

Scilab Textbook Companion for
Microprocessor Architecture, Programming &
Applications with the 8085
by R. S. Goankar¹

Created by
Rishabh Jain
B.Tech
Electronics Engineering
uptu
College Teacher
None
Cross-Checked by
K. V. P. Pradeep

May 15, 2014

¹Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website <http://scilab.in>

Book Description

Title: Microprocessor Architecture, Programming & Applications with the
8085

Author: R. S. Goankar

Publisher: Penram International, Mumbai

Edition: 4

Year: 1999

ISBN: 978-81-900828-7-7

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes	4
2 Microprocessor Architecture and Microcomputer Systems	10
3 8085 Microprocessor Architecture And Memory Interfacing	17
6 Introduction To 8085 Instructions	23
7 Programming Techniques With Additional Instructions	33
9 Stack And Subroutines	52
10 Code Conversion BCD Arithmetic and 16 bit Data Operations	59
12 Interrupts	72
14 Programmable Interface Devices	76
15 General Purpose Programmable Peripheral Devices	79
19 Appendix A Number System	94

List of Scilab Codes

Exa 2.1	MEMORY ADDRESS RANGE	10
Exa 2.2	MEMORY ADDRESS RANGE	11
Exa 2.3	CALCULATING ADDRESS LINES	13
Exa 2.4	CALCULATING NO OF CHIPS	14
Exa 2.5	FETCHING AN INSTRUCTION	15
Exa 3.2	EXECUTING THE INSTRUCTION	17
Exa 3.3	TIME REQUIRED FOR EXECUTION	18
Exa 3.5	MEMORY ADDRESS RANGE OF 6116	19
Exa 3.6	MEMORY ADDRESS RANGE OF 8155	20
Exa 6.1	LOAD A DATA TO ONE REGISTER AND MOVE IT TO ANOTHER	23
Exa 6.2	TO SWITCH ON SOME DEVICES	23
Exa 6.3	ADDITION OF TWO NUMBERS	25
Exa 6.4	CONTINUATION OF PREVIOUS EXAMPLE	25
Exa 6.5	INCRIMENTING ACCUMULATOR CONTENT	26
Exa 6.6	SUBTRACTION OF TWO NUMBERS	27
Exa 6.7	PERFORMING LOGICAL OPERATIONS	29
Exa 6.8	KEEPING THE RADIO ON	30
Exa 6.9	TURN OFF THE AIR CONDITIONER	31
Exa 7.1	STEPS TO ADD 10 BYTES OF DATA	33
Exa 7.2	LOADING 16 BIT NUMBER	34
Exa 7.3	TRANSFER OF DATA BYTES TO ACCUMULATOR	35
Exa 7.4	USE OF ADDRESSING MODES	36
Exa 7.5	INCREMENT A NUMBER	38
Exa 7.6	ARITHMETIC OPERATIONS	40
Exa 7.7	INCREMENT AND DECREMENT	41
Exa 7.8	LEFT ROTATION RLC OF BITS	43
Exa 7.9	LEFT ROTATION RAL OF BITS	45

Exa 7.10	RIGHT ROTATION RRC AND RAR OF BITS	47
Exa 7.11	COMPARISION OF DATA	48
Exa 9.1	PUSH POP AND DELAY INSTRUCTIONS	52
Exa 9.2	EXCHANGE OF DATA USING STACK	53
Exa 9.3	EXCHANGE INFORMATION BETWEEN STACK AND PROGRAM COUNTER	57
Exa 10.1	BCD TO BINARY	59
Exa 10.2	ADDITION OF PACKED BCD NUMBERS	60
Exa 10.3	EXCHANGE OF DATA	62
Exa 10.4	ADDITION OF TWO 16 BIT NUMBERS	62
Exa 10.5	SUBTRACTION OF TWO 16 BIT NUMBERS	64
Exa 10.6	DISPLAY CONTENTS OF STACK	67
Exa 10.7	SUBROUTINE TO SET THE ZERO FLAG	68
Exa 10.8	TRANSFER A PROGRAM TO AN ADDRESS IN HL REGISTER	69
Exa 12.1	ENABLE INTERRUPTS	72
Exa 12.2	RESET INTERRUPT	73
Exa 12.3	CHECK PENDING INTERRUPT	73
Exa 14.1	INITIALIZE HYPOTHETICAL CHIP AS OUTPUT BUFFER	76
Exa 14.2	ADDRESS DETERMINATION OF GIVEN FIGURE	77
Exa 15.1	PORT ADDRESS CONTROL WORD ADDRESS AND READ THE DIP SWITCHES	79
Exa 15.2	BSR CONTROL WORD SUBROUTINE	82
Exa 15.3	INSTRUCTIONS TO GENERATE A PULSE FROM COUNTER 0	83
Exa 15.4	INSTRUCTIONS TO GENERATE SQUARE WAVE PULSE FROM COUNTER 1	84
Exa 15.5	SUBROUTINE TO GENERATE AN INTERRUPT	87
Exa 15.6	EXPLANATION OF INSTRUCTIONS	89
Exa 15.8	INITIALIZATION INSTRUCTIONS FOR DMA	90
Exa 19.1	BINARY INTO HEX AND OCTAL	94
Exa 19.2	SUBTRACTION OF TWO NUMBERS	94
Exa 19.3	SUBTRACTION OF TWO NUMBERS	96
Exa 19.4	2s COMPLIMENT OF BINARY NUMBER	98
Exa 19.5	SUBTRACTION OF TWO NUMBERS	99
Exa 19.6	SUBTRACTION OF TWO NUMBERS	100
Exa 19.7	SUBTRACTION OF UNSIGNED NUMBERS	100

Exa 19.8	SUBTRACTION OF SIGNED NUMBERS	102
Exa 19.9	ADDITION OF TWO POSITIVE NUMBERS	103

List of Figures

2.1	MEMORY ADDRESS RANGE	12
2.2	MEMORY ADDRESS RANGE	14
2.3	CALCULATING ADDRESS LINES	14
2.4	CALCULATING NO OF CHIPS	15
2.5	FETCHING AN INSTRUCTION	16
3.1	EXECUTING THE INSTRUCTION	18
3.2	TIME REQUIRED FOR EXECUTION	19
3.3	MEMORY ADDRESS RANGE OF 6116	20
3.4	MEMORY ADDRESS RANGE OF 8155	22
6.1	LOAD A DATA TO ONE REGISTER AND MOVE IT TO ANOTHER	24
6.2	TO SWITCH ON SOME DEVICES	24
6.3	ADDITION OF TWO NUMBERS	26
6.4	CONTINUATION OF PREVIOUS EXAMPLE	26
6.5	INCRIMENTING ACCUMULATOR CONTENT	28
6.6	SUBTRACTION OF TWO NUMBERS	29
6.7	PERFORMING LOGICAL OPERATIONS	31
6.8	KEEPING THE RADIO ON	31
6.9	TURN OFF THE AIR CONDITIONER	32
7.1	STEPS TO ADD 10 BYTES OF DATA	34
7.2	LOADING 16 BIT NUMBER	35
7.3	TRANSFER OF DATA BYTES TO ACCUMULATOR	37
7.4	USE OF ADDRESSING MODES	39
7.5	USE OF ADDRESSING MODES	39
7.6	INCREMENT A NUMBER	40
7.7	ARITHMETIC OPERATIONS	42

7.8	INCREMENT AND DECREMENT	44
7.9	LEFT ROTATION RLC OF BITS	46
7.10	LEFT ROTATION RAL OF BITS	47
7.11	RIGHT ROTATION RRC AND RAR OF BITS	49
7.12	COMPARISION OF DATA	51
9.1	PUSH POP AND DELAY INSTRUCTIONS	53
9.2	EXCHANGE OF DATA USING STACK	56
9.3	EXCHANGE INFORMATION BETWEEN STACK AND PRO- GRAM COUNTER	58
10.1	BCD TO BINARY	60
10.2	ADDITION OF PACKED BCD NUMBERS	61
10.3	EXCHANGE OF DATA	63
10.4	ADDITION OF TWO 16 BIT NUMBERS	65
10.5	SUBTRACTION OF TWO 16 BIT NUMBERS	67
10.6	DISPLAY CONTENTS OF STACK	68
10.7	SUBROUTINE TO SET THE ZERO FLAG	70
10.8	TRANSFER A PROGRAM TO AN ADDRESS IN HL REG- ISTER	71
12.1	ENABLE INTERRUPTS	73
12.2	RESET INTERRUPT	74
12.3	CHECK PENDING INTERRUPT	75
14.1	INITIALIZE HYPOTHETICAL CHIP AS OUTPUT BUFFER	77
14.2	ADDRESS DETERMINATION OF GIVEN FIGURE	78
15.1	PORT ADDRESS CONTROL WORD ADDRESS AND READ THE DIP SWITCHES	81
15.2	BSR CONTROL WORD SUBROUTINE	83
15.3	INSTRUCTIONS TO GENERATE A PULSE FROM COUNTER 0	85
15.4	INSTRUCTIONS TO GENERATE SQUARE WAVE PULSE FROM COUNTER 1	87
15.5	SUBROUTINE TO GENERATE AN INTERRUPT	89
15.6	EXPLANATION OF INSTRUCTIONS	91
15.7	INITIALIZATION INSTRUCTIONS FOR DMA	93
15.8	INITIALIZATION INSTRUCTIONS FOR DMA	93

19.1 BINARY INTO HEX AND OCTAL	95
19.2 SUBTRACTION OF TWO NUMBERS	96
19.3 SUBTRACTION OF TWO NUMBERS	98
19.4 2s COMPLIMENT OF BINARY NUMBER	99
19.5 SUBTRACTION OF TWO NUMBERS	100
19.6 SUBTRACTION OF TWO NUMBERS	101
19.7 SUBTRACTION OF UNSIGNED NUMBERS	102
19.8 SUBTRACTION OF SIGNED NUMBERS	104
19.9 ADDITION OF TWO POSITIVE NUMBERS	105

Chapter 2

Microprocessor Architecture and Microcomputer Systems

Scilab code Exa 2.1 MEMORY ADDRESS RANGE

```
1 //page no 39
2 //example no 2.1
3 //MEMORY ADDRESS RANGE.
4 clc;
5 printf('A7–A0 are address lines for register select
   . \n');
6 printf('A15–A8 are address lines for chip select. \n
   \n');
7 printf('A15 A14 A13 A12 A11 A10 A9 A8 \n');
8 printf(' 0 0 0 0 0 0 0 0 =00H \n \n');
   //chip select bits have to be active low always
   to select that chip.
9 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
10 printf(' 0 0 0 0 0 0 0 0 =00H \n'); //this
   selects the register 00.
11 printf('The above combination selects the memory
   address 0000H. \n \n');
12 printf('A15 A14 A13 A12 A11 A10 A9 A8 \n');
13 printf(' 0 0 0 0 0 0 0 0 =00H \n \n');
```

```

        //chip select bits have to be active low always
        to select that chip.
14 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
15 printf(' 1 1 1 1 1 1 1 1  =FFH \n'); //this
    selects the register FF.
16 printf('The above combination selects the memory
    address 00FFH. \n \n');
17 //thus this chip can select any memory location from
    0000H to 00FFH.
18 //the memory addressed of the chip can be changed by
    modifying the hardware.For example if we remove
    the inverter on line A15.
19 printf('A15 A14 A13 A12 A11 A10 A9 A8 \n');
20 printf(' 1 0 0 0 0 0 0 0  =80H \n \n');
    //chip select bits have to be active low always
    to select that chip.
21 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
22 printf(' 0 0 0 0 0 0 0 0  =00H \n'); //this
    selects the register 00.
23 printf('The above combination selects the memory
    address 8000H. \n \n');
24 //The memory address range from above change will be
    8000H to 80FFH.
25 //Thus a memory can be assigned address in various
    locations over the entire map of 0000H to FFFFH.

```

Scilab code Exa 2.2 MEMORY ADDRESS RANGE

```

1 //page no 41
2 //example no 2.2
3 //MEMORY ADDRESS RANGE.
4 clc;
5 printf('A9-A0 are address lines for register select

```

```
Scilab 5.4.1 Console
A7-A0 are address lines for register select.
A15-A8 are address lines for chip select.

A15 A14 A13 A12 A11 A10 A9 A8
0 0 0 0 0 0 0 0 =00H

A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 0 0 0 =00H
The above combination selects the memory address 0000H.

A15 A14 A13 A12 A11 A10 A9 A8
0 0 0 0 0 0 0 0 =00H

A7 A6 A5 A4 A3 A2 A1 A0
1 1 1 1 1 1 1 1 =FFH
The above combination selects the memory address 00FFH.

A15 A14 A13 A12 A11 A10 A9 A8
1 0 0 0 0 0 0 0 =80H

A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 0 0 0 =00H
The above combination selects the memory address 8000H.
```

Figure 2.1: MEMORY ADDRESS RANGE

```

        . \n');
6 printf('A15–A10 are address lines for chip select. \
    n \n');
7 printf('A15 A14 A13 A12 A11 A10 \n');
8 printf(' 0 0  0  0  0  0 \n \n'); //chip select
    bits have to be active low always to select that
    chip.
9 printf('A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 \n');
10 printf(' 0 0 0 0 0 0 0 0 0 0 \n'); //this
    selects the register
11 printf('The above combination selects the memory
    address 0000H. \n \n');
12 printf('A15 A14 A13 A12 A11 A10 \n');
13 printf(' 0 0  0  0  0  0 \n \n'); //chip select
    bits have to be active low always to select that
    chip.
14 printf('A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 \n');
15 printf(' 1 1 1 1 1 1 1 1 1 1 \n'); //this
    selects the register
16 printf('The above combination selects the memory
    address 03FFH. \n \n');
17 //thus this chip can select any memory location from
    0000H to 03FFH.
18 //the memory addressed of the chip can be changed by
    modifying the hardware.Like we did in the
    previous example.

```

Scilab code Exa 2.3 CALCULATING ADDRESS LINES

```

1 //page no 43
2 //example no 2.3
3 //CALCULATING ADDRESS LINES
4 clc;

```

```

Scilab 5.4.1 Console
A9-A0 are address lines for register select.
A15-A10 are address lines for chip select.

A15 A14 A13 A12 A11 A10
0 0 0 0 0 0

A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 0 0 0 0 0
The above combination selects the memory address 0000H.

A15 A14 A13 A12 A11 A10
0 0 0 0 0 0

A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
1 1 1 1 1 1 1 1 1 1
The above combination selects the memory address 03FFH.

```

Figure 2.2: MEMORY ADDRESS RANGE

```

Scilab 5.4.1 Console
Number of address lines=
13.

```

Figure 2.3: CALCULATING ADDRESS LINES

```

5 //number of address lines are given by x
6 x={log(8192)}/{log(2)};
7 printf('Number of address lines= ')
8 disp(x);

```

Scilab code Exa 2.4 CALCULATING NO OF CHIPS

```

1 //page no 43
2 //example no 2.4
3 //CALCULATING NO OF CHIPS.

```



```
Scilab 5.4.1 Console
No of chips=
  64.
-->
```

Figure 2.4: CALCULATING NO OF CHIPS

```
4 clc;
5 //chip 1024*1 has 1024(1k) registers & each register
   can store one bit with one data line. We need 8
   data lines for byte size memory. Therefore 8
   chips are necessary for 1k byte memory. For 1k
   byte memory we will need 64 chips. We can arrive
   at the same ans by dividing 8k byte by 1k*1 as
   follows:
6 no=(8192*8)/(1024*1);
7 printf('No of chips= ');
8 disp(no);
```

Scilab code Exa 2.5 FETCHING AN INSTRUCTION

```
1 //page no 44
2 //example no 2.5
3 //FETCHING AN INSTRUCTION.
4 clc;
5 printf('Memory Location 2005H= 4FH \n');
6 printf('Address bus= 2005H \n') //program counter
   places the 16-bit address on the address bus.
7 printf('Control bus—> (MEMR) \n'); //control bus
   sends memory read control signal.
8 printf('Data bus= 4FH \n'); //instruction 4FH is
   fetched and transferred to instruction decoder.
```



```
Scilab 5.4.1 Console
Memory Location 2005H= 4FH
Address bus= 2005H
Control bus--> (MEMR)
Data bus= 4FH
-->
```

Figure 2.5: FETCHING AN INSTRUCTION

Chapter 3

8085 Microprocessor Architecture And Memory Interfacing

Scilab code Exa 3.2 EXECUTING THE INSTRUCTION

```
1 //page no 78
2 //example no 3.2
3 //EXECUTING THE INSTRUCTION.
4 clc;
5 A=82; //contents of the accumulator.
6 printf('Accumulator= ');
7 disp(A);
8 TR=A; //contents of the accumulator tranferred to
        the temporary register.
9 printf('Temporary Register= ');
10 disp(TR);
11 C=TR; //contents of the temporary register are
        transferred to register C.
12 printf('Register C= ');
13 disp(C);
```



```
Scilab 5.4.1 Console
Accumulator=
  82.
Temporary Register=
  82.
Register C=
  82.
-->|
```

Figure 3.1: EXECUTING THE INSTRUCTION

Scilab code Exa 3.3 TIME REQUIRED FOR EXECUTION

```
1 //page no 82
2 //example no 3.3
3 //TIME REQUIRED FOR EXECUTION.
4 clc;
5 A=32; // MVI A,32H loads the value 32 in accumulator
6
7 printf('Accumulator= ');
8 disp(A);
9 //calculating the execution time for instruction.
10 f=2; // clock frequency.
11 printf('clock frequency= %f MHz \n',f);
12 t=1/f; // T-state=clock period
13 printf('T-state=clock period= %f microsec \n',t);
14 t1=4*t; // execution time for opcode fetch.
15 printf('Execution time for opcode fetch= %f microsec
16 \n',t1);
17 t2=3*t; // execution time for memory read.
18 printf('Execution time for memory read= %f microsec
19 \n',t2);
20 t3=7*t; // execution time for instruction.
21 printf('Execution time for instruction= %f microsec
```

```

Scilab 5.4.1 Console
Accumulator=
  32.
clock frequency= 2.000000 MHz
T-state=clock period= 0.500000 microsec
Execution time for opcode fetch= 2.000000 microsec
Execution time for memory read= 1.500000 microsec
Execution time for instruction= 3.500000 microsec

-->|

```

Figure 3.2: TIME REQUIRED FOR EXECUTION

```
\n',t3);
```

Scilab code Exa 3.5 MEMORY ADDRESS RANGE OF 6116

```

1  ///page no 91
2  //example no 3.5
3  //MEMORY ADDRESS RANGE OF 6116.
4  clc;
5  printf('A10-A0 are address lines for register
        select. \n');
6  printf('A15-A11 are address lines for chip select. \
        n \n');
7  printf('A15 A14 A13 A12 A11 \n');
8  printf(' 1  0  0  0  1 \n \n'); //chip select
        bits have to be active low always to select that
        chip.
9  printf('A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 \n');
10 printf('0  0  0  0  0  0  0  0  0  0  0 \n'); //
        this selects the register
11 printf('The above combination selects the memory
        address 8800H. \n \n');
12 printf('A15 A14 A13 A12 A11 \n');

```

```

Scilab 5.4.1 Console
A10-A0 are address lines for register select.
A15-A11 are address lines for chip select.

A15 A14 A13 A12 A11
1 0 0 0 1

A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 0 0 0 0 0 0 0
The above combination selects the memory address 8800H.

A15 A14 A13 A12 A11
1 0 0 0 1

A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
1 1 1 1 1 1 1 1 1 1 1 1
The above combination selects the memory address 88FFH.

-->

```

Figure 3.3: MEMORY ADDRESS RANGE OF 6116

```

13 printf(' 1 0 0 0 1 \n \n'); //chip select
    bits have to be active low always to select that
    chip.
14 printf('A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 \n');
15 printf('1 1 1 1 1 1 1 1 1 1 \n'); //
    this selects the register
16 printf('The above combination selects the memory
    address 88FFH. \n \n');
17 //thus this chip can select any memory location from
    8800H to 88FFH.

```

Scilab code Exa 3.6 MEMORY ADDRESS RANGE OF 8155

```

1 ///page no 95
2 //example no 3.6

```

```

3 //MEMORY ADDRESS RANGE OF 8155.
4 clc;
5 printf('A7-A0 are address lines for register select.
        \n');
6 printf('A10-A8 address lines are dont care
        conditions. \n');
7 printf('A15-A11 are address lines for chip select. \
        n \n');
8 printf('A15 A14 A13 A12 A11 \n');
9 printf(' 0 0 1 0 0 \n \n'); //chip select
        bits have to be active low always to select that
        chip.
10 printf('A10 A9 A8 \n');
11 printf('0 0 1 \n \n'); //this is the don't care
        condition.
12 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
13 printf('0 0 0 0 0 0 0 0 \n'); //this selects
        the register
14 printf('The above combination selects the memory
        address 2100H. \n \n');
15 printf('A15 A14 A13 A12 A11 \n');
16 printf(' 0 0 1 0 0 \n \n'); //chip select
        bits have to be active low always to select that
        chip.
17 printf('A10 A9 A8 \n');
18 printf('0 0 1 \n \n'); //this is the don't care
        condition.
19 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
20 printf('1 1 1 1 1 1 1 1 \n'); //this selects
        the register
21 printf('The above combination selects the memory
        address 21FFH. \n \n');
22 //thus this chip can select any memory location from
        2100H to 21FFH.

```

```
Scilab 5.4.1 Console
A7-A0 are address lines for register select.
A10-A8 address lines are dont care conditions.
A15-A11 are address lines for chip select.

A15 A14 A13 A12 A11
0 0 1 0 0

A10 A9 A8
0 0 1

A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 0 0 0
The above combination selects the memory address 2100H.

A15 A14 A13 A12 A11
0 0 1 0 0

A10 A9 A8
0 0 1

A7 A6 A5 A4 A3 A2 A1 A0
1 1 1 1 1 1 1 1
The above combination selects the memory address 21FFH.
```

Figure 3.4: MEMORY ADDRESS RANGE OF 8155

Chapter 6

Introduction To 8085 Instructions

Scilab code Exa 6.1 LOAD A DATA TO ONE REGISTER AND MOVE IT TO ANOTHER

```
1 //page no 164
2 //example no 6.1
3 //LOAD A DATA TO ONE REGISTER AND MOVE IT TO ANOTHER
4 .
5 clc;
6 A=hex2dec(['82']); //storing the decimal value of
   hexadecimal no 82 in accumulator A
7 B=dec2hex([A]); //storing the hexadecimal value of A
   in B
8 print(%io(2),B); //displaying the hexadecimal number
   in register B
```

Scilab code Exa 6.2 TO SWITCH ON SOME DEVICES


```
Scilab 5.4.1 Console
B =
82
-->
```

Figure 6.1: LOAD A DATA TO ONE REGISTER AND MOVE IT TO ANOTHER

```
Scilab 5.4.1 Console
At output port 01H:
1001111
Value 1s are showing the devices are ON.
Value 0s are showing the devices are switched OFF.
-->
```

Figure 6.2: TO SWITCH ON SOME DEVICES

```
1 //page no 164
2 //example 6.2
3 //TO SWITCH ON SOME DEVICES
4 //let the switches which are ON are at bit no D0,D1,
   D2,D3,D6;
5 clc;
6 x=hex2dec(['4F']); //hexadecimal to decimal
   conversion
7 y=dec2bin(x); //decimal to binary conversion
8 printf('At output port 01H: '); //same input appears
   at the putput
9 disp(y);
10 printf('Value 1s are showing the devices are ON. \n'
   )
11 printf('Value 0s are showing the devices are
   switched OFF. ');
```

Scilab code Exa 6.3 ADDITION OF TWO NUMBERS

```
1 //page no 174
2 //example 6.3
3 //ADDITION OF TWO NUMBERS.
4 //93H is stored in accumulator. Converting it into
   decimal.
5 clc;
6 A=hex2dec(['93']);
7 //B7H is stored in register C. Converting it into
   decimal.
8 C=hex2dec(['B7']);
9 X=A+C; // the result comes out to be 330
10 Z=X-256;
11 //X=330; // this is a decimal value. Converting it
   into hexadecimal
12 Y=dec2hex(Z);
13 printf('Sum= ')
14 disp(Y);
15 if X>255 then
16     printf('CY=1')
17 else
18     printf('CY=0')
19 end
```

Scilab code Exa 6.4 CONTINUATION OF PREVIOUS EXAMPLE

```
1 //page no 175
2 //example 6.4
3 //CONTINUATION OF PREVIOUS EXAMPLE.
```

```
Scilab 5.4.1 Console
Sum=
4A
CY=1
-->
```

Figure 6.3: ADDITION OF TWO NUMBERS

```
Scilab 5.4.1 Console
Sum=
7F
CY=0
-->
```

Figure 6.4: CONTINUATION OF PREVIOUS EXAMPLE

```
4 //the sum of previous example is added to 35H
5 clc;
6 S=hex2dec(['4A']); //4AH is converted into decimal
  value.
7 A=hex2dec(['35']); //35H is converted into decimal
  value
8 s=A+S; //the result comes out to be 127. it is a
  decimal value
9 Y=dec2hex(s);
10 printf('Sum= ')
11 disp(Y);
12 if s>255 then
13     printf('CY=1')
14 else
15     printf('CY=0')
16 end
```

Scilab code Exa 6.5 INCRIMENTING ACCUMULATOR CONTENT

```
1 //page no 175
2 //example no 6.5
3 //INCRIMENTING ACCUMULATOR CONTENT.
4 //accumulator holds the data FFH
5 clc;
6 A=hex2dec(['FF']); //converting FFH into decimal
   value
7 //decimal value of 01H is 01. Adding 01 to A
8 Y=A+1; //the result comes out to be 256
9 Z=Y-256;
10 X=dec2hex(Z);
11 printf('Sum =')
12 disp(X);
13 if Y>255 then
14     printf('CY=1 \n')
15 else
16     printf('CY=0 \n')
17 end
18 if Z>127 then
19     printf('S=1 \n')
20 else
21     printf('S=0 \n')
22 end
23 if Z>0 then
24     printf('Z=0 \n')
25 else
26     printf('Z=1 \n')
27 end
```

Scilab code Exa 6.6 SUBTRACTION OF TWO NUMBERS

```
Scilab 5.4.1 Console
Sum =
0
CY=1
S=0
Z=1
-->
```

Figure 6.5: INCRIMENTING ACCUMULATOR CONTENT

```
1 //page no 179
2 //example 6.6
3 //SUBTRACTION OF TWO NUMBERS.
4 //accumulator has 97H. Converting it into decimal
   value
5 clc;
6 A=hex2dec(['97']);
7 //register B has 65H. Finding 2's compliment of 65H.
8 B=hex2dec(['65']);
9 X=256-B;
10 Y=A+X;
11 S=Y-256;
12 Z=dec2hex(S);
13 printf('Subtraction= ')
14 disp(Z);
15 if Y>255 then
16     CY=1;
17     printf('The result is positive. \n');
18 else
19     CY=0;
20     printf('The result is negative. \n')
21 end
22 if S>127 then
23     printf('S=1 \n')
24 else
25     printf('S=0 \n')
26 end
27 if S>0 then
```

```
Scilab 5.4.1 Console
Subtraction=
32
The result is positive.
S=0
Z=0
-->
```

Figure 6.6: SUBTRACTION OF TWO NUMBERS

```
28     printf('Z=0 \n')
29 else
30     printf('Z=1 \n')
31 end
```

Scilab code Exa 6.7 PERFORMING LOGICAL OPERATIONS

```
1 //page no 185
2 //example no 6.7
3 //PERFORMING LOGICAL OPERATIONS.
4 //register B holds 93H. Binary of 93H is 10010011
5 //register A holds 15H. Binary of 15H is 00010101.
6 clc;
7 B=[1 0 0 1 0 0 1 1]; //taking the value of A in
   matrix form.
8 A=[0 0 0 1 0 1 0 1]; //taking the value of B in
   matrix form.
9 Y= bitor(A,B); // getting OR of A & B
10 printf('OR of A & B is ')
11 disp(Y);
12 if Y(1,1)==1 then
13     printf('S=1 \n');
14 else
15     printf('S=0 \n');
```

```

16 end
17 if Y==0 then
18     printf('Z=1 \n');
19 else
20     printf('Z=0 \n');
21 end
22 printf('CY=0 \n');
23 R=bitxor(A,B); //getting XOR of A & B
24 printf('XOR of A & B is ')
25 disp(R);
26 if R(1,1)==1 then
27     printf('S=1 \n');
28 else
29     printf('S=0 \n');
30 end
31 if R==0 then
32     printf('Z=1 \n');
33 else
34     printf('Z=0 \n');
35 end
36 printf('CY=0 \n');
37 K=bitcmp(A,1); //getting the compliment of A
38 printf('Compliment of A is: \n');
39 disp(K);

```

Scilab code Exa 6.8 KEEPING THE RADIO ON

```

1 //page no 186
2 //example no 6.8
3 //KEEPING THE RADIO ON.
4 //to keep the radio on without affecting the other
   appliances , the D4 bit should always be 1
5 //assuming an input input binary 10101010

```

```

Scilab 5.4.1 Console
OR of A & B is
  1.  0.  0.  1.  0.  1.  1.  1.
S=1
Z=0
CY=0
XOR of A & B is
  1.  0.  0.  0.  0.  1.  1.  0.
S=1
Z=0
CY=0
Compliment of A is:
  1.  1.  1.  0.  1.  0.  1.  0.
-->

```

Figure 6.7: PERFORMING LOGICAL OPERATIONS

```

Scilab 5.4.1 Console
  1.  0.  1.  1.  1.  0.  1.  0.
D4 bit will always be one without affecting the other bits
-->|

```

Figure 6.8: KEEPING THE RADIO ON

```

6  clc;
7  A=[1 0 1 0 1 0 1 0];
8  B=[0 0 0 1 0 0 0 0];
9  Y=bitor(A,B); //ORing input (A) with B to keep the
   D4 bit always set
10 disp(Y);
11 printf('D4 bit will always be one without affecting
   the other bits');

```

Scilab code Exa 6.9 TURN OFF THE AIR CONDITIONER


```
Scilab 5.4.1 Console
0.  0.  1.  0.  1.  0.  1.  0.
D7 bit will always be zero without affecting the other bits
-->
```

Figure 6.9: TURN OFF THE AIR CONDITIONER

```
1 //page no 187
2 //example no 6.9
3 //TURN OFF THE AIR CONDITIONER.
4 //to turn OFF the air conditioner, reset bit D7
5 //Assuming the same input as earlier as it is a
  continuation of previous example.
6 clc;
7 A=[1 0 1 0 1 0 1 0];
8 B=[0 1 1 1 1 1 1 1];
9 Y=bitand(A,B); //ANDing input (A) with B to keep the
  D4 bit always set
10 disp(Y);
11 printf('D7 bit will always be zero without affecting
  the other bits');
```

Chapter 7

Programming Techniques With Additional Instructions

Scilab code Exa 7.1 STEPS TO ADD 10 BYTES OF DATA

```
1 //page no 216
2 //example no 7.1
3 //STEPS TO ADD 10 BYTES OF DATA.
4 clc;
5 disp('The micriprocessor needs :');
6 disp('a counter to count 10 data bytes');
7 disp('an index or a memory pointer to locate where
      data bytes are stored');
8 disp('to transfer data from a memory location to the
      microprocessor');
9 disp('to perform addition');
10 disp('registers for temporary storage of partial
      answers');
11 disp('a flag to indicate the completion of the task'
      );
12 disp('to store or output the result');
```

```
Scilab 5.4.1 Console ? ↗ ✕

The micriprocessor needs :

a counter to count 10 data bytes

an index or a memory pointer to locate where data bytes are stored

to transfer data from a memory location to the microprocessor

to perform addition

registers for temporary storage of partial answers

a flag to indicate the completion of the task

to store or output the result

-->|
```

Figure 7.1: STEPS TO ADD 10 BYTES OF DATA

Scilab code Exa 7.2 LOADING 16 BIT NUMBER

```
1 //page no 219
2 //example no 7.2
3 //LOADING 16-BIT NUMBER.
4 //working of LXI instruction.
5 clc;
6 disp('LXI H,2050H'); // loads HL register pair.
7 disp('L=50H'); // 50H in L register.
8 disp('H=20H'); //20H in H register pair.
9 disp('LXI instruction takes 3 bytes of memory and 10
    clock periods.')
```

```
10 //working of MVI instruction.
11 disp('MVI H,20H');
12 disp('H=20H'); // load 20H in register H.
13 disp('MVI L,50H'); // load 50H in register L.
14 disp('L=50H');
```

```
Scilab 5.4.1 Console

LXI H,2050H

L=50H

H=20H

LXI instruction takes 3 bytes of memory and 10 clock periods.

MVI H,20H

H=20H

MVI L,50H

L=50H

2 MVI instructions take 4 bytes of memory and 14 clock periods.

-->
```

Figure 7.2: LOADING 16 BIT NUMBER

15 `disp('2 MVI instructions take 4 bytes of memory and
14 clock periods.')`

Scilab code Exa 7.3 TRANSFER OF DATA BYTES TO ACCUMULATOR

```
1 //page no 220
2 //example no 7.3
3 // TRANSFER OF DATA BYTES TO ACCUMULATOR.
4 // Memory location 2050H has the data F7H.
5 clc;
6
7 // using MOV instruction.
8 //indirect addressing mode.
9 disp('LXI H,2050H');
```

```

10 printf('H=20H   L=50H \n \n'); // the 16-bit address
    of the data is loaded in HL register pair.
11 M=hex2dec(['F7']); // M is the memory location
    pointer of address 2050H.
12 printf('MOV A,M \n');
13 A=dec2hex(M);
14 printf('A= ');
15 disp(A); // the contents of the HL register pair are
    used as memory pointer to the location 2050H.
16
17 // using LDAX instruction.
18 // indirect addressing mode.
19 disp('LXI B,2050H');
20 printf('B=20H   C=50H \n \n'); // the 16-bit address
    of the data is loaded in BC register pair.
21 M=hex2dec(['F7']); // M is the memory location
    pointer of address 2050H.
22 printf('LDAX B \n');
23 A=dec2hex(M);
24 printf('A= ');
25 disp(A); // the contents of the BC register pair are
    used as memory pointer to the location 2050H.
26
27 // using LDA instruction.
28 // direct addressing mode.
29 printf('\n LDA 2050H \n'); //directly sends the data
    of memory location 2050H to accumulator.
30 printf('A= ');
31 disp(A);

```

Scilab code Exa 7.4 USE OF ADDRESSING MODES

```
1 //page no 222
```

```

Scilab 5.4.1 Console

LXI H,2050H
H=20H  L=50H

MOV A,M
A=
  F7

LXI B,2050H
B=20H  C=50H

LDAX B
A=
  F7

LDA 2050H
A=
  F7

-->

```

Figure 7.3: TRANSFER OF DATA BYTES TO ACCUMULATOR

```

2 //example no 7.4
3 // USE OF ADDRESSING MODES.
4 clc;
5 //register B contains 32H
6 B=32;
7
8 //using indirect addressing modes
9 printf('B= %d \n',B);
10 disp(' 1) LXI H,8000H'); // loads HL register pair.
11 disp('H=80H      L=00H');
12 disp('MOV M,B'); // contents of register B are moved
    in memory location pointed by HL register pair.
13 M=B;
14 printf(' \n 8000H --> %d \n \n',M);
15
16 disp('LXI D,8000H'); //loads the memory location
    8000H in DE register pair.
17 disp('D=80H      E=00H');
18 disp('MOV A,B');

```

```

19 A=B;
20 printf('A= %d \n',A);
21 disp('STAX D'); //stores the value of accumulator in
    the memory location pointer by DE register pair.
22 printf('\n 8000H --> %d \n \n',A);
23
24 //using direct addressing mode.
25 disp('2) A= F2H');
26 disp('STA 8000H'); //this instruction stores the
    value of accumulator in the memory location 8000H
.
27 disp('8000H --> F2H');
28
29 //using indirect addressing mode.
30 disp('3) LXI H,8000H'); // loads HL register pair.
31 disp('H=80H      L=00H');
32 disp('MVI M,F2H'); //moving the data in the memory.
33 disp('8000H --> F2H');

```

Scilab code Exa 7.5 INCREMENT A NUMBER

```

1 //page no 224
2 //example no 7.5
3 // INCREMENT A NUMBER.
4 clc;
5 disp('LXI B,2050H'); //loads the data 2050H in BC
    register pair.
6 disp('B=20H      C=50H');
7 B=20;
8 C=50;
9 disp('INX B');

```

```
Scilab 5.4.1 Console
B= 32

1) LXI H,8000H

H=80H    L=00H

MOV M,B

8000H --> 32

LXI D,8000H

D=80H    E=00H

MOV A,B
A= 32

STAX D

8000H --> 32
```

Figure 7.4: USE OF ADDRESSING MODES

```
2) A= F2H

STA 8000H

8000H --> F2H

3) LXI H,8000H

H=80H    L=00H

MVI M,F2H

8000H --> F2H

-->
```

Figure 7.5: USE OF ADDRESSING MODES


```
Scilab 5.4.1 Console ? ↗ ✕

LXI B,2050H

B=20H C=50H

INX B
B= 20 C= 51

The contents of BC register pair will be 2051H

INR B
B= 21

INR C
C= 51

The contents of BC register pair will be 2151H

-->
```

Figure 7.6: INCREMENT A NUMBER

```
10 C=C+1;
11 printf('B= %d C= %d \n',B,C);
12 disp('The contents of BC register pair will be 2051H
    ');
13 disp('INR B');
14 B=B+1;
15 printf('B= %d \n',B);
16 disp('INR C');
17 C=50;
18 C=C+1;
19 printf('C= %d \n',C);
20 disp('The contents of BC register pair will be 2151H
    ');
```

Scilab code Exa 7.6 ARITHMETIC OPERATIONS

```
1 //page no 228
2 //example no 7.6
3 // ARITHMETIC OPERATIONS.
4 clc;
5 disp('A-->30H');
6 disp('2040H-->68H');
7 disp('2041H-->7FH');
8 disp('LXI H,2040H'); // loads HL register pair.
9 disp('H=20H      L=40H   M=68H');
10 disp('ADD M');
11 A=hex2dec(['30']);
12 M=hex2dec(['68']);
13 S=A+M; // adds the contents of A and data at memory
        location 2040H.
14 s=dec2hex(S);
15 printf('\n Content of A after addition with 2040H= ');
        );
16 disp(s);
17 disp('INX H'); // takes the program to the next
        memory location.
18 disp('H=20H      L=41H   M=7FH');
19 disp('SUB M');
20 M=hex2dec(['7F']);
21 D=S-M; // subtracts the contents of A from the data
        at memory location 2041H.
22 d=dec2hex(D);
23 printf('\n Content of A after subtraction with 2041H
        = ');
24 disp(d);
```

Scilab code Exa 7.7 INCREMENT AND DECREMENT

```
Scilab 5.4.1 Console ? ↗ ✕

A-->30H

2040H-->68H

2041H-->7FH

LXI H,2040H

H=20H    L=40H    M=68H

ADD M

Content of A after addition with 2040H=
98

INX H

H=20H    L=41H    M=7FH

SUB M

Content of A after subtraction with 2041H=
19

-->
```

Figure 7.7: ARITHMETIC OPERATIONS

```

1 //page no 229
2 //example no 7.7
3 // INCREMENT & DECREMENT.
4 clc;
5 disp('LXI H,2040H'); // loads HL register pair.
6 disp('H=20H      L=40H');
7 disp('MVI M,59H');
8 M=59;
9 M=hex2dec(['59']);
10 disp('2040H-->59H');
11 disp('INR M');
12 M=M+1; // increments the value at the memory
        location by 1.
13 m=dec2hex(M);
14 printf('\n Content of 2040H after increment= ');
15 disp(m);
16 disp('INX H'); //takes the program to the next
        memory location.
17 disp('H=20H      L=41H');
18 disp('MVI M,90H');
19 M=90;
20 M=hex2dec(['90']);
21 disp('2041H-->90H');
22 disp('DCR M');
23 M=M-1; //decrements the value at the memory location
        by 1.
24 m=dec2hex(M);
25 printf('\n Content of 2041H after decrement= ');
26 disp(m);

```

Scilab code Exa 7.8 LEFT ROTATION RLC OF BITS

```

1 //page no 233

```

```
Scilab 5.4.1 Console ? ? X
LXI H,2040H
H=20H    L=40H
MVI M,59H
2040H-->59H
INR M
Content of 2040H after increment=
5A
INX H
H=20H    L=41H
MVI M,90H
2041H-->90H
DCR M
Content of 2041H after decrement=
8F
-->
```

Figure 7.8: INCREMENT AND DECREMENT

```

2 // example no 7.8
3 // LEFT ROTATION (RLC) OF BITS.
4 clc;
5 // initially
6 printf('Accumulator= AAH \n');
7 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
8 printf(' 1 0 1 0 1 0 1 0   =AAH \n \n');
9 printf('CY= 0 \n \n');
10 printf('RLC \n \n');
11 printf('CY= 1 \n \n');
12 // carry flag is set because D7 bit was 1.
13 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
14 printf(' 0 1 0 1 0 1 0 1   =55H \n \n'); //
    after the execuuiou of first RLC.
15 // RLC instruction places D7 bit in CY flag as well
    as in D0 bit.
16 printf('RLC \n \n');
17 printf('CY= 0 \n \n');
18 // carry flag is reset because D7 bit was 0.
19 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
20 printf(' 1 0 1 0 1 0 1 0   =AAH \n \n'); //
    after the execuuiou of second RLC.
21 // RLC instruction places D7 bit in CY flag as well
    as in D0 bit.

```

Scilab code Exa 7.9 LEFT ROTATION RAL OF BITS

```

1 //page no 234
2 // example no 7.9
3 // LEFT ROTATION (RAL) OF BITS.
4 clc;
5 // initially
6 printf('Accumulator= AAH \n');

```

```

Scilab 5.4.1 Console
Accumulator= AAH
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 0 1 0 1 0   =AAH

CY= 0

RLC

CY= 1

D7 D6 D5 D4 D3 D2 D1 D0
0 1 0 1 0 1 0 1   =55H

RLC

CY= 0

D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 0 1 0 1 0   =AAH

-->

```

Figure 7.9: LEFT ROTATION RLC OF BITS

```

7 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
8 printf(' 1 0 1 0 1 0 1 0   =AAH \n \n');
9 printf('CY= 0 \n \n');
10 printf('RAL \n \n');
11 printf('CY= 1 \n \n');
12 // carry flag is set because D7 bit was 1.
13 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
14 printf(' 0 1 0 1 0 1 0 0   =54H \n \n'); //
    after the execuatuion of first RAL.
15 // RAL instruction places D7 bit in CY flag & CY
    flags bit is send to D0 bit.
16 printf('RAL \n \n');
17 printf('CY= 0 \n \n');
18 // carry flag is reset because D7 bit was 0.
19 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
20 printf(' 1 0 1 0 1 0 0 1   =A9H \n \n'); //
    after the execuatuion of second RAL.

```

```

Scilab 5.4.1 Console
Accumulator= AAH
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 0 1 0 1 0   =AAH

CY= 0

RAL

CY= 1

D7 D6 D5 D4 D3 D2 D1 D0
0 1 0 1 0 1 0 0   =54H

RAL

CY= 0

D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 0 1 0 0 1   =A9H

-->|

```

Figure 7.10: LEFT ROTATION RAL OF BITS

21 // RAL instruction places D7 bit in CY flag & CY
 flag bit is send to D0 bit.

Scilab code Exa 7.10 RIGHT ROTATION RRC AND RAR OF BITS

```

1 //page no 235
2 // example no 7.10
3 // RIGHT ROTATION (RRC & RAR) OF BITS.
4 clc;
5 // initially
6 printf('Accumulator= 81H \n');
7 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
8 printf('1 0 0 0 0 0 0 1   =81H \n \n');
9 printf('CY= 0 \n \n');

```



```

10 printf('RRC \n \n');
11 printf('CY= 1 \n \n');
12 // carry flag is set because D0 bit was 1.
13 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
14 printf(' 1  1  0  0  0  0  0  0   =C0H \n \n'); //
    after the execuion of RRC.
15 // RRC instruction places D0 bit in CY flag as well
    as in D7 bit.

16
17
18
19 // initially
20 printf('Accumulator= 81H \n');
21 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
22 printf(' 1  0  0  0  0  0  0  1   =81H \n \n');
23 printf('CY= 0 \n \n');
24 printf('RAR \n \n');
25 printf('CY= 1 \n \n');
26 // carry flag is set because D0 bit was 1.
27 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
28 printf(' 0  1  0  0  0  0  0  0   =40H \n \n'); //
    after the execuion of RAR.
29 // RAR instruction places D0 bit in CY flag & CY
    flags bit is send to D7 bit.

```

Scilab code Exa 7.11 COMPARISION OF DATA

```

1 //page no 241
2 //example no 7.11
3 // COMPARISION OF DATA.
4 clc;
5 disp('MVI A,64H'); //loads accumulator with 64H.
6 disp('A-->64H');

```

```
Scilab 5.4.1 Console
Accumulator= 81H
D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 0 0 0 0 1 =81H

CY= 0

RRC

CY= 1

D7 D6 D5 D4 D3 D2 D1 D0
1 1 0 0 0 0 0 0 =C0H

Accumulator= 81H
D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 0 0 0 0 1 =81H

CY= 0

RAR

CY= 1

D7 D6 D5 D4 D3 D2 D1 D0
0 1 0 0 0 0 0 0 =40H

-->
```

Figure 7.11: RIGHT ROTATION RRC AND RAR OF BITS

```

7  disp('LXI H,2050H'); // loads HL register pair.
8  disp('H=20H      L=50H');
9  disp('M-->9AH'); //assumed in the solution.
10 disp('CMP M');
11 //this command compares the contents of A with M by
    subtracting M from A.
12 A=hex2dec(['64']);
13 //register M has 9AH. Finding 2's compliment of 9AH.
14 M=hex2dec(['9A']);
15 a=dec2bin(A);
16 m=dec2bin(M);
17 t=isequalbitwise(a,m); //compares the two datas
    bitwise.
18 if(A==M) // Jump condition
19     printf('\n Result after comparision is= ');
20     printf('OUT1');
21 else
22     printf('\n Result after comparision is= ');
23     disp(t); //this shows the false condition of the
        bitwise comparison.

```

```
Scilab 5.4.1 Console ? ? X
MVI A,64H
A-->64H
LXI H,2050H
H=20H    L=50H
M-->9AH
CMP M
Result after comparision is=
F
-->|
```

Figure 7.12: COMPARISION OF DATA

Chapter 9

Stack And Subroutines

Scilab code Exa 9.1 PUSH POP AND DELAY INSTRUCTIONS

```
1 // page no 283
2 // example no 9.1
3 // PUSH POP AND DELAY INSTRUCTIONS
4 clc;
5 printf('LXI SP,2099H \n \n'); // the stack pointer
   is located at 2099H.
6 printf('LXI H,42F2H \n ');
7 printf('H--> 42      L-->F2 \n \n');
8 printf('PUSH H \n'); // sends the data of HL
   register pair in the stack.
9 // stack pointer is decremented by one to 2098H and
   the contents of the H register are copied to
   memory location 2098H
10 printf('2098H--> 42 \n');
11 // stack pointer is again decremented by one to 2097
   H and the contents of the L register are copied
   to memory location 2097H
12 printf('2097H--> F2 \n \n');
13 printf('Delay Counter \n \n');
14
15
```

```

Scilab 5.4.1 Console
LXI SP,2099H

LXI H,42F2H
H--> 42      L-->F2

PUSH H
2098H--> 42
2097H--> F2

Delay Counter

POP H
L--> F2H
H--> 42H

-->|

```

Figure 9.1: PUSH POP AND DELAY INSTRUCTIONS

```

16 n=hex2dec(['42F2']);
17 for i=1:n                               // DELAY LOOP
18     {
19     }
20 end
21
22 printf(' POP H \n'); // sends the data in the stack
    back to the HL register pair.
23 // the contents of the top of the stack are copied
    to L register and the stack pointer is
    incremented by one to 2098H
24 printf('L--> F2H \n');
25 // the contents of the current location of stack are
    copied to H register and the stack pointer is
    again incremented by one to 2099H.
26 printf('H--> 42H \n');

```

Scilab code Exa 9.2 EXCHANGE OF DATA USING STACK

```

1 // page no 285
2 // example no 9.2
3 // EXCHANGE OF DATA USING STACK.
4 clc;
5 printf('LXI SP,2400H \n \n'); // the stack pointer
   is located at 2400H.
6 printf('LXI H,2150H \n ');
7 printf('H--> 21      L-->50 \n \n');
8 printf('LXI B,2280H \n ');
9 printf('B--> 22      C-->80 \n \n');
10 printf('PUSH H \n'); // sends the data of HL
   register pair in the stack.
11 // stack pointer is decremented by one to 23FFH and
   the contents of the H register are copied to
   memory location 23FFH
12 printf('23FFH--> 21 \n');
13 // stack pointer is again decremented by one to 23
   FEH and the contents of the L register are copied
   to memory location 23FEH
14 printf('23FEH--> 50 \n \n');
15 printf('PUSH B \n'); // sends the data of BC
   register pair in the stack.
16 // stack pointer is decremented by one to 23FDH and
   the contents of the H register are copied to
   memory location 23FDH
17 printf('23FDH--> 22 \n');
18 // stack pointer is again decremented by one to 23
   FCH and the contents of the L register are copied
   to memory location 23FCH
19 printf('23FCH--> 80 \n \n');
20 printf('PUSH PSW \n'); // sends the data of
   accumulator & flag register in the stack.
21 // stack pointer is decremented by one to 23FBH and
   the contents of the H register are copied to
   memory location 23FBH
22 printf('23FBH--> contents of accumulator \n');
23 // stack pointer is again decremented by one to 23
   FAH and the contents of the L register are copied

```

```

        to memory location 23FAH
24 printf('23FAH--> contents of flag register \n \n');
25
26 printf('To exchange the data. \n \n')
27 printf(' POP PSW \n'); // sends the data in the
        stack back to the accumulator & flag register.
28 // the contents of the top of the stack are copied
        to A register and the stack pointer is
        incremented by one to 23FBH
29 printf('A--> contents of accumulator \n');
30 // the contents of the current location of stack are
        copied to flag register and the stack pointer is
        again incremented by one to 23FCH.
31 printf('F--> contents of flag register \n \n');
32 printf(' POP H \n'); // sends the data in the stack
        back to the HL register pair.
33 // the contents of the current location of the stack
        are copied to L register and the stack pointer
        is incremented by one to 23FDH
34 printf('L--> 80H \n');
35 // the contents of the current location of stack are
        copied to H register and the stack pointer is
        again incremented by one to 23FEH.
36 printf('H--> 22H \n \n');
37 printf(' POP B \n'); // sends the data in the stack
        back to the BC register pair.
38 // the contents of the current location of the stack
        are copied to C register and the stack pointer
        is incremented by one to 23FFH
39 printf('C--> 50H \n');
40 // the contents of the current location of stack are
        copied to B register and the stack pointer is
        again incremented by one to 2400H.
41 printf('B--> 21H \n');

```

```
Scilab 5.4.1 Console
LXI SP,2400H

LXI H,2150H
H--> 21    L-->50

LXI B,2280H
B--> 22    C-->80

PUSH H
23FFH--> 21
23FEH--> 50

PUSH B
23FDH--> 22
23FCH--> 80

PUSH PSW
23FBH--> contents of accumulator
23FAH--> contents of flag register

To exchange the data.

POP PSW
A--> contents of accumulator
F--> contents of flag register

POP H
L--> 80H
H--> 22H
```

Figure 9.2: EXCHANGE OF DATA USING STACK

Scilab code Exa 9.3 EXCHANGE INFORMATION BETWEEN STACK AND PROGRAM COUNTER

```
1 // page no 292
2 // example no 9.3
3 // EXCHANGE INFORMATION BETWEEN STACK AND PROGRAM
  COUNTER
4 clc;
5 printf('After the CALL instruction \n \n');
6 printf('STACK MEMORY: \n \n');
7 printf('23FFH--> 20 \n');
8 printf('23FEH-->43 \n');
9 printf('Stack pointer--> 23FEH \n');
10 printf('Program counter--> 2070H \n \n');
11 printf('After RET instruction \n \n');
12 printf('Program counter--> 2043H \n');
13 printf('Stack pointer--> 2400H');
```

```
Scilab 5.4.1 Console ? ? x
After the CALL instruction

STACK MEMORY:

23FFH--> 20
23FEH-->43
Stack pointer--> 23FEH
Program counter--> 2070H

After RET instruction

Program counter--> 2043H
Stack pointer--> 2400H
-->
```

Figure 9.3: EXCHANGE INFORMATION BETWEEN STACK AND PROGRAM COUNTER

Chapter 10

Code Conversion BCD Arithmetic and 16 bit Data Operations

Scilab code Exa 10.1 BCD TO BINARY

```
1 // page no 310
2 // example 10.1
3 // BCD TO BINARY
4 // BCD into its binary equivalent.
5 // given BCD no is 72
6 clc;
7 a=72;
8     x=modulo(a,10); // seperating the units digit
9     printf('Unpacked BCD1 ')
10    disp(dec2bin(x,8));
11 a=a/10; // seperating the tens place
    digit
12 a=floor(a);
13 printf(' \n \n Unpacked BCD2');
14 disp(dec2bin(a,8));
15 printf(' \n \n Multiply BCD2 by 10 and add BCD1');
```

```
Scilab 5.4.1 Console
Unpacked BCD1
00000010

Unpacked BCD2
00000111

Multiply BCD2 by 10 and add BCD1
-->
```

Figure 10.1: BCD TO BINARY

Scilab code Exa 10.2 ADDITION OF PACKED BCD NUMBERS

```
1 // page no 321
2 // example no 10.2
3 // ADDITION OF PACKED BCD NUMBERS
4 clc;
5 a=77;
6 b=48;
7 x=modulo(a,10);
8 y=modulo(b,10);
9 z=x+y;
10 if z>9 then
11     f=z+6;
12
13     printf('After addition BCD1 is: ')
14     disp(dec2bin(f));
15     printf('MSB of this sequence is the carry
16         generated after addition. \n \n')
17 else
18     printf('After addition BCD1 is: ')
19 end
```

```
Scilab 5.4.1 Console
After addition BCD1 is:
10101
MSB of this sequence is the carry generated after addition.

After addition BCD2 is:
10010
MSB of this sequence is the carry generated after addition.

BCD1 : 0101
BCD2: 0010
-->
```

Figure 10.2: ADDITION OF PACKED BCD NUMBERS

```
18     disp(z);
19 end
20 x=a/10;
21 x=floor(x);
22 y=b/10;
23 y=floor(y);
24 z=x+y;
25 if z>9 then
26     f=z+6;
27     f=f+1; // this 1 is the carry of BCD1.
28     printf('After addition BCD2 is: ')
29     disp(dec2bin(f));
30     printf('MSB of this sequence is the carry
           generated after addition.')
```

```
31 else
32     printf('After addition BCD1 is: ')
33     disp(z);
34 end
35 printf('\n \n BCD1 : 0101 \n \n');
36 printf('BCD2: 0010')
```

Scilab code Exa 10.3 EXCHANGE OF DATA

```
1 // page no 325
2 // example no 10.3
3 // EXCHANGE OF DATA
4 clc;
5 printf('2050H—> 3FH \n \n');
6 printf('2051H—> 42H \n \n');
7 printf('DE—> 856FH \n');
8 printf('D—> 85H      E—> 6FH \n \n');
9 printf('LHLD 2050H \n'); // loads the HL register
   pair with data on 2050H & 2051H.
10 printf('H—> 42H      L—> 3FH \n \n');
11 printf('XCHG \n'); // exchange the data of HL
   register pair with DE register pair.
12 printf('D<—>H      E<—>L \n');
13 printf('D—> 42H      E—> 3FH \n H—> 85H      L—>
   6FH \n \n');
14 printf('SHLD 2050H \n'); // stores the 16bit dat in
   HL register pair on memory location 2051H & 2050H
   .
15 printf('2050H—> 6FH \n');
16 printf('2051H—> 85H');
```

Scilab code Exa 10.4 ADDITION OF TWO 16 BIT NUMBERS

```
1 // page no 326
2 // examle no 10.4
3 // ADDITION OF TWO 16 BIT NUMBERS
4 clc;
5 printf('B—> 27H      C—> 93H \n');
6 printf('D—> 31H      E—> 82H \n \n');
7 b=hex2dec(['27']);
```

```

Scilab 5.4.1 Console
2050H--> 3FH

2051H--> 42H

DE--> 856FH
D--> 85H      E--> 6FH

LHLD 2050H
H--> 42H      L--> 3FH

XCHG
D<-->H      E<-->L
D--> 42H      E--> 3FH
H--> 85H      L--> 6FH

SHLD 2050H
2050H--> 6FH
2051H--> 85H
-->|

```

Figure 10.3: EXCHANGE OF DATA

```

8 c=hex2dec(['93']);
9 d=hex2dec(['31']);
10 e=hex2dec(['82']);
11 printf('MOV A,C \n \n');
12 a=c;
13 printf('ADD E \n');
14 a=a+e;
15 Z=a-256;
16 X=dec2hex(Z);
17 printf('Sum =')
18 disp(X);
19 if a>255 then
20     printf('CY=1 \n \n');
21     CY=1;
22 else
23     printf('CY=0 \n');
24     CY=0;
25 end
26 printf('MOV L,A \n');
27 printf('L-->');

```



```

28 disp(X);
29 printf('\n \n MOV A,B \n \n');
30 a=b;
31 printf('ADC D \n');
32 a=a+d+CY; // CY is added because of the previous
           carry as per the instructions ADC (add with carry
           )
33 T=dec2hex(a);
34 printf('Sum =')
35 disp(T);
36 if a>255 then
37     printf('CY=1 \n \n')
38 else
39     printf('CY=0 \n \n')
40 end
41 printf('MOV H,A \n');
42 printf('H-->');
43 disp(T);
44 printf('\n \n SHLD 2050H \n'); // stores the
           contents of HL register pair on memory locations
           2051H & 2050H.
45 printf('2050H--> ');
46 disp(X);
47 printf('2051H--> ');
48 disp(T);

```

Scilab code Exa 10.5 SUBTRACTION OF TWO 16 BIT NUMBERS

```

1 // page no 326
2 // examle no 10.5
3 // SUBTRACTION OF TWO 16 BIT NUMBERS
4 clc;
5 printf('B--> 85H      C--> 38H \n');

```

```
Scilab 5.4.1 Console
B--> 27H      C--> 93H
D--> 31H      E--> 82H

MOV A,C

ADD E
Sum =
 15
CY=1

MOV L,A
L-->
 15

MOV A,B

ADC D
Sum =
 59
CY=0

MOV H,A
H-->
 59

SHLD 2050H
2050H-->
 15
2051H-->
 59
```

Figure 10.4: ADDITION OF TWO 16 BIT NUMBERS

```

6 printf('D—> 62H      E—> A5H \n \n');
7 b=hex2dec(['85']);
8 c=hex2dec(['38']);
9 d=hex2dec(['62']);
10 e=hex2dec(['A5']);
11 printf('MOV A,C \n \n');
12 a=c;
13 printf('SUB E \n');
14 a=a-e;
15 Z=a+256;
16 X=dec2hex(Z);
17 printf('Difference =')
18 disp(X);
19 if a<0 then
20     printf('Borrow=1 \n \n');
21     B=1;
22 else
23     printf('Borrow=0 \n')
24     B=0;
25 end
26 printf('MOV C,A \n');
27 printf('C—>');
28 disp(X);
29 printf('\n \n MOV A,B \n \n');
30 a=b;
31 printf('SBB D \n');
32 a=a-d-B; // 1 is subtracted because of the previous
           borrow as per the instructions SBB (subtract with
           borrow)
33 T=dec2hex(a);
34 printf('Difference =')
35 disp(T);
36 if a<0 then
37     printf('Borrow=1 \n \n')
38 else
39     printf('Borrow=0 \n \n')
40 end
41 printf('MOV B,A \n');

```

```

Scilab 5.4.1 Console
B--> 85H      C--> 38H
D--> 62H      E--> A5H

MOV A,C

SUB E
Difference =
 93
Borrow=1

MOV C,A
C-->
 93

MOV A,B

SBB D
Difference =
 22
Borrow=0

MOV B,A
B-->
 22
-->

```

Figure 10.5: SUBTRACTION OF TWO 16 BIT NUMBERS

```

42 printf('B-->');
43 disp(T);

```

Scilab code Exa 10.6 DISPLAY CONTENTS OF STACK

```

1 // page no 327
2 // example no 10.6
3 // DISPLAY CONTENTS OF STACK
4 clc;
5 printf('LXI H,0000H \n'); // clears the HL
   register pair

```

```

Scilab 5.4.1 Console
LXI H,0000H
H--> 00H      L--> 00H

DAD SP
H--> higher bytes of stack pointer register
L--> lower bytes of stack pointer register

MOV A,H
H--> A

OUT PORT1

MOV A,L
L--> A

OUT PORT2
-->

```

Figure 10.6: DISPLAY CONTENTS OF STACK

```

6 printf('H--> 00H      L--> 00H \n \n');
7 printf('DAD SP \n'); // place the stack pointer
  content in HL
8 printf('H--> higher bytes of stack pointer register
  \n');
9 printf('L--> lower bytes of stack pointer register \
  n \n');
10 printf('MOV A,H \n'); // copies the contents of H in
  A.
11 printf('H--> A \n \n');
12 printf('OUT PORT1 \n \n');
13 printf('MOV A,L \n'); // copies the contents of L in
  A.
14 printf('L--> A \n \n');
15 printf('OUT PORT2');

```

Scilab code Exa 10.7 SUBROUTINE TO SET THE ZERO FLAG

```

1 // page no 327
2 // example no 10.7
3 // SUBROUTINE TO SET THE ZERO FLAG
4 clc;
5 printf('CHECK:  PUSH H \n \n'); // sends the
    contents of H to the location pointed by the
    stack pointer.
6 printf('          MVI L,FFH \n');
7 l=hex2dec(['FF']);
8 l=dec2bin(l,8);
9 printf('          L—> '); // set all bits in L to
    logic 1.
10 disp(l);
11 printf('\n \n          PUSH PSW \n \n'); // save
    flags on top of the stack
12 printf('          XTHL \n \n'); // set all bits in
    the top stack location.
13 printf('          POP PSW \n \n'); // now the zero
    flag is set.
14 printf('          JZ NOEROR \n \n');
15 printf('          JMP ERROR \n \n');
16 printf('NOEROR:  POP H \n \n'); // retrieves the data
    from the stack into H if zero flag is set
17 printf('          RET');

```

Scilab code Exa 10.8 TRANSFER A PROGRAM TO AN ADDRESS IN HL REGISTER

```

1 // page no 328
2 // example no 10.8
3 // TRANSFER A PROGRAM TO AN ADDRESS IN HL REGISTER
4 clc;
5 printf('\n \nThe program can be transfered using

```

```
Scilab 5.4.1 Console
CHECK:  PUSH H

        MVI L, FFH
        L-->
11111111

        PUSH PSW

        XTHL

        POP PSW

        JZ NOEROR

        JMP ERROR

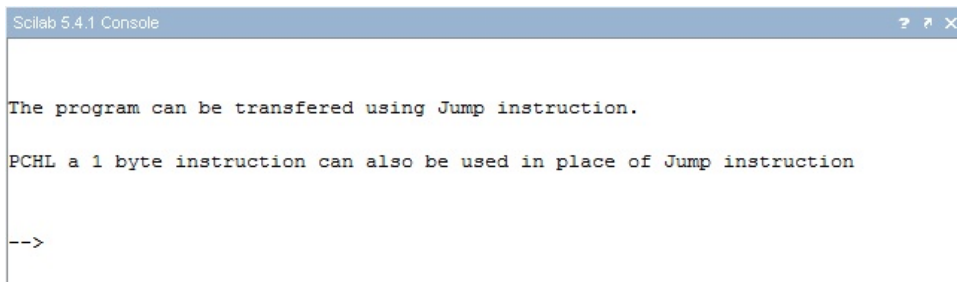
NOEROR: POP H

        RET

-->
```

Figure 10.7: SUBROUTINE TO SET THE ZERO FLAG

```
Jump instruction. \n \n');
6 printf('PCHL a 1 byte instruction can also be used
in place of Jump instruction \n \n');
```

A screenshot of a Scilab 5.4.1 Console window. The window title bar reads "Scilab 5.4.1 Console" and has standard window control buttons (minimize, maximize, close) on the right. The console area contains the following text:

```
The program can be transfered using Jump instruction.  
PCHL a 1 byte instruction can also be used in place of Jump instruction  
  
-->
```

Figure 10.8: TRANSFER A PROGRAM TO AN ADDRESS IN HL REGISTER

Chapter 12

Interrupts

Scilab code Exa 12.1 ENABLE INTERRUPTS

```
1 // page no 374
2 // example no 12.1
3 // ENABLE INTERRUPTS
4 clc;
5 printf('EI \n \n'); // enable interrupts
6 printf('MVI A,08H \n');
7 a=hex2dec(['8']);
8 b=dec2bin(a,8);
9 printf('A--> ')
10 disp(b);
11 printf('\n SIM \n \n'); // enable RST 7.5,6.5, and
    5.5
12 printf('D3=1      SIM functional \n');
13 printf('D2=0      Enable RST 7.5 \n');
14 printf('D1=0      Enable RST 6.5 \n');
15 printf('D0=0      Enable RST 5.5 \n');
```

```
Scilab 5.4.1 Console
EI
MVI A,08H
A-->
00001000

SIM
D3=1    SIM functional
D2=0    Enable RST 7.5
D1=0    Enable RST 6.5
D0=0    Enable RST 5.5

-->
```

Figure 12.1: ENABLE INTERRUPTS

Scilab code Exa 12.2 RESET INTERRUPT

```
1 // page no 374
2 // example no 12.2
3 // RESET 7.5 INTERRUPT
4 clc;
5 printf('MVI A,18H \n'); // set D4=1
6 a=hex2dec(['18']);
7 b=dec2bin(a,8);
8 printf('A--> ');
9 disp(b);
10 printf('\n \n SIM'); // Reset 7.5 interrupt
    flip flop
```

Scilab code Exa 12.3 CHECK PENDING INTERRUPT

```
1 // page no 375
2 // example no 12.3
3 // CHECK PENDING INTERRUPT
```

```

Scilab 5.4.1 Console
MVI A,18H
A-->
00011000

SIM
-->

```

Figure 12.2: RESET INTERRUPT

```

4  clc;
5  printf('RIM instruction interpretation \n \n');
6  printf('D7=SID                               Serial input
   data if any \n');
7  printf('D6,D5,D4= I7.5 ,I6.5 ,I5.5          Pending
   interrupts: 1= pending \n');
8  printf('D3=IE                               Interrupt enable
   flag: 1= enabled \n');
9  printf('D2,D1,D0= M7.5 ,M6.5 ,M5.5          Interrupt masks:
   1= masked \n \n \n');
10
11
12 printf('Instructions \n \n');
13 printf('          RIM \n');                // Read
   interrupt mask
14 printf('          MOV B,A \n');            // save mask
   information
15 printf('          ANI 20H \n');            // check
   whether RST 6.5 is pending
16 printf('          JNZ NEXT \n');
17 printf('          EI \n');
18 printf('          RET \n');                // RST 6.5 is
   not pending, return to main program
19 printf('NEXT:  MOV A,B \n');            // get bit
   pattern; RST 6.5 is pending
20 printf('          ANI 0DH \n');            // enables RST
   6.5 by setting D1=0

```

```

Scilab 5.4.1 Console
RIM instruction interpretation

D7=SID                Serial input data if any
D6,D5,D4= I7.5,I6.5,I5.5  Pending interrupts: 1= pending
D3=IE                Interrupt enable flag: 1= enabled
D2,D1,D0= M7.5,M6.5,M5.5  Interrupt masks: 1= masked

Instructions

    RIM
    MOV B,A
    ANI 20H
    JNZ NEXT
    EI
    RET
NEXT:  MOV A,B
       ANI 0DH
       ORI 08H
       SIM
       JMP SERV
-->

```

Figure 12.3: CHECK PENDING INTERRUPT

```

21 printf('          ORI 08H\n');          // enable SIM by
    setting D3=1
22 printf('          SIM \n');
23 printf('          JMP SERV \n');        // jump to
    service routine for RST 6.5

```

Chapter 14

Programmable Interface Devices

Scilab code Exa 14.1 INITIALIZE HYPOTHETICAL CHIP AS OUTPUT BUFFER

```
1 // page no 414
2 // example no 14.1
3 // INITIALIZE HYPOTHETICAL CHIP AS OUTPUT BUFFER
4 clc;
5 printf('MVI A,01H \n'); // Set D0=1, D1 through
   D7 are don't care lines.
6 a=hex2dec(['1']);
7 b=dec2bin(a,8);
8 printf('A--> ')
9 disp(b);
10 printf('\n \n OUT FFH \n \n'); // write in the
   control register.
11 printf('MVI A,BYTE1 \n \n'); // load data byte.
12 printf('OUT FEH'); // send data out.
```

```

Scilab 5.4.1 Console
MVI A,01H
A-->
00000001

OUT FFH

MVI A,BYTE1

OUT FEH
-->

```

Figure 14.1: INITIALIZE HYPOTHETICAL CHIP AS OUTPUT BUFFER

Scilab code Exa 14.2 ADDRESS DETERMINATION OF GIVEN FIGURE

```

1 // page no 420
2 // example no 14.2
3 // ADDRESS DETERMINATION OF GIVEN FIGURE
4 clc;
5 printf('To select the chip: \n \n');
6 printf('A15 A14 A13 A12 A11 \n');
7 printf(' 0  0  0  1  0 \n \n');
8 printf('A15,A14          Enable lines of 8205 \n'
9       ');
9 printf('A13,A12,A11      Input logic to activate
10        the putput line O4 of the 8205 \n \n');
10 printf('A15,A14,A13,A12,A11 = A7,A6,A5,A4,A3, = 2H \
11        n \n');
11 printf('AD2  AD1  AD0 = Address Ports \n');
12 printf(' 0    0    0 = 20H Control or status
13        register \n');
13 printf(' 0    0    1 = 21H Port A \n');
14 printf(' 0    1    0 = 22H Port B \n');
15 printf(' 0    1    1 = 23H Port C \n');

```

```
Scilab 5.4.1 Console
To select the chip:

A15 A14 A13 A12 A11
0 0 0 1 0

A15,A14          Enable lines of 8205
A13,A12,A11      Input logic to activate the putput line O4 of the 8205

A15,A14,A13,A12,A11 = A7,A6,A5,A4,A3, = 2H

AD2  AD1  AD0 = Address Ports
0    0    0 = 20H Control or status register
0    0    1 = 21H Port A
0    1    0 = 22H Port B
0    1    1 = 23H Port C
1    0    0 = 24H Timer LSB
1    0    1 = 25H Timer MSB

Port numbers in given figure thus range from 20H-25H
-->
```

Figure 14.2: ADDRESS DETERMINATION OF GIVEN FIGURE

```
16 printf(' 1      0      0 = 24H Timer LSB \n');
17 printf(' 1      0      1 = 25H Timer MSB \n \n');
18 printf('Port numbers in given figure thus range from
      20H-25H');
```

Chapter 15

General Purpose Programmable Peripheral Devices

Scilab code Exa 15.1 PORT ADDRESS CONTROL WORD ADDRESS
AND READ THE DIP SWITCHES

```
1 // page no 449
2 // example no 15.1
3 // PORT ADDRESS CONTROL WORD ADDRESS AND READ THE
  DIP SWITCHES
4 clc;
5 printf('1 Port Address \n \n');
6 printf('Port A           =      8000H  (A1=0,A0
   =0) \n');
7 printf('Port B           =      8001H  (A1=0,A0
   =1) \n');
8 printf('Port C           =      8002H  (A1=1,A0
   =0) \n');
9 printf('Control Register =      8003H  (A1=1,A0
   =1) \n \n');
10
11
```



```

12 printf('2 Control Word \n \n');
13 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
14 printf('1 0 0 0 0 0 1 1          = 83H \n \n')
    ;
15 printf('D7=1      I/O Function \n');
16 printf('D6,D5=0   Port A in Mode 0 \n');
17 printf('D4=0      Port A= output \n');
18 printf('D3=0      Port C upper= output \n');
19 printf('D2=0      Port B in Mode 0 \n');
20 printf('D1=1      Port B= input \n');
21 printf('D0=1      Port C1= input \n \n');
22
23
24 printf('3 Program \n \n');
25 printf('MVI A,83H \n'); // load accumulator with
    the control word.
26 printf('STA 8003H \n'); // write word in the
    control register to initialize the ports.
27 printf('LDA 8001H \n'); // reads switches at port B
    .
28 printf('STA 8000H \n'); // display the reading at
    port A.
29 printf('LDA 8002H \n'); // read switches at port C.
30 printf('ANI 0FH \n'); // mask the upper four bits
    of port C, these bits are not input data.
31 printf('RLC \n'); // rotate and place the
    data in the upper half of the accumulator.
32 printf('RLC \n');
33 printf('RLC \n');
34 printf('RLC \n');
35 printf('STA 8002H \n'); // display data at port C
    upper.
36 printf('HLT \n');

```

```

Scilab 5.4.1 Console
1 Port Address

Port A           = 8000H (A1=0,A0=0)
Port B           = 8001H (A1=0,A0=1)
Port C           = 8002H (A1=1,A0=0)
Control Register = 8003H (A1=1,A0=1)

2 Control Word

D7 D6 D5 D4 D3 D2 D1 D0
1  0  0  0  0  0  1  1      = 83H

D7=1    I/O Function
D6,D5=0 Port A in Mode 0
D4=0    Port A= output
D3=0    Port C upper= output
D2=0    Port B in Mode 0
D1=1    Port B= input
D0=1    Port C1= input

3 Program

MVI A,83H
STA 8003H
LDA 8001H
STA 8000H
LDA 8002H
ANI 0FH
RLC
RLC
RLC
RLC
STA 8002H
HLT

```

Figure 15.1: PORT ADDRESS CONTROL WORD ADDRESS AND READ THE DIP SWITCHES

Scilab code Exa 15.2 BSR CONTROL WORD SUBROUTINE

```
1 // page no 453
2 // example no 15.2
3 // BSR CONTROL WORD SUBROUTINE
4 clc;
5 printf('BSR Control Words \n \n');
6 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
7 printf('0 0 0 0 1 1 1 1          = 0FH      To
   set bit PC7 \n');
8 printf('0 0 0 0 1 1 1 0          = 0EH      To
   reset bit PC7 \n');
9 printf('0 0 0 0 0 1 1 1          = 07H      To
   set bit PC3 \n');
10 printf('0 0 0 0 0 1 1 0         = 06H      To
   reset bit PC3 \n \n');
11
12
13 printf('Port Address \n \n');
14 printf('Control Register Address = 83H \n \n');
15
16
17 printf('Subroutine \n \n');
18 printf('MVI A,0FH \n'); // load byte in
   accumulator to set PC7
19 printf('OUT 83H \n'); // set PC7=1
20 printf('MVI A,07H \n'); // load byte in
   accumulator to set PC3.
21 printf('OUT 83H \n'); // set PC3=1.
22 printf('CALL DELAY \n'); // this is a 10
   microsec delay.
23 printf('MVI A,06H \n'); // load byte in
   accumulator to reset PC3
24 printf('OUT 83H \n'); // reset PC3
25 printf('MVI A,0EH \n'); // load byte in
   accumulator to reset PC7.
26 printf('OUT 83H \n'); // reset PC7
27 printf('RET');
```

```

Scilab 5.4.1 Console
BSR Control Words

D7 D6 D5 D4 D3 D2 D1 D0
0 0 0 0 1 1 1 1      = 0FH      To set bit PC7
0 0 0 0 1 1 1 0      = 0EH      To reset bit PC7
0 0 0 0 0 1 1 1      = 07H      To set bit PC3
0 0 0 0 0 1 1 0      = 06H      To reset bit PC3

Port Address

Control Register Address = 83H

Subroutine

MVI A,0FH
OUT 83H
MVI A,07H
OUT 83H
CALL DELAY
MVI A,06H
OUT 83H
MVI A,0EH
OUT 83H
RET
-->

```

Figure 15.2: BSR CONTROL WORD SUBROUTINE

Scilab code Exa 15.3 INSTRUCTIONS TO GENERATE A PULSE FROM COUNTER 0

```

1 // page no 483
2 // example no 15.3
3 // INSTRUCTIONS TO GENERATE A PULSE FROM COUNTER 0
4 clc;
5 printf('Control Word \n \n');
6 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
7 printf('0 0 0 1 0 1 0 0          = 14H \n \n')
  ;

```

```

8 printf('D7,D6=0          Select counter 0 \n');
9 printf('D5,D4=01        Load 8 bit count \n');
10 printf('D3,D2,D1=010    Mode 2 \n');
11 printf('D0=0            Binary Count \n \n');
12
13
14 printf('Count \n \n');
15 count=(50*10^-6)/(0.5*10^-6);
16 printf('Count= ');
17 disp(count);
18 disp(dec2hex(count));
19 printf('in hexadecimal \n \n');
20
21
22 printf('Instructions \n \n');
23 printf('PULSE: \n')
24 printf('MVI A,00010100B \n');    // control word
    mode 2 & counter 0.
25 printf('OUT 83H \n');          // write in 8254
    control register.
26 printf('MVI A,64H \n');        // low order byte
    of the count.
27 printf('OUT 80H \n');          // load counter 0
    with low order byte
28 printf('HLT');

```

Scilab code Exa 15.4 INSTRUCTIONS TO GENERATE SQUARE WAVE PULSE FROM COUNTER 1

```

1 // page no 484
2 // example no 15.4
3 // INSTRUCTIONS TO GENERATE SQUARE WAVE PULSE FROM
    COUNTER 1

```

```
Scilab 5.4.1 Console ? ? x
Control Word
D7 D6 D5 D4 D3 D2 D1 D0
0 0 0 1 0 1 0 0      = 14H

D7,D6=0      Select counter 0
D5,D4=01     Load 8 bit count
D3,D2,D1=010 Mode 2
D0=0         Binary Count

Count

Count=
  100.

  64
in hexadecimal

Instructions

PULSE:
MVI A,00010100B
OUT 83H
MVI A,64H
OUT 80H
HLT
-->
```

Figure 15.3: INSTRUCTIONS TO GENERATE A PULSE FROM COUNTER 0

```

4  clc;
5  printf('Control Word \n \n');
6  printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
7  printf('0 1 1 1 0 1 1 0           = 76H \n \n')
   ;
8  printf('D7,D6=01           Select counter 1 \n');
9  printf('D5,D4=11           Load 16 bit count \n');
10 printf('D3,D2,D1=011       Mode 3 \n');
11 printf('D0=0               Binary Count \n \n');
12
13
14 printf('Count \n \n');
15 count=(1*10-3)/(0.5*10-6);
16 printf('Count= ');
17 disp(count);
18 b=dec2hex(2000);
19 disp(b);
20 printf('in hexadecimal \n \n');
21
22
23 printf('Instructions \n \n');
24 printf('SQWAVE: \n');
25 printf('MVI A,01110110B \n');    // control word
   mode 3 & counter 1.
26 printf('OUT 83H \n');          // write in 8254
   control register.
27 printf('MVI A,D0H \n');        // low order byte
   of the count.
28 printf('OUT 81H \n');          // load counter 1
   with low order byte.
29 printf('MVI A,07H \n');        // high order byte
   of the count.
30 printf('OUT 81H \n');          // load counter 1
   with high order byte
31 printf('HLT');

```

```

Scilab 5.4.1 Console
Control Word

D7 D6 D5 D4 D3 D2 D1 D0
0 1 1 1 0 1 1 0      = 76H

D7,D6=01      Select counter 1
D5,D4=11      Load 16 bit count
D3,D2,D1=011  Mode 3
D0=0          Binary Count

Count

Count=
  2000.

  7D0
in hexadecimal

Instructions

SQWAVE:
MVI A,01110110B
OUT 83H
MVI A,D0H
OUT 81H
MVI A,07H
OUT 81H
HLT
-->

```

Figure 15.4: INSTRUCTIONS TO GENERATE SQUARE WAVE PULSE FROM COUNTER 1

Scilab code Exa 15.5 SUBROUTINE TO GENERATE AN INTERRUPT

```

1 // page no 486
2 // example no 15.5
3 // SUBROUTINE TO GENERATE AN INTERRUPT
4 clc;
5 printf('Control Word \n \n');

```



```

6 printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
7 printf('0 1 1 1 0 1 0 0          = 74H
   Counter 1 \n');
8 printf('1 0 0 1 0 1 0 0          = 94H
   Counter 2 \n \n');
9 printf('D7,D6          Select counter 1 \n');
10 printf('D5,D4          Load count \n');
11 printf('D3,D2,D1=010    Mode 2 \n');
12 printf('D0=0           Binary Count \n \n');
13
14
15
16 printf('Instructions \n \n');
17 printf('CNT1LO EQU 50H \n');
18 printf('CNT1HI EQU C3H \n');
19 printf('COUNT2 EQU 40H \n');
20 printf('SECOND MVI A,01110100B \n'); // control
   word mode 2 & counter 1.
21 printf('          OUT 83H \n'); // write in
   8254 control register.
22 printf('          MVI A,10010100B \n'); // control
   word mode 2 & counter 2.
23 printf('          OUT 83H \n'); // write in
   8254 control register.
24 printf('          MVI A,CNT1LO \n'); // Low
   order byte of count 50000
25 printf('          OUT 81H \n'); // Load
   counter 1 with low order byte
26 printf('          MVI A,CNT1HI \n'); // high
   order byte of count 50000.
27 printf('          OUT 81H \n'); // load
   counter 1 with high order byte
28 printf('          MVI A,COUNT2 \n'); // Count
   for Counter 2.
29 printf('          OUT 82H \n'); // load
   counter 2.
30 printf('          RET');

```

```

Scilab 5.4.1 Console
Control Word

D7 D6 D5 D4 D3 D2 D1 D0
0 1 1 1 0 1 0 0      = 74H      Counter 1
1 0 0 1 0 1 0 0      = 94H      Counter 2

D7,D6      Select counter 1
D5,D4      Load count
D3,D2,D1=010 Mode 2
D0=0       Binary Count

Instructions

CNT1LO EQU 50H
CNT1HI EQU C3H
COUNT2 EQU 40H
SECOND MVI A,01110100B
      OUT 83H
      MVI A,10010100B
      OUT 83H
      MVI A,CNT1LO
      OUT 81H
      MVI A,CNT1HI
      OUT 81H
      MVI A,COUNT2
      OUT 82H
      RET
-->

```

Figure 15.5: SUBROUTINE TO GENERATE AN INTERRUPT

Scilab code Exa 15.6 EXPLANATION OF INSTRUCTIONS

```

1 // page no 493
2 // example no 15.6
3 // EXPLANATION OF INSTRUCTIONS
4 clc;
5 printf('1) DI instruction disables the interrupts. \
      n \n');
6 printf('2) Command word 76H specifies the following

```

```

        parameters \n');
7 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
8 printf('0 1 1 1 0 1 1 0          =76H \n');
9 printf('A7,A6,A5      Low order address bits \n');
10 printf('A3          Edge triggered \n');
11 printf('A2          Call address interval is four
        locations \n');
12 printf('A1          Single 8259A \n \n');
13 printf('Low order byte of the IR0 call address \n');
14 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
15 printf('0 1 1 0 0 0 0 0          =60H \n');
16 printf('The address bits A4–A0 are supplied by 8259A
        . \n');
17 printf('Subsequent addresses are four locations
        apart eg. IR1=64H')
18 printf('3) Port address of the 8259SA for ICW1 is 80
        H, A0 should be at \n logic 0 & the other bits
        are determined by the decoder. \n \n');
19 printf('4) Command word ICW2 is 20H. \n \n');
20 printf('5) Port address of ICW2 is 81H, A0 should be
        at logic 1. ');

```

Scilab code Exa 15.8 INITIALIZATION INSTRUCTIONS FOR DMA

```

1 // page no 502
2 // example no 15.8
3 // INITIALIZATION INSTRUCTIONS FOR DMA
4 clc;
5 printf('MVI A,00000100B \n \n');
6 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
7 printf('0 0 0 0 0 1 0 0 \n');
8 printf('A2=1          Disable DMA \n \n');
9 printf('OUT 08H \n');

```

```

Scilab 5.4.1 Console
1) DI instruction disables the interrupts.

2) Command word 76H specifies the following parameters
A7 A6 A5 A4 A3 A2 A1 A0
0 1 1 1 0 1 1 0      =76H
A7,A6,A5      Low order address bits
A3            Edge triggered
A2            Call address interval is four locations
A1            Single 8259A

Low order byte of the IRO call address
A7 A6 A5 A4 A3 A2 A1 A0
0 1 1 0 0 0 0 0      =60H
The address bits A4-A0 are supplied by 8259A.
Subsequent addresses are four locations apart eg. IR1=64H3) Port address of the
logic 0 & the other bits are determined by the decoder.

4) Command word ICW2 is 20H.

5) Port address of ICW2 is 81H, A0 should be at logic 1.
-->

```

Figure 15.6: EXPLANATION OF INSTRUCTIONS

```

10 printf('MVI A,00000111B \n \n');
11 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
12 printf('0 0 0 0 0 1 1 1 \n');
13 printf('A7,A6=00      Demand mode \n');
14 printf('A5=0          Increment address \n');
15 printf('A4=0          Disable auto load \n');
16 printf('A3,A2=01      Write \n');
17 printf('A1,A0=11      Ch 3 \n \n');
18 printf('OUT 0BH \n');          // Send to mode reg
.
19 printf('MVI A,75H \n');          // Low order byte
    of starting address
20 printf('OUT 06H \n');          // Output to CH3
    memory address reg.
21 printf('MVI A,40H \n');          // High order byte
    of starting address
22 printf('OUT 06H \n');          // Output to CH3
    memory address reg.
23 printf('MVI A,FFH \n');          // Low order byte

```

```

    of the count 03FFH
24 printf('OUT 07H \n');           // Output to CH3
    count reg.
25 printf('MVI A,03H \n');         // High order byte
    of the count 03FFH
26 printf('OUT 07H \n');         // Output to CH3
    count reg.
27 printf('MVI A,10000000B \n \n');
28 printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
29 printf('1 0 0 0 0 0 0 0 \n');
30 printf('A7,A6=10    DACK DREQ High \n');
31 printf('A5=0       Late write \n');
32 printf('A4=0       Fixed priority \n');
33 printf('A3=0       Normal time \n');
34 printf('A2=0       DMA enable \n');
35 printf('A0=0       Disable mem to mem \n \n');
36 printf('OUT 08H \n');

```

```

Scilab 5.4.1 Console
MVI A,00000100B

A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 1 0 0
A2=1      Disable DMA

OUT 08H
MVI A,00000111B

A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 1 1 1
A7,A6=00  Demand mode
A5=0      Increment address
A4=0      Disable auto load
A3,A2=01  Write
A1,A0=11  Ch 3

OUT 0BH
MVI A,75H
OUT 06H
MVI A,40H
OUT 06H
MVI A,FFH
OUT 07H
MVI A,03H
OUT 07H
MVI A,10000000B

```

Figure 15.7: INITIALIZATION INSTRUCTIONS FOR DMA

```

A7 A6 A5 A4 A3 A2 A1 A0
1 0 0 0 0 0 0 0
A7,A6=10  DACK DREQ High
A5=0      Late write
A4=0      Fixed priority
A3=0      Normal time
A2=0      DMA enable
A0=0      Disable mem to mem

OUT 08H

-->|

```

Figure 15.8: INITIALIZATION INSTRUCTIONS FOR DMA

Chapter 19

Appendix A Number System

Scilab code Exa 19.1 BINARY INTO HEX AND OCTAL

```
1 // page no 622
2 // example no A.1
3 // BINARY INTO HEX AND OCTAL
4 clc;
5 printf('Binary no= 10011010 \n \n');
6 str='10011010';
7 h=bin2dec(str);
8 H=dec2hex(h);
9 printf('Hex Equivalent= ');
10 disp(H);
11 O=dec2oct(h);
12 printf('\n Octal Equivalent= ')
13 disp(O);
```

Scilab code Exa 19.2 SUBTRACTION OF TWO NUMBERS

```
1 // page no 623
```

```
Scilab 5.4.1 Console
Binary no= 10011010

Hex Equivalent=
9A

Octal Equivalent=
232

-->
```

Figure 19.1: BINARY INTO HEX AND OCTAL

```
2 // example no A.2
3 // SUBTRACTION OF TWO NUMBERS
4 clc;
5 printf('Minuend: 52 \n');
6 printf('Subtrahend: 23 \n \n');
7 printf('BORROW METHOD \n \n');
8
9 m=5*10+2; // minuend
10 s=2*10+3; // subtrahend
11 // to subtract 3 from 2, 10 must be borrowed from
    the second place of the minuend.
12
13 m=4*10+12;
14
15 sub=m-s;
16 printf('Subtraction= ')
17 disp(sub);
18
19 printf('\n \n 10s COMPLEMENT METHOD \n \n');
20
21 // 9's complement of 23 is
22
23 n=99-23;
24 // add 1 to the 9's complement to find the 10's
    complement
25 t=n+1;
```



```
Scilab 5.4.1 Console
Minuend: 52
Subtrahend: 23

BORROW METHOD

Subtraction=
  29.

10s COMPLEMENT METHOD

Subtraction=
  29.

-->
```

Figure 19.2: SUBTRACTION OF TWO NUMBERS

```
26 // add the 10's complement of the subtrahend(23) to
    minuend(52) to subtract 23 from 52
27 a=m+t;
28 // subtract 100 from a to compensate for the 100
    that was added to find the 10's complement of 23
29 sub=a-100;
30 printf('Subtraction= ');
31 disp(sub);
```

Scilab code Exa 19.3 SUBTRACTION OF TWO NUMBERS

```
1 // page no 624
2 // example no A.3
3 // SUBTRACTION OF TWO NUMBERS
4 clc;
5 printf('Minuend: 23 \n');
6 printf('Subtrahend: 52 \n \n');
```

```

7 printf('BORROW METHOD \n \n');
8
9 m=2*10+3; // minuend
10 s=5*10+2; // subtrahend
11 // subtraction of the digits in the first place
    results in
12 a=3-2;
13 // to subtract the digits in the second place a
    borrow is required from the third place. assuming
    1 at third place.
14
15 x=12-5; // with a borrowed 1 from the third place
16
17 sub=10*x+a;
18 printf('Subtraction= ');
19 disp(sub);
20 printf('this is negative 29, expressed in 10s
    complement. \n negative sign is verified by the
    borrowed 1 from the third place.');
```

21

```

22 printf('\n \n 10s COMPLEMENT METHOD \n \n');
```

23

```

24 // 9's complement of 52 is
25
26 n=99-52;
27 // add 1 to the 9's complement to find the 10's
    complement
28 t=n+1;
29 // add the 10's complement of the subtrahend(23) to
    minuend(52) to subtract 23 from 52
30 a=m+t;
31
32 printf('Subtraction= ');
33 disp(a);
34 printf('this is negative 29, expressed in 10s
    complement');
```

```
Scilab 5.4.1 Console
Minuend: 23
Subtrahend: 52

BORROW METHOD

Subtraction=
    71.
this is negative 29, expressed in 10s complement.
negative sign is verified by the borrowed 1 from the third place.

10s COMPLEMENT METHOD

Subtraction=
    71.
this is negative 29, expressed in 10s complement
-->|
```

Figure 19.3: SUBTRACTION OF TWO NUMBERS

Scilab code Exa 19.4 2s COMPLIMENT OF BINARY NUMBER

```
1 // page no 625
2 // example no A.4
3 // 2's COMPLIMENT OF BINARY NUMBER
4 clc;
5 printf('Given binary no= 00011100 \n \n');
6 str='00011100'
7 d=bin2dec(str);
8 x=bitcmp(d,8);
9 s=x+1;
10 y=dec2bin(s);
11 printf('2s complement=');
12 disp(y);
```

```
Scilab 5.4.1 Console
Given binary no= 00011100

2s complement=
11100100

-->
```

Figure 19.4: 2s COMPLIMENT OF BINARY NUMBER

Scilab code Exa 19.5 SUBTRACTION OF TWO NUMBERS

```
1 // page no 626
2 // example no A.5
3 // SUBTRACTION OF TWO NUMBERS
4 clc;
5 printf('Subtrahend= 32H \n');
6 printf('Minuend= 45H \n \n');
7 // finding 2's complement of subtrahend (32H);
8 m=hex2dec(['45']);
9 x=hex2dec(['32']);
10 y=bitcmp(x,8); // 1's compliment of 32H
11 z=y+1; // 2's compliment of 32H
12 s=m+z;
13 f=s-256; // to compensate the effect of 2's
    compliment
14 e=dec2hex(f);
15 printf('Subtraction= ');
16 disp(e);
```

```
Scilab 5.4.1 Console
Subtrahend= 32H
Minuend= 45H

Subtraction=
13

-->
```

Figure 19.5: SUBTRACTION OF TWO NUMBERS

Scilab code Exa 19.6 SUBTRACTION OF TWO NUMBERS

```
1 // page no 626
2 // example no A.6
3 // SUBTRACTION OF TWO NUMBERS
4 clc;
5 printf('Subtrahend= 45H \n');
6 printf('Minuend= 32H \n \n');
7 // finding 2's complement of subtrahend (32H);
8 m=hex2dec(['32']);
9 x=hex2dec(['45']);
10 y=bitcmp(x,8); // 1's compliment of 32H
11 z=y+1; // 2's compliment of 32H
12 s=m+z;
13 r=dec2hex(s);
14 printf('Subtraction= ');
15 disp(r);
16 printf('The result is negative & it is expressed in
    2s complement.')
```

Scilab code Exa 19.7 SUBTRACTION OF UNSIGNED NUMBERS

```
1 // page no 628
```

```
Scilab 5.4.1 Console
Subtrahend= 45H
Minuend= 32H

Subtraction=
ED
The result is negative & it is expressed in 2s complement.
-->
```

Figure 19.6: SUBTRACTION OF TWO NUMBERS

```
2 // example no A.7
3 // SUBTRACTION OF UNSIGNED NUMBERS
4 clc;
5 printf('Part a \n \n')
6 printf('Subtrahend= 62H \n');
7 printf('Minuend= FAH \n \n');
8 // finding 2's complement of subtrahend (62H);
9 m=hex2dec(['FA']);
10 x=hex2dec(['62']);
11 y=bitcmp(x,8); // 1's compliment of 62H
12 z=y+1; // 2's compliment of 62H
13 s=m+z;
14 f=s-256; // to compensate the effect of 2's
    compliment
15 e=dec2hex(f);
16 printf('Subtraction= ');
17 disp(e);
18 printf('This result is positive \n \n');
19
20
21 printf('Part b \n \n')
22 printf('Subtrahend= FAH \n');
23 printf('Minuend= 62H \n \n');
24 // finding 2's complement of subtrahend (FAH);
25 m=hex2dec(['62']);
26 x=hex2dec(['FA']);
27 y=bitcmp(x,8); // 1's compliment of FAH
28 z=y+1; // 2's compliment of FAH
```

```
Scilab 5.4.1 Console
Part a
Subtrahend= 62H
Minuend= FAH
Subtraction=
 98
This result is positive

Part b
Subtrahend= FAH
Minuend= 62H
Subtraction=
 68
The result is negative & it is expressed in 2s complement.
-->
```

Figure 19.7: SUBTRACTION OF UNSIGNED NUMBERS

```
29 s=m+z;
30 r=dec2hex(s);
31 printf('Subtraction= ');
32 disp(r);
33 printf('The result is negative & it is expressed in
      2s complement.')
```

Scilab code Exa 19.8 SUBTRACTION OF SIGNED NUMBERS

```
1 // page no 629
2 // example no A.8
3 // SUBTRACTION OF SIGNED NUMBERS
4 clc;
5 printf('Part a \n \n');
6 printf('Minuend= FAH \n \n');
7 printf('It is a negative no since D7= 1 for FAH,
```

```

    this must be represented in \n2s compliment form.
    \n');
8 // finding 2's complement of subtrahend (FAH);
9 m=hex2dec(['FA']);
10 x=hex2dec(['62']);
11 y=bitcmp(m,8); // 1's compliment of FAH
12 z=y+1; // 2's compliment of FAH
13 printf('2s compliment of minuend is= ');
14 disp(z);
15
16 printf('\n \n Subtrahend= 62H \n');
17 printf('It is a positive no since D7= 0 for 62H. \n'
    );
18 // subtraction can be represented as
19 // FAH-62H= (-06H)-(+62H)
20 s=-x-z;
21 a=-s;
22 d=dec2hex(a);
23 printf('Subtraction= ');
24 disp(s);
25 disp(d);
26 printf('in hexadecimal with a negative sign \n \n');
27 g=bitcmp(a,8); // 1's compliment of result
28 q=g+1; // 2's compliment of result
29 e=dec2hex(q);
30 printf('2s compliment of result would be= ');
31 disp(e);

```

Scilab code Exa 19.9 ADDITION OF TWO POSITIVE NUMBERS

```

1 //page no 629
2 //example no A.9
3 // ADDITION OF TWO POSITIVE NUMBERS

```



```

Scilab 5.4.1 Console
Part a
Minuend= FAH

It is a negative no since D7= 1 for FAH, this must be represented in
2s compliment form.
2s compliment of minuend is=
    6.

Subtrahend= 62H
It is a positive no since D7= 0 for 62H.
Subtraction=
- 104.

68
in hexadecimal with a negative sign

2s compliment of result would be=
98
-->

```

Figure 19.8: SUBTRACTION OF SIGNED NUMBERS

```

4  clc;
5  //the given numbers are 41H & 54H.
6  A=hex2dec(['41']);
7  B=hex2dec(['54'])
8  Y=A+B;
9  X=dec2hex(Y);
10 printf('Sum =')
11 disp(X);
12 if Y>255 then // checking the carry flag
13     printf('CY=1 \n \n')
14 else
15     printf('CY=0 \n \n')
16 end
17 if Y>127 then // checking the sign flag.
18     printf('S=1 \n \n')
19 else
20     printf('S=0 \n \n')

```

```
Scilab 5.4.1 Console
Sum =
95
CY=0

S=1

Z=0

-->
```

Figure 19.9: ADDITION OF TWO POSITIVE NUMBERS

```
21 end
22 if Y>0 then // checking the zero flag.
23     printf('Z=0 \n \n')
24 else
25     printf('Z=1 \n \n')
26 end
```
