Examples of Assembly Language Programs

6.1 INTRODUCTION

To learn assembly language programming the beginner should write simple programs given in this chapter and try to execute them on Intel 8085 based microprocessor kit. The memory addresses given in the program are for a particular microprocessor kit. These addresses can be changed to suit the microprocessor kit available in the laboratory. Before writing assembly language program, one should learn some important Intel 8085 instructions such as MOV, MVI, ADD, SUB, LXI, LDA, INX, INR, HLT etc. described in chapter 4. While learning programs, one should gradually pick up new instructions which have not been used earlier.

6.2 SIMPLE EXAMPLES

The use of some important instructions are described below with very simple examples.

The memory addresses given in these examples are for Vinytics' kit.

Example 1. Object: Place 05 in register B.

PROGRAM

Memory address	*Machine codes*	*Mnemonics*	*Operands*	*Comments*
FC00	06,05	MVI	B, 05	Get 05 in register B.
FC02	76	HLT		Stop

The instruction MVI B, 05 moves 05 to register B. HLT halts the program. A program is fed to the microprocessor kit in machine codes. The machine code for the instruction MVI B, 05 is 06, 05. The 1st byte of the instruction is 06 which is the machine code for the instruction MVI B. The second byte of the instruction, 05 is the data which is to be moved to register B. The code for HLT is 76. The machine codes for a program are entered in the memory. In the above program the memory addresses from FC00 to FC02 have been used. The machine code 06 is entered in the memory location FC00 H; 05 in FC01 H and 76 in FC02 H. This program can be executed on a microprocessor kit and the register B can be examined. After the execution of the program the register B will contain 05. The memory address can be changed to suit the microprocessor kit available in the laboratory. The address and data used for a microprocessor based system are in hexadecimal system. The symbol H after a digit denotes that it is in hexadecimal system.

Example 2

Object: Get 05 in register A; then move it to register B.

PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
FC00	3E, 05	MVI	A, 05	Get 05 in register A.
FC02	47	MOV	B,A	Transfer 05 from register A to B.
FC03	76	HLT		Stop.

Note: Microprocessors are designed to process hexadecimal numbers. Hence, data and address given in an assembly language program are hexadecimal numbers.

The instruction MVI A, 05 will move 05 to register A. In the code from it is written as 3E, 05. The 1st byte of the instruction is 3E. This code is for MVI A. The second byte 05 is the data which is to be placed in A. The instruction MOV B, A transfers the content of register A to register B. After the execution of the above program regiser B will contain 05. The program can be executed on a microprocessor kit and the register B can be examined.

Example 3

Object: Load the content of the memory location FC50 H directly to the accumulator, then transfer it to regiser B. The content of the memory location FC50 H is 05.

PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
FC00	3A, 50, FC	LDA	FC50	Get the content of the memory location FC50 H into accumulator.
FC03	47	MOV	B,A	Move the content of register A to B.
FC04	76	HLT	Halt.	

DATA

FC50 — 05

The instruction LDA loads the accumulator directly with the content of the memory location specified in the instruction. Thus the instruction LDA FC50 H will load the accumulator with the content of the memory location FC50 H. The instruction MOV B, A will move the content of the accumulator to B. The content of the memory location FC50 H is 05. It is fed to the microprocessor kit as data. After the execution of the above program the register B will contain 05.

Example 4

Object: Move the content of the memory location FC50 H to register C. The content of the memory location FC50 H is 08.

PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
FC00	21, 50, FC	LXI	H,FC50	Get the memory address FC50 H in H-L pair.
FC03	4E	MOV	C,M	Move the content of the memory location, whose address is in the H-L pair, to register C.
FC04	76	HLT		Halt.

DATA

FC50 — 08

The instruction LXI H, FC50 H will place FC50 H in register pair H-L. FC50 H is the address of the memory location from where the data is to be transferred to register C. In the code form the instruction is written as 21, 50, FC. The 1st byte of the instruction is 21. It is the machine code for the instruction LXI H. The operand is FC50 which is to be placed in H-L pair. The 2nd byte of the instruction is 50. It is 8 LSBs of the operand. The 3rd byte is FC which is 8 MSBs of the operand. In the code form the LSB is written first then the MSB. Due to this reason the operand FC50 has been written in the code form as 50, FC. The instruction MOV C, M transfers the content of the memory location, whose address is in H-L pair, to register C. The execution of the previous instruction has placed FC50

H in H-L pair. Therefore, MOV C, M will move the content of FC50 H to register C. After the execution of the above program the regiseter C will contain 08.

Example 5

Object : Place the content of the memory location FC50 H in register B and that of FC51 H in register C. The contents of FC50 and FC51 H are 11 H and 12 H respectively.

PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
FC00	21, 50, FC	LXI	H, FC50 H	Get FC50 H in H-L pair.
FC03	46	MOV	B, M	Move the content of FC50 to B.
FC04	23	INX	H	Increment H-L pair by one.
FC05	4E	MOV	C, M	Move the content of FC51 to C.
FC06	76	HLT		Halt.

DATA

FC50 — 11 H

FC51 — 12 H.

The instruction LXI H, FC50 H will place FC50 H in H-L pair. FC50 H is the address of the memory location which contains 11 H. MOV B, M will move the content of FC50 H to register B. The instruction INX H increases the content of H-L pair by 1. The execution of the instruction INX H will increase the content of H-L pair from FC50 H to FC51 H. MOV C, M will move the content of FC51 H to register C. Thus, after the execution of the above program, the register B and C will contain 11 H and 12 H respectively.

Example 6

Object : Place 05 in the accumulator. Increment it by one and store the result in the memory location FC50 H.

PROGRAM

Memory address	Machine Codes	*Mnemonics*	*Operands*	*Comments*
FC00	3E,05	MVI	A,05	Get 05 in the accumulator.
FC02	3C	INR	A	Increment the content of accumulator by one.
FC03	32,50,FC	STA	FC50 H	Store result in FC50 H.
FC06	76	HLT		Halt

The instruction MVI A, 05 moves 05 to the accumulator. INR A increases the content of the accumulator from 05 to 06. STA FC50 stores the content of the accumulator in the memory location FC50 H. After the execution of the above program the memory location FC50 H will contain 06.

A number of important and useful assembly language programs are given in the subsequent subsections. Memory addresses used for them are for Professional's kits/Vinytics' kits.

6.3. ADDITION OF TWO 8-BIT NUMBERS: SUM 8-BITS

Problem:

Add 49 H and 56 H.

The 1st number 49 H is in the memory location 2501 H.
The 2nd number 56 H is in the memory location 2502 H.
The result is to be stored in the memory location 2503 H.
Numbers are represented in hexadecimal system.

PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25	LXI	H, 2501 H	Get address of 1st number in H-L pair.
2003	7E	MOV	A,M	1st number in accumulator.
2004	23	INX	H	Increment content of H-L pair.
2005	86	ADD	M	Add 1st and 2nd numbers.
2006	32, 03,25	STA	2503 H	Store sum in 2503 H.
2009	76	HLT		Stop

DATA

2501 — 49 H
2502 — 56 H
The sum is stored in the memory location 2503 H.

Result

2503 — 9F H.

2501 H is the address of memory location for the 1st number. 2501 is placed in H-L pair by the instruction LXI H, 2501 H. The next instruction is MOV A, M which moves the content of the memory location addressed by H-L pair to the accumulator. In this case H-L pair contains 2501 H and, therefore, the content of the memory location 2501 H is moved to the accumulator. Thus the 1st number 49 H has been moved to the accumulator. The instruction INX H increases the content of H-L pair by one. Previously, the content of H-L pair was 2501 H. After the execution of INX H it becomes 2502 H. ADD M adds the contents of the accumulator and the content of the memory location addressed by H-L pair. The content of 2502 H is the 2nd number 56 H. So 56 H is added to 49 H. Sum resides in the accumulator. The instruction STA 2503 H stores the sum in the memory location 2503 H. The instruction HLT ends the program.

6.4. 8-BIT SUBTRACTION*

Example 1

	49 H	1st number
	− 32 H	2nd number
Result	17 H	

The 1st number 49 H is in the memory location 2501 H.
The 2nd number 32 H is in the memory location 2502 H.
The result is to be stored in the memory location 2503 H.

*Note: If the second number is greater than the first number, the processor will give result in 2's complement i.e. negative result. See Section 6.39.

PROGRAM

Memory address	Machine Codes	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25	LXI	H, 2501 H	Get address of 1st number in H-L pair.
2003	7E	MOV	A, M	1st number in accumulator.
2004	23	INX	H	Content of H-L pair increases from 2501 to 2502 H.
2005	96	SUB	M	1st number - 2nd number.
2006	23	INX	H	Content of H-L pair becomes 2503 H.
2007	77	MOV	M, A	Store result in 2503 H.
2008	76	HLT		Halt

Example 1

DATA

2501 — 49 H

2502 — 32 H

Result is stored in the Memory location 2503 H

2503 — 17 H

Example 2

DATA

2501 — F8 H

2502 — 9B H

Result

2503 — 5D H

The 1st number is in the memory location 2501 H. 2501 is placed in H-L pair by the execution of the instruction LXI H, 2501 H. The instruction MOV A, M moves the content of the memory location addressed by H-L pair to the accumulator. Thus the 1st number 49 H (Example 1) which is in 2501 H is placed in the accumulator. INX H increases the content of H-L pair from 2501 to 2502 H. The instruction SUB M subtracts the content of the memory location addressed by H-L pair from the accumulator. The 2nd number which is in the memory location 2502 H is subtracted from the 1st number which is in the accumulator. The result resides in the accumulator. The instruction INX H increases the content of H-L pair from 2502 to 2503 H. The instruction MOV M, A transfers the content of the accumulator to the memory location addressed by H-L pair. So the result which is in the accumulator is sotred in the memory location 2503 H. The instruction HLT ends the program.

6.5 ADDITION OF TWO 8-BIT NUMBERS; SUM: 16-BITS

Example 1

Add 98 H and 9A H.

SUM = 01, 32 H.

The 1st number 98 H is in the memory location 2501 H.

The 2nd number 9A H is in the memory location 2502 H.

The results are to be stored in 2503 and 2504 H.

Numbers are represented in hexadecimal.

In this case the sum is to be stored in two consecutive memory locations. The LSBs of the sum is 32 H and it will be stored in the memory location 2503 H. The MSB of the sum is 01 which will be stored in 2504 H.

Note. The program given below is also applicable for addition of 8-bit numbers: sum: 8 bits. The MSBs of the sum for the such a case will be 00.

PROGRAM

Memory address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25		LXI	H, 2501 H	Address of 1st number in H-L pair.
2003	0E, 00		MVI	C,00	MSBs of sum in register C. Initial value = 00.
2005	7E		MOV	A, M	1st number in accumulator.
2006	23		INX	H	Address of 2nd number 2502 in H-L pair.
2007	86		ADD	M	1st number + 2nd number.
2008	D2, 0C, 20		JNC	AHEAD	Is carry? No, go to the label AHEAD.
200B	0C		INR	C	Yes, increment C.
200C	32, 03, 25	AHEAD	STA	2503 H	LSBs of sum in 2503 H.
200F	79		MOV	A, C	MSBs of sum in accumulator.
2010	32, 04, 25		STA	2504 H	MSBs of sum in 2504 H.
2013	76		HLT		Halt

Example 1

DATA

2501 — 98 H

2502 — 9A H

Result

2503 — 32 H, LSBs of sum.

2504 — 01 H, LSBs of sum.

Example 2

DATA

2501 — F5 H

2502 — 8A H

Result

2503 — 7F H, LSBs of sum.

2504 — 01 H, MSBs of sum.

The 1st instruction of the program LXI H, 2501 H gets the address of the 1st number in H-L pair. The MSBs of the sum is placed in register C. The initial value of the MSBs of the sum is kept 00. MOV A, M transfers 1st number from the memory location 2501 H to the accumulator. INX H increases the content of H-L pair from 2501 to 2502 H. The 2nd number is in 2502 H. ADD M adds 1st and 2nd numbers. After the execution of the instruction JNC the instruction INR C is executed if the addition of two numbers produces a carry. In binary system carry is equal to one and, therefore, the content of the register C is increased from 00 to 01. The value 01 is nothing but the MSBs of the sum. Now the register C contains the MSBs of the sum. The accumulator contains the LSBs of the sum. The instruction STA 2503 H places the LSBs of the sum in memory location 2503 H. The instruction MOV A, C moves the MSBs of the sum from register C to the accumulator. The instruction STA 2504 H stores the MSBs of the sum in 2504 H.

After the execution of the instruction JNC the program jumps to label AHEAD and executes STA 2503 H, if the addition of two numbers does not produce a carry. The LSBs of the sum is placed in the memory location 2503 H. Register C contains 00, so the execution of the instruction MOV A, C trans-

fers 00 in the accumulator. The instruction STA 2504 H stores 00 in the memory location 2504 H. This is the MSB of the sum. Such a situation will arise when the sum of two numbers is of 8 bits, and hence there is no carry.

6.6. DECIMAL ADDITION OF TWO 8-BIT NUMBERS, SUM: 16 BITS

In the previous programs for addition the data (numbers) and sum are in hexadecimal. A program can be developed for decimal addition, in which numbers and sum are in decimal system. Instruction DAA is used in the program for decimal adjust.

Example 1

Add 84 D and 75 D, D stands for decimal number.

Sum = 159 = 01, 59 D

59 is the LSDs of the sum.

01 is the MSDs of the sum.

The 1st number 84 D is in the memory location 2501 H.

The 2nd number 75 D is in the memory location 2502 H.

The sum is to be stored in 2503 and 2504 H.

PROGRAM

Memory address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25		LXI	H, 2501 H	Address of 1st number in H-L pair.
2003	0E, 00		MVI	C, 00	MSDs of sum in register C. Initial value = 00.
2005	7E		MOV	A, M	1st number in accumulator.
2006	23		INX	H	Address of 2nd number 2502 in H-L pair.
2007	86		ADD	M	1st number + 2nd number.
2008	27		DAA		Decimal adjust.
2009	D2, 0D, 20		JNC	AHEAD	Is carry? No, go to the label AHEAD.
200C	0C		INR	C	Yes, increment C.
200D	32, 03, 25	AHEAD	STA	2503 H	LSDs of sum in 2503 H.
2010	79		MOV	A, C	MSDs of sum in accumulator.
2011	32, 04, 25		STA	2504 H	MSDs of sum in 2504 H.
2014	76		HLT		

Example 1

DATA

2501 — 84 D

2502 — 75 D

Result

2503 — 59 D, LSDs of the sum.

2504 — 01 D, MSDs of the sum.

Example 2

DATA

2501 — 96 D

2502 — 69 D

Result

2503 — 65 D, LSDs of the sum.

2504 — 01 D, MSDs of the sum.

The program for "Decimal Addition of Two 8-Bit Numbers" is same as that for "Addition of Two 8-Bit numbers; sum 16 Bits" except that DAA instruction has been used for decimal adjustment after

the instruction ADD M. After the execution of the instruction ADD M the sum is in the hexadecimal. The DAA instruction makes correction and the final result is obtained in decimal system. The following examples will make clear how DAA instruction makes correction.

Case 1. When the sum of the number comes in between 10 and 15 (or A and F in hexadecimal), the DAA instruction adds 6 to give correct result. 6 is added to both LSBs and MSBs.

Example

```
 96 D =    1001  0110   (BCD)
+69 D = + 0110  1001   (BCD)
-------------------------------------------------
FF (Hex) = 1111  1111   (Binary representation of hexadecimal)
   DAA   + 0110  0110   (+ 6 corrections to LSBs and + 6 corrections to MSBs).
-------------------------------------------------
01, 65 D = 1,0110 0101 (BCD)
```

Case 2

```
                84 D = 1000 0100 (BCD)
                92 D = 1001 0010 (BCD)
-------------------------------------------------
116 (hex) = 0001, 0001 0110
                  ↑
                CARRY
              DAA + 0110
-------------------------------------------------
01, 76 D = 0001, 0111, 0110 (BCD) = Correct result.
```

A correction of + 6 is required when there is carry after addition. In this case Auxiliary Carry is zero, so there is no correction in LSBs of the sum.

Case 3

```
                98 D = 1001 1000 (BCD)
                99 D = 1001 1001 (BCD)
-------------------------------------------------
01, 31, (hex) = 0001, 0011 0001
                  ↑    ↑
                Carry  Auxiliary carry
DAA                  + 0110 + 0110 (Correction + 6 and + 6)
-------------------------------------------------
01, 97 D = 0001, 1001 0111 (BCD) Correct result
```

A correction of + 6 is also made when there is auxiliary carry.

Conclusion

(*i*) When sum lies between 10 to 15 (or A to F hexadecimal) a correction of + 6 is made by DAA instruction. 6 is added to both LSBs and MSBs.

(*ii*) If there is an auxiliary carry after addition, a correction of + 6 is made to the LSBs of the sum.

(*iii*) If there is a carry after addition, a correction of + 6 is made to the MSBs of the sum.

Thus the DAA instruction checks whether there is a carry, auxiliary carry or the sum lies between A to F and makes correction accordingly.

6.7. ADDITION OF TWO 16-BIT NUMBERS, SUM: 16 BITS OR MORE

Example 1

5 B 9 8 H	1st number
+ 8 E 4 C H	2nd number
Sum: 00, E 9 E 4 H	

The 1st number is in the memory locations 2501 and 2502 H.
The 2nd number is in the memory locations 2503 and 2504 H.
The sum is stored in the memory locations 2505 to 2507 H.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	2A, 01, 25		LHLD	2501 H	1st 16-bit number in H-L pair.
2003	EB		XCHG		Get 1st number in D-E pair.
2004	2A, 03, 25		LHLD	2503 H	2nd 16-bit number in H-L pair.
2007	0E, 00		MVI	C, 00	MSBs of the sum in register C. Initial value = 00.
2009	19		DAD	D	1st number + 2nd number
200A	D2, 0E, 20		JNC	AHEAD	Is carry? No, go to the label AHEAD.
200D	0C		INR	C	Yes, increment C.
200E	22, 05, 25	AHEAD	SHLD	2505 H	Store LSBs of sum in 2505 and 2506 H.
2011	79		MOV	A, C	MSBs of sum in accumulator.
2012	32, 07, 25		STA	2507 H	Store MSBs of the sum in 2507 H.
2015	76		HLT		Halt

Example 1

DATA

2501 — 98 H, LSBs of 1st Number.
2502 — 5B H, MSBs of 1st number.
2503 — 4C H, LSBs of 2nd number.
2504 — 8E H, MSBs of 2nd number.

Result

2505 — E4, LSBs of sum.
2506 — E9, LSBs of sum.
2507 — 00, MSBs of sum.

Example 2

DATA

2501 — 45 H, LSBs of 1st number.
2502 — A6 H, MSBs of 1st number.
2503 — 23 H, LSBs of 2nd number.
2504-9B H, MSBs of 2nd number.

Result

A6 45
+ 9B 23
Sum = 01, 41 68

2505 — 68 H, LSBs of sum.
2506 — 41 H, LSBs of sum.
2507 — 01 H, MSBs of sum.

The instruction LHLD 2501 H transfers the content of 2501 H to register L, and the content of 2502 H to register H. Thus the LSBs of the 1st number is placed in register L and the MSBs of the 1st number in register H. Now H-L pair contains 1st number which is a 16-bit number. The instruction XCHG exchanges the content of D-E pair with H-L pair. Therefore, the 1st number is transferred from H-L pair to D-E pair. Again the instruction LHLD 2503 H places the 2nd number in H-L pair. MVI C, 00 places the MSBs of the sum in register C, and the initial value is 00. DAD D adds the content of D-E and H-L pairs and places the sum in H-L pair. This instruction is for 16-bit addition. If there is a carry after 16-bit addition, the content of the register C is incremented. If there is no carry, the program jumps after JNC to the lable AHEAD and executes SHLD 2505 H. The LSBs of the sum is stored in the memory location 2505 and 2506. The MSBs of the sum is transferred to the accumulator from register C and then it is stored in the memory location 2507 H. In case of the carry the MSBs of the sum is 01. In case of no carry it is 00.

6.8. 8-BIT DECIMAL SUBTRACTION*

Example 1

96 D 1st number
– 38 D 2nd number
Result: 58 D

DAA instruction cannot be used after SUB or SBB instruction for decimal subtraction. It is used, after ADD, ADC etc. Therefore, for decimal subtraction the number which is to be subtracted is converted into 10's complement. In the above example 38 D is to be subtracted. 10's complement of 38 is first obtained and then it is added to 96 D.

$$96 - 38 = 96 + (-38)$$
$$= 96 + \text{10's complement of 38.}$$

98 is stored in the memory location 2501 H and 38 in 2502 H. The result is to be stored in 2503 H. The program flow chart is shown in Fig. 6.1

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	21, 02, 25	LXI	H, 2502 H	Get address of 2nd number in H-L pair.
2003	3E, 99	MVI	A, 99	Place 99 in accumulator.
2005	96	SUB	M	9's complement of 2nd number.
2006	3C	INR	A	10's complement of 2nd number.
2007	2B	DCX	H	Get address of 1st number.

*Note: If the second number is greater than the first number, the processor will give result in 10's complement i.e. negative result. See Section 6.40.

2008	86	ADD	M	Add 1st number and 10's complement of 2nd number.
2009	27	DAA		Decimal adjustment.
200A	32, 03, 25	STA	2503 H	Store result in 2503 H.
200D	76	HLT		Halt

Example 1
DATA
2501 — 96
2502 — 38
Result
2503 — 58

Example 2
DATA
2501 — 99
2502 — 48
Result
2503 — 51

Example 3
DATA
2501 — 50
2502 — 10
Result
2503 — 40

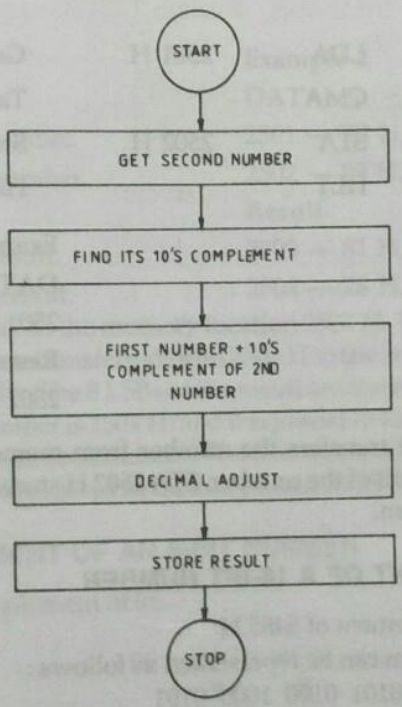

Fig. 6.1. Program Flow Chart for Decimal Subtraction.

The instruction LXI H, 2502 H places the address of the 2nd number in H-L pair. MVI A, 99 places 99 D in the accumulator. SUB M subtracts the 2nd number (placed in 2502 H) from 99 D to get 9's complement of the 2nd number. INR A increments the content of the accumulator. Thus 10's complement of the 2nd number is obtained. DCX H decrements the content of H-L pair to obtain the address of the 1st number *i.e.* 2501 H. ADD M adds 10's complement of the 2nd number to the 1st number. After addition the result is in hexadecimal. The instruction DAA is used to get the result in decimal system. The result is stored in the memory location 2503 H by the instruction STA 2503 H. The instruction HLT ends the program.

6.9. FIND ONE'S COMPLEMENT OF AN 8-BIT NUMBER

Example 1. Find one's complement of 96 H. The number in the binary form is represented as follows :

96 H = 1001 0110
(9) (6)

One's complement = 0110 1001 = 69 H.
(6) (9)

To obtain one's complement of a number its 0 bits are replaced by 1 and 1 by 0.
The number is placed in the memory location 2501 H.
The result is stored in the memory location 2502 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 01, 25	LDA	2501 H	Get data in accumulator.
2003	2F	CMA		Take its complement.
2004	32, 02, 25	STA	2502 H	Store result in 2502 H.
2007	76	HLT		Halt

Example 1
DATA
2501 — 96 H
Result
2502 — 69 H

Example 2
DATA
2501 — E4 H
Result
2502 — 1B H

The instruction LDA 2501 H transfers the number from memory location 2501 H to the accumulator. CMA takes complement of the number. STA 2502 H stores the result in the memory location 2502 H. HLT ends the program.

6.10. FIND ONE'S COMPLEMENT OF A 16-BIT NUMBER

Example 1. Find one's complement of 5485 H.
The number in the binary form can be represented as follows :

5485 = 0101 0100 1000 0101
(5) (4) (8) (5)

One's complement = 1010 1011 0111 1010 = AB7A H
(A) (B) (7) (A)

The number is in the memory locations 2501 H and 2502 H.
The result is to be stroed in the memory locations 2503 H and 2504 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25	LXI	H, 2501 H	Address of LSBs of the number.

2003	7E	MOV	A, M	8 LSBs of the number in accumulator.
2004	2F	CMA		Complement of 8 LSBs of the number.
2005	32, 03, 25	STA	2503 H	Store 8 LSBs of result.
2008	23	INX	H	Address of 8 MSBs of the number.
2009	7E	MOV	A, M	8 MSBs of the number in accumulator.
200A	2F	CMA		Complement of 8 MSBs of the number.
200B	32, 04, 25	STA	2504	Store 8 MSBs of the result.
200E	76	HLT		Halt

Example 1

DATA

2501 — 85 H, LSBs of the number.

2502 — 54 H, MSBs of the number.

Result

2503 — 7A H, LSBs of the result.

2504 — AB H, MSBs of the result.

Example 2

DATA

2501 — 7E H, LSBs of the number.

2502 — 89 H, MSBs of the number

Result

2503 — 81 H, LSBs of the result.

2504 — 76 H, MSBs of the result.

The 8 LSBs of the number are in the memory location 2501 H. The address 2501 is placed in H-L pair. The 8 LSBs of the number are transferred from 2501 H to the accumulator. The instruction CMA takes one's complement of 8 LSBs. The 8 LSBs of the result are stored in the memory location 2503 H. The address of 8 MSBs of the number is 2502 H, and it is placed in H-L pair. The 8 MSBs of the number are transferred from 2502 H to the accumulator. The instruction CMA takes one's complement of 8 MSBs. The 8 MSBs of the result are stored in memory location 2504 H.

6.11 FIND TWO'S COMPLEMENT OF AN 8-BIT NUMBER

Example 1. Find two's complement of 96.

	96 = 1001	0110	
	(9)	(6)	
1's complement	= 0110	1001	= 69
	(6)	(9)	
	+ 0000	0001	
2's complement	= 0110	1010	= 6A
	(6)	(A)	

Two's complement of a number is obtained by adding 1 to the 1's complement of the number.
The number is placed in the memory location 2501 H.
The result is to be stored in the memory location 2502 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 01, 25	LDA	2501 H	Get data in accumulator.
2003	2F	CMA		Take its 1's complement.
2004	3C	INR	A	Take 2's complement.
2005	32, 02, 25	STA	2502 H	Store result in 2502 H.
2008	76	HLT		Stop

Example 1
DATA
2501 — 96 H
Result
2502 — 6A H

Example 2
DATA
2501 — E4 H
Result
2502 — 1C H

6.12. FIND TWO'S COMPLEMENT OF A 16-BIT NUMBER

Example 1. Find two's complement of 5B8C.

5B8C	= 0101	1011	1000	1100	
	(5)	(B)	(8)	(C)	
1's complement	= 1010	0100	0111	0011	= A473
	(A)	(4)	(7)	(3)	
	+ 0000	0000	0000	0001	
2's complement	= 1010	0100	0111	0100	= A474

Two's complement of a number is obtained by adding 1 to the one's complement of the number.
The number is stored in the memory location 2501 and 2502 H.
The result is to be stored in the memory location 2503 and 2504 H.

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25		LXI	H, 2501 H	Address of 8 LSBs of the number.
2003	06, 00		MVI	B, 00	Use register B to store carry.
2005	7E		MOV	A, M	8 LSBs in accumulator.
2006	2F		CMA		1's complement of 8 LSBs of the number.
2007	C6, 01		ADI	01	2's complement of 8 LSBs of the number.
2009	32, 03, 25		STA	2503 H	Store 8 LSBs of the result.
200C	D2,10, 20		JNC	GO	
200F	04		INR	B	Store carry.
2010	23	GO	INX	H	Address of 8 MSBs of the number.

2011	7E	MOV	A, M	8 MSBs in accumulator.
2012	2F	CMA		1's complement of 8 MSBs of the number.
2013	80	ADD	B	Add carry.
2014	32, 04, 25	STA	2504 H	Store 8 MSBs of the result.
2017	76	HLT		Stop.

Example 1

DATA

2501 — 8C, LSBs of the number.

2502 — 5B, MSBs of the number.

Result

2503 — 74, LSBs of the result.

2504 — A4, MSBs of the result.

2's complement of the number is A474.

Example 2. Find 2's complement of 5B00.

5B00	= 0101	1011	0000	0000
	(5)	(B)	(0)	(0)
1's complement	= 1010	0100	1111	1111
	(A)	(4)	(F)	(F)
	+ 0000	0000	0000	0001
2's complement	= 1010	0101	0000	0000
	(A)	(5)	(0)	(0)

2's complement of the number is A500.

For the problem given above the data and results are as follows :

DATA

2501 — 00, LSBs of the number.

2502 — 5B, MSBs of the number.

Result

2503 — 00, LSBs of the result.

2504 — A5, MSBs of the result.

The 8 LSBs of the number are in the memory location 2501 H. They are taken first and 1's complement is obtained. To obtain 2's complement 1 is added to 1's complement. 8 LSBs of 2's complement are stored in 2503 H. The carry resulting from the addition of 1 to 1's complement is stored in register B. After this 8 MSBs of the number are taken and 1's complement is obtained. The carry is added to it. The 8 MSBs of the result are stored in 2504 H. In case of no carry the program jumps from JNC GO to INX H and the content of register B is not incremented. It remains zero. The addition of zero to 1's complement of 8 MSBs does not affect the result.

6.13. SHIFT AN 8-BIT NUMBER LEFT BY ONE BIT

Example. Shift 65 left by one bit.

The binary representation of 65 is given below :

65 =	0110	0101
	(6)	(5)

Result of shifting 65 left by one bit

= 1100 1010 = CA
(C) (A)

To shift a number left by one bit the number is added to itself. If 65 is added to 65, the result is CA as shown below.

```
  65 = 0110 0101
+ 65 = 0110 0101
-----------------
       1100 1010 = CA
```

The number is placed in memory 2501 H.
The result is to be stored in memory 2502 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 01, 25	LDA	2501 H	Get data in accumulator.
2003	87	ADD	A	Shift it left by one bit.
2004	32, 02, 25	STA	2502 H	Store result in 2502 H.
2007	76	HLT		Halt

DATA
2501 — 65 H
Result
2502 — CA H

The instruction LDA 2501 H transfers the number from memory location 2501 H to the accumulator. ADD A adds the contents of the accumulator to itself. The result is twice the number and thus the number is shifted left by one bit. This program does not take carry into account after ADD instruction. If numbers to be handled are likely to produce carry the program may be modified to store it.

6.14 SHIFT AN 8-BIT NUMBER LEFT BY 2 BITS

Example. Shift 15 left by two bits.

```
                          15 = 0001 0101
                               (1)  (5)
Shift left by one bit        = 0010 1010 = 2A
                               (2)  (A)
Shift the result left by one bit
                             = 0101 0100 = 54
i.e. Shift the number left by  (5)  (4)
     two bits
```

The shifting of a number left by two bits can be achieved by adding the number to itself and again adding the sum to itself. This is obtained using the instruction ADD A twice.
The number is placed in the memory location 2501 H.
The result is to be stored in the memory location 2502 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 01, 25	LDA	2501 H	Get data in accumulator.
2003	87	ADD	A	Shift it left by one bit.
2004	87	ADD	A	Again shift left by one bit.
2005	32, 02, 25	STA	2502 H	Store result in 2502 H.
2008	76	HLT		Stop

DATA
2501 — 15 H
Result
2502 — 54 H

6.15. SHIFT A 16-BIT NUMBER BY ONE BIT

Example 1
Shift 7596 H left by one bit. 7596 = 0111 0101 1001 0110
(7) (5) (9) (6)
Result of shifting left by one bit = 1110 1011 0010 1100 = EB2C
(E) (B) (2) (C)
The number is placed in the memory locations 2501 and 2502 H.
The result is to be stored in the memory locations 2503 and 2504 H.
PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	2A, 01, 25	LHLD	2501 H	Get data in H-L pair.
2003	29	DAD	H	Shift left by one bit.
2004	22, 03, 25	SHLD	2503 H	Store result in 2503 and 2504 H.
2007	76	HLT		Stop

Example 1
DATA
2501 — 96 H, LSBs of the number.
2502 — 75 H, MSBs of the number.
Result
2502 — 2C, LSBs of the result.
2504 — EB, MSBs of the result.

Example 2
DATA
2501 — BF, LSBs of the number.
2502 — 00, MSBs of the number.
Result
2503 — 7E, LSBs of the result.
2504 — 01, MSBs of the result.

The 16-bit number has been placed in two consecutive memory locations 2501 and 2502 H. The instruction LHLD 2501 H transfers the 16-bit number from 2501 and 2502 H to H-L pair. DAD H is an instruction for 16-bit addition. It adds the contents of H-L pair to itself. Thus the 16-bit number is shifted left by one bit. The 16-bit result is stored in the memory locations 2503 and 2504 H by SHLD instruction. In some cases there may be carry after the execution of instruction DAD H. In that case carry may be stored in some register. The program may be modified accordingly.

If the shifting of an 8-bit number gives a result which is more than 8-bits, the problem can be tackled using the technique of shifting 16-bit number (See Example 2).

6.16 SHIFTING OF A 16-BIT NUMBER LEFT BY 2 BITS

Example. Shift 1596 H left by 2 bits.

1596 =	0001	0101	1001	0110	
	(1)	(5)	(9)	(6)	
Result of shifting left by one bit =	0010	1011	0010	1100	= 2B2C
	(2)	(B)	(2)	(C)	
Result of shifting 2 bits left =	0101	0110	0101	1000	= 5658
	(5)	(6)	(5)	(8)	

The number is stored in the memory locations 2501 and 2502 H.
The result is to be stored in the memory locations 2503 and 2504 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	2A, 01, 25	LHLD	2501 H	Get data in H-L pair.
2003	29	DAD	H	Shift left by one bit.
2004	29	DAD	H	Again shift left by one bit.
2005	22, 03, 25	SHLD	2503 H	Store result in 2503 and 2504 H.
2008	76	HLT		Stop

DATA
2501 — 96 H, LSBs of the number.
2502 — 15 H, MSBs of the number.
Result
2503 — 58 H, LSBs of the result.
2504 — 56 H, MSBs of the result.

6.17 MASK OFF LEAST SIGNIFICANT 4 BITS OF AN 8-BIT NUMBER

Example.

Number = A6			
=	1010	0110	
	(A)	(6)	
Result = A0 =	1010	0000	
	(A)	(0)	

We want to mask off the least significant 4 bits of a given number. The LSD of the given number A6 is 6. It is to be cleared (masked off) *i.e.* it is to be made equal to zero. The MSD of the number A6 is A. In the binary form it is 1010. It is not to be affected. If this number is ANDed with 1111 *i.e.* F, it will not be affected. Similarly the LSD of the number is 6. In the binary form it is represented by 0110. If it is ANDed with 0000, it becomes 0000 *i.e.* it is cleared. Thus if the number A6 is ANDed with F0, the LSD of the number is masked off.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 01, 25	LDA	2501 H	Get data in accumulator.
2003	E6, F0	ANI	F0	Mask off the least significant 4 bits.
2005	32, 02, 25	STA	2502 H	Store result in 2502 H.
2008	76	HLT		Stop

DATA
2501 — A6
Result
2502 — A0

The instruction LDA 2501 H transfers the content of memory location 2501 H *i.e.* the given number to the accumulator. ANI F0 logically ANDs the content of the accumulator with F0 to clear the least significant 4 bits of the number. STA 2502 H stores the result in memory location 2502 H. HLT stops the program.

6.18. MASK OFF MOST SIGNIFICANT 4 BITS OF AN 8-BIT NUMBER

Example.

Number = A6
= 1010 0110
(A) (6)
Result = 06 = 0000 0110
(0) (6)

To mask off 4 most significant bits of a number, 4 MSBs are ANDed with 0000. The least significant bits are not to be affected and, therefore, they are ANDed with 1111 *i.e.* F. Thus if an 8-bit number is ANDed with 0F, the 4 most significant bits are cleared.

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 01, 25	LDA	2501 H	Get data in accumulator.
2003	E6, 0F	ANI	0F	Mask off the most significant 4 bits.
2005	32, 02, 25	STA	2502 H	Store result in 2502 H.
2008	76	HLT		Stop

DATA
2501 — A6
Result
2502 — 06

The instruction LDA 2501 H transfers the content of memory location 2501 H to the accumulator. ANI 0F logically ANDs the content of the accumulator with 0F to clear the most significant 4 bits of the number. STA 2502 H stores the result in 2502 H. HLT stops the program.

6.19. TO FIND SQUARE FROM LOOK-UP TABLE

Example. Find square of 07 (decimal) using look-up table technique.

The number 07 D is in the memory location 2500 H.

The result is to be stored in the memory location 2501 H.

The table for square is stored from 2600 to 2609 H.

The address of the memory locations is in hexadecimal.

The number and squares are to be entered in decimal.

The squares of data are stored in certain memory locations in the tabular form. This is called look-up table. For this example the squares of numbers from 00 to 09 are stored in memory locations 2600 to 2609 H. The values of squares are in decimal. The data form the index and it is transferred from memory to the accumulator and then to register L. It forms the LSBs of the memory location where square of the data is placed. The MSBs of the address is moved to register H. Now the address of the desired memory location where the square of the data resides is in H-L pair. The square of the data is now moved to the accumulator and then it is stored.

In the program LDA 2500 H moves data (07) to the accumulator. MOV L, A moves 07 to register L. MVI H, 26 H moves 26 H in register H. Now in H-L pair 2607 H is residing which is the address of the memory location where the square of 07 D is placed. MOV A, M transfers 49 D from memory location 2607 to the accumulator. This is stored in 2501 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 00, 25	LDA	2500 H	Get data in accumulator.
2003	6F	MOV	L, A	Get data in register L.
2004	26, 26	MVI	H, 26 H	Get 26 in register H.
2006	7E	MOV	A, M	Square of data in accumulator.
2007	32, 01, 25	STA	2501 H	Store square in 2501 H.
200A	76	HLT		Stop

DATA

2500 — 07 D

Result

2501 — 49 D

Look-up Table

Address (Hex)		*Square (Decimal)*
2600	—	00
2601	—	01
2602	—	04
2603	—	09
2604	—	16
2605	—	25
2606	—	36

6.19. TO FIND SQUARE FROM LOOK-UP TABLE

Example. Find square of 07 (decimal) using look-up table technique.

The number 07 D is in the memory location 2500 H.

The result is to be stored in the memory location 2501 H.

The table for square is stored from 2600 to 2609 H.

The address of the memory locations is in hexadecimal.

The number and squares are to be entered in decimal.

The squares of data are stored in certain memory locations in the tabular form. This is called look-up table. For this example the squares of numbers from 00 to 09 are stored in memory locations 2600 to 2609 H. The values of squares are in decimal. The data form the index and it is transferred from memory to the accumulator and then to register L. It forms the LSBs of the memory location where square of the data is placed. The MSBs of the address is moved to register H. Now the address of the desired memory location where the square of the data resides is in H-L pair. The square of the data is now moved to the accumulator and then it is stored.

In the program LDA 2500 H moves data (07) to the accumulator. MOV L, A moves 07 to register L. MVI H, 26 H moves 26 H in register H. Now in H-L pair 2607 H is residing which is the address of the memory location where the square of 07 D is placed. MOV A, M transfers 49 D from memory location 2607 to the accumulator. This is stored in 2501 H.

PROGRAM

Address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	3A, 00, 25	LDA	2500 H	Get data in accumulator.
2003	6F	MOV	L, A	Get data in register L.
2004	26, 26	MVI	H, 26 H	Get 26 in register H.
2006	7E	MOV	A, M	Square of data in accumulator.
2007	32, 01, 25	STA	2501 H	Store square in 2501 H.
200A	76	HLT		Stop

DATA

2500 — 07 D

Result

2501 — 49 D

Look-up Table

Address (Hex)		*Square (Decimal)*
2600	—	00
2601	—	01
2602	—	04
2603	—	09
2604	—	16
2605	—	25
2606	—	36

2607	—	49
2608	—	64
2609	—	81

6.20 TO FIND LARGER OF TWO NUMBERS

Example 1. Find the larger of 98 H and 87 H.

The first number 98 H is placed in the memory locaticn 2501 H.

The 2nd number 87 H is placed in the memory location 2502 H.

The result is stored in the memory location 2503 H.

The numbers are represented in hexadecimal system. The 1st number is moved from its memory location to the accumulator. It is compared with 2nd number. The larger of the two is then placed in the accumulator. From the accumulator the larger number is moved to the desired memory location.

PROGRAM

Memory address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25		LXI	H, 2501 H	Address of 1st number in H-L pair.
2003	7E		MOV	A, M	1st number in accumulator.
2004	23		INX	H	Address of 2nd number in H-L pair.
2005	BE		CMP	M	Compare 2nd number with 1st number. Is the 2nd number > 1st ?
2006	D2, 0A, 20		JNC	AHEAD	No, larger number is in accumulator. Go to AHEAD.
2009	7E		MOV	A, M	Yes, get 2nd number in accumulator.
200A	32, 03, 25	AHEAD	STA	2503 H	Store larger number in 2503 H.
200D	76		HLT		Stop

Example 1

DATA

2501 — 98 H

2502 — 87 H

Result is 98 H and it is stored in memory location 2503 H.

2503 — 98 H

Example 2

DATA

2501 — A9 H

2502 — EB H

Result

2503 — EB H

The 1st step of the program is LXI H, 2501 H. The address of the memory location of the 1st number is 2501 H. This address *i.e.* 2501 is placed in H-L pair. The next instruction of the program is MOV A, M. The execution of this instruction moves the content of the memory (which has been previously defined *i.e.* 2501) to the accumulator. Now the 1st number *i.e.* 98 (in example 1) is in the

accumulator. INX H increases the content of H-L pair by 1. The content of H-L pair is increased from 2501 to 2502. The address of the memory location for 2nd number is 2502 H. Now M associated with any instruction will mean the memory location 2502 H. The instruction CMP M means that the content of the memory location 2502 is compared with the content of the accumulator. CMP M sets the flags as if the content of the memory location addressed by H-L pair had been subtracted from the contents of the accumulator. However, the contnets of the accumulator remains unchanged for further processing. If the 1st number (which is in accumulator) is larger than the 2nd number there is no carry (see example 1). The instruction JNC is executed and the program jumps to the instruction STA 2503 H. The larger number is in the accumulator. The instruction STA 2503 H stores the larger number *i.e.* 98 in the memory location 2503 H. The instruction HLT will end the program.

If the 1st number (which is in the accumulator) is smaller than the 2nd number; (which is in memory) there is a carry. In example 2, 1st the number A9 is smaller than 2nd number. After the execution of instruction JNC the program will not jump to STA 2503 H as there is a carry. The program takes up the next instruction MOV A, M. The 2nd number which is greater is moved to the accumulator. Now STA 2503 H is executed and the larger number is stored in the memory location 2503 H. The instruction HLT will end the program.

6.21 TO FIND THE LARGEST NUMBER IN A DATA ARRAY

Example 1. The number in a series are: 98, 75 and 99.

As there are three numbers in the series, the count = 03. The count is placed in the memory location 2500 H. The numbers are placed in the memory locations 2501 to 2503 H. The result is to be stored in the memory location 2450 H.

The 1st number of the series is placed in the accumulator and it is compared with the 2nd number residing in the memory. The larger of the two numbers is placed in the accumulator. Again this number which is in the accumulator is compared with the 3rd number of series and larger number is placed in the accumulator. This process of comparison is repeated till all the numbers of the series are compared and the largest number is stored in the desired memory location. The program flow chart is shown in Fig. 6.2

PROGRAM

Memory address	Machine Codes	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 00, 25		LXI	H, 2500 H	Address for count in H-L pair.
2003	4E		MOV	C, M	Count in register C.
2004	23		INX	H	Address of 1st number in H-L pair.
2005	7E		MOV	A, M	1st number in accumulator.
2006	0D		DCR	C	Decrement count.
2007	23	LOOP	INX	H	Address of next number.
2008	BE		CMP	M	Compare next number with previous maximum. Is next number > previous maximum?

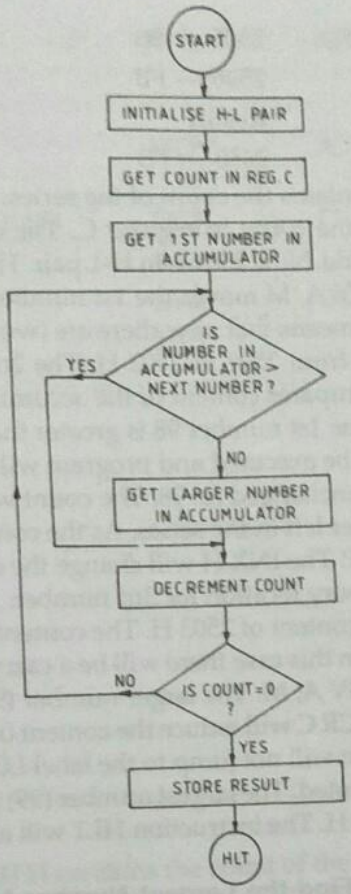

Fig. 6.2. Program Flow Chart to find Largest Number from a Series of Numbers.

2009	D2, 0D, 20		JNC	AHEAD	No, larger number is in accumulator. Go to the lable AHEAD
200C	7E		MOV	A, M	Yes, get larger number in accumulator.
200D	0D	AHEAD	DCR	C	Decrement count.
200E	C2, 07, 20		JNZ	LOOP	
2011	32, 50, 24		STA	2450 H	Store result in 2450 H.
2014	76		HLT		Stop.

Example 1
DATA
2500 — 03
2501 — 98
2502 — 75
2503 — 99

Example 2
DATA
2500 — 06
2501 — 38
2502 — 94
2503 — EB
2504 — A8

Result

2450 — 99

2505 — B5

2506 — FB

Result

2450 — FB

The memory location 2500 H contains the count of the series. 2500 H is placed in H-L pair. The next instruction MOV C, M places the count in register C. The count in Example 1 is 03. INX H increases the content of H-L pair by one. Now 2501 is in H-L pair. The 1st number of the series resides in the memory location 2501 H. MOV A, M moves the 1st number (98) in the accumulator. DCR C decreases the count from 03 to 02. It means that now there are two more numbers in the series. INX H increases the content of H-L pair from 2501 to 2502 H. The 2nd number of the series is in the memory location 2502 H. CMP M compares content of the accumulator (1st number) with the content of 2502 H (2nd number). Since the 1st number 98 is greater than the 2nd number 75, there will be no carry. The instruction JNC will be executed and program will move to the label AHEAD. The content of the accumulator remains unchanged *i.e.* 98. The count will be decreased from 02 to 01. It indicates that now there is one number left in the series. As the content of register C is not zero, the program will move to the label LOOP. The INX H will change the content of H-L pair from 2502 to 2503 H. 2503 H is the address of memory location for 3rd number. Again CMP M will compare the content of accumulator (98) with the content of 2503 H. The content of 2503 H is 99 which is greater than the content of the accumulator. In this case there will be a carry. After JNC instruction the program will take up the instruction MOV A, M. The larger number (99) will move from the memory location 2503 H to the accumulator. DCR C will reduce the content of register C from 01 to 00. As the content of register C is 00, the program will not jump to the label LOOP. It will proceed further and the instruction STA 2450 H will be executed. The largest number (99) which is in the accumulator will be stored in the memory location 2450 H. The instruction HLT will end the program.

6.21.1. An Alternative Program to Find the Largest Number from a Series of Numbers.

Example 1. The numbers of a series are: 98, 75 and 99. As there are three numbers in the series, count = 03. The count is stored in the memory location 2500 H. The numbers are stored in the memory location 2501 H to 2503 H.

The result is to be stored in the memory location 2450 H.

Memory address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2400	21, 00, 25		LXI	H, 2500 H	Address for count in H-L pair.
2403	4E		MOV	C, M	Count in register C.
2404	97		SUB	A	Get 00 in accumulator.
2405	23	LOOP	INX	H	Address of the next number of the series.
2406	BE		CMP	M	Compare next number with previous maximum. Is next number > previous maximum?

2407	D2, 0B, 24		JNC	AHEAD	No, larger number is in the accumulator. Go to the label AHEAD.
240A	7E		MOV	A, M	Yes, get larger number in the accumulator.
240B	0D	AHEAD	DCR	C	Decrement count.
240C	C2, 05, 24		JNZ	LOOP	
240F	32, 50, 24		STA	2450 H	Store result in 2450 H.
2412	76		HLT		Stop

Example 1

DATA

2500 — 03

2501 — 98

2502 — 75

2503 — 99

Result

2450 — 99

Example 2

DATA

2500 — 06

2501 — 38

2502 — 94

2503 — EB

2504 — AB

2505 — B5

2506 — FB

Result

2450 — FB

The memory location 2500 H contains the count of the series. 2500 H is placed in H-L pair. The instruction MOV C, M places the count in register C. In example 1 count is 03. The instruction SUB A makes the content of the accumulator 00. As 00 is the smallest 8-bit number, it is taken as initial value for comparison. INX H points out the 1st number of the series. CMP M compares 1st number with 00. As the 1st number is greater than 00, there is a carry, and after the execution of JNC, the instruction MOV A, M is executed. The first number *i.e.* 98 is now transferred to the accumulator. Then the count in register C is decremented and the program is transferred to the label LOOP. Then INX H points out the address of the 2nd number of the series (75). The 1st number (98) which is in the accumulator is compared with the 2nd number. As the 1st number (98) is larger than the 2nd number, there is no carry, and after the execution of the instruction JNC the program jumps to the label AHEAD. Thus the 1st number being larger number is retained in accumulator. Then the count is decremented and program jumps to LOOP again to process the third number of the series. In this way at every step of comparison, if the next number of the series is larger one, it is transferred to the accumulator. If the next number is smaller one, it is ignored and the previous larger number which remains in the accumulator is retained. Thus the larger number of the series is obtained and stored in the memory location 2450 H.

6.22 TO ARRANGE A SERIES OF NUMBERS IN DESCENDING ORDER

Example 1. Arrange 54, EB, 85, 9B and A8 in descending order. These numbers are stored in the memory locations 2501 to 2505 H. The count = 05 and it is stored in 2500 H. Results are to be stored in 2601 to 2605 H.

First of all the largest number is selected from the given series of numbers and it is stored in 2601

H. This number is deleted from the series and 00 is stored in its position. Again the largest number is selected second time from the modified series. Since 00 is the smallest 8-bit number it does not affect the result while selecting the largest number second time. The largest number is now stored in 2602 H. Again this number is also replaced by 00 and from the modified series of numbers the largest number is selected again and stored in 2603 H. This process is repeated till all the numbers of the string are arranged in descending order.

SUBROUTINE-1 is to find the largest number in a data array. SUBROUTINE-2 is to check which number is the largest, and to replace it by zero.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	11, 01, 26		LXI	D, 2601	Memory locations to store result.
2003	21, 00, 25		LXI	H, 2500	Count address in H-L pair.
2006	46		MOV	B, M	Count in register B to check whether all numbers have been arranged in descending order.
2007	CD, 00, 22	START	CALL	2200	Call subroutine-1 to find largest number.
200A	12		STAX	D	Store result.
200B	CD, 50, 20		CALL	2050	Call subroutine-2 to check which number is largest.
200E	13		INX	D	
200F	05		DCR	B	Have all numbers been arranged in descending order?
2010	C2, 07, 20		JNZ	START	No, repeat process.
2013			HLT		Stop

SUBROUTINE-1. To Find Largest Number

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2200	21, 00, 25		LXI	H, 2500	Count address in H-L pair.
2203	4E		MOV	C, M	Count in register C to check whether all numbers have been compared to find largest number.
2204	97		SUB	A	Make content of accumulator zero.
2205	23	LOOP	INX	H	

2206	BE		CMP	M	Compare next number with previous largest. Is next number > previous largest?
2207	D2, 0B, 22		JNC	AHEAD	No, larger number is in the accumulator. Go to label AHEAD.
220A	7E		MOV	A, M	Yes, get larger number in accumulator.
220B	0D	AHEAD	DCR	C	
220C	C2, 05, 22		JNZ	LOOP	
220F	C9		RET		

SUBROUTINE-2

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2050	21, 00, 25		LXI	H, 2500 H	
2053	4E		MOV	C, M	Count to check which number was selected as largest one.
2054	23	BEHIND	INX	H	Get next number.
2055	BE		CMP	M	Compare the next number with largest number which is in accumulator.
2056	CA, 5D, 20		JZ	FORWARD	Is the present number larger one? Yes, go to FORWARD.
2059	0D		DCR	C	Decrement count.
205A	C2, 54, 20		JNZ	BEHIND	No, jump to take up next number.
205D	3E, 00	FORWARD	MVI	A, 00	
205F	77		MOV	M, A	Replace largest number by 00.
2060	C9		RET		

Example 1

DATA

2500 — 05
2501 — 54
2502 — EB
2503 — 85
2504 — 9B
2505 — A8

Example 2

DATA

2500 — 06
2501 — 99
2502 — B4
2503 — 34
2504 — FF
2505 — A6

Result

2601 — EB
2602 — A8
2603 — 9B
2604 — 85
2605 — 54
2506 — E8

Result

2601 — FF
2602 — E8
2603 — B4
2604 — A6
2605 — 99
2606 — 34

In the program there are three counts. Count 1 is stored in register B and it checks whether all the numbers of the series have been arranged in descending order. Count 2 is stored in register C to check whether all the numbers of the series have been compared while finding the largest number of the series. After the selection of the largest number it is to be ascertained that which number was the largest one. At this stage register C is free. The register C is again used in the SUBROUTINE-2 to store count 3 to check which number has been selected as the largest number. The largest number is replaced by 00 in the series. Again, the program to find the largest number is taken up to select the largest number in the modified series. The program flow chart is shown in Fig. 6.3.

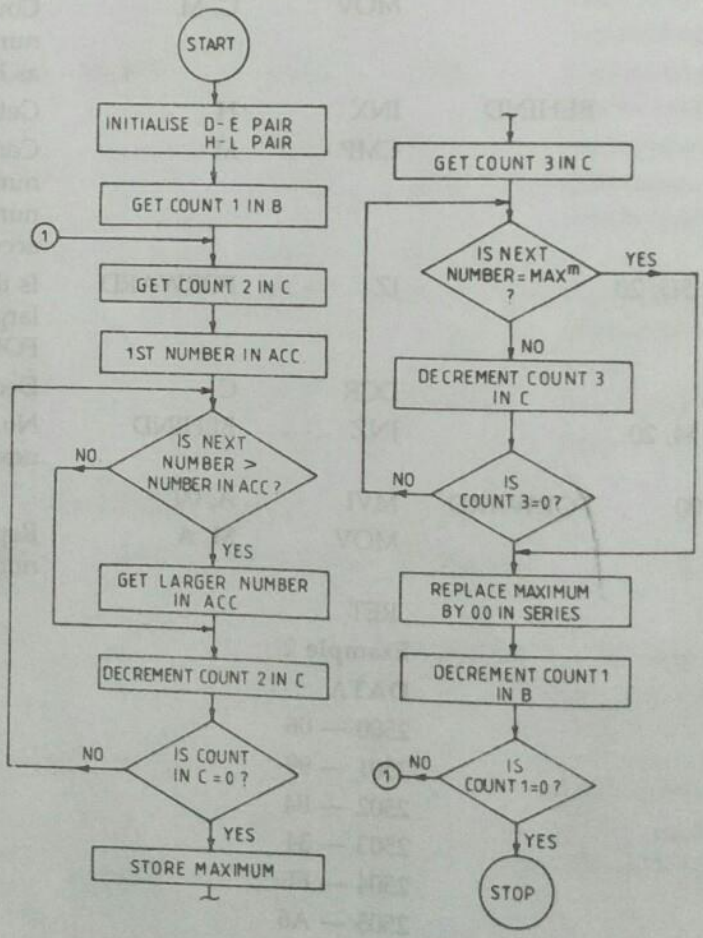

Fig. 6.3 Program Flow Chart for Arranging Numbers in Descending order

6.22.1 An Alternative Program to Arrange an Array of Data in Ascending (or Descending) Order

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 00, 25		LXI	H, 2500	Address for count.
2003	4E		MOV	C, M	Count for number of passes in register C.
2004	21, 00, 25	BACK	LXI	H, 2500	
2007	56		MOV	D, M	Count for number of Comparisons in register D.
2008	23		INX	H	
2009	7E		MOV	A, M	1st number in accumulator
200A	23	LOOP	INX	H	Address of next number
200B	46		MOV	B, M	Next number in Register B.
200C	B8		CMP	B	Compare next number with previous greatest number.
200D	D2, 16, 20		JNC	AHEAD	If previous greatest number > next number, go to AHEAD
2010	2B		DCX	H	
2011	77		MOV	M, A	Place smaller of the two compared numbers in memory
2012	78		MOV	A, B	Place greater of the two numbers in accumulator.
2013	C3, 18, 20		JMP	GO	
2016	2B	AHEAD	DCX	H	
2017	70		MOV	M, B	Place smaller of the two compared numbers in memory
2018	23	GO	INX	H	
2019	15		DCR	D	Decrease the count for comparisons.
201A	C2, 0A, 20		JNZ	LOOP	
201D	77		MOV	M, A	Place the greatest number after a pass in the memory
201E	0D		DCR	C	Decrease the count for passes
201F	C2, 04, 20		JNZ	BACK	
2022	76		HLT		Stop

DATA

2500 — Count (04)
2501 — 60
2502 — 40
2503 — 50
2504 — 15
2505 — 25

Result

15
25
40
50
60

2503 — 50	40
2504 — 15	50
2505 — 25	60

Note: When the instruction JNC at the memory location 200D is replaced by JC (code: DA), th above program becomes a program for arranging an array of data in descending order.

In this program the microprocessor first compares the first two numbers of the data array. I keeps the larger of the two numbers in the accumulator and stores the smaller number in the memory. Then it proceeds further and takes up the third number and compares it with the number which is in the accumulator. If the number in the accumulator is smaller than the 3rd number, it stores 3rd number in the accumulator. It stores the smaller number (which was in the accumulator) in the memory. In this way it processes all the numbers of the array. Finally, it stores the largest number in the last memory location. In the example given above the last memory location was 2505 H. This is all in the 1st pass. Now the processor again goes in the loop for the 2nd pass. In 2nd pass again it selects the largest number from the remaining numbers and it stores it in the last but one memory location (in above example, the memory location is 2504 H. The processor goes through several passes and arranges numbers in ascending order. Both the number of passes and the number of comparisons will be less than the number of data in the array by one. In the above example there are five data and hence both counts are 04. When the processor arranges numbers in descending order, it selects the smallest number from the data array and stores it in the last memory location. In this way it proceeds further.

6.23 TO FIND SMALLER OF TWO NUMBERS

Example 1. Find the smaller of 84 H and 99 H.

The first number 84 H is placed in the memory location 2501 H.

The 2nd number 99 H is placed in the memory location 2502 H.

Store the result in the memory location 2503 H.

The numbers are represented in hexadecimal system. The 1st number is moved from its memory location to the accumulator. It is compared with the 2nd number. The smaller of the two is then placed in the accumulator. From the accumulator the smaller number is moved to the desired memory location where it is to be stored.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25		LXI	H, 2501 H	Address of 1st number in H-L pair.
2003	7E		MOV	A, M	1st number in accumulator.
2004	23		INX	H	Address of 2nd number in H-L pair.
2005	BE		CMP	M	Compare 2nd number with 1st. Is 1st number < 2nd number?
2006	DA, 0A, 20		JC	AHEAD	Yes, smaller number is in accumulator. Go to AHEAD.

2009	7E		MOV	A, M	No, Get 2nd number in accumulator
200A	32, 03, 25	AHEAD	STA	2503 H	Store smaller number in 2503 H.
200D	76			HLT	Stop

Example 1	Example 2
DATA	**DATA**
2501 — 84 H	2501 — B9 H
2502 — 99 H	2502 — 8A H
Result	**Result**
2503 — 84 H	2503 — 8A H

2501 H is the address of the memory location for the 1st number. The instruction LXI H, 2501 H places 2501 H in H-L pair. the next instruction MOV A, M moves the content of the memory location 2501 H to the accumulator. Thus the 1st number 84 H (Example 1) is now in the accumulator. The instruction INX H increases the content of H-L pair from 2501 to 2502 H. The instruction CMP M compares the content of the accumulator (1st number) with the content of the memory location 2502 H (2nd number). The instruction CMP M sets the flags as if the contents of the memory location addressed by H-L pair has been subtracted from the contents of the accumulator. However, the contents of the accumulator remain unchanged. If the 1st number (which is in the accumulator) is smaller than the 2nd number (which is in the memory), there is a carry (see Example 1). The instruction JC is executed and the program jumps to STA 2503 H. The smaller number (84 H) is in the accumulator and the instruction STA 2503 H stores it in the memory location 2503 H. The instruction HLT will end the program.

If the 1st number (which is in the accumulator) is greater than the 2nd number (which is in the memory), there is no carry (see Example 2). In this case after the execution of instruction JC the program does not jump to the instruction STA 2503 H. The program takes up the next instruction MOV A, M. Since the 2nd number is smaller and it is in the memory, it is moved to the accumulator. Now the instruction STA 2503 H is executed and the smaller number is stored in the memory location 2503 H. The instruction HLT will end the program.

6.24 TO FIND THE SMALLEST NUMBER IN A DATA ARRAY

Example 1. The numbers of a series are: 86, 58 and 75.

As there are three numbers in the series, count = 03.

The count is placed in the memory location 2500 H.

The numbers are placed in the memory location 2501 to 2503 H.

The result is to be stored in the memory location 2450 H.

The 1st number of the series is placed in the accumulator and it is compared with the 2nd number which is in the memory. The smaller of the two is placed in the accumulator. Again this number which is in the accumulator is compared with the 3rd number of the series and smaller number is placed in the accumulator. This process of comparison is repeated till all the numbers of the series are compared and the smallest number is stored in the desired memory location.

The memory location 2500 H contains the count of the series. In Example 1 count is 03. 2500 is placed in H-L pair. The instruction MOV C, M places the count in register C. INX H increases the content of H-L pair from 2500 to 2501 H. The 1st number of the series resides in the memory location 2501 H. MOV A, M moves the 1st number (86) in the accumulator. DCR C decreases the count from

03 to 02, which means that now there are 2 more numbers in the series. INX H increases the content of H-L pair from 2501 to 2502 H. The 2nd number of the series is in 2502 H. CMP M compares the content of the accumulator (1st number) with the content of memory location 2502 H (2nd number). Since the 1st number 86 is greater than the 2nd number 58, there will be no carry. As there is no carry, the instruction MOV A, M will be executed. The smaller number 58 will be placed in the accumulator. DCR C will decrease count from 02 to 01. It indicates that there is one more number left in the series. As the content of register C is not zero, the program will jump to the label LOOP. The INX H will change the content of H-L pair from 2502 to 2503 H. The 3rd number of the series is in 2503 H. Again CMP M will compare the content of the accumulator (58) with the content of 2503 H. The content of 2503 H is 75 which is greater than the content of the accumulator. In this case there will be a carry. After the execution of the instruction JC AHEAD the program will jump to the label AHEAD. The content of the accumulator will remain as it is. DCR C will reduce the count from 01 to 00. As the content of register C is zero, the program will not jump. It will proceed further to execute STA 2450 H. The smallest number (58) which is in the accumulator will be stored in the memory location 2450 H. The instruction HLT will end the program.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 00, 25		LXI	H, 2500 H	Get address for count in H-L pair.
2003	4E		MOV	C, M	Count in register C.
2004	23		INX	H	Get address of 1st number in H-L pair.
2005	7E		MOV	A, M	1st number in accumulator.
2006	0D		DCR	C	Decrement count.
2007	23	LOOP	INX	H	Address of next number in H-L pair.
2008	BE		CMP	M	Compare next number with previous smallest. Is previous smallest less than next number?
2009	DA, 0D, 20		JC	AHEAD	Yes, smaller number in accumulator. Go to AHEAD.
200C	7E		MOV	A, M	No, get next number in accumulator.
200D	0D	AHEAD	DCR	C	Decrement count.
200E	C2, 07, 20		JNZ	LOOP	
2011	32, 50, 24		STA	2450 H	Store smallest number in 2450 H.
2014	76		HLT		Stop.

Example 1
DATA
2500 — 03 H
2501 — 86 H

Example 2
DATA
2500 — 05 H
2501 — EB H

2502 — 58 H
2503 — 75 H
Result
2450 — 58 H

2502 — D4 H
2503 — 3C H
2504 — 0F H
2505 — A8 H
Result
2450 — 0F H

6.24.1 An Alternative Program to Find the Smallest Number from a Series of Numbers

Example 1. The numbers of a series are : 86, 58 and 75. As there are three numbers in the series, count = 03. The count is stored in the memory location 2500 H. The numbers are stored in the memory location 2501 to 2503 H.

The result is to be stored in the memory location 2450 H.

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2000	21, 00, 25		LXI	H, 2500 H	Get address for count in H-L pair.
2003	4E		MOV	C, M	Count in register C.
2004	3E, FF		MVI	A, FF	Get FF in accumulator.
2006	23	LOOP	INX	H	Address of the next number of the series.
2007	BE		CMP	M	Compare next number wth previous smallest. Is next number < previous smallest?
200A	DA, 0C, 20		JC	AHEAD	No, smallest number is in accumulator. Go to AHEAD.
200B	7E		MOV	A, M	Yes, get smaller number in accumulator.
200C	0D	AHEAD	DCR	C	Decrement count.
200D	C2, 06, 20		JNZ	LOOP	
2010	32, 50, 24		STA	2450 H	Store result in 2450 H.
2013	76		HLT		Stop.

Example 1
DATA
2500 — 03 H
2501 — 86 H
2502 — 58 H
2503 — 75 H
Result
2450 — 58 H

Example 2
DATA
2500 — 05 H
2501 — EB
2502 — D4
2503 — 3C
2504 — 0F
2505 — A8

Result

2450 — 0F

The memory location 2500 H contains the count of the series. 2500 H is placed in H-L pair. MOV C, M transfers the count to register C. MVI A, FF brings FF into the accumulator. As FF is the largest 8-bit number, it is taken as initial number for comparison. INX H points out the address of the 1st number. CMP M compares the content of the accumulator *i.e.* FF with the 1st number (86 in Example 1). As FF is greater than 86, there is no carry. After the execution of the instruction JC, the instruction MOV A, M is executed to transfer the 1st number (86) to the accumulator. Then the count is decremented and the program jumps to the label LOOP. Now INX H points out the address of the 2nd number (58 H). As 58 H is smaller than 86 H, 58 H is transferred to the accumulator. Again the count is decremented and the program goes back to the label LOOP. This time INX H points out the 3rd number (75 H). As 58 is smaller than 75, there will be a carry. After the execution of JC the program will jump to AHEAD, and the count will be decremented. 58 H will remain in the accumulator. In this way the smallest number of the series is obtained and stored in the memory location 2450 H.

6.25 TO ARRANGE A DATA ARRAY IN ASCENDING ORDER

Example 1. Arrange E5, A9, 96, B4 and 15 in ascending order. These numbers are in the memory locations 2501 to 2505. H.

Count = 05, it is stored in 2500 H.

Results are to be stored in 2601 to 2605 H.

First of all the smallest number is selected from the given series of numbers and it is stored in 2601 H. This number is deleted from the series and FF is stored in its position. Again the smallest number is selected from the modified series. Since FF is the highest 8-bit number, it does not affect the result when the smallest number is selected second time. The smallest is stored in 2602 H. This number is also replaced by FF, and from the modified series of numbers the smallest number is selected again and stored in 2603 H. This procedure is repeated till all the numbers of the series are arranged in ascending order.

SUBROUTINE-1 is to find the smallest number in a data array. SUBROUTINE-2 is to check which number is the smallest, and to replace it by FF.

In the program three counts have been used. Count 1 is stored in register B to check whether all the numbers of the series have been arranged in ascending order. Count 2 is stored in register C to check whether all the numbers of the series have been compared while selecting the smallest number of the series. After the selection of the smallest number it is to be ascertained which number was the smallest. At this stage register C is free. The register C is again used in SUBROUTINE-2 to store count 3 to check which number has been selected as the smallest number. The smallest number is replaced by FF in the series. Again the program to find the smallest number is taken up to select the smallest number in the modified series.

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2000	11, 01, 26		LXI	D, 2601	Memory locations to store result.
2003	21, 00, 25		LXI	H, 2500	Count address in H-L pair

2006	46		MOV	B, M	Count in register B to check whether all numbers have been arranged in ascending order.
2007	CD, 00, 22	START	CALL	2200	Call Subroutine-1 to find smallest number.
200A	12		STAX	D	Store the result.
200B	CD, 50, 20		CALL	2050	Call Subroutine-2 to check which number is smallest.
200E	13		INX	D	
200F	05		DCR	B	Have all numbers been arranged in ascending order?
2010	C2, 07, 20		JNZ	START	No, repeat process.
2013	76		HLT		Stop

SUBROUTINE-1 To Find the Smallest Number

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2200	21, 00, 25		LXI	H, 2500	Count address in H-L pair.
2203	4E		MOV	C, M	Count in register C.
2204	3E, FF		MVI	A, FF	Get FF in accumulator.
2206	23	LOOP	INX	H	
2207	BE		CMP	M	Compare next number with previous smallest. Is next number < previous smallest?
2208	DA, 0C, 22		JC	AHEAD	No, smallest number is in the accumulator. Go to AHEAD.
220B	7E		MOV	A, M	Yes, get smaller number in accumulator.
220C	0D	AHEAD	DCR	C	
220D	C2, 06, 22		JNZ	LOOP	
2210	C9		RET		

SUBROUTINE-2

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2050	21, 00, 25		LXI	H, 2500 H	
2053	4E		MOV	C, M	Count to check which number was smallest.
2054	23	BEHIND	INX	H	Get next number.

2055	BE		CMP	M	Compare the next number with smallest number which is in the accumulator.
2056	CA, 5D, 20		JZ	FORWARD	Is the present number the smallest one? Yes, go to FORWARD.
2059	0D		DCR	C	Decrement count.
205A	C2, 54, 20		JNZ	BEHIND	No, jump to take up next number.
205D	3E, FF	FORWARD	MVI	A, FF	
205F	77		MOV	M, A	Replace the smallest number by FF.
2060	C9		RET		

Example 1

DATA

2500 — 05
2501 — E5
2502 — A9
2503 — 96
2504 — B4
2505 — 15

Result

2601 — 15
2602 — 96
2603 — A9
2604 — B4
2605 — E5

Example 2

DATA

2500 — 06
2501 — 45
2502 — B8
2503 — FF
2504 — E8
2505— 98
2506 — 30

Result

2601 — 30
2602 — 45
2603 — 98
2604 — B8
2605 — E8
2606 — FF

6.26 SUM OF A SERIES OF 8-BIT NUMBERS; SUM; 8-BIT

Example. Add 16, 2B, 39 and 12 H.

The numbers are placed in the memory locations 2501 to 2504 H.

The sum is to be stored in the memory location 2450 H.

As there are 4 numbers in the series, count = 04.

The initial value of the sum is made 00. The number of the series are taken one by one and added to the sum. The program flow chart is shown in Fig. 6.4.

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2400	21, 00, 25		LXI	H, 2500 H	Address for the count in H-L pair.
2403	4E		MOV	C, M	The count in register C.

2404	3E, 00		MVI	A, 00	Initial value of sum = 00.
2406	23	LOOP	INX	H	Address of next data in H-L pair.
2407	86		ADD	M	Previous sum + next number.
2408	0D		DCR	C	Decrement count.
2409	C2, 06, 24		JNZ	LOOP	Is count = 0 ? No, jump to LOOP.
240C	32, 50, 24		STA	2450 H	Store sum in 2450 H.
240F	76		HLT		Stop.

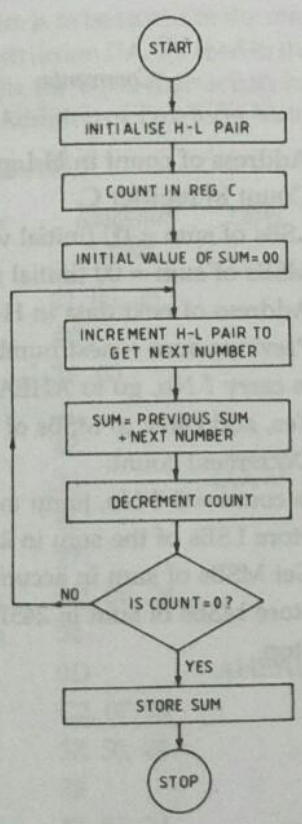

Fig. 6.4 Flow Chart to find sum of a series of 8-Bit numbers; sum : 8-bit

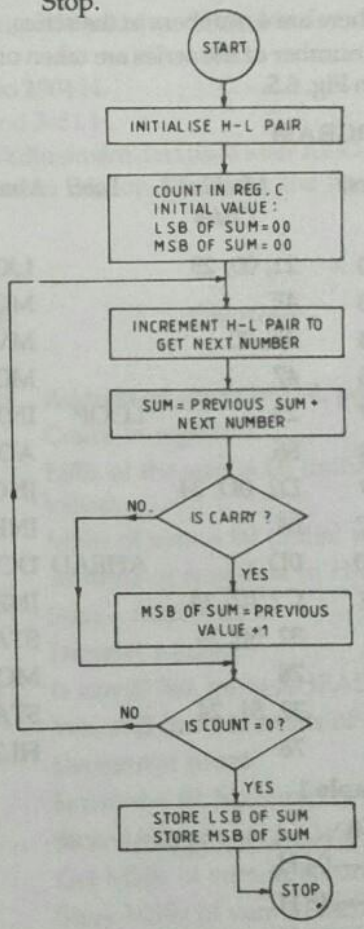

Fig. 6.5. Flow Chart to find sum of a series of 8-bit numbers; sum : 16-bit

DATA

2500 — 04
2501 — 16
2502 — 2B
2503 — 39
2504 — 12

Result

2504 — 8C

6.27 SUM OF A SERIES OF 8-BIT NUMBERS, SUM; 16-BIT

Example 1. Add 45, 98, 8A and E2 H ; these are hexadecimal numbers.

The numbers are placed in the memory locations 2501 to 2504 H, and the count in 2500 H.

The sum is to be stored in the memory locations 2450 and 2451 H.

As there are 4 numbers in the series, count = 04. The initial value of the sum is made 00.

The number of the series are taken one by one and added to the sum. The program flow chart is shown in Fig. 6.5.

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2400	21, 00, 25		LXI	H, 2500 H	Address of count in H-L pair.
2403	4E		MOV	C, M	Count in register C.
2404	3E, 00		MVI	A, 00	LSBs of sum = 00 (initial value).
2406	47		MOV	B, A	MSBs of sum = 00 (initial value).
2407	23	LOOP	INX	H	Address of next data in H-L pair.
2408	86		ADD	M	Previous sum + next number.
2409	D2, 0D, 24		JNC	AHEAD	Is carry ? No, go to AHEAD.
240C	04		INR	B	Yes, add carry to MSBs of sum.
240D	0D	AHEAD	DCR	C	Decrement count.
240E	C2, 07, 24		JNZ	LOOP	Is count = 0 ? No, jump to LOOP.
2411	32, 50, 24		STA	2450 H	Store LSBs of the sum in 2450 H.
2414	78		MOV	A, B	Get MSBs of sum in accumulator.
2415	32, 51, 24		STA	2451 H	Store MSBs of sum in 2451 H.
2418	76		HLT		Stop

Example 1

DATA

2500 — 04 H

2501 — 45 H

2502 — 98 H

2503 — 8A H

2504 — E2 H

Result

2450 — 49 H, LSBs of sum.

2451 — 02 H, MSBs of sum.

Example 2

DATA

2500 — 05 H

2501 — 98 H

2502 — 24 H

2503 — 35 H

2504 — 46 H

2505 — 39 H

Result

2450 — 70 H, LSBs of sum.

2451 — 01 H, MSBs of sum.

The count is placed in register C. The LSBs of the sum reside in the accumulator, and MSBs of the

sum in register B. Initial values of the LSBs and MSBs of the sum are made 00. The 1st number is added to the initial sum. Count is decreased by one and the program jumps to LOOP. The address of the 2nd number is placed in H-L pair. ADD M adds 2nd number to the previous sum which is in the accumulator. If there is any carry after addition, that is stored in register B. This is nothing but the MSB of the sum. The process is repeated and the remaining numbers of the series are added. The LSBs of the sum is stored in memory location 2450 H and MSBs of the sum in 2451 H.

6.28 SUM OF A SERIES OF 8-BIT DECIMAL NUMBERS, SUM: 16-BIT

Example 1. Add 65, 46, 35 and 98 D, these are decimal numbers. D stands for decimal. Sum = 0244 D

The numbers are placed in the memory locations 2501 to 2504 H.

The sum is to be stored in the memory locations 2450 and 2451 H.

The instruction DAA is used in the program for decimal adjustment. It is used after ADD instruction. Details for DAA instruction have already been given in Section 6.6 under the heading of "Decimal Addition of Two 8-Bit Number."

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2400	21, 00, 25		LXI	H, 2500 H	Address of count in H-L pair.
2403	4E		MOV	C, M	Count in register C.
2404	3E, 00		MVI	A, 00	LSBs of the sum = 00 (initial value).
2406	47		MOV	B, A	MSBs of sum = 00 (initial value).
2407	23	LOOP	INX	H	Address of next data in H-L pair.
2408	86		ADD	M	Sum = Sum + data.
2409	27		DAA		Decimal adjust.
240A	D2, 0E, 24		JNC	AHEAD	Is carry? No, go to AHEAD.
240D	04		INR	B	Yes, add carry to MSBs of sum.
240E	0D	AHEAD	DCR	C	Decrement count.
240F	C2, 07, 24		JNZ	LOOP	Is count = 0? No, jump to LOOP.
2412	32, 50, 24		STA	2450 H	Store LSBs of sum in 2450 H.
2415	78		MOV	A, B	Get MSBs of sum in accumulator.
2416	32, 51, 24		STA	2451 H	Store MSBs of sum in 2451 H.
2419	76		HLT		Stop.

Example 1

DATA

2500 — 04 H
2501 — 65 D
2502 — 46 D
2503 — 35 D
2504 — 98 D

Example 2

DATA

2500 — 05 H
2501 — 96 D
2502 — 98 D
2503 — 85 D
2504 — 89 D

Result	2505 — 93 D
2450 — 44 D, LSDs of the sum.	Sum = 0461
2451 — 02 D, MSDs of the sum.	2450 — 61 D, LSDs of the sum.
	2451 — 04 D, MSDs of the sum.

The program is exectly same as that for "sum of a series of 8-Bit Numbers, sum 16-bit," Section 6.27; except that the instruction DAA has been incorporated after the instruction ADD M. The instruction ADD M gives the sum in hexadecimal system. The instruction DAA makes correction and gives the result in decimal system. Similarly the program for "sum of a series of 8-bit decimal numbers, sum 8-bit" can be obtained incorporting the instruction DAA in the program for "sum of a series of 8-bit Numbers, sum 8-bit" given in Section 6.26.

6.29 8-BIT MULTIPLICATION ; PRODUCT 16-BIT

Decimal Multiplication. For decimal multiplication the following procedure is followed :

Example

```
      36 D, Multiplicand
    × 29 D, Multiplier
    -------------------
      32  4
      72
    -------------------
Product:  104  4 D
```

D stands for decimal number

$$36 \times 29 = 9 \times 36 + 20 \times 36$$

$$324 + 720 = 1044 \text{ D.}$$

First of all the multiplicand 36 is multiplied by 9 and the result 324 is written on the paper. Again 36 is multipled by 2 and the result 72 is shifted left to make it 720. 324 and 720 are added. Either 72 is shifted left or 324 right with respect to 72 the final result will be same.

Binary Multiplication

Example. Multiply 7 by 5

First of all 7 and 5 are represented in binary form;

```
    0111 = 7, Multiplicand
  × 0101 = 5, Multiplier
  ----------------------
    0111
   0000
  0111
 0000
  ----------------------
Product = 010, 0011 = 35
```

It should be noted that the above product is in the binary form. It is not in BCD.

In binary multiplication we see that when a multiplicand is multipled by 1 the product is equal to the multiplicand. When a multiplicand is multiplied by zero, the product is zero. The procedure for multiplication is that first the multiplicand is multipled by the LSB of the multiplier and the partial product is stored and shifted right. Again the multiplicand is multipled by the 2nd bit and the result is added to the previous shifted partial product. The procedure is repeated. If the bit of the multiplier is 1 the multiplicand is added to the previous partial product. In case of 0 bit there is

nothing to be added to the partial product but it will be simply shifted right by one bit. In case of binary multiplication by computer if the partial product is shifted left instead of right, and we take bits of multiplier from MSB side instead of LSB side the final product will remain same.

Conclusion. The conclusion is that each bit of multiplier is taken one by one and it is checked whether it is 1 or 0. If the bit of the multipler is one the multiplicand is added to the product and the product is shifted left. If the bit of multipler is zero, the product is simply shifted left by one bit.

Example 1. Multiply 84 H by 56 H.

In this example multiplicand is 84 H. It is extended to 16-bits and stored in the two consecutive memory locations 2501 and 2502 H. The multiplier is 56 H and it is stored in 2503 H. The product is a 16-bit number and it is stored in 2504 H and in 2505 H. Data and results are in hexadecimal. The program flow chart is shown in Fig. 6.6.

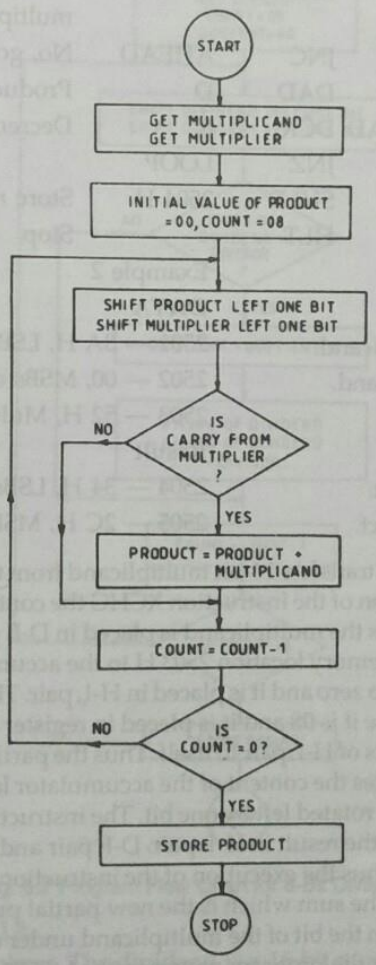

Fig. 6.6 Program Flow Chart for 8-Bit Multiplication

8-bit multiplication; product 16-bit

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2000	2A, 01, 25		LHLD	2501 H	Get multiplicand in H-L pair.
2003	EB		XCHG		Multiplicand in D-E pair.
2004	3A, 03, 25		LDA	2503 H	Multiplier in accumulator.
2007	21, 00, 00		LXI	H, 0000	Initial value of product = 00 in H-L pair.
200A	0E, 08		MVI	C, 08	Count = 8 in register C.
200C	29	LOOP	DAD	H	Shift partial product left by 1 bit.
200D	17		RAL		Rotate multiplier left one bit. Is multipler's bit = 1?
200E	D2, 12, 20		JNC	AHEAD	No, go to AHEAD.
2011	19		DAD	D	Product = Produt + Multiplicand.
2312	0D	AHEAD	DCR	C	Decrement count.
2013	C2, 0C, 20		JNZ	LOOP	
2016	22, 04, 25		SHLD	2504 H	Store result.
2019	76		HLT		Stop

Example 1

DATA

2501 — 84 H, LSBs of multiplicand.

2502 — 00, MSBs of multiplicand.

2503 — 56 H, Multiplier.

Result

2504 — 58 H, LSBs of product.

2505 — 2C H, MSBs of product.

Example 2

DATA

2501 — 8A H, LSBs of multiplicand.

2502 — 00, MSBs of multiplicand.

2503 — 52 H, Multiplier.

Result

2504 — 34 H, LSBs of product.

2505 — 2C H, MSBs of product.

The instruction LHLD 2501 H transfers 16-bit multiplicand from the memory locations 2501 and 2502 H to H-L pair. By the execution of the instruction XCHG the contents of H-L pair are exchanged with the contents of D-E pair. Thus the multiplicand is placed in D-E pair. The instruction LDA 2503 H transfers multiplier from the memory location 2503 H to the accumulator. LXI H, 0000 makes the initial value of the product equal to zero and it is placed in H-L pair. The count is equal to the number of bits of the multiplier. In this case it is 08 and it is placed in register C. DAD H is an instruction for 16-bit addition. It adds the contents of H-L pair to itself. Thus the partial product which is in H-L pair is shifted left by one bit. RAL rotates the content of the accumulator left by one bit. The accumulator contains multiplier and hence it is rotated left by one bit. The instruction DAD D adds the content of D-E pair and H-L pair and places the result in H-L pair. D-E pair and H-L pair contain multiplicand and partial product respecively. Thus the execution of the instruction DAD D adds the multiplicand to the partial product and places the sum which is the new partial product in H-L pair. The instruction DAD D is executed only when the bit of the multiplicand under consideration is one, otherwise it is not executed. To get the result the program moves in the LOOP 8 times as there are 8-bit in the multiplier.

6.30. 8-BIT DIVISION

The computer performs division by trial subtractions. The divisor is subtracted from the 8 most significant bits of the dividend. If there is no borrow, the bit of the quotient is set to 1; otherwise 0. To line up the dividend and quotient properly the dividend is shifted left by one bit before each trial of subtraction. The dividend and quotient share a 16-bit register. Due to shift of dividend one bit of the register falls vacant in each step. The quotient is stored in vacant bit positions. The program flow chart is shown in Fig. 6.7.

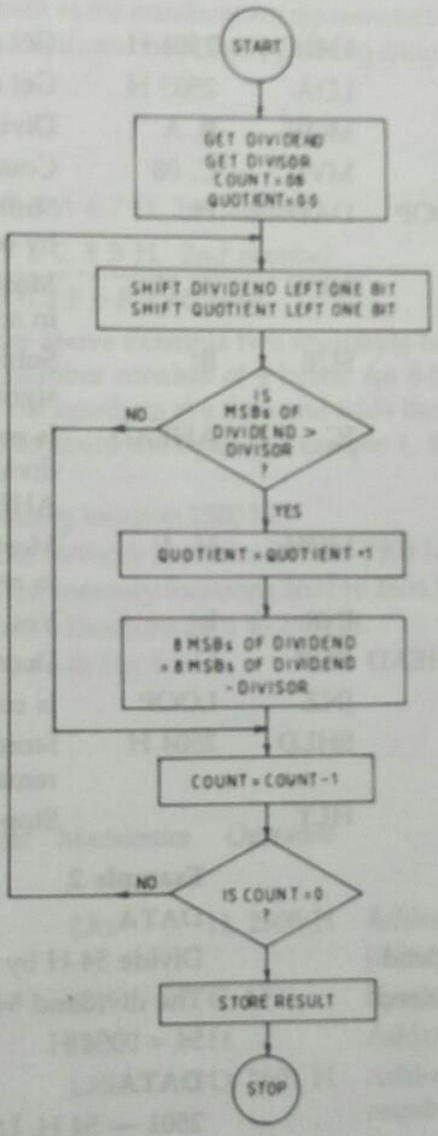

Fig. 6.7 Program Flow Chart for 8-Bit Division.

Example 1. Divide 489B by 1A.

These are hexadecimal numbers. The dividend is a 16-bit number and divisor 8-bit number. If the dividend of a problem is an 8-bit number, it is extended to a 16-bit number by placing zeros in MSBs positions.

The dividend is placed in the memory locations 2501 and 2502 H.

The divisor is placed in the memory location 2503 H.
The results are stored in the memory locations 2504 and 2505 H.
The quotient is stored in the memory locations 2504 H.
The remainder is stored in the memory location 2505 H.

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2400	2A, 01, 25		LHLD	2501 H	Get dividend in H-L pair.
2403	3A, 03, 25		LDA	2503 H	Get divisor from 2503 H.
2406	47		MOV	B, A	Divisor in register B.
2407	0E, 08		MVI	C, 08	Count = 08 in register C.
2409	29	LOOP	DAD	H	Shift dividend and quotient left by one bit.
240A	7C		MOV	A, H	Most significant bits of dividend in accumulator.
240B	90		SUB	B	Subtract divisor from most significant bits of dividend.
240C	DA, 11, 24		JC	AHEAD	Is most siginificant part of dividend > divisor? No, go to AHEAD.
240F	67		MOV	H, A	Most significant bits of dividend in register H.
2410	2C		INR	L	Yes, add 1 to quotient.
2411	0D	AHEAD	DCR	C	Decrement count.
2412	C2, 09, 24		JNZ	LOOP	Is count = 0? No, jump to LOOP.
2415	22, 04, 25		SHLD	2504 H	Store quotient in 2504 and remainder in 2505 H.
2418	76		HLT		Stop.

Example 1

DATA

2501 — 9B H, LSBs of dividend.
2502 — 48 H, MSBs of dividend.
2503 — 1A H, Divisor.

Result

2504 — F2, Quotient.
2505 — 07, Remainder.

Example 2

DATA

Divide 54 H by 09.
The dividend 54 is extended to 16 bits.
54 = 0054 H

DATA

2501 — 54 H, LSBs of dividend.
2502 — 00, MSBs of dividend.
2503 — 09, Divisor.

Result

2504 — 09, Quotient
2505 — 03, Remainder

The count in register C is kept 08. The trial subtraction is done 8 times and an 8-bit quotient is obtained. The instruction DAD H shifts dividend and quotient left by one bit. Due to shift of dividend the bit positions in register L fall vacant. In the vacant bit positions quotient is stored. Note that the dividend is shifted prior to trial subtraction. The MSB of the dividend should be zero, otherwise it will be shifted to carry bit. If a problem contains MSB not equal to zero, it will be solved by splitting it in two parts. Shifting of dividend before subtraction is not done in ordinary division by pen and paper, but the computer method gives correct result as the numbers are represented in binary coded hexadecimal system. You can check this by taking simple numerical examples.

6.31 MULTIBYTE ADDITION

Example 1.

3 A 9 C 8 A 6 7 H, 1st number
+ 9 B 4 7 6 C 8 B H, 2nd number
D 5 E 3 F 6 F 2 H, Sum

A byte consists of 8-bits. In the above example two multibyte hex numbers are to be added. Each number consists of 4 bytes. An 8-bit microcomputer takes one byte of the numbers at a time and adds them with carry. A counter is initiated to count the byte. In Example 1, the count = 4

The count is placed in the memory location 2500 H.
The 1st number is placed in the memory locations 2501 to 2504 H.
The 2nd number is placed in the memory locations 2601 to 2604 H.
The sum is placed in the memory locations 2501 to 2504 H.
The program flow chart is shown in Fig. 6.8.

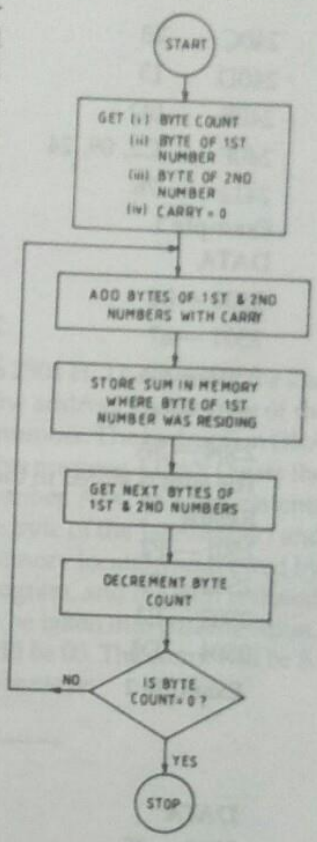

Fig. 6.8 Program Flow Chart for Multibyte Addition.

PROGRAM

Address	*Machine Codes*	*Label*	*Mnemonics*	*Operands*	*Comments*
2400	21, 00, 25		LXI	H, 2500 H	Address of byte count in H-L pair.
2403	4E		MOV	C, M	Byte count in register C.
2404	23		INX	H	Address of 1st byte of 1st number.
2405	11, 01, 26		LXI	D, 2601 H	Address of 1 st byte of 2nd number.
2408	B7		ORA	A	Clear carry.
2409	1A	LOOP	LDAX	D	Get byte of 2nd number in accumulator.
240A	8E		ADC	M	Byte of 2nd number + byte of 1st number + carry.

240B	77	MOV	M, A	Store sum in memory addressed by H-L pair.
240C	23	INX	H	Increment the content of H-L pair.
240D	13	INX	D	Increment the content of D-E pair
240E	0D	DCR	C	Decrement count.
240F	C2, 09, 24	JNZ	LOOP	Is count = 0? No, jump to LOOP.
2412	76	HLT		Stop.

Example 1

DATA

2500 — 04
2501 — 67
2502 — 8A
2503 — 9C
2504 — 3A
2601 — 8B
2602 — 6C
2603 — 47
2604 — 9B

The sum is stored in the memory locations 2501 to 2504 H.

Result

2501 — F2
2502 — F6
2503 — E3
2504 — D5

Example 2

00B8968B7E
009C8A5C46
015520E7C4

DATA

2500 — 05
2501 — 7E
2502 — 8B
2503 — 96
2504 — B8
2505 — 00
2601 — 46
2502 — 5C
2603 — 8A
2604 — 9C
2605 — 00

Result

2501 — C4
2502 — E7
2503 — 20
2504 — 55
2505 — 01

Example 3

00156987 H
00982142 H
00AD8AC9 H

DATA

2500 — 04
2501 — 87
2502 — 69
2503 — 15
2504 — 00

2601 — 42
2602 — 21
2603 — 98
2604 — 00

Result

2501 — C9
2502 — 8A
2503 — AD
2504 — 00

The byte of 1st number are placed in the memory locations 2501 to 2504 H. The bytes of the 2nd number are in 2601 to 2604 H. H-L pair has been initiated to contain the address for the byte of the 1st number and D-E pair to contain the address for the byte of the 2nd number. The instruction ORA A clears the carry bit. Any other logical operation can also be used for this purpose. LDAX D gets the content of the memory addressed by D-E pair *i.e.* the byte of the 2nd number. ADC M adds contents of the accumulator, contents of the memory addressed by H-L pair (the byte of the 1st number) and the carry of the previous byte addition. MOV M, A stores the sum in memory location addressed by H-L pair. Thus the 1st number is destroyed after the execution of the program, and the sum is stored in its place. If there is any carry after the addition of the last byte, it can be taken into consideration. This can be done by extending the numbers to 5th byte. The 5th byte will be 00. The count will be 5. The 5th byte of the sum will be 01. It is better to extend numbers one byte more.

6.32 MULTIBYTE DECIMAL ADDITION

Example 1

0096876958	1st number
0087358769	2nd number
0184235727	Sum

In this example there are two four-byte decimal numbers. Each number has been extended to 5th byte.

The count is stored in the memory location 2500 H.
The bytes of the 1st number are placed in the memory locations 2501 to 2505 H.
The bytes of the 2nd numbers are stored in the memory locations 2601 to 2605 H.
The sum is stored in the memory locations 2501 to 2505 H.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2400	21, 00, 25		LXI	H, 2500 H	Address of count in H-L pair.
2403	4E		MOV	C, M	Count in register C.
2404	23		INX	H	Address of 1st byte of 1st number.

2405	11, 01, 26		LXI	D, 2601 H	Address of 1st byte of 2nd number.
2408	B7		ORA	A	Clear carry.
2409	1A	LOOP	LDAX	D	Get byte of 2nd number in accumulator.
240A	8E		ADC	M	Byte of 2nd number + byte of 1st number + carry.
240B	27		DAA		Decimal adjust.
240C	77		MOV	M, A	Store sum in memory addressed by H-L pair.
240D	13		INX	D	Increment D-E pair.
240E	23		INX	H	Increment H-L pair.
240F	0D		DCR	C	Decrement count.
2410	C2, 09, 24		JNZ	LOOP	Is count 00? No, go to LOOP.
2413	76		HLT		Stop.

The program is exactly same as that for "Multibyte Addition" Section 6.31, except that DAA instruction has been incorporated and placed after instruction ADC M. After the execution of the program 1st number is destroyed. DAA also produces carry which is taken into account while adding the next byte by the instruction ADC M. Details of DAA are given in Section 6.6, under the heading "Decimal addition of Two 8-Bit Numbers."

Example 1

DATA

2500 — 05
2501 — 58 2601 — 69
2502 — 69 2602 — 87
2503 — 87 2603 — 35
2504 — 96 2604 — 87
2505 — 00 2605 — 00

Sum is stored in the memory locations 2501 to 2505 H as given below :

SUM

2501 — 27
2502 — 57
2503 — 23
2504 — 84
2505 — 01

Example 2

0035879984
0057246389
0093126373

DATA

2500 — 05
2501 — 84 2601 — 89

2502 — 99
2503 — 87
2504 — 35
2505 — 00

2602 — 63
2603 — 24
2604 — 57
2605 — 00

SUM

2501 — 73
2502 — 63
2503 — 12
2504 — 93
2505 — 00

6.33 SUM OF A SERIES OF MULTIBYTE DECIMAL NUMBERS

Example:

$$\begin{array}{r} 0024568945 \\ +\ 0054863564 \\ +\ 0032718907 \\ \hline 0112151416 \end{array}$$

In this example 3 four-byte decimal numbers are to be added. Each number has been extended to 5th byte. To solve such problem two counts are required. One of them counts the length of the series. As there are 3 numbers, so length of the series is 03. The other count is for the number of bytes of the numbers. As each number has been extended to 5th byte, the byte count is = 05.

First of all initial sum is made zero. The 1st number is added to this. The process of addition is repeated and other numbers are added to the previous sum one by one.

The count for the length of series is placed in the memory location 24FF.

The count for bytes is placed in the memory location 2500 H.

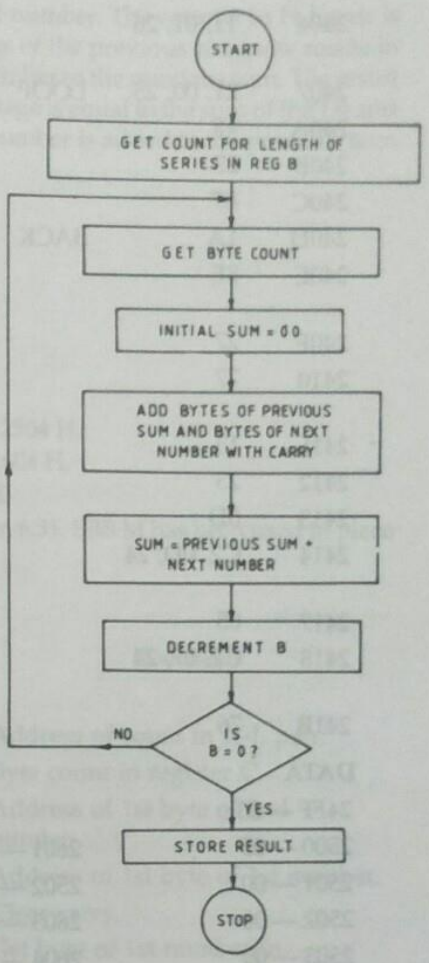

Fig. 6.9. Program Flow Chart for Sum of a Series of Multibyte Decimal Numbers.

The initial sum = 00 is placed in the memory locations from 2501 to 2505 H.

The 1st number is placed in the memory locations from 2601 to 2605 H.

The 2nd number from the memory locations 2606 to 260A.

The 3rd number from the memory locations 260B to 260F. The program flow chart is shown in Fig. 6.9.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2400	3A, FF, 24		LDA	24FF H	Length of the series.
2403	47		MOV	B, A	Length of series in B.

2404	11, 01, 26		LXI	D, 2601 H	Address for 1st byte of 1st number.
2407	21, 00, 25	LOOP	LXI	H, 2500 H	Address for byte count.
240A	4E		MOV	C, M	Byte count in C.
240B	23		INX	H	Get byte of the previous sum.
240C	B7		ORA	A	Clear carry.
240D	1A	BACK	LDAX	D	Get byte of next number.
240E	8E		ADC	M	Add byte of next number to byte of previous sum.
240F	27		DAA		Decimal adjust.
2410	77		MOV	M, A	Store result in memory addressed by H-L pair.
2411	13		INX	D	Increment D-E pair.
2412	23		INX	H	Increment H-L pair.
2413	0D		DCR	C	Decrement byte count.
2414	C2, 0D, 24		JNZ	BACK	Is byte count zero? No, go to BACK.
2417	05		DCR	B	Decrement count for series length.
2418	C2, 07, 24		JNZ	LOOP	Are all numbers of series added? No, go to LOOP.
241B	76		HLT		Stop.

DATA

24FF — 03
2500 — 05
2501 — 00
2502 — 00
2503 — 00
2504 — 00
2505 — 00
2601 — 45
2502 — 89
2603 — 56
2604 — 24
2605 — 00
2606 — 64
2607 — 35
2608 — 86
2609 — 54
260A — 00
260B — 07
260C — 89
260D — 71
260E — 32
260F — 00

Result

2501 — 16
2502 — 14
2503 — 15
2504 — 12
2505 — 01

The initial sum is made zero and it is stored in memory locations from 2501 to 2505 H. When the program moves from 240D to 2416, the 1st number from memory locations 2601 to 2605 H are added to the initial sum. The result is stored in memory locations 2501 to 2505 H. At this stage sum is equal to the 1st number as the previous sum was zero. Now the series length in register B is decreased from 03 to 02. It means that 2 more numbers are to be added. The program jumps to the label LOOP at the address 2407 H and the process for addition of the 2nd number is started. Now the counter in D-E

pair stands at 2606 H, so it will take up the bytes of the second number. The counter in H-L pair is again initiated from 2500 H to take up the byte count. The bytes of the previous sum now reside in 2501 to 2505 H. The program moves further and adds the 2nd number to the previous sum. The result is stored in memory locations 2501 to 2505 H. The result at this stage is equal to the sum of the 1st and 2nd numbers. The process is repeated once again and the 3rd number is added to the previous sum to get the final result.

6.34 MULTIBYTE SUBTRACTION

Example 1

EC8B945A, 1st number
− 4B3CA8B9, 2nd number
A14EEBA1

The count is placed in the memory location 2500 H.
The 2nd number is placed in the memory locations 2501 to 2504 H.
The 1st number is placed in the memory locations 2601 to 2604 H.
The result is placed in the memory locations 2501 to 2504 H.
The program is similar to that of multibyte addition, Section 6.31. SBB M has been used in place in ADC M.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2400	21, 00, 25		LXI	H, 2500 H	Address of count in H-L pair.
2403	4E		MOV	C, M	Byte count in register C.
2404	23		INX	H	Address of 1st byte of 2nd number.
2405	11, 01, 26		LXI	D, 2601 H	Address of 1st byte of 1st number.
2408	B7		ORA	A	Clear carry.
2409	1A	LOOP	LDAX	D	Get byte of 1st number in accumulator.
240A	9E		SBB	M	Subtract byte of 2nd number from byte of 1st number with borrow.
240B	77		MOV	M, A	Store result in memory addressed by H-L pair.
240C	23		INX	H	Increment H-L pair.
240D	13		INX	D	Increment D-E pair.
240E	0D		DCR	C	Decrement count.
240F	C2, 09, 24		JNZ	LOOP	Is count = 0? No, go to LOOP.
2412	76		HLT		Stop

Example 1
DATA
2500 — 04
2501 — B9 2601 — 5A

2502 — A8 2602 — 94
2503 — 3C 2603 — 8B
2504 — 4B 2604 — EC

Result

2501 — A1
2502 — EB
2503 — 4E
2504 — Al

Example 2

976354 H
– 758238 H
21E11C H

DATA

2500 — 03
2501 — 38 2601 — 54
2502 — 82 2602 — 63
2503 — 75 2603 — 97

Result

2501 — 1C
2502 — E1
2503 — 21

In case of multibyte subtraction it should be noted that the number which is to be subtracted is placed in the memory locations addressed by H-L pair.

6.35 MULTIBYTE DECIMAL SUBTRACTION

It has already been explained under the heading of "8-Bit Decimal Subtraction" that DAA is not used after SUB or SBB instruction for decimal adjustment. It is used only after ADD, ADI, ACI or ADC instruction. Therefore, for decimal subtraction the number which is to be subtracted is converted into 10's complement and then it is added to the other number.

For multibyte subtraction the least significant byte is subtracted using 10's complemnet. In the subtraction of other bytes borrow is also taken into consideration. Now see Example 1, for the subtraction of the 1st byte no borrow is required from the 2nd byte. The application of 10's complement and DAA technique gives correct result and finally produce a carry bit. For the subtraction of 2nd byte, the carry bit is added to 9's complement of 37. Thus 10's complement is obtained and the technique gives correct result. There is no borrow from 3rd byte for subtraction of 2nd byte. For the subtraction of 3rd byte there is borrow from 4th byte (see ordinary decimal subtraction in Example 1). The application of 10's complement does not produce any carry in case of 3rd byte subtraction. For the subtraction of 4th byte 0 is added to 9's complement of 25 because there is no carry from 3rd byte operation. This technique gives correct result for 4th byte subtraction. Actually in this byte also there is 10's complement but due to borrow the final result looks like 9's complement. For 4th byte subtraction this can be explained as follows:

10's complement of 25 –1 (Correction for borrow)
= (9's complement of 25 + 1) – 1
= 9's complement.

Example 1

	4	3	2	1	Bytes
	86	87	38	65	Ordinary decimal
−	25	96	37	54	subtraction
	60	91	01	11	

10's complement and DAA technique for each byte is shown below :

1st Byte of Subtraction

Subtract 1st byte of 2nd number from 99	99	
	− 54	
9's complement of 54	45	
	+ 01	
10's complement of 54	46	
Add 1st byte of 1st number	65	
	A B	result in hexadecimal
DAA correction	+ 6 + 6	
	1 1	
carry	1	

2nd Byte Subtraction

Subtract 2nd byte of 2nd number from 99	99
	− 37
9's complement of 37	62
Add carry from previous stage	+ 01
10's complement of 37	63
Add 2nd byte of 1st number	+ 38
	9B
DAA correction	+6
	01
carry	1

3nd Byte Subtraction

Subtract 3rd byte of 2nd number from 99	9 9
	− 9 6
9's complement of 96	0 3
Add carry from previous operation	+ 1
10's complement of 96	0 4
Add 3rd byte of 1st number	+ 8 7
	9 1
DAA correction	+ 0 0
	9 1
carry	0

4th Byte subtraction

Subtract 4th byte of 2nd number from 99	99
	− 25
	74
9's complement of 25	
Add carry from previous operation	+ 0
	74
9's complement of 25	
Add 4th byte of 1st number	+ 86
	FA
DAA correction	+ 6+6
	60
carry	1

Note. The last carry is not considered in the result. The carry are being considered only in taking 10's complement of the number which is to be subtracted. This has been explained as the basis to develop a program for multibyte decimal subtraction.

Conclusion. From example 1 it is clear that if there is any borrow from a byte (to the adjacent byte in previous operation), only 9's complement of the byte which is to be subtracted, gives correct result. If there is a borrow from the adjacent byte, 10's complement and DAA operation does not produce any carry, otherwise there will be a carry. Thus ADC M instruction at memory location 201F in the program satisfies above requirements. The program flow chart is shown in Fig. 6.10.

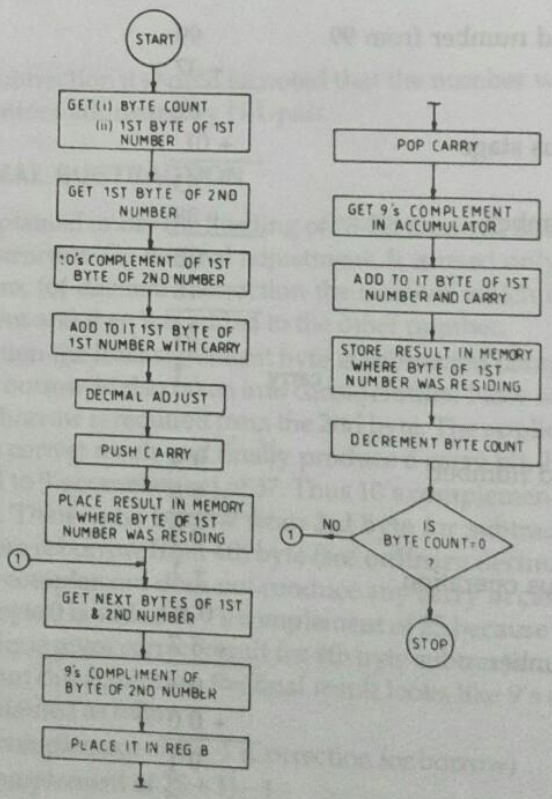

Fig. 6.10. Program Flow Chart for Multibyte Decimal Subtraction.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	31, 00, 27		LXI	SP, 2700	Initialise stack pointer.
2003	21, 00, 25		LXI	H, 2500 H	Address for Byte count in H-L pair.
2006	4E		MOV	C, M	Byte count in register C.
2007	23		INX	H	Address for 1st byte of 1st number.
2008	11, 01, 26		LXI	D, 2601 H	Address for 1st byte of 2nd number in D-E pair.
200B	1A		LDAX	D	1st byte of 2nd number in accumulator.
200C	47		MOV	B, A	Get it in register B.
200D	3E, 99		MVI	A, 99	Get 99 in accumulator.
200F	90		SUB	B	Take 9's complement of 1st byte.
2010	3C		INR	A	10's complement.
2011	86		ADD	M	Add it to 1st byte of 1st number.
2012	27		DAA		Decimal adjust.
2013	F5		PUSH	PSW	Save carry.
2014	77		MOV	M, A	Store result.
2015	23	LOOP	INX	H	Next byte of 1st number.
2016	13		INX	D	Address for next byte of 2nd number.
2017	1A		LDAX	D	Next byte of 2nd number in accumulator.
2018	47		MOV	B, A	Get it in register B.
2019	3E,99		MVI	A, 99	99 in accumulator.
201B	90		SUB	B	Take its 9's complement.
201C	47		MOV	B, A	
201D	F1		POP	PSW	Get carry of previous byte calculation.
201E	78		MOV	A, B	9's complement in accumulator.
201F	8E		ADC	M	Add to it byte of 1st number and carry.
2020	27		DAA		Decimal adjust.
2021	F5		PUSH	PSW	Save carry.
2022	77		MOV	M, A	Store result.
2023	0D		DCR	C	Decrement count.
2024	C2, 15, 20		JNZ	LOOP	
2027	76		HLT		Stop

Example 1

DATA	DATA	Result
2500 — 04		
2501 — 65	2601 — 54	2501 — 11
2502 — 38	2602 — 37	2502 — 01
2503 — 87	2603 — 96	2503 — 91
2504 — 86	2604 — 25	2504 — 60

Example 2

DATA	DATA	Result
2500 — 04		
2501 — 93	2601 — 23	2501 — 70
2502 — 28	2602 — 19	2502 — 09
2503 — 03	2603 — 85	2503 — 18
2504 — 85	2604 — 14	2504 — 70

Example 3

57849900
−28975899
28874001

DATA	DATA	Result
2500 — 04		
2501 — 00	2601 — 99	2501 — 01
2502 — 99	2602 — 58	2502 — 40
2503 — 84	2603 — 97	2503 — 87
2504 — 57	2604 — 28	2504 — 28

Example 4

2500 — 04		
2501 — 90	2601 — 09	2501 — 81
2502 — 35	2602 — 57	2502 — 78
2503 — 00	2603 — 00	2503 — 99
2504 — 87	2604 — 42	2504 — 44

6.36 TO FIND SQUARE-ROOT OF A NUMBER

Suppose that X is the square root of number N. To find a suitable equation to be used by the computer for iteration the following manipulation is done.

$$X^2 = N$$

or $$2X^2 = N + X^2$$

or $$X^2 = \frac{N + X^2}{2}$$

or $$X = \frac{N + X^2}{2X}$$

or $$X = \left(\frac{N}{X} + X\right) / 2$$

or $$X_{new} = \left(\frac{N}{X} + X\right) / 2$$

To find the square root of a given number we provide an initial value of the root, which may be very approximate. In the above equation X is the initial value of the root given by the programmer. The computer calculates X_{new} and compares it with X. When $X_{new} = X$, it gives the result = X_{new}. A program is given below. To find the square root within certain tolerance, limits for comparison may also be provided and approximate square root can be obtained. The program given below is for integer number.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2600	3E, X		MVI	A, X	X is the first approximation.
2602	57	BACK	MOV	D, A	
2603	2A, 00, 25		LHLD	2500	N in H-L pair.
2606	CD, 06, 24		CALL	2406	Division Subroutine, N/X in register L.
2609	7A		MOV	A, D	X in accumulator.
260A	85		ADD	L	Get (X + N/X)
260B	6F		MOV	L, A	
260C	26, 00		MVI	H, 00	
260E	3E, 02		MVI	A, 02	
2610	CD, 06, 24		CALL	2406	(N/X + X)/2 in L = X_{new}.
2613	7D		MOV	A, L	
2614	BA		CMP	D	Compare X_{new} with X
2615	CA, 1B, 26		JZ	BELOW	Jump to BELOW, if $X_{new} = X$
2618	C3, 02, 26		JMP	BACK	Jump to BACK, if $X_{new} \neq X$
261B	32, 50, 25	BELOW	STA	2550	
261E	76		HLT		Stop

Note. This program is valid for numbers upto 3F01 hex or 16129 decimal. *i.e.* N ≤ 3F01.

Division-Subroutine

2406	4F		MOV	C, A
2407	06, 08		MVI	B, 08
2409	29	DIV	DAD	H
240A	7C		MOV	A, H
240B	91		SUB	C
240C	DA, 11, 24		JC	AHEAD

or $$X = \left(\frac{N}{X} + X\right) / 2$$

or $$X_{new} = \left(\frac{N}{X} + X\right) / 2$$

To find the square root of a given number we provide an initial value of the root, which may be very approximate. In the above equation X is the initial value of the root given by the programmer. The computer calculates X_{new} and compares it with X. When $X_{new} = X$, it gives the result = X_{new}. A program is given below. To find the square root within certain tolerance, limits for comparison may also be provided and approximate square root can be obtained. The program given below is for integer number.

PROGRAM

Address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2600	3E, X		MVI	A, X	X is the first approximation.
2602	57	BACK	MOV	D, A	
2603	2A, 00, 25		LHLD	2500	N in H-L pair.
2606	CD, 06, 24		CALL	2406	Division Subroutine, N/X in register L.
2609	7A		MOV	A, D	X in accumulator.
260A	85		ADD	L	Get (X + N/X)
260B	6F		MOV	L, A	
260C	26, 00		MVI	H, 00	
260E	3E, 02		MVI	A, 02	
2610	CD, 06, 24		CALL	2406	(N/X + X)/2 in L = X_{new}.
2613	7D		MOV	A, L	
2614	BA		CMP	D	Compare X_{new} with X
2615	CA, 1B, 26		JZ	BELOW	Jump to BELOW, if $X_{new} = X$
2618	C3, 02, 26		JMP	BACK	Jump to BACK, if $X_{new} \neq X$
261B	32, 50, 25	BELOW	STA	2550	
261E	76		HLT		Stop

Note. This program is valid for numbers upto 3F01 hex or 16129 decimal. *i.e.* N ≤ 3F01.

Division-Subroutine

2406	4F		MOV	C, A
2407	06, 08		MVI	B, 08
2409	29	DIV	DAD	H
240A	7C		MOV	A, H
240B	91		SUB	C
240C	DA, 11, 24		JC	AHEAD

240F	67		MOV	H, A
2410	2C		INR	L
2411	05	AHEAD	DCR	B
2412	C2, 09, 24		JNZ	DIV
2415	22, 03, 25		SHLD	2503
2418	C9		RET	

Example 1
DATA
2500 — 10 H
2501 — 00
Result
2550 — 04 H

Example 2
DATA
2500 — 09 H
2501 — 04
Result
2550 — 23 H

Example 3
DATA
2500 — 19 H
2501 — 00
Result
2550 — 05 H

6.37 MOVE A BLOCK OF DATA FROM ONE SECTION OF MEMORY TO ANOTHER SECTION OF MEMORY

PROGRAM

2400	21,00,20	LXI H, 2000H	; Get memory address of count
2403	4E	MOV C, M	; Count in Register C
2404	23	INX H	; Source address of data in H-L pair.
2405	11,01,22	LXI D, 2201	; Destiny address of data in D-E pair
2408	7E LOOP	MOV A, M	; Data from source address to ACC
2409	EB	XCHG	; Destiny address in H-L pair
240A	77	MOV M, A	; Data to destiny address
240B	EB	XCHG	; Source address in H-L pair
240C	23	INX H	; Source address of next data
240D	13	INX D	; Destiny address of next data
240E	0D	DCR C	; Decrement count
240F	C2, 08, 24	JNZ LOOP	; Jump to label LOOP
2412	76	HLT	; Stop

DATA	**Result**
2000–05	
2001–01	2201–01
2002–02	2202–02
2003–03	2203–03
2004–04	2204–04
2005–05	2205–05

Alternative Program

PROGRAM

2400	11,00,20	LXI D, 2000 H	; Get memory address of count
2403	1A	LDAX D	; Count in ACC
2404	4F	MOV C, A	; Count in Register C
2405	13	INX D	; Source address of data in D-E pair
2406	21,01,22	LXI H, 2201 H	; Destiny address in H-L pair
2409	1A LOOP	LDAX D	; Data from source address to ACC
240A	77	MOV M, A	; Data to destiny address
240B	23	INX H	; Destiny address of next data
240C	13	INX D	; Source address of next data
240D	0D	DCR C	; Decrement Count
240E	C2,09,24	JNZ LOOP	; Jump to Label LOOP
2411	76	HLT	; Stop

DATA	Result
2000–05	
2001–01	2201–01
2002–02	2202–02
2003–03	2203–03
2004–04	2204–04
2005–05	2205–05

6.38 SIGNED ARITHMETIC OPERATION

In signed arithmetic operation, signed binary numbers are used. To represent the sign of a number, the MSB is used as a sign bit. It represents sign of a number, and the remaining bits of the data word are used to represent the magnitude of the number. When a number is positive, the sign bit is kept 0. When the number is negative, the sign bit is kept 1. The result of signed operation is a signed number. For unsigned operation all the bits of the data word are used to represent the magnitude. The result of unsigned arithmetic operation is also an unsigned number.

In the examples given below from 6.38.1 to 6.38.4, the results are of 7 bits. If in any such example, the result comes in more than 7 bits (i.e. in 8 bits), there will be overflow. The microprocessor Intel 8085 does not give any indication of overflow because its MSB is used to represent sign bit. The carry bit of Intel 8085 will not help in this situation. The microprocessor Intel 8086 and microcontroller 8051 provide overflow flag besides the carry flag. If overflow flag is set to 1, result needs corrections. Both the sign bit as well as magnitude need correction. The sign bit has to be complemented to know the correct sign of the result. Then, for +ve result +128 is added to the 7-bit result given by the processor, and for – ve result – 128 is added to the 7-bit result given by the processor.

6.38.1 Signed Arithmetic Addition

Example 1. Add 14H and – 9H

14 = 001 0100 (7-bit representation)

Since 14 is positive number, the sign bit (MSB) is 0.

+14 = 0001 0100 = 14H (8-bit signed representation)

To represent a negative number its 2's complement is used.

9 = 000 1001 (7-bit representation)

Its 1's complement = 111 0110

Its 2's complement = 111 0111

Its 8-bit signed representation = 1111 0111 = F7 hex.

Now 14H and F7H are to be added.

```
  0001  0100 (+14H, signed 8-bit representation)
+ 1111  0111 (2's complement of 9)
-------------
  0000  1011 (Result neglecting carry)
```

The sign bit of the result is 0, hence it is positive. Its magnitude is B hex which is equal to 11 decimal. Reader should note that when the 2's complement of a number is added to a positive number, it is equivalent to subtraction. Microprocessors perform subtraction by adding 2's complement of a number which is to be subtracted.

Now assembly language program will be used for this addition.

PROGRAM

Memory Address	*Machine Code*	*Mnemonics Operands*	*Comments*
2000	21,01,25	LXI H, 2501H	; Get address of 1st number in H-L pair.
2003	7E	MOV A,M	; First number in accumulator
2004	23	INX H	; Address of next number in H-L pair
2005	86	ADD M	; Add 1st and 2nd number
2006	32,03,25	STA 2503 H	; Store sum in 2503H
2009	76	HLT	; Stop

DATA	**Result**
2501–14H	2503 – 0B
2502–F7H	

Example 2. Add – 14H and + 9H.

In this example the bigger number is negative, and therefore, the result will be negative. The negative result comes in signed 8-bit representation of 2's complement.

14H = 001 0100 (7-bit representation)

Its 1's complement = 110 1011

Its 2's complement = 110 1011 + 1 = 110 1100

Its 8-bit signed representation = 1110 1100 = EC
9 = 000 1001 (7-bit representation)
Since it is a positive number, its sign bit is 0.
Its 8-bit signed representation is = 0000 1001

Now EC hex and 9H are to be added.

```
  1110 1100 (8-bit signed representation of 2's complement of 14H)
+ 0000 1001 (8-bit signed representation of 9)
-------------
  1111 0101 (Result)
```

As the sign bit of the result is 1, it is negative. Its magnitude is 75H. The result is in 2's complement. To get the correct result, its 2's complement is taken again.
2's complement of the result (i.e. 75H) = 0B hex
Therefore, result = –11 decimal.

Now this addition can be performed using assembly language program given in Example 1. Data and result for Example 2 are as follows :

DATA	Result
2501–EC	2503–F5 (including sign bit)
2502–09	

2's complement of 75H = 0B (take 7 bits)

Result = –11 decimal.

Example 3. Add –14H and –9H

In this example both numbers are negative, and therefore, the result is negative and it is in 2's complement.

-14H = EC, 8-bit signed representation of 2's complement of 14H.
-9H = F7, 8-bit signed representation of 2's complement of 9H
Now EC and F7 are to be added.

```
EC =  1110 1100 (8-bit signed representation)
F7 =  1111 0111 (8-bit signed representation)
-------------
1     1110 0011 (Result)
↑
```

Carry to be neglected

The sign-bit i.e. MSB of the 8-bit result is 1, and hence the result is negative and it is in 2's complement. The carry resulting from the addition of sign bits is ignored.

The 2's complement of the result = 1D hex = 29 decimal.

The Addition can be performed using assembly language program given in Example 1. The data and result are as follows:

DATA	Result
2501–EC	2503-E3
2502–F7	

2's complement of E3 hex = 1D hex = 29 decimal.

Example 4. Add 14H and 9H

14H = 00001 0100 (8-bit signed representation. The MSB is the sign bit)

09H = 0000 1001 (8-bit signed representation)

00001 1101 (Result, 8-bit signed representation)

The sign bit (MSB) of the result is 0, and therefore, the result is positive. Its magnitude is 1D hex = 29 decimal.

The addition can be performed using assembly language program given in Example 1. The data and result are as follows :

DATA	Result
2501–14H	2503–1DH
2502–09H	

6.39 8-BIT UNSIGNED SUBTRACTION TO CONSIDER POSITIVE AS WELL AS NEGATIVE RESULT

When a smaller number is subtracted from a larger one, the result is positive. But when a larger number is subtracted from a smaller one, the result is negative. A program has been developed which considers both types of cases to give positive or negative result depending on the case.

PROGRAM

Memory Address	*Machine codes*	*Mnemonics /operands*	*Comments*
2000	21, 01, 25	LXI H, 2501	; address of 1st number in H-L pair.
2003	7E	MOV A, M	; 1st number in accumulator.
2004	23	INX H	; address of 2nd number in H-L pair.
2005	96	SUB M	; 1st number – 2nd number.
2006	DA, 50, 20	JC GO	; jump to label GO, if carry.
2009	23	INX	; memory address in H-L pair to store result.
200A	77	MOV M, A	; store result in 2503H. Case of +ve result when there is no carry.
200B	3E, 00	MOV A, 00	; get 00 in accumulator.
200D	32, 04, 25	STA 2504	; store carry (00) in 2504H for +ve result.
2010	76	HLT	; stop.
2050	GO 2F	CMA	; 1's complement of the result.
2051	3C	INR A	; 2's complement of result.
2052	32, 03, 25	STA 2503	; magnitude of result in 2503H. Case of –ve result.
2055	3E, 01	MVI 01	; get 01 in accumulator.
2057	32, 04, 25	STA 2504	; store carry (01) in 2504H for – ve result.
205A	76	HLT	; stop.

Example 1	Example 2
DATA	**DATA**
2501–95H	2501–34H
2502–3AH	2502–40H
Result	**Result**
2503–5B	2503–0C
Carry in 2504 is 00,	Carry in 2504 is 01,
So result is +ve i.e. +5B.	therefore result is – ve i.e. – 0C.

Alternative program using 2's complement technique. The addition of 2's complement of a number is equivalent to subtraction of the number.

```
PROGRAM
LXI H , 2502H      ; address of 2nd number in H-L pair
MOV A , M          ; 2nd number in accumulator
CMA                ; 1's complement of 2nd number
ADI 01             ; 2's complement of 2nd number
DCX H              ; address of 1st number in H-L pair
ADD M              ; add 1st number and 2's complement of 2nd number
JNC  AHEAD         ; jump on no carry to label AHEAD. Case of –ve result
STA  2503H         ; store result. Case of carry, +ve result
MVI A, 00          ; accumulator = 00
RAL                ; rotate accumulator left with carry to get carry in accumulator

STA 2504H          ; store carry for +ve result
HLT                ; stop
AHEAD  CMA         ; 1's complement of result
    INR A          ; 2's complement of result
    STA 2503H      ; store magnitude of –ve result
    MVI A, 00      ; accumulator = 00
    STA 2504H      ; store carry (=00) for –ve result
    HLT            ; stop
```

Data	Result
2501H – 1st number	2503 – result
2502H – 2nd number	2504 – carry

Examine carry and result. When carry is 1, result is positive.
When carry is zero, result is negative.

In the above program to subtract the 2nd number from the first number, 2's complement technique has been used. The 2's compliment of a number gives the negative value of the number. Therefore, to subtract the 2nd number from the 1st number, the 2's complement of the 2nd number is added

to the 1st number. When the 2nd number is less than the first number, the result of subtraction is positive. When the 2nd number is greater than the 1st number, the result of the subtraction is negative. If 2's compliment technique is used for subtraction, there will be carry for positive result and no carry for negative result. Therefore, carry is also stored to judge whether the result is positive or negative. When the result is negative, it is the 2's compliment of the result. To get its correct value, 2's compliment of the negative result is taken again.

6.40 8-BIT UNSIGNED DECIMAL SUBTRACTION TO CONSIDER POSITIVE AS WELL AS NEGATIVE RESULT

When a smaller number is subtracted from a larger one, the result is positive. But when a larger number is subtracted from a smaller one, the result is negative. A program has been developed to consider both types of cases to give positive or negative result depending on the case.

PROGRAM

```
LXI H , 2502H      ; address of 2nd number in H-L pair
MOV A , 99         ; get 99 in accumulator
SUB M              ; 9's complement of 2nd number
INR A              ; 10's complement of 2nd number
DCX H              ; address of 1st number in H-L pair
ADD M              ; add 1st number and 10's complement of 2nd number
JNC  AHEAD         ; jump on no carry to label AHEAD. Case of –ve result
STA  2503H         ; store +ve result. Case of carry, result is +ve
MVI A, 00          ; accumulator = 00
RAL                ; rotate accumulator left with carry to get carry in accumulator

STA 2504H          ; store carry for +ve result
HLT                ; stop

AHEAD   MOV B , A  ; result in register B
MVI A , 99         ; get 99 in accumulator
SUB B              ; 9's compliment of result
ADI 01             ; 10's compliment of result
STA 2503H          ; store magnitude of –ve result
MVI A , 00         ; accumulator = 00
STA 2504H          ; store carry (=00) for –ve result
HLT
```

DATA	**RESULT**
2501H – 1st number	2503 – result
2502H – 2nd number	2504 – carry

Examine carry and result. When carry is = 1, result is positive. When carry is zero, result is negative.

1st number = x
2nd number = y
9's complement of 2nd number = 99 – y
10's complement of 2nd number = 99 – y + 1
= 100 – y

R = result = x – y
1st number + 2's complement of 2nd number = R
R = result given by the processor = 100 +x – y

If y is less than x, R is positive and greater than 100, which is in 3 digits. In an 8-bit processor, the accumulator contains only up to 8-bits or two decimal digits. Therefore, the third digit will result a carry. The result will be equal to x – y and carry equal to 1.

If y is greater than x, R is 99 or less, Which is of only two digits. Therefore, there will not be any carry.

Since y is greater than x, R = y – x
Result given by processor = 100 + x – y
= 100 – (y – x)
= 10's complement of the result.

If y = x, R = 100 (3 digits), there will be a carry. So the result will be equal to 00 and the carry will be 1.

PROBLEMS

1. Write an assembly language program to add two 8-bit numbers, the sum may be of 16 bits.
2. Write an assembly language program to add two 8-bit decimal numbers; sum may be of 16 bits.
3. Write and assembly language program to get 2's complement of a 16-bit number.
4. Write an assembly language program to find the largest number in a data array.
5. Write an assembly language program to get the smallest number in a data array.
6. Write an assembly language program to find the sum of a series of 8-bit numbers, sum may be of 16 bits.
7. Write an assembly language program to get the sum of a series of 8-bit decimal numbers, sum may be of 16 bits.
8. Write an assembly language program for 8-bit multiplication, product being of 16 bits.
9. Write an assembly language program for 8-bit division, dividend being a 16-bit number.

R = result = x – y
1st number + 2's complement of 2nd number = R
R = result given by the processor = 100 +x – y

If y is less than x, R is positive and greater than 100, which is in 3 digits. In an 8-bit processor, the accumulator contains only up to 8-bits or two decimal digits. Therefore, the third digit will result a carry. The result will be equal to x – y and carry equal to 1.

If y is greater than x, R is 99 or less, Which is of only two digits. Therefore, there will not be any carry.

Since y is greater than x, R = y – x
Result given by processor = 100 + x – y
= 100 – (y – x)
= 10's complement of the result.

If y = x, R = 100 (3 digits), there will be a carry. So the result will be equal to 00 and the carry will be 1.

PROBLEMS

1. Write an assembly language program to add two 8-bit numbers, the sum may be of 16 bits.
2. Write an assembly language program to add two 8-bit decimal numbers; sum may be of 16 bits.
3. Write and assembly language program to get 2's complement of a 16-bit number.
4. Write an assembly language program to find the largest number in a data array.
5. Write an assembly language program to get the smallest number in a data array.
6. Write an assembly language program to find the sum of a series of 8-bit numbers, sum may be of 16 bits.
7. Write an assembly language program to get the sum of a series of 8-bit decimal numbers, sum may be of 16 bits.
8. Write an assembly language program for 8-bit multiplication, product being of 16 bits.
9. Write an assembly language program for 8-bit division, dividend being a 16-bit number.