         : "Terminate program execution"
```

Finding one's complement of a number

`Statement:` Find the l's complement of the number stored at memory location 4400H and store the complemented number at memory location 4300H.

```
Sample problem:
(4400H) = 55H
Result = (4300B) = AAB
Source program:
LDA 4400B : "Get the number"
CMA       : "Complement number"
STA 4300H : "Store the result"
HLT       : "Terminate program execution"
```

Finding Two's complement of a number

`Statement:` Find the 2's complement of the number stored at memory location 4200H and store the complemented number at memory location 4300H

```
Sample problem:
(4200H) = 55H
Result = (4300H) = AAH + 1 = ABH
Source program:
LDA 4200H  : "Get the number"
CMA        : "Complement the number"
ADI, 01 H  : "Add one in the number"
STA 4300H  : "Store the result"
HLT        : "Terminate program execution"
```

Pack the unpacked BCD numbers

Statement: Pack the two unpacked BCD numbers stored in memory locations 4200H and 4201H and store result in memory location 4300H. Assume the least significant digit is stored at 4200H.

```
Sample problem:
(4200H) = 04
(4201H) = 09
Result = (4300H) = 94
Source program:
LDA 4201H   : "Get the Most significant BCD digit"
RLC
RLC
RLC
RLC         : "Adjust the position of the second digit (09 is changed to 90)"
ANI FOH     : "Make least significant BCD digit zero"
MOV C, A    : "store the partial result"
LDA 4200H   : "Get the lower BCD digit"
ADD C       : "Add lower BCD digit"
STA 4300H   : "Store the result"
HLT         : "Terminate program execution"
```

Unpack a BCD number

Statement: Two digit BCD number is stored in memory location 4200H. Unpack the BCD number and store the two digits in memory locations 4300H and 4301H such that memory location 4300H will have lower BCD digit.

```
Sample problem:
(4200H) = 58
Result = (4300H) = 08 and
     (4301H) = 05
Source program:
LDA 4200H  : "Get the packed BCD number"
ANI FOH    : "Mask lower nibble"
RRC
RRC
RRC
```

```
RRC         : "Adjust higher BCD digit as a lower digit"
STA 4301H   : "Store the partial result"
LDA 4200H   : "Get the original BCD number"
ANI OFH     : "Mask higher nibble"
STA 4201H   : "Store the result"
HLT         : "Terminate program execution"
```

Execution format of instructions

`Statement:` Read the program given below and state the contents of all registers after the execution of each instruction in sequence.

```
Main program:
4000H     LXI SP, 27FFH
4003H     LXI H, 2000H
4006H     LXI B, 1020H
4009H     CALL SUB
400CH     HLT
Subroutine program:
4100H     SUB: PUSH B
4101H     PUSH H
4102H     LXI B, 4080H
4105H     LXI H, 4090H
4108H     SHLD 2200H
4109H     DAD B
410CH     POP H
410DH     POP B
410EH     RET
```

Right shift, bit of data(8 bit and 16 bit)

`Statement:` Write a program to shift an eight bit data four bits right. Assume data is in register C.

```
Sample problem:
(4200H) = 58
Result = (4300H) = 08 and
```

```
4.      (4301H) = 05
5.
6.  Source program 1:
7.  MOV A, C
8.  RAR
9.  RAR
10. RAR
11. RAR
12. MOV C, A
13. HLT
```

`Statement:` Write a program to shift a 16 bit data, 1 bit right. Assume that data is in BC register pair.

```
1.  Source program 2
2.  MOV A, B
3.  RAR
4.  MOV B, A
5.  MOV A, C
6.  RAR
7.  MOV C, A
8.  HLT
```

Left Shifting of a 16-bit data

`Statement:` Program to shift a 16-bit data 1 bit left. Assume data is in the HL register

```
1.  HL = 1025 = 0001 0000 0010 0101
2.
3.      HL = 0001 0000 0010 0101
4.    + HL = 0001 0000 0010 0101
5.  ----------------------------
6.  Result = 0010 0000 0100 1010
```

Alter the contents of flag register in 8085

`Statement:` Write a set of instructions to alter the contents of flag register in 8085.

```
1. PUSH PSW  : "Save flags on stack"
2. POP H     : "Retrieve flags in ‘L’"
3. MOV A, L  : "Flags in accumulator"
4. CMA       : "Complement accumulator"
5. MOV L, A  : "Accumulator in ‘L’"
6. PUSH H    : "Save on stack"
7. POP PSW   : "Back to flag register"
8. HLT       : "Terminate program execution"
```

Program Level – 2

Programs For 8085 Microprocessor Level 2

Count number of one's in a number

`Statement:` Write a program to count number of l's in the contents of D register and store the count in the B register.

```
Sample problem
(2200H) =  04
(2201H) = 34H
(2202H) = A9H
(2203H) = 78H
(2204H) = 56H
Result = (2202H) = A9H
```

```
1. MVI B, 00H
2. MVI C, 08H
3. MOV A, D
4. BACK: RAR
5. JNC SKIP
6. INR B
7. SKIP: DCR C
8. JNZ BACK
9. HLT
```

Arrange in ascending order

`Statement:` Write a program to sort given 10 numbers from memory location 2200H in the ascending order.

```
1. MVI B, 09      :"Initialize counter"
2. START          :"LXI H, 2200H: Initialize memory pointer"
3. MVI C, 09H     :"Initialize counter 2"
```

```
4. BACK: MOV A, M :"Get the number"
5. INX H          :"Increment memory pointer"
6. CMP M          :"Compare number with next number"
7. JC SKIP        :"If less, don't interchange"
8. JZ SKIP        :"If equal, don't interchange"
9. MOV D, M
10.MOV M, A
11.DCX H
12.MOV M, D
13.INX H          :"Interchange two numbers"
14.SKIP:DCR C     :"Decrement counter 2"
15.JNZ BACK       :"If not zero, repeat"
16.DCR B          :"Decrement counter 1"
17.JNZ START
18.HLT            :"Terminate program execution"
```

Calculate the sum of series of even numbers

`Statement:` Calculate the sum of series of even numbers from the list of numbers. The length of the list is in memory location 2200H and the series itself begins from memory location 2201H. Assume the sum to be 8 bit number so you can ignore carries and store the sum at memory location 2210H.

```
Sample problem

2200H= 4H

2201H= 20H

2202H= l5H

2203H= l3H

2204H= 22H

Result 22l0H= 20 + 22 = 42H

= 42H
```

```
1. LDA 2200H
2. MOV C, A        :"Initialize counter"
3. MVI B, 00H      :"sum = 0"
4. LXI H, 2201H    :"Initialize pointer"
5. BACK: MOV A, M  :"Get the number"
6. ANI 0lH         :"Mask Bit l to Bit7"
7. JNZ SKIP        :"Don't add if number is ODD"
8. MOV A, B        :"Get the sum"
9. ADD M           :"SUM = SUM + data"
10.MOV B, A        :"Store result in B register"
```

```
11.SKIP: INX H      :"increment pointer"
12.DCR C            :"Decrement counter"
13.JNZ BACK         :"if counter  0 repeat"
14.STA 2210H        :"store sum"
15.HLT              :"Terminate program execution"
```

Calculate the sum of series of odd numbers

`Statement:` Calculate the sum of series of odd numbers from the list of numbers. The length of the list is in memory location 2200H and the series itself begins from memory location 2201H. Assume the sum to be 16-bit. Store the sum at memory locations 2300H and 2301H.

```
Sample problem

2200H = 4H

2201H= 9AH

2202H= 52H

2203H= 89H

2204H= 3FH

Result = 89H + 3FH = C8H

2300H= H Lower byte

2301H = H Higher byte
```

```
1. Source program :
2. LDA 2200H
3. MOV C, A         :"Initialize counter"
4. LXI H, 2201H     :"Initialize pointer"
5. MVI E, 00        :"Sum low = 0"
6. MOV D, E         :"Sum high = 0"
7. BACK: MOV A, M   :"Get the number"
8. ANI 01H          :"Mask Bit 1 to Bit7"
9. JZ SKIP          :"Don't add if number is even"
10.MOV A, E         :"Get the lower byte of sum"
11.ADD M            :"Sum = sum + data"
12.MOV E, A         :"Store result in E register"
13.JNC SKIP
14.INR D            :"Add carry to MSB of SUM"
15.SKIP: INX H      :"Increment pointer"
```

Find the square of given number

`Statement:` Find the square of the given numbers from memory location 6100H and store the result from memory location 7000H.

```
Sample problem

2200H = 4H

2201H= 9AH

2202H= 52H

2203H= 89H

2204H= 3FH

Result = 89H + 3FH = C8H

2300H= H Lower byte

2301H = H Higher byte
```

```
1. LXI H, 6200H  :"Initialize lookup table pointer"
2. LXI D, 6100H  :"Initialize source memory pointer"
3. LXI B, 7000H  :"Initialize destination memory pointer"
4. BACK: LDAX D  :"Get the number"
5. MOV L, A      :"A point to the square"
6. MOV A, M      :"Get the square"
7. STAX B        :"Store the result at destination memory location"
8. INX D         :"Increment source memory pointer"
9. INX B         :"Increment destination memory pointer"
10.MOV A, C
11.CPI 05H       :"Check for last number"
12.JNZ BACK      :"If not repeat"
13.HLT           :"Terminate program execution"
```

Search a byte in a given number

Statement: Search the given byte in the list of 50 numbers stored in the consecutive memory locations and store the address of memory location in the memory locations 2200H and 2201H. Assume byte is in the C register and starting address of the list is 2000H. If byte is not found store 00 at 2200H and 2201H.

```
1. LX I H, 2000H      :"Initialize memory pointer 52H"
2. MVI B, 52H         :"Initialize counter"
3. BACK: MOV A, M     :"Get the number"
4. CMP C              :"Compare with the given byte"
5. JZ LAST            :"Go last if match occurs"
6. INX H              :"Increment memory pointer"
7. DCR B              :"Decrement counter"
8. JNZ B              :"If not zero, repeat"
9. LXI H, 0000H
10.SHLD 2200H
11.JMP END            :"Store 00 at 2200H and 2201H"
12.LAST: SHLD 2200H   :"Store memory address"
13.END:  HLT          :"Stop"
```

Add two decimal numbers of 6 digit each

Statement: Two decimal numbers six digits each, are stored in BCD package form. Each number occupies a sequence of byte in the memory. The starting address of first number is 6000H Write an assembly language program that adds these two numbers and stores the sum in the same format starting from memory location 6200H.

```
1. LXI H, 6000H     :"Initialize pointer l to first number"
2. LXI D, 6l00H     :"Initialize pointer2 to second number"
3. LXI B, 6200H     :"Initialize pointer3 to result"
4. STC
5. CMC              :"Carry = 0"
6. BACK: LDAX D     :"Get the digit"
7. ADD M            :"Add two digits"
8. DAA              :"Adjust for decimal"
9. STAX.B           :"Store the result"
10.INX H            :"Increment pointer 1"
11.INX D            :"Increment pointer2"
12.INX B            :"Increment result pointer"
13.MOV A, L
14.CPI 06H          :"Check for last digit"
15.JNZ BACK         :"If not last digit repeat"
16.HLT              :"Terminate program execution"
```

Separate even numbers from given numbers

`Statement:` Write an assembly language program to separate even numbers from the given list of 50 numbers and store them in the another list starting from 2300H. Assume starting address of 50 number list is 2200H.

```
1. LXI H, 2200H     :"Initialize memory pointer 1"
2. LXI D, 2300H     :"Initialize memory pointer2"
3. MVI C, 32H       :"Initialize counter"
4. BACK:MOV A, M    :"Get the number"
5. ANI 01H          :"Check for even number"
6. JNZ SKIP         :"If ODD, don't store"
7. MOV A, M         :"Get the number"
8. STAX D           :"Store the number in result list"
9. INX D            :"Increment pointer 2"
10.SKIP: INX H      :"Increment pointer 1"
11.DCR C            :"Decrement counter"
12.JNZ BACK         :"If not zero, repeat"
13.HLT              :"Stop
```

Transfer contents to overlapping memory blocks

`Statement:` Write assembly language program with proper comments for the following:
A block of data consisting of 256 bytes is stored in memory starting at 3000H. This block is to be shifted (relocated) in memory from 3050H onwards. Do not shift the block or part of the block anywhere else in the memory.

```
Two blocks (3000 – 30FF and 3050 – 314F) are overlapping. Therefore it is necessary to tran
sfer last byte first and first byte last.
```

```
1. MVI C, FFH       :"Initialize counter"
2. LX I H, 30FFH    :"Initialize source memory pointer 3l4FH"
3. LXI D, 314FH     :"Initialize destination memory pointer"
4. BACK: MOV A, M   :"Get byte from source memory block"
5. STAX D           :"Store byte in the destination memory block"
6. DCX H            :"Decrement source memory pointer"
7. DCX              :"Decrement destination memory pointer"
```

```
8. DCR C            :"Decrement counter"
9. JNZ BACK         :"If counter   0 repeat"
10.HLT              :"Stop execution"
```