Scilab Textbook Companion for Microprocessor Architecture, Programming & Applications with the 8085 by R. S. Goankar[1]

Created by
Rishabh Jain
B.Tech
Electronics Engineering
Dr. A.P.J. Abdul Kalam Technical University
College Teacher
None
Cross-Checked by
K. V. P. Pradeep

June 2, 2016

[1]Funded 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

List of Figures

Chapter 2

Microprocessor Architecture and Microcomputer Systems

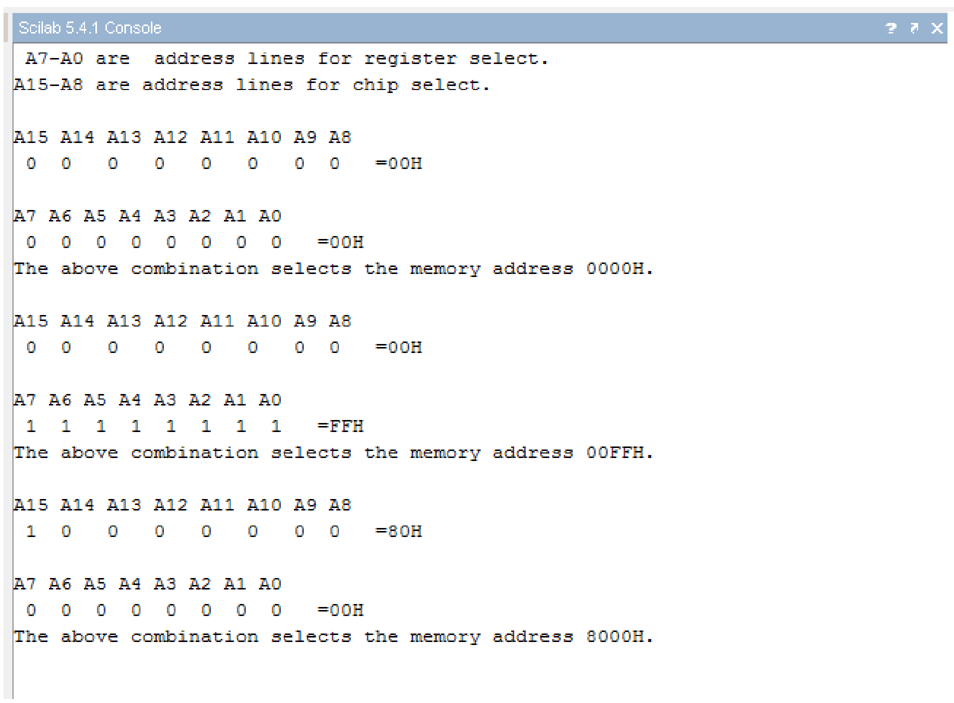
Scilab code Exa 2.1 MEMORY ADDRESS RANGE

```
//page no 39
//example no 2.1
//MEMORY ADDRESS RANGE.
clc;
printf('A7-A0 are  address lines for register select
    . \n');
printf('A15-A8 are address lines for chip select. \n
    \n');
printf('A15 A14 A13 A12 A11 A10 A9 A8 \n');
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.
printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf(' 0   0   0   0   0   0   0   0    =00H \n'); //this
    selects the register 00.
printf('The above combination selects the memory
    address 0000H. \n \n');
printf('A15 A14 A13 A12 A11 A10 A9 A8 \n');
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.
printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf(' 1  1  1  1  1  1  1  1    =FFH \n'); //this selects the register FF.
printf('The above combination selects the memory address 00FFH. \n \n');
//thus this chip can select any memory location from 0000H to 00FFH.
//the memory addressed of the chip can be changed by modifying the hardware.For example if we remove the inverter on line A15.
printf('A15 A14 A13 A12 A11 A10 A9 A8 \n');
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.
printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf(' 0  0  0  0  0  0  0  0    =00H \n'); //this selects the register 00.
printf('The above combination selects the memory address 8000H. \n \n');
//The memory address range from above change will be 8000H to 80FFH.
//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

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



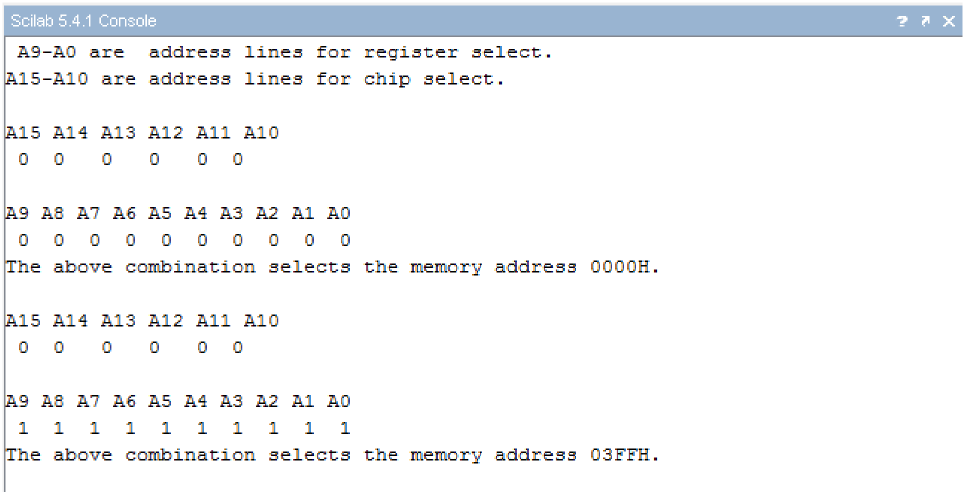
```
Scilab 5.4.1 Console
 A7-A0 are  address lines for register select.
A15-A8 are address lines for chip select.

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');
printf('A15-A10 are address lines for chip select. \n \n');
printf('A15 A14 A13 A12 A11 A10 \n');
printf(' 0  0   0   0   0  0 \n \n'); //chip select bits have to be active low always to select that chip.
printf('A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf(' 0  0  0  0  0  0  0  0  0  0  \n'); //this selects the register
printf('The above combination selects the memory address 0000H. \n \n');
printf('A15 A14 A13 A12 A11 A10 \n');
printf(' 0  0   0   0   0  0 \n \n'); //chip select bits have to be active low always to select that chip.
printf('A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf(' 1  1  1  1  1  1  1  1  1  1  \n'); //this selects the register
printf('The above combination selects the memory address 03FFH. \n \n');
//thus this chip can select any memory location from 0000H to 03FFH.
//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

```
//page no 43
//example no 2.3
//CALCULATING ADDRESS LINES
clc;
```



```
 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

```
 Number of address lines=
    13.
```

Figure 2.3: CALCULATING ADDRESS LINES

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

Scilab code Exa 2.4 CALCULATING NO OF CHIPS

```
//page no 43
//example no 2.4
//CALCULATING NO OF CHIPS.
```

Figure 2.4: CALCULATING NO OF CHIPS

```
clc;
//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:
no=(8192*8)/(1024*1);
printf('No of chips= ');
disp(no);
```

Scilab code Exa 2.5 FETCHING AN INSTRUCTION

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

Figure 2.5: FETCHING AN INSTRUCTION

Chapter 3

8085 Microprocessor Architecture And Memory Interfacing

Scilab code Exa 3.2 EXECUTING THE INSTRUCTION

```
//page no 78
//example no 3.2
//EXECUTING THE INSTRUCTION.
clc;
A=82; //contents of the accumulator.
printf('Accumulator= ');
disp(A);
TR=A; //contents of the accumulator tranferred to
   the temporary register.
printf('Temporary Register= ');
disp(TR);
C=TR; //contents of the temporary register are
   transferred to register C.
printf('Register C= ');
disp(C);
```

Figure 3.1: EXECUTING THE INSTRUCTION

Scilab code Exa 3.3 TIME REQUIRED FOR EXECUTION

```
//page no 82
//example no 3.3
//TIME REQUIRED FOR EXECUTION.
clc;
A=32; // MVI A,32H loads the value 32 in accumulator.
printf('Accumulator= ');
disp(A);
//calculating the execution time for instruction.
f=2; // clock frequncy.
printf('clock frequency= %f MHz \n',f);
t=1/f; // T-state=clock period
printf('T-state=clock period= %f microsec \n',t);
t1=4*t; // execution time for opcode fetch.
printf('Execution time for opcode fetch= %f microsec \n',t1);
t2=3*t; // execution time for memory read.
printf('Execution time for memory read= %f microsec \n',t2);
t3=7*t; // execution time for instruction.
printf('Execution time for instruction= %f microsec
```

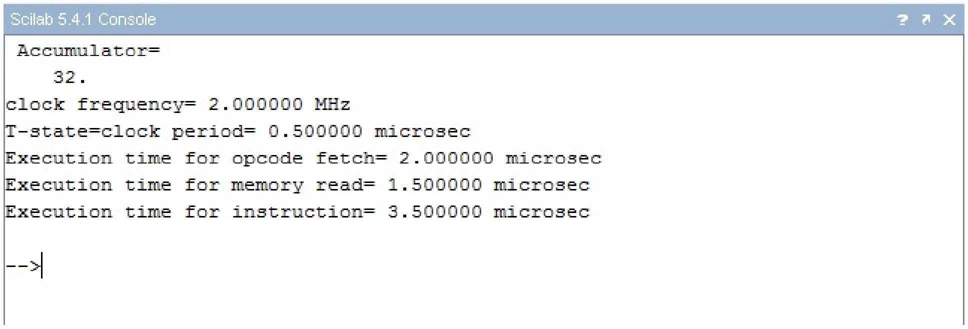


Figure 3.2: TIME REQUIRED FOR EXECUTION

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

Scilab code Exa 3.5 MEMORY ADDRESS RANGE OF 6116

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

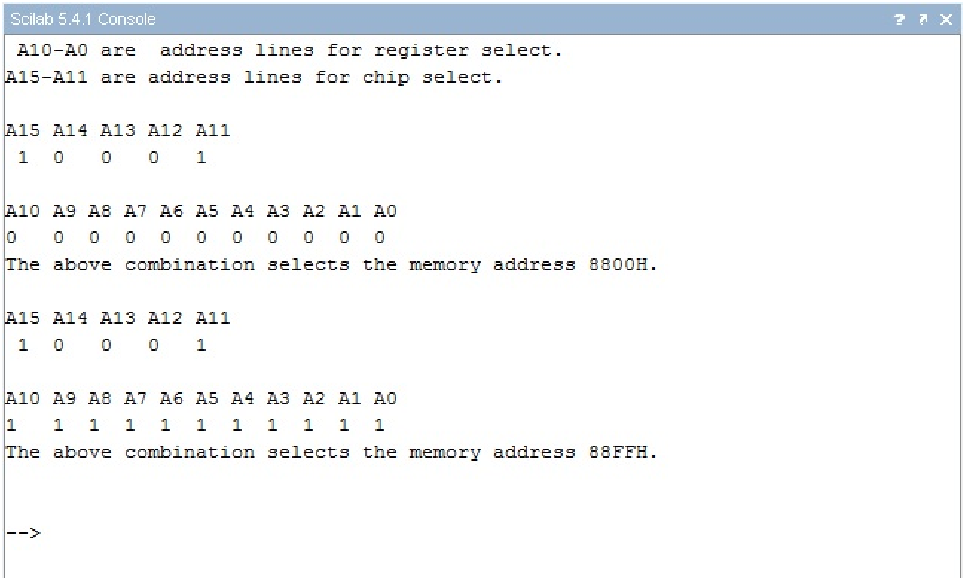


Figure 3.3: MEMORY ADDRESS RANGE OF 6116

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

Scilab code Exa 3.6 MEMORY ADDRESS RANGE OF 8155

```
////page no 95
//example no 3.6
```

```
//MEMORY ADDRESS RANGE OF 8155.
clc;
printf('A7-A0 are address lines for register select.\n');
printf('A10-A8 address lines are dont care conditions. \n');
printf('A15-A11 are address lines for chip select.\n \n');
printf('A15 A14 A13 A12 A11 \n');
printf(' 0  0   1   0   0 \n \n'); //chip select bits have to be active low always to select that chip.
printf('A10 A9 A8 \n');
printf('0  0  1  \n \n'); //this is the don't care condition.
printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf('0  0  0  0  0  0  0  0  \n'); //this selects the register
printf('The above combination selects the memory address 2100H. \n \n');
printf('A15 A14 A13 A12 A11 \n');
printf(' 0  0   1   0   0 \n \n'); //chip select bits have to be active low always to select that chip.
printf('A10 A9 A8 \n');
printf('0  0  1  \n \n'); //this is the don't care condition.
printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf('1  1  1  1  1  1  1  1  \n'); //this selects the register
printf('The above combination selects the memory address 21FFH. \n \n');
//thus this chip can select any memory location from 2100H to 21FFH.
```

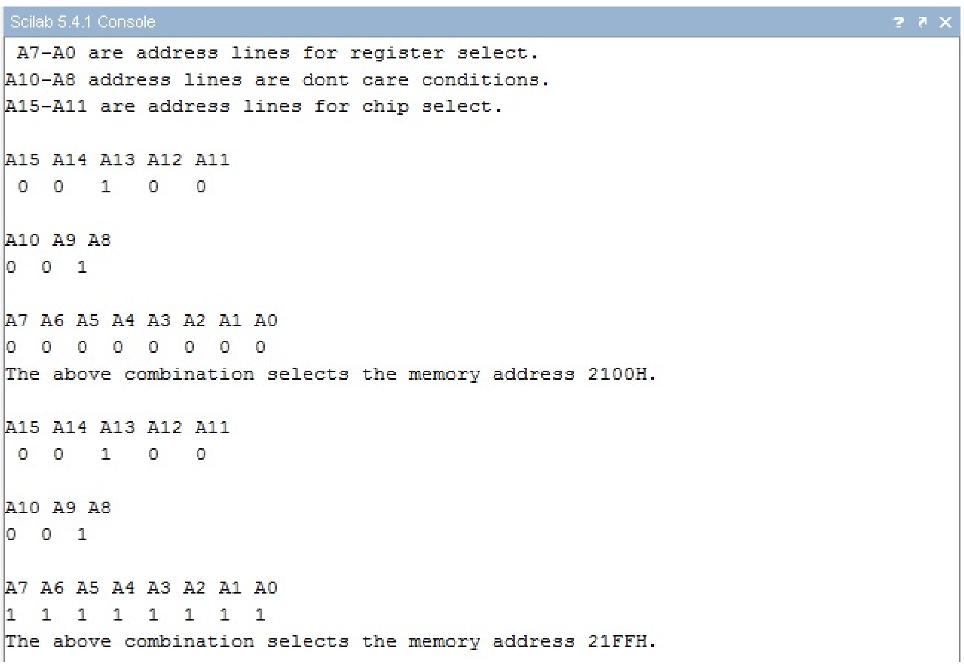


Figure 3.4: MEMORY ADDRESS RANGE OF 8155

Chapter 6

Introduction To 8085 Instructions

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

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

Scilab code Exa 6.2 TO SWITCH ON SOME DEVICES

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

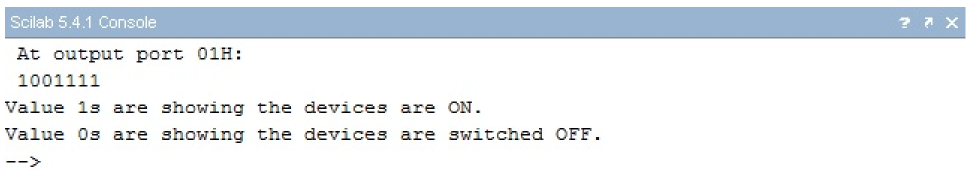


Figure 6.2: TO SWITCH ON SOME DEVICES

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

Scilab code Exa 6.3 ADDITION OF TWO NUMBERS

```
//page no 174
//example 6.3
//ADDITION OF TWO NUMBERS.
//93H is stored in accumulator. Converting it into decimal.
clc;
A=hex2dec(['93']);
//B7H is stored in register C. Converting it into decimal.
C=hex2dec(['B7']);
X=A+C; // the result comes out to be 330
Z=X-256;
//X=330; // this is a decimal value. Converting it into hexadecimal
Y=dec2hex(Z);
printf('Sum= ')
disp(Y);
if X>255 then
    printf('CY=1')
else
    printf('CY=0')
end
```

Scilab code Exa 6.4 CONTINUATION OF PREVIOUS EXAMPLE

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

Figure 6.3: ADDITION OF TWO NUMBERS

Figure 6.4: CONTINUATION OF PREVIOUS EXAMPLE

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

Scilab code Exa 6.5 INCRIMENTING ACCUMULATOR CONTENT

```
//page no 175
//example no 6.5
//INCRIMENTING ACCUMULATOR CONTENT.
//accumulator holds the data FFH
clc;
A=hex2dec(['FF']); //converting FFH into decimal
    value
//decimal value of 01H is 01. Adding 01 to A
Y=A+1; //the result comes out to be 256
Z=Y-256;
X=dec2hex(Z);
printf('Sum =')
disp(X);
if Y>255 then
    printf('CY=1 \n')
else
    printf('CY=0 \n')
end
if Z>127 then
    printf('S=1 \n')
else
    printf('S=0 \n')
end
if Z>0 then
    printf('Z=0 \n')
else
    printf('Z=1 \n')
end
```

Scilab code Exa 6.6 SUBTRACTION OF TWO NUMBERS

Figure 6.5: INCRIMENTING ACCUMULATOR CONTENT

```
//page no 179
//example 6.6
//SUBTRACTION OF TWO NUMBERS.
//accumulator has 97H. Converting it into decimal
    value
clc;
A=hex2dec(['97']);
//register B has 65H. Finding 2's compliment of 65H.
B=hex2dec(['65']);
X=256-B;
Y=A+X;
S=Y-256;
Z=dec2hex(S);
printf('Subtraction= ')
disp(Z);
if Y>255 then
    CY=1;
    printf('The result is positive. \n');
else
    CY=0;
    printf('The result is negative. \n')
end
if S>127 then
    printf('S=1 \n')
else
    printf('S=0 \n')
end
if S>0 then
```

Figure 6.6: SUBTRACTION OF TWO NUMBERS

```
    printf('Z=0 \n')
else
    printf('Z=1 \n')
end
```

Scilab code Exa 6.7 PERFORMING LOGICAL OPERATIONS

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

```
end
if Y==0 then
    printf('Z=1 \n');
else
    printf('Z=0 \n');
end
printf('CY=0 \n');
R=bitxor(A,B); //getting XOR of A & B
printf('XOR of A & B is')
disp(R);
if R(1,1)==1 then
    printf('S=1 \n');
else
    printf('S=0 \n');
end
if R==0 then
    printf('Z=1 \n');
else
    printf('Z=0 \n');
end
printf('CY=0 \n');
K=bitcmp(A,1); //getting the compliment of A
printf('Compliment of A is: \n');
disp(K);
```

Scilab code Exa 6.8 KEEPING THE RADIO ON

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

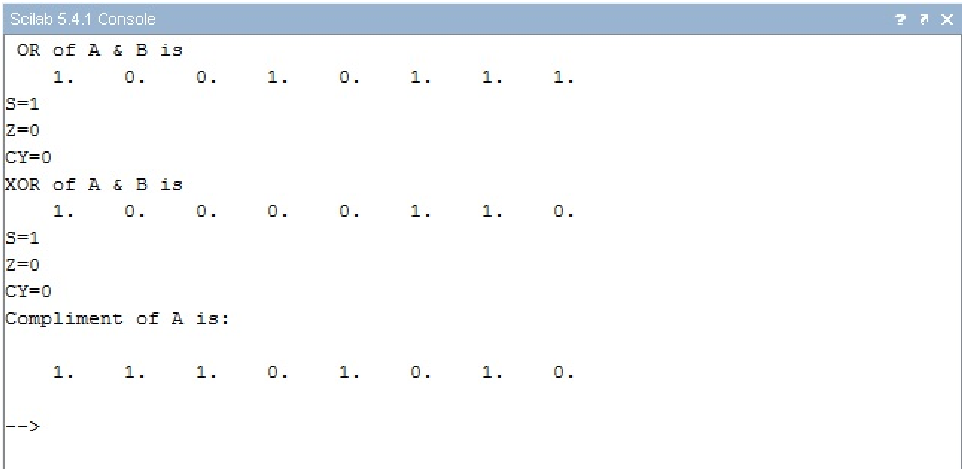


Figure 6.7: PERFORMING LOGICAL OPERATIONS

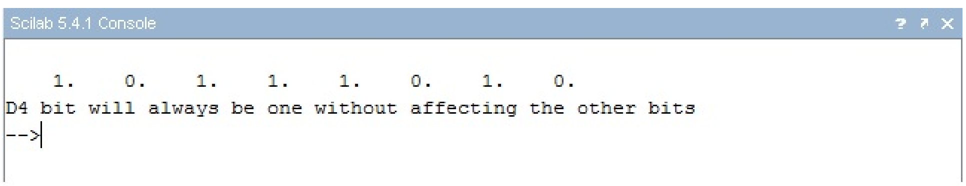


Figure 6.8: KEEPING THE RADIO ON

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

Scilab code Exa 6.9 TURN OFF THE AIR CONDITIONER

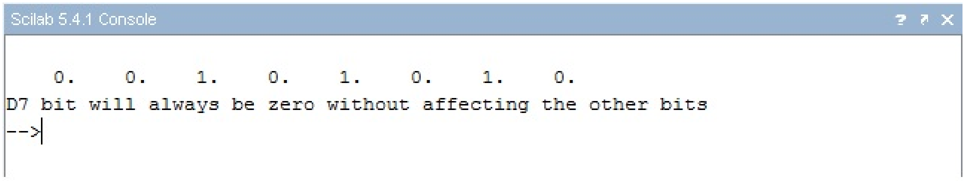


Figure 6.9: TURN OFF THE AIR CONDITIONER

```
//page no 187
//example no 6.9
//TURN OFF THE AIR CONDITIONER.
//to turn OFF the air conditioner, reset bit D7
//Assuming the same input as earlier as it is a
   continuation of previous example.
clc;
A=[1 0 1 0 1 0 1 0];
B=[0 1 1 1 1 1 1 1];
Y=bitand(A,B); //ANDing input (A) with B to keep the
    D4 bit always set
disp(Y);
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

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

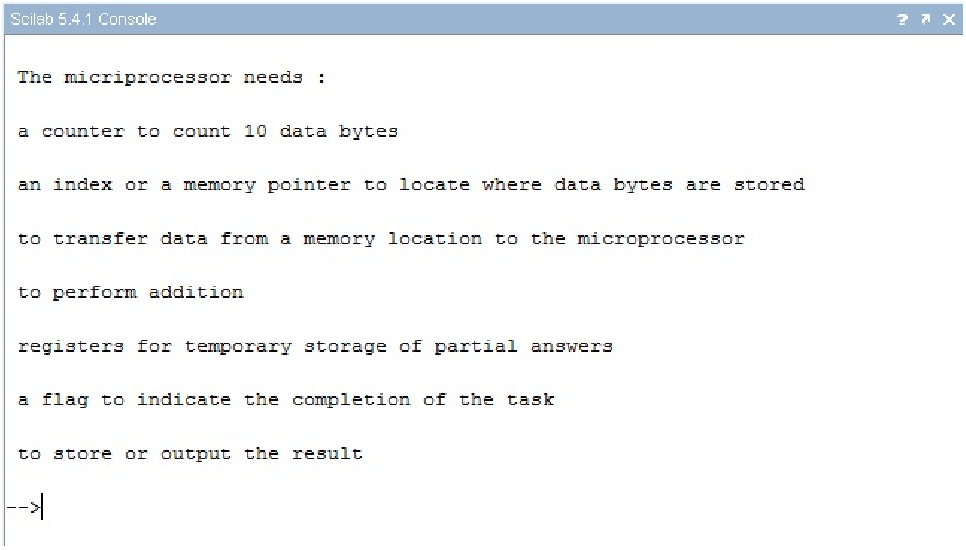


Figure 7.1: STEPS TO ADD 10 BYTES OF DATA

Scilab code Exa 7.2 LOADING 16 BIT NUMBER

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

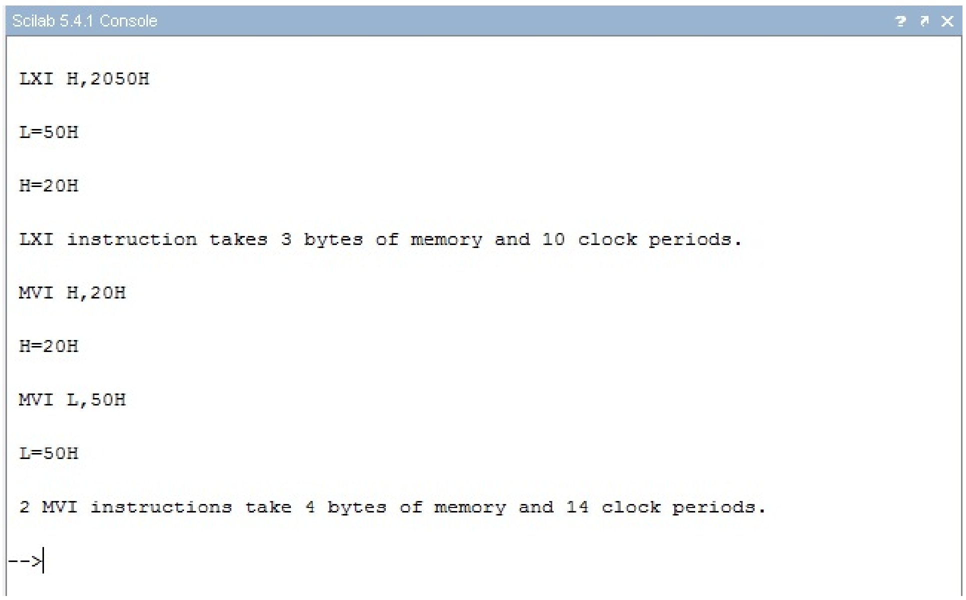


Figure 7.2: LOADING 16 BIT NUMBER

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

Scilab code Exa 7.3 TRANSFER OF DATA BYTES TO ACCUMULATOR

```
//page no 220
//example no 7.3
// TRANSFER OF DATA BYTES TO ACCUMULATOR.
// Memory location 2050H has the data F7H.
clc;

// using MOV instruction.
//indirect addressing mode.
disp('LXI H,2050H');
```

```
printf('H=20H    L=50H \n \n'); // the 16-bit address of the data is loaded in HL register pair.
M=hex2dec(['F7']); // M is the memory location pointer of address 2050H.
printf('MOV A,M \n');
A=dec2hex(M);
printf('A= ');
disp(A); // the contents of the HL register pair are used as memory pointer to the location 2050H.

// using LDAX instruction.
// indirect addressing mode.
disp('LXI B,2050H');
printf('B=20H    C=50H \n \n'); // the 16-bit address of the data is loaded in BC register pair.
M=hex2dec(['F7']); // M is the memory location pointer of address 2050H.
printf('LDAX B \n');
A=dec2hex(M);
printf('A= ');
disp(A); // the contents of the BC register pair are used as memory pointer to the location 2050H.

// using LDA instruction.
// direct addressing mode.
printf('\n LDA 2050H \n'); //directly sends the data of memory location 2050H to accumulator.
printf('A= ');
disp(A);
```

Scilab code Exa 7.4 USE OF ADDRESSING MODES

```
//page no 222
```

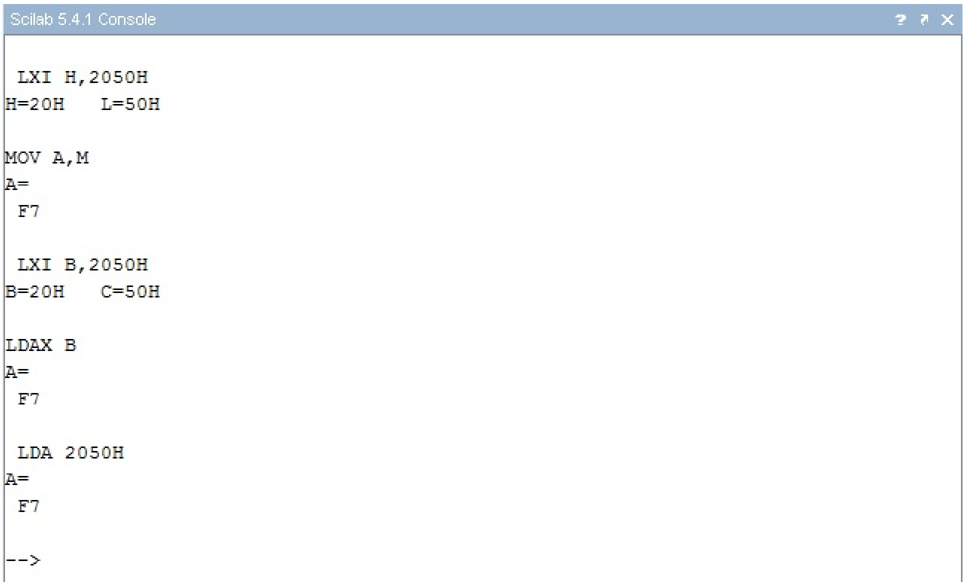


Figure 7.3: TRANSFER OF DATA BYTES TO ACCUMULATOR

```
//example no 7.4
// USE OF ADDRESSING MODES.
clc;
//register B contains 32H
B=32;

//using indirect addressing modes
printf('B= %d \n',B);
disp(' 1) LXI H,8000H'); // loads HL register pair.
disp('H=80H      L=00H');
disp('MOV M,B'); // contents of register B are moved in memory location pointed by HL register pair.
M=B;
printf('\n 8000H --> %d \n \n',M);

disp('LXI D,8000H'); //loads the memory location 8000H in DE register pair.
disp('D=80H      E=00H');
disp('MOV A,B');
```

```
A=B;
printf('A= %d \n',A);
disp('STAX D'); //stores the value of accumulator in the memory location pointer by DE register pair.
printf('\n 8000H --> %d \n \n',A);

//using direct addressing mode.
disp('2) A= F2H');
disp('STA 8000H'); //this instruction stores the value of accumulator in the memory location 8000H.
disp('8000H --> F2H');

//using indirect addressing mode.
disp('3) LXI H,8000H'); // loads HL register pair.
disp('H=80H      L=00H');
disp('MVI M,F2H'); //moving the data in the memory.
disp('8000H --> F2H');
```

Scilab code Exa 7.5 INCREMENT A NUMBER

```
//page no 224
//example no 7.5
// INCREMENT A NUMBER.
clc;
disp('LXI B,2050H'); //loads the data 2050H in BC register pair.
disp('B=20H    C=50H');
B=20;
C=50;
disp('INX B');
```

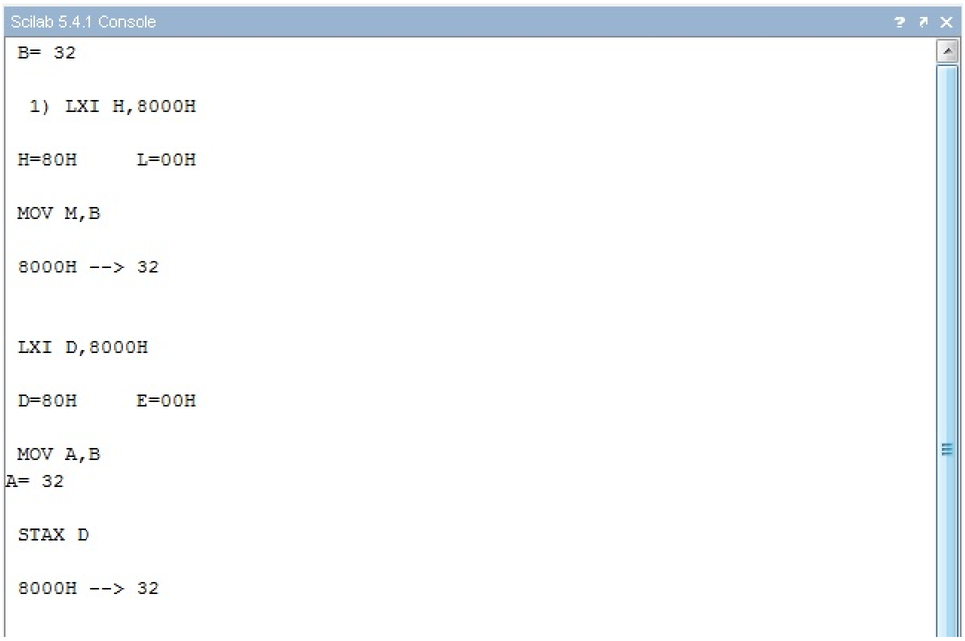


Figure 7.4: USE OF ADDRESSING MODES

Figure 7.5: USE OF ADDRESSING MODES

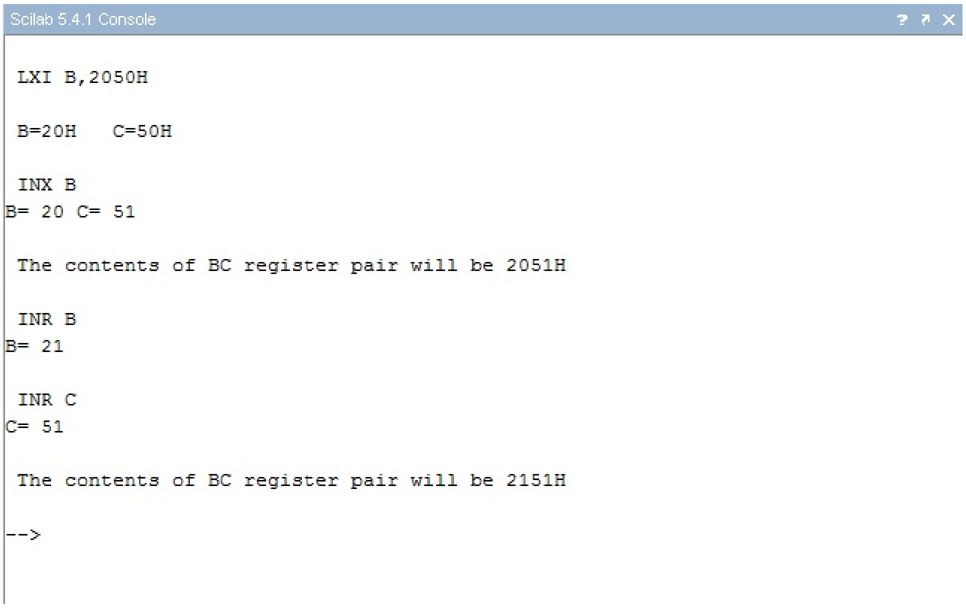


Figure 7.6: INCREMENT A NUMBER

```
C=C+1;
printf('B= %d C= %d \n',B,C);
disp('The contents of BC register pair will be 2051H');
disp('INR B');
B=B+1;
printf('B= %d \n',B);
disp('INR C');
C=50;
C=C+1;
printf('C= %d \n',C);
disp('The contents of BC register pair will be 2151H');
```

Scilab code Exa 7.6 ARITHEMETIC OPERATIONS

```
//page no 228
//example no 7.6
// ARITHEMETIC OPERATIONS.
clc;
disp('A-->30H');
disp('2040H-->68H');
disp('2041H-->7FH');
disp('LXI H,2040H'); // loads HL register pair.
disp('H=20H      L=40H   M=68H');
disp('ADD M');
A=hex2dec(['30']);
M=hex2dec(['68']);
S=A+M; // adds the contents of A and data at memory location 2040H.
s=dec2hex(S);
printf('\n Content of A after addition with 2040H= ');
disp(s);
disp('INX H'); // takes the program to the next memory location.
disp('H=20H      L=41H   M=7FH');
disp('SUB M');
M=hex2dec(['7F']);
D=S-M; // subtracts the contents of A from the data at memory location 2041H.
d=dec2hex(D);
printf('\n Content of A after subtraction with 2041H = ');
disp(d);
```

Scilab code Exa 7.7 INCREMENT AND DECREMENT

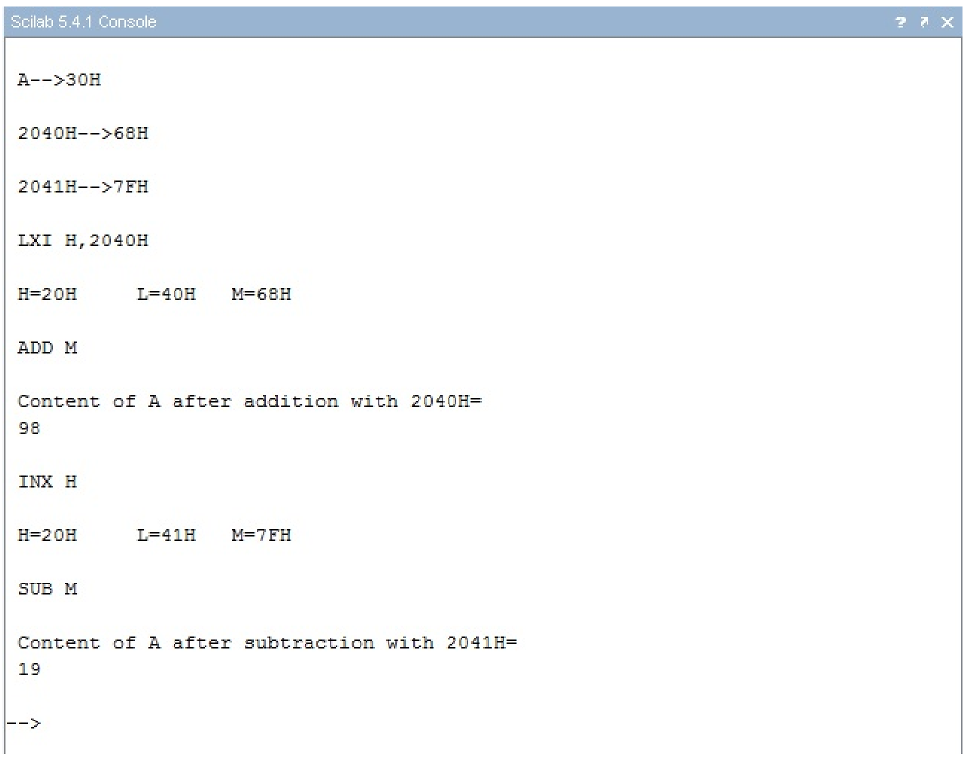


Figure 7.7: ARITHEMETIC OPERATIONS

```
//page no 229
//example no 7.7
// INCREMENT & DECREMENT.
clc;
disp('LXI H,2040H'); // loads HL register pair.
disp('H=20H       L=40H');
disp('MVI M,59H');
M=59;
M=hex2dec(['59']);
disp('2040H-->59H')
disp('INR M');
M=M+1; // increments the value at the memory location by 1.
m=dec2hex(M);
printf('\n Content of 2040H after increment= ');
disp(m);
disp('INX H'); //takes the program to the next memory location.
disp('H=20H       L=41H');
disp('MVI M,90H');
M=90;
M=hex2dec(['90']);
disp('2041H-->90H');
disp('DCR M');
M=M-1; //decrements the value at the memory location by 1.
m=dec2hex(M);
printf('\n Content of 2041H after decrement= ');
disp(m);
```

Scilab code Exa 7.8 LEFT ROTATION RLC OF BITS

```
//page no 233
```

```
Scilab 5.4.1 Console

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

```
// example no 7.8
// LEFT ROTATION (RLC) OF BITS.
clc;
// initially
printf('Accumulator= AAH \n');
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 1  0  1  0  1  0  1  0   =AAH \n \n');
printf('CY= 0 \n \n');
printf('RLC \n \n');
printf('CY= 1 \n \n');
// carry flag is set because D7 bit was 1.
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 0  1  0  1  0  1  0  1   =55H \n \n'); //
    after the executuion of first RLC.
// RLC instruction places D7 bit in CY flag as well
    as in D0 bit.
printf('RLC \n \n');
printf('CY= 0 \n \n');
// carry flag is reset because D7 bit was 0.
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 1  0  1  0  1  0  1  0   =AAH \n \n'); //
    after the executuion of second RLC.
// RLC instruction places D7 bit in CY flag as well
    as in D0 bit.
```

Scilab code Exa 7.9 LEFT ROTATION RAL OF BITS

```
//page no 234
// example no 7.9
// LEFT ROTATION (RAL) OF BITS.
clc;
// initially
printf('Accumulator= AAH \n');
```

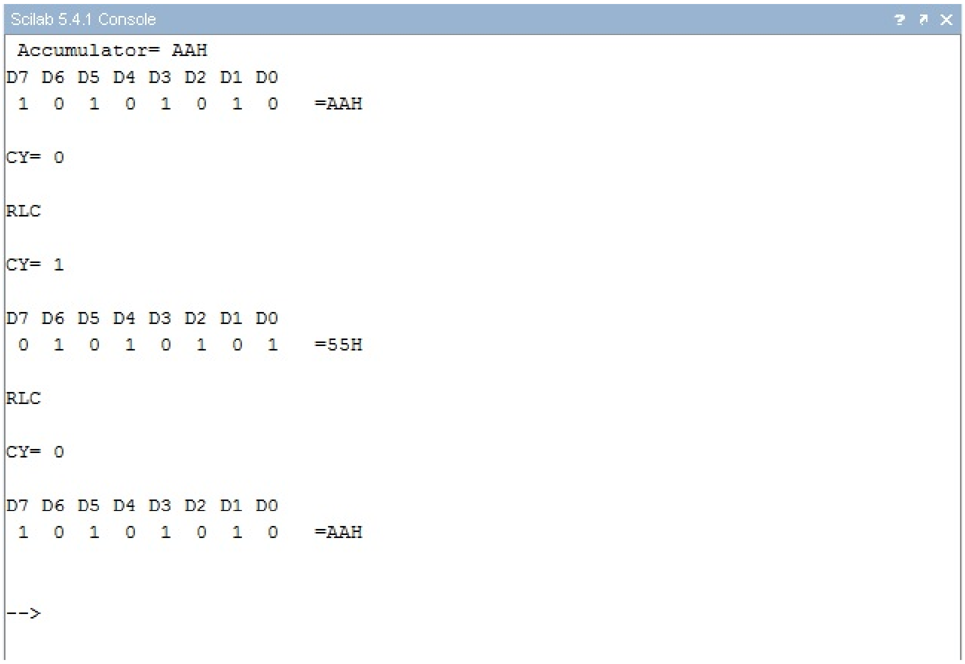
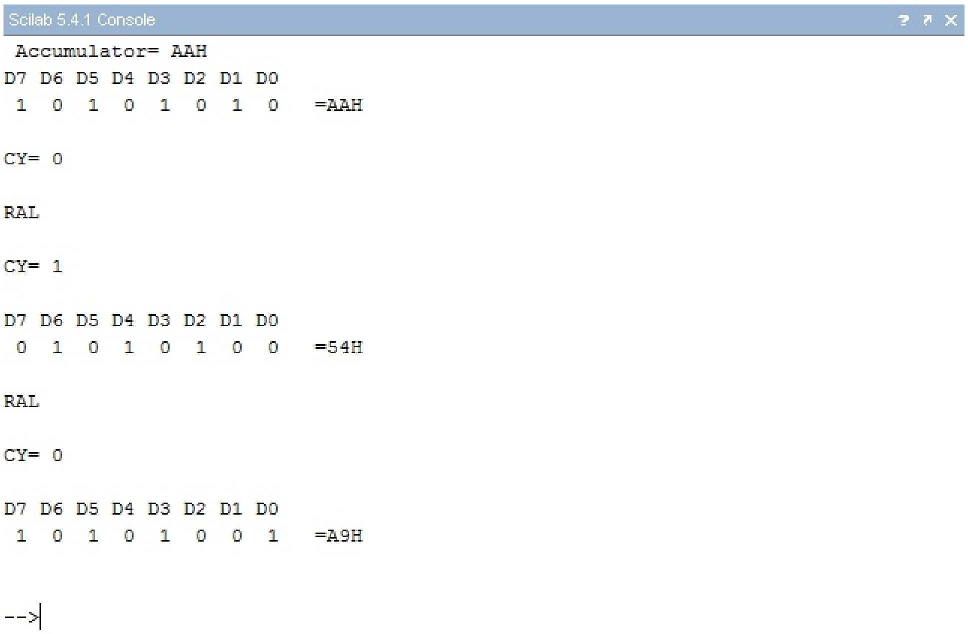


Figure 7.9: LEFT ROTATION RLC OF BITS

```
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 1  0  1  0  1  0  1  0   =AAH \n \n');
printf('CY= 0 \n \n');
printf('RAL \n \n');
printf('CY= 1 \n \n');
// carry flag is set because D7 bit was 1.
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 0  1  0  1  0  1  0  0   =54H \n \n'); // after the executuion of first RAL.
// RAL instruction places D7 bit in CY flag & CY flags bit is send to D0 bit.
printf('RAL \n \n');
printf('CY= 0 \n \n');
// carry flag is reset because D7 bit was 0.
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 1  0  1  0  1  0  0  1   =A9H \n \n'); // after the executuion 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

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

Scilab code Exa 7.10 RIGHT ROTATION RRC AND RAR OF BITS

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

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



// initially
printf('Accumulator= 81H \n');
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 1  0  0  0  0  0  0  1   =81H \n \n');
printf('CY= 0 \n \n');
printf('RAR \n \n');
printf('CY= 1 \n \n');
// carry flag is set because D0 bit was 1.
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf(' 0  1  0  0  0  0  0  0   =40H \n \n'); // after the executuion of RAR.
// RAR instruction places D0 bit in CY flag & CY flags bit is send to D7 bit.
```

Scilab code Exa 7.11 COMPARISION OF DATA

```
//page no 241
//example no 7.11
// COMPARISION OF DATA.
clc;
disp('MVI A,64H'); //loads accumulator with 64H.
disp('A-->64H');
```

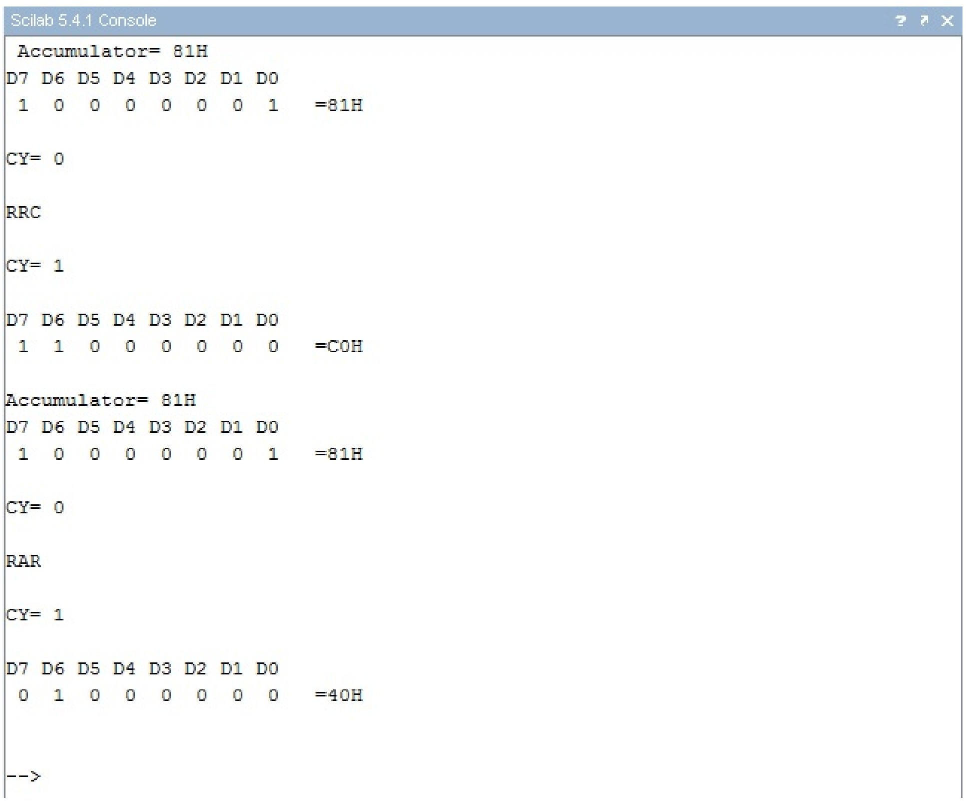


Figure 7.11: RIGHT ROTATION RRC AND RAR OF BITS

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

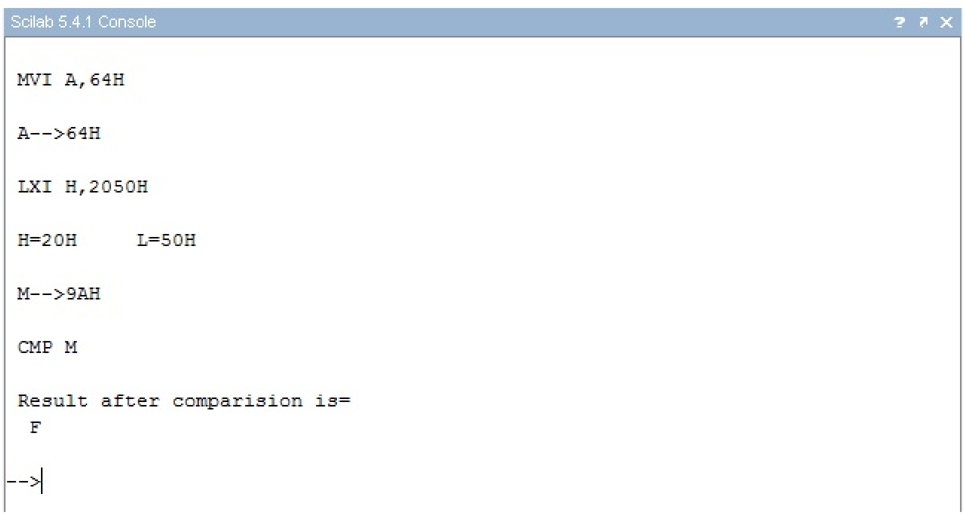


Figure 7.12: COMPARISION OF DATA

Chapter 9

Stack And Subroutines

Scilab code Exa 9.1 PUSH POP AND DELAY INSTRUCTIONS

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


```

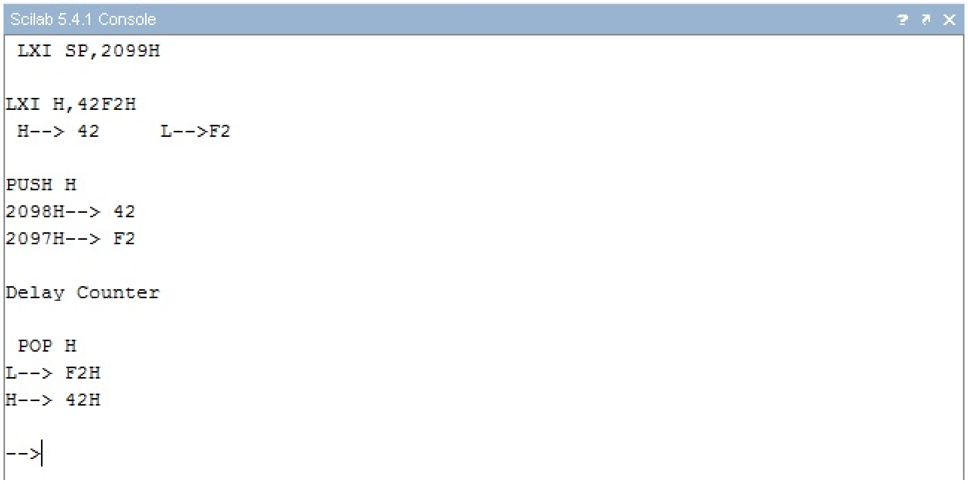


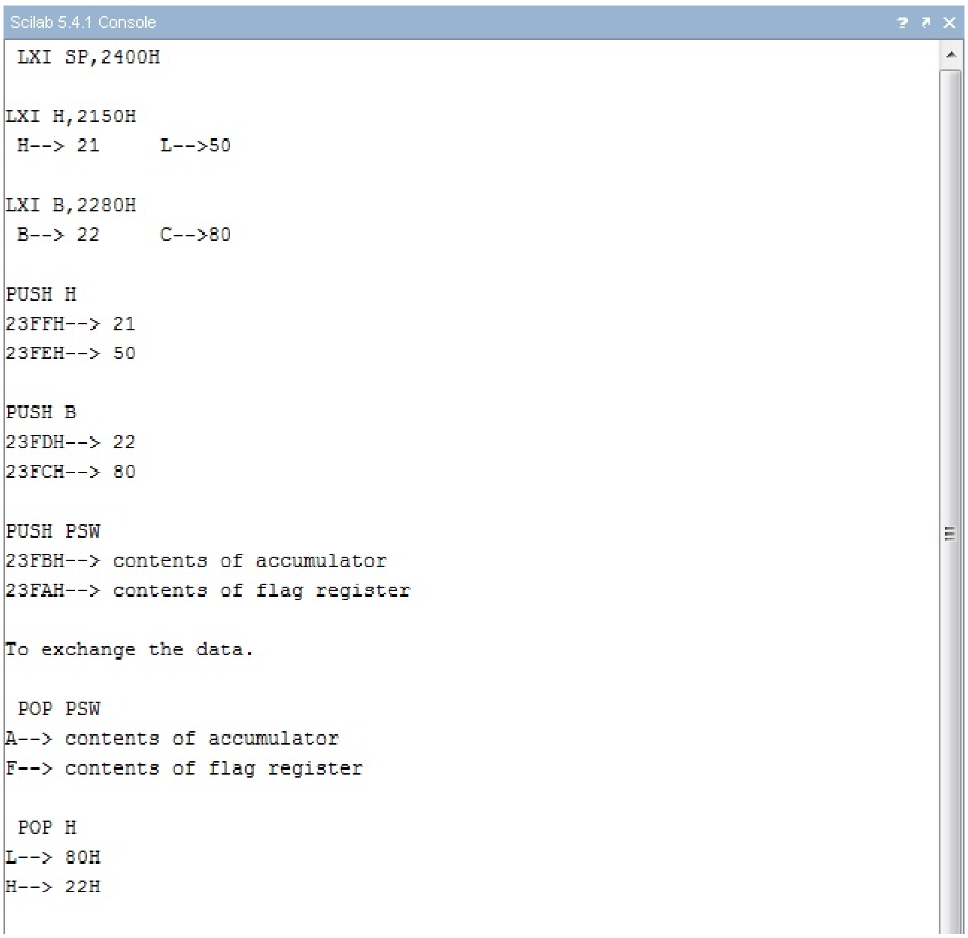
Figure 9.1: PUSH POP AND DELAY INSTRUCTIONS

```
n=hex2dec(['42F2']);
for i=1:n                                          // DELAY LOOP
    {
    }
end

printf(' POP H \n'); // sends the data in the stack back to the HL register pair.
// the contents of the top of the stack are copied to L register and the stack pointer is incremented by one to 2098H
printf('L—> F2H \n');
// the contents of the current location of stack are  copied to H register and the stack pointer is again incremented by one to 2099H.
printf('H—> 42H \n');
```

Scilab code Exa 9.2 EXCHANGE OF DATA USING STACK

```
// page no 285
// example no 9.2
// EXCHANGE OF DATA USING STACK.
clc;
printf('LXI SP,2400H \n \n'); // the stack pointer is located at 2400H.
printf('LXI H,2150H \n ');
printf('H--> 21      L-->50 \n \n');
printf('LXI B,2280H \n ');
printf('B--> 22      C-->80 \n \n');
printf('PUSH H \n'); // sends the data of HL register pair in the stack.
// stack pointer is decremented by one to 23FFH and the contents of the H register are copied to memory location 23FFH
printf('23FFH--> 21 \n');
// stack pointer is again decremented by one to 23FEH and the contents of the L register are copied to memory location 23FEH
printf('23FEH--> 50 \n \n');
printf('PUSH B \n'); // sends the data of BC register pair in the stack.
// stack pointer is decremented by one to 23FDH and the contents of the H register are copied to memory location 23FDH
printf('23FDH--> 22 \n');
// stack pointer is again decremented by one to 23FCH and the contents of the L register are copied to memory location 23FCH
printf('23FCH--> 80 \n \n');
printf('PUSH PSW \n'); // sends the data of accumulator & flag register in the stack.
// stack pointer is decremented by one to 23FBH and the contents of the H register are copied to memory location 23FBH
printf('23FBH--> contents of accumulator \n');
// stack pointer is again decremented by one to 23FAH and the contents of the L register are copied
```

```
to memory location 23FAH
printf('23FAH—> contents of flag register \n \n');

printf('To exchange the data. \n \n')
printf(' POP PSW \n'); // sends the data in the stack back to the accumulator & flag register.
// the contents of the top of the stack are copied to A register and the stack pointer is incremented by one to 23FBH
printf('A—> contents of accumulator \n');
// the contents of the current location of stack are copied to flag register and the stack pointer is again incremented by one to 23FCH.
printf('F—> contents of flag register \n \n');
printf(' POP H \n'); // sends the data in the stack back to the HL register pair.
// the contents of the current location of the stack are copied to L register and the stack pointer is incremented by one to 23FDH
printf('L—> 80H \n');
// the contents of the current location of stack are copied to H register and the stack pointer is again incremented by one to 23FEH.
printf('H—> 22H \n \n');
printf(' POP B \n'); // sends the data in the stack back to the BC register pair.
// the contents of the current location of the stack are copied to C register and the stack pointer is incremented by one to 23FFH
printf('C—> 50H \n');
// the contents of the current location of stack are copied to B register and the stack pointer is again incremented by one to 2400H.
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

```
// page no 292
// example no 9.3
// EXCHANGE INFORMATION BETWEEN STACK AND PROGRAM
   COUNTER
clc;
printf('After the CALL instruction \n \n');
printf('STACK MEMORY: \n \n');
printf('23FFH--> 20 \n');
printf('23FEH-->43 \n');
printf('Stack pointer--> 23FEH \n');
printf('Program counter--> 2070H \n \n');
printf('After RET instruction \n \n');
printf('Program counter--> 2043H \n');
printf('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

```
// page no 310
// example 10.1
// BCD TO BINARY
// BCD into its binary equivalent.
// given BCD no is 72
clc;
a=72;
    x=modulo(a,10);   // seperating the units digit
    printf('Unpacked BCD1 ')
    disp(dec2bin(x,8));
a=a/10;                    // seperating the tens place
   digit
a=floor(a);
printf('\n \n Unpacked BCD2');
disp(dec2bin(a,8));
printf('\n \n Multiply BCD2 by 10 and add BCD1');
```

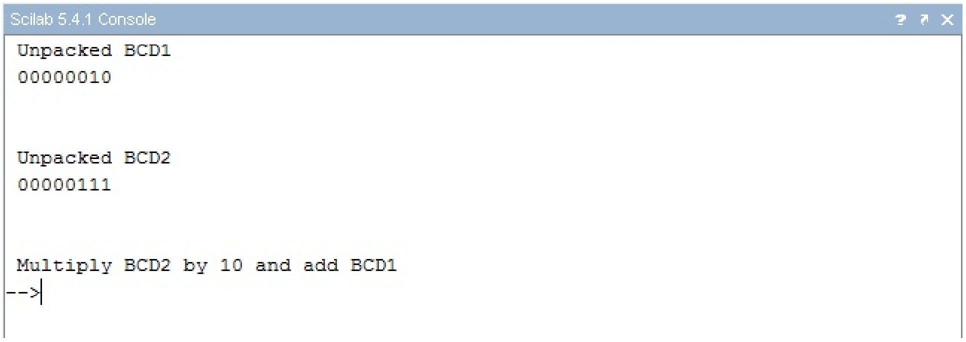


Figure 10.1: BCD TO BINARY

Scilab code Exa 10.2 ADDITION OF PACKED BCD NUMBERS

```
// page no 321
// example no 10.2
// ADDITION OF PACKED BCD NUMBERS
clc;
a=77;
b=48;
x=modulo(a,10);
y=modulo(b,10);
z=x+y;
if z>9 then
    f=z+6;

    printf('After addition BCD1 is: ')
    disp(dec2bin(f));
    printf('MSB of this sequence is the carry
       generated after addition. \n \n')
else
    printf('After addition BCD1 is: ')
```

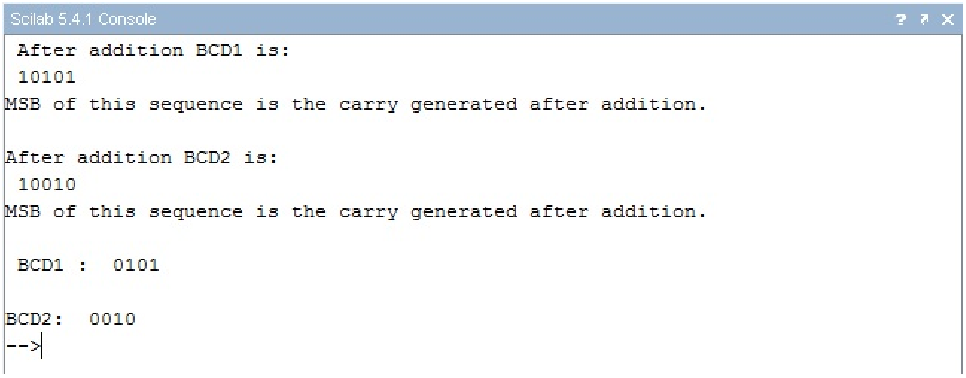


Figure 10.2: ADDITION OF PACKED BCD NUMBERS

```
    disp(z);
end
x=a/10;
x=floor(x);
y=b/10;
y=floor(y);
z=x+y;
if z>9 then
    f=z+6;
    f=f+1;  // this 1 is the carry of BCD1.
    printf('After addition BCD2 is: ')
    disp(dec2bin(f));
    printf('MSB of this sequence is the carry
       generated after addition.')
else
    printf('After addition BCD1 is: ')
    disp(z);
end
printf('\n \n BCD1 :  0101 \n \n');
printf('BCD2:  0010')
```

Scilab code Exa 10.3 EXCHANGE OF DATA

```
// page no 325
// example no 10.3
// EXCHANGE OF DATA
clc;
printf('2050H--> 3FH \n \n');
printf('2051H--> 42H \n \n');
printf('DE--> 856FH \n');
printf('D--> 85H        E--> 6FH \n \n');
printf('LHLD 2050H \n'); // loads the HL register pair with data on 2050H & 2051H.
printf('H--> 42H        L--> 3FH \n \n');
printf('XCHG \n'); // exchange the data of HL register pair with DE register pair.
printf('D<--->H         E<--->L \n');
printf('D--> 42H        E--> 3FH \n H--> 85H        L--> 6FH \n \n');
printf('SHLD 2050H \n'); // stores the 16bit dat in HL register pair on memory location 2051H & 2050H.
printf('2050H--> 6FH \n');
printf('2051H--> 85H');
```

Scilab code Exa 10.4 ADDITION OF TWO 16 BIT NUMBERS

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

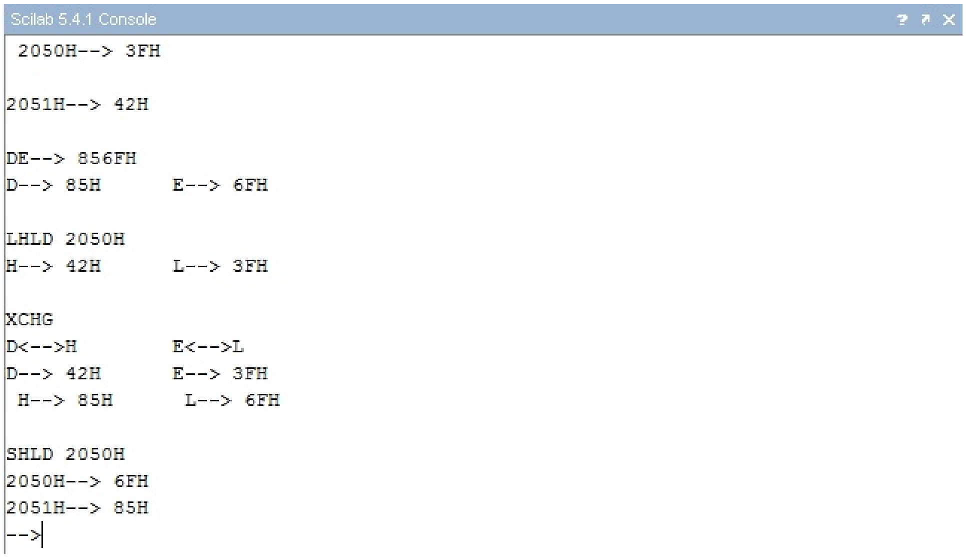


Figure 10.3: EXCHANGE OF DATA

```
c=hex2dec(['93']);
d=hex2dec(['31']);
e=hex2dec(['82']);
printf('MOV A,C \n \n');
a=c;
printf('ADD E \n');
a=a+e;
Z=a-256;
X=dec2hex(Z);
printf('Sum =')
disp(X);
if a>255 then
    printf('CY=1 \n \n');
    CY=1;
else
    printf('CY=0 \n');
    CY=0;
end
printf('MOV L,A \n');
printf('L-->');
```

```
disp(X);
printf('\n \n MOV A,B \n \n');
a=b;
printf('ADC D \n');
a=a+d+CY; // CY is added because of the previous
    carry as per the instructions ADC (add with carry
    )
T=dec2hex(a);
printf('Sum =')
disp(T);
if a>255 then
    printf('CY=1 \n \n')
else
    printf('CY=0 \n \n')
end
printf('MOV H,A \n');
printf('H—>');
disp(T);
printf('\n \n SHLD 2050H \n'); // stores the
    contents of HL register pair on memory locations
    2051H & 2050H.
printf('2050H—> ');
disp(X);
printf('2051H—> ');
disp(T);
```

Scilab code Exa 10.5 SUBTRACTION OF TWO 16 BIT NUMBERS

```
// page no 326
// examle no 10.5
// SUBTRACTION OF TWO 16 BIT NUMBERS
clc;
printf('B—> 85H        C—> 38H \n');
```

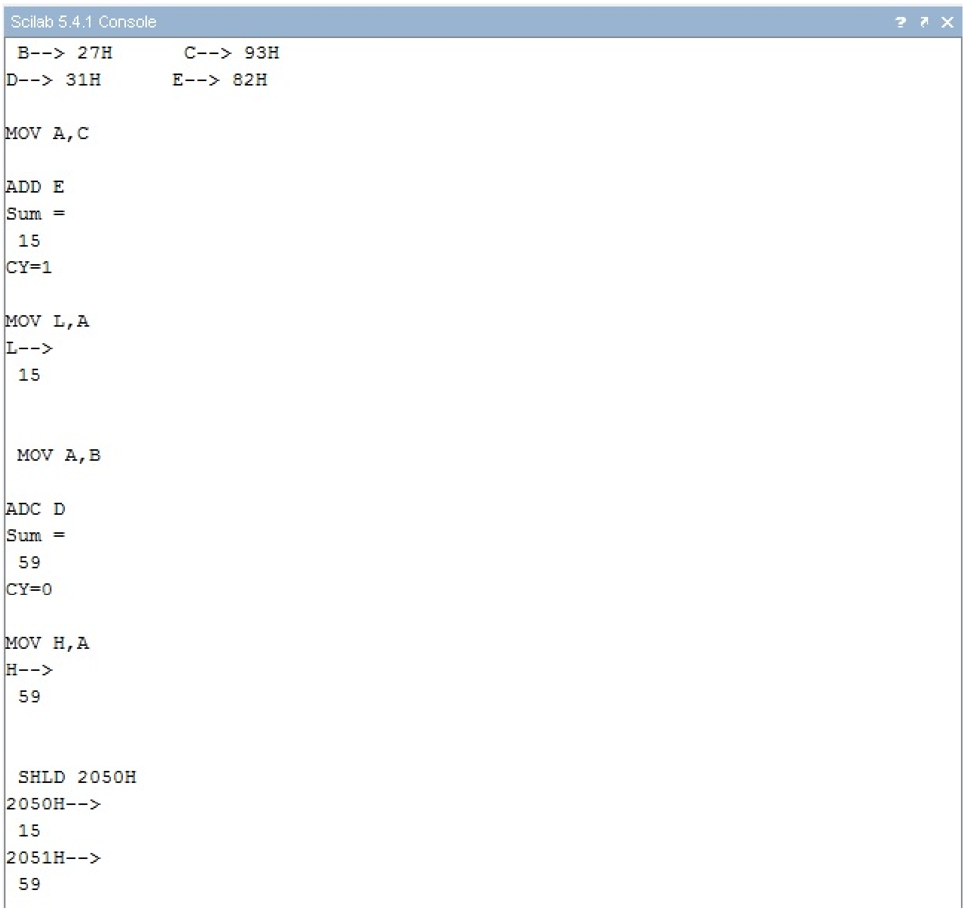


Figure 10.4: ADDITION OF TWO 16 BIT NUMBERS

```
printf('D—> 62H      E—> A5H \n \n');
b=hex2dec(['85']);
c=hex2dec(['38']);
d=hex2dec(['62']);
e=hex2dec(['A5']);
printf('MOV A,C \n \n');
a=c;
printf('SUB E \n');
a=a-e;
Z=a+256;
X=dec2hex(Z);
printf('Difference =')
disp(X);
if a<0 then
    printf('Borrow=1 \n \n');
    B=1;
else
    printf('Borrow=0 \n')
    B=0;
end
printf('MOV C,A \n');
printf('C—>');
disp(X);
printf('\n \n MOV A,B \n \n');
a=b;
printf('SBB D \n');
a=a-d-B; // 1 is subtracted because of the previous
   borrow as per the instructions SBB (subtract with
    borrow)
T=dec2hex(a);
printf('Difference =')
disp(T);
if a<0 then
    printf('Borrow=1 \n \n')
else
    printf('Borrow=0 \n \n')
end
printf('MOV B,A \n');
```

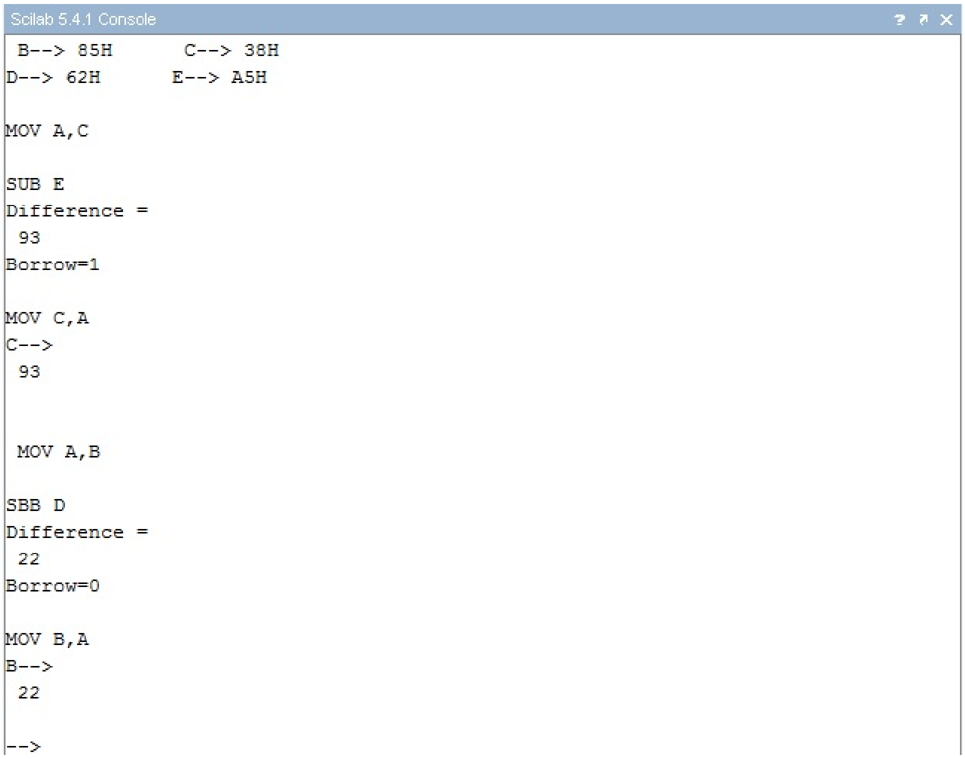


Figure 10.5: SUBTRACTION OF TWO 16 BIT NUMBERS

```
printf('B—>');
disp(T);
```

Scilab code Exa 10.6 DISPLAY CONTENTS OF STACK

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

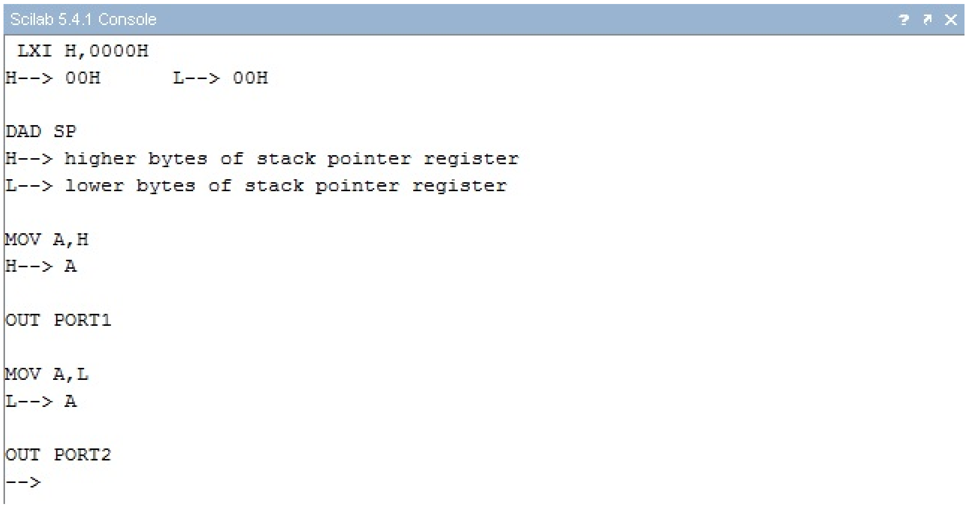


Figure 10.6: DISPLAY CONTENTS OF STACK

```
printf('H--> 00H        L--> 00H \n \n');
printf('DAD SP \n'); // place the stack pointer content in HL
printf('H--> higher bytes of stack pointer register \n');
printf('L--> lower bytes of stack pointer register \n \n');
printf('MOV A,H \n'); // copies the contents of H in A.
printf('H--> A \n \n');
printf('OUT PORT1 \n \n');
printf('MOV A,L \n'); // copies the contents of L in A.
printf('L--> A \n \n');
printf('OUT PORT2');
```

Scilab code Exa 10.7 SUBROUTINE TO SET THE ZERO FLAG

```
// page no 327
// example no 10.7
// SUBROUTINE TO SET THE ZERO FLAG
clc;
printf('CHECK:    PUSH H \n \n'); // sends the contents of H to the location pointed by the stack pointer.
printf('          MVI L,FFH \n');
l=hex2dec(['FF']);
l=dec2bin(l,8);
printf('          L--> '); // set all bits in L to logic 1.
disp(l);
printf('\n \n          PUSH PSW \n \n'); // save flags on top of the stack
printf('          XTHL \n \n'); // set all bits in the top stack location.
printf('          POP PSW \n \n'); // now the zero flag is set.
printf('          JZ NOEROR \n \n');
printf('          JMP ERROR \n \n');
printf('NOEROR:   POP H \n \n'); // retrives the data from the stack into H if zero flag is set
printf('          RET');
```

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

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

Figure 10.7: SUBROUTINE TO SET THE ZERO FLAG

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

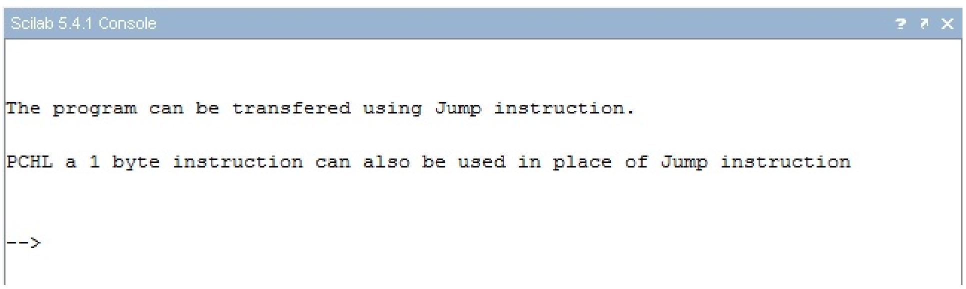


Figure 10.8: TRANSFER A PROGRAM TO AN ADDRESS IN HL REGISTER

Chapter 12

Interrupts

Scilab code Exa 12.1 ENABLE INTERRUPTS

```
// page no 374
// example no 12.1
// ENABLE INTERRUPTS
clc;
printf('EI \n \n');    // enable interrupts
printf('MVI A,08H \n');
a=hex2dec(['8']);
b=dec2bin(a,8);
printf('A--> ')
disp(b);
printf('\n SIM \n \n');  // enable RST 7.5,6.5, and
    5.5
printf('D3=1      SIM functional \n');
printf('D2=0      Enable RST 7.5 \n');
printf('D1=0      Enable RST 6.5 \n');
printf('D0=0      Enable RST 5.5 \n');
```

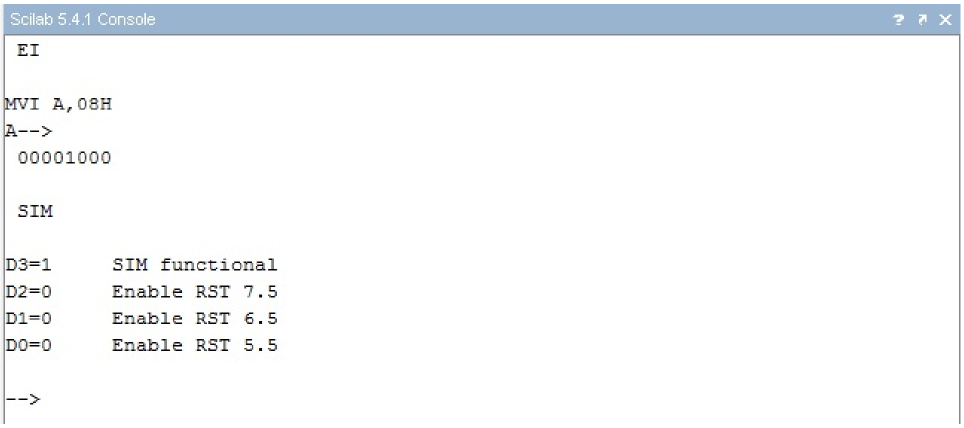


Figure 12.1: ENABLE INTERRUPTS

Scilab code Exa 12.2 RESET INTERRUPT

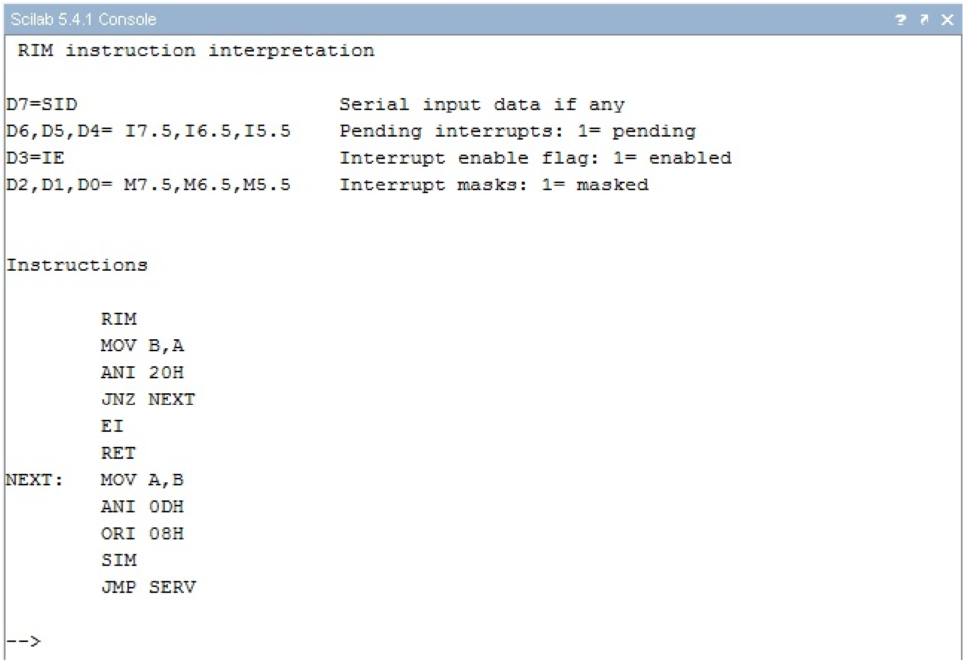
```
// page no 374
// example no 12.2
// RESET 7.5 INTERRUPT
clc;
printf('MVI A,18H \n');      // set D4=1
a=hex2dec(['18']);
b=dec2bin(a,8);
printf('A--> ')
disp(b);
printf('\n \n SIM');         // Reset 7.5 interrupt flip flop
```

Scilab code Exa 12.3 CHECK PENDING INTERRUPT

```
// page no 375
// example no 12.3
// CHECK PENDING INTERRUPT
```

Figure 12.2: RESET INTERRUPT

```
clc;
printf('RIM instruction interpretation \n \n');
printf('D7=SID                              Serial input data if any \n');
printf('D6,D5,D4= I7.5,I6.5,I5.5      Pending interrupts: 1= pending \n');
printf('D3=IE                               Interrupt enable flag: 1= enabled \n');
printf('D2,D1,D0= M7.5,M6.5,M5.5      Interrupt masks: 1= masked \n \n \n');


printf('Instructions \n \n');
printf('          RIM \n');              // Read interrupt mask
printf('          MOV B,A \n');          // save mask information
printf('          ANI 20H \n');          // check whether RST 6.5 is pending
printf('          JNZ NEXT \n');
printf('          EI \n');
printf('          RET \n');              // RST 6.5 is not pending, return to main program
printf('NEXT:     MOV A,B \n');          // get bit pattern; RST 6.5 is pending
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

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

Chapter 14

Programmable Interface Devices

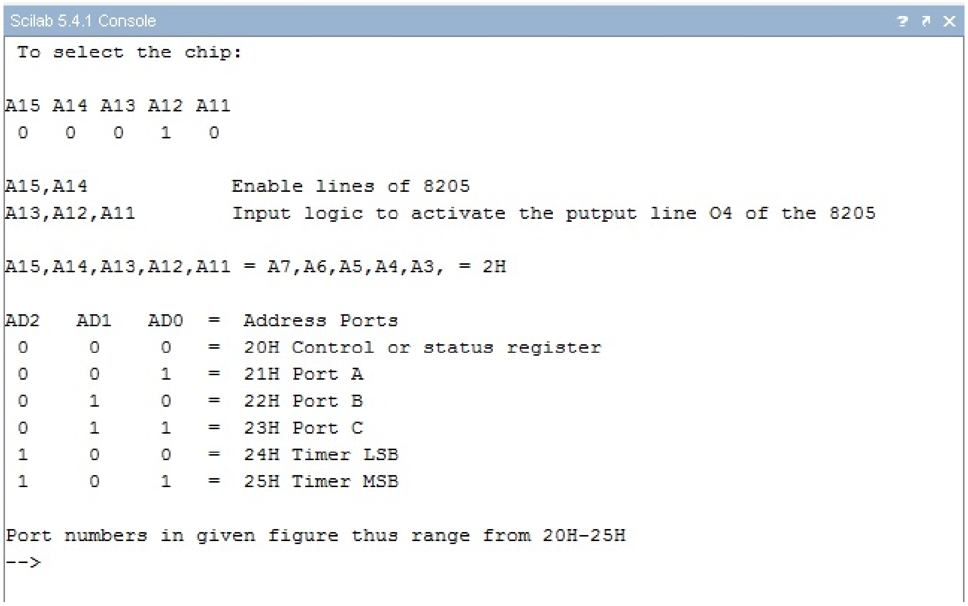
Scilab code Exa 14.1 INITIALIZE HYPOTHETICAL CHIP AS OUTPUT BUFFER

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

Figure 14.1: INITIALIZE HYPOTHETICAL CHIP AS OUTPUT BUFFER

Scilab code Exa 14.2 ADDRESS DETERMINATION OF GIVEN FIGURE

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

```
printf(' 1      0      0    =   24H Timer LSB \n');
printf(' 1      0      1    =   25H Timer MSB \n \n');
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

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


```

```
printf('2 Control Word \n \n');
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf('1  0  0  0  0  0  1  1        = 83H \n \n');
printf('D7=1     I/O Function \n');
printf('D6,D5=0  Port A in Mode 0 \n');
printf('D4=0     Port A= output \n');
printf('D3=0     Port C upper= output \n');
printf('D2=0     Port B in Mode 0 \n');
printf('D1=1     Port B= input \n');
printf('D0=1     Port C1= input \n \n');


printf('3 Program \n \n');
printf('MVI A,83H \n');   // load accumulator with the control word.
printf('STA 8003H \n');   // write word in the control register to initialize the ports.
printf('LDA 8001H \n');   // reads switches at port B.
printf('STA 8000H \n');   // display the reading at port A.
printf('LDA 8002H \n');   // read switches at port C.
printf('ANI 0FH \n');     // mask the upper four bits of port C, these bits are not input data.
printf('RLC \n');         // rotate and place the data in the upper half of the accumulator.
printf('RLC \n');
printf('RLC \n');
printf('RLC \n');
printf('STA 8002H \n');   // display data at port C upper.
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

```
// page no 453
// example no 15.2
// BSR CONTROL WORD SUBROUTINE
clc;
printf('BSR Control Words \n \n');
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf('0  0  0  0  1  1  1  1        = 0FH      To
    set bit PC7 \n');
printf('0  0  0  0  1  1  1  0        = 0EH      To
    reset bit PC7 \n');
printf('0  0  0  0  0  1  1  1        = 07H      To
    set bit PC3 \n');
printf('0  0  0  0  0  1  1  0        = 06H      To
    reset bit PC3 \n \n');


printf('Port Address \n \n');
printf('Control Register Address = 83H \n \n');


printf('Subroutine \n \n');
printf('MVI A,0FH \n');          // load byte in
   accumulator to set PC7
printf('OUT 83H \n');            // set PC7=1
printf('MVI A,07H \n');          // load byte in
   accumulator to set PC3.
printf('OUT 83H \n');            // set PC3=1.
printf('CALL DELAY \n');         // this is a 10
   microsec delay.
printf('MVI A,06H \n');          // load byte in
   accumulator to reset PC3
printf('OUT 83H \n');            // reset PC3
printf('MVI A,0EH \n');          // load byte in
   accumulator to reset PC7.
printf('OUT 83H \n');            // reset PC7
printf('RET');
```

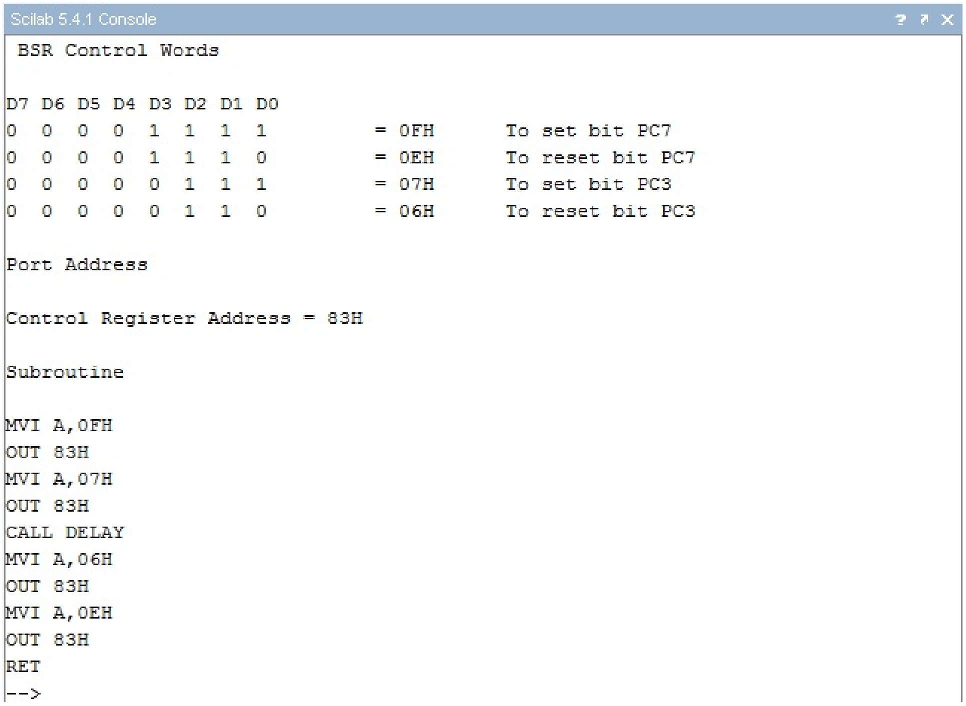


Figure 15.2: BSR CONTROL WORD SUBROUTINE

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

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

```
printf('D7,D6=0          Select counter 0 \n');
printf('D5,D4=01         Load 8 bit count \n');
printf('D3,D2,D1=010     Mode 2 \n');
printf('D0=0             Binary Count \n \n');


printf('Count \n \n');
count=(50*10^-6)/(0.5*10^-6);
printf('Count= ');
disp(count);
disp(dec2hex(count));
printf('in hexadecimal \n \n');


printf('Instructions \n \n');
printf('PULSE: \n')
printf('MVI A,00010100B \n');     // control word mode 2 & counter 0.
printf('OUT 83H \n');             // write in 8254 control register.
printf('MVI A,64H \n');           // low order byte of the count.
printf('OUT 80H \n');             // load counter 0 with low order byte
printf('HLT');
```

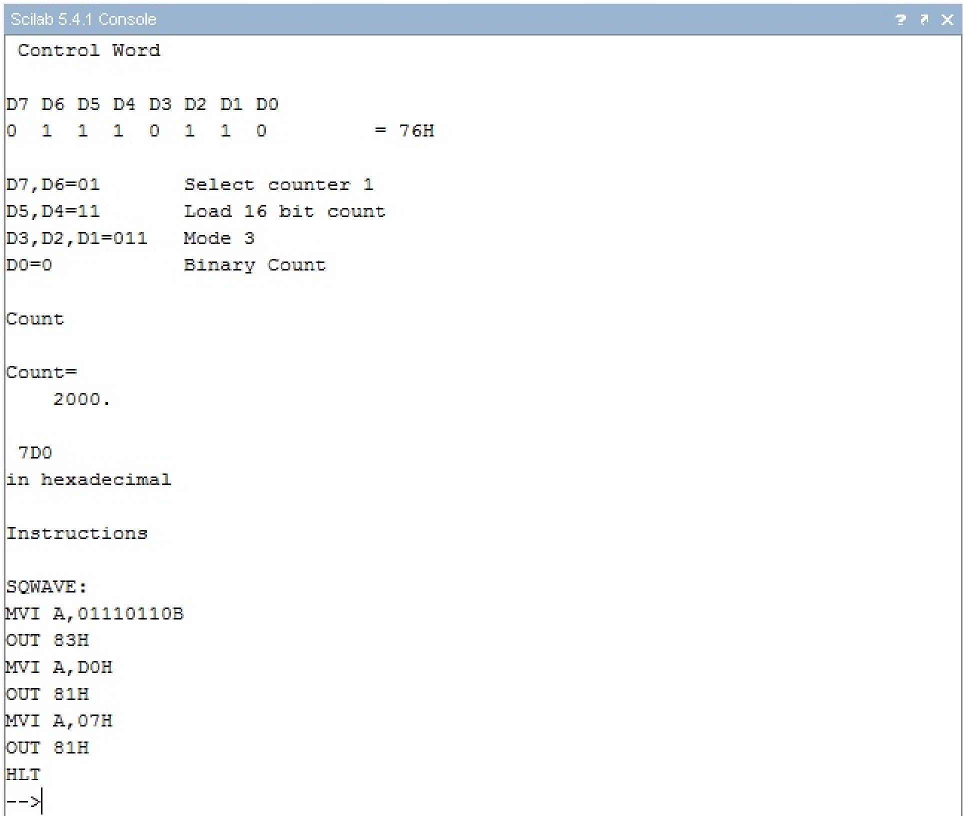
Scilab code Exa 15.4 INSTRUCTIONS TO GENERATE SQUARE WAVE PULSE FROM COUNTER 1

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

```
Scilab 5.4.1 Console
 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

```
clc;
printf('Control Word \n \n');
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf('0  1  1  1  0  1  1  0        = 76H \n \n');
printf('D7,D6=01       Select counter 1 \n');
printf('D5,D4=11       Load 16 bit count \n');
printf('D3,D2,D1=011   Mode 3 \n');
printf('D0=0           Binary Count \n \n');


printf('Count \n \n');
count=(1*10^-3)/(0.5*10^-6);
printf('Count= ');
disp(count);
b=dec2hex(2000);
disp(b);
printf('in hexadecimal \n \n');


printf('Instructions \n \n');
printf('SQWAVE: \n');
printf('MVI A,01110110B \n');    // control word mode 3 & counter 1.
printf('OUT 83H \n');            // write in 8254 control register.
printf('MVI A,D0H \n');          // low order byte of the count.
printf('OUT 81H \n');            // load counter 1 with low order byte.
printf('MVI A,07H \n');          // high order byte of the count.
printf('OUT 81H \n');            // load counter 1 with high order byte
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

```
// page no 486
// example no 15.5
// SUBROUTINE TO GENERATE AN INTERRUPT
clc;
printf('Control Word \n \n');
```

```
printf('D7 D6 D5 D4 D3 D2 D1 D0 \n');
printf('0  1  1  1  0  1  0  0       = 74H  Counter 1 \n');
printf('1  0  0  1  0  1  0  0       = 94H  Counter 2 \n \n');
printf('D7,D6          Select counter 1 \n');
printf('D5,D4          Load count \n');
printf('D3,D2,D1=010   Mode 2 \n');
printf('D0=0           Binary Count \n \n');



printf('Instructions \n \n');
printf('CNT1LO  EQU 50H \n');
printf('CNT1HI  EQU C3H \n');
printf('COUNT2  EQU 40H \n');
printf('SECOND  MVI A,01110100B \n');      // control word mode 2 & counter 1.
printf('        OUT 83H \n');              // write in 8254 control register.
printf('        MVI A,10010100B \n');      // control word mode 2 & counter 2.
printf('        OUT 83H \n');              // write in 8254 control register.
printf('        MVI A,CNT1LO \n');         // Low order byte of count 50000
printf('        OUT 81H \n');              // Load counter 1 with low order byte
printf('        MVI A,CNT1HI \n');         // high order byte of count 50000.
printf('        OUT 81H \n');              // load counter 1 with high order byte
printf('        MVI A,COUNT2 \n');         // Count for Counter 2.
printf('        OUT 82H \n');              // load counter 2.
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

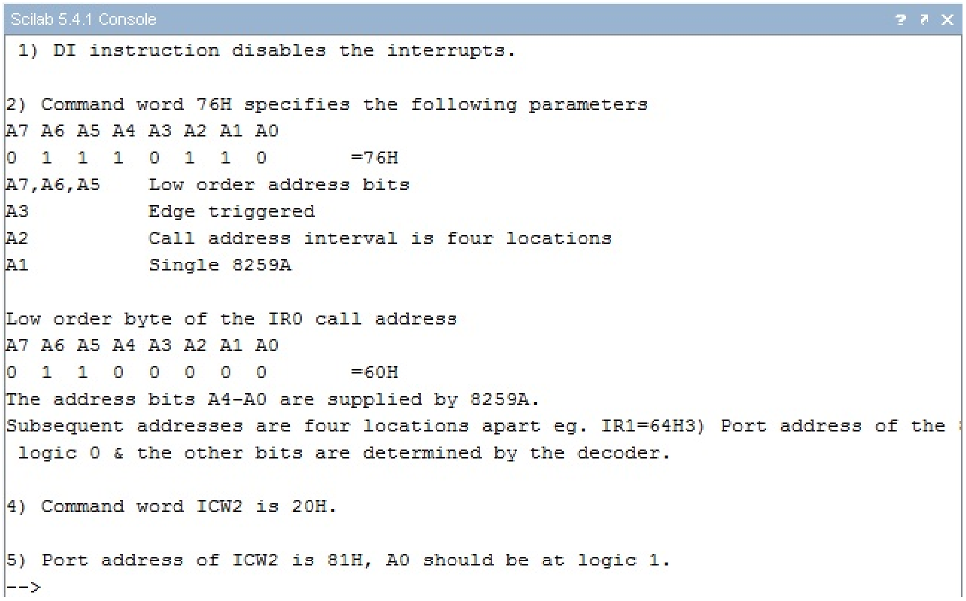
Scilab code Exa 15.6 EXPLANATION OF INSTRUCTIONS

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

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

Scilab code Exa 15.8 INITIALIZATION INSTRUCTIONS FOR DMA

```
// page no 502
// example no 15.8
// INITIALIZATION INSTRUCTIONS FOR DMA
clc;
printf('MVI A,00000100B \n \n');
printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf('0 0 0 0 0 1 0 0 \n');
printf('A2=1        Disable DMA \n \n');
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 IR0 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

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

```
    of the count 03FFH
printf('OUT 07H \n');              // Output to CH3 count reg.
printf('MVI A,03H \n');            // High order byte of the count 03FFH
printf('OUT 07H \n');              // Output to CH3 count reg.
printf('MVI A,10000000B \n \n');
printf('A7 A6 A5 A4 A3 A2 A1 A0 \n');
printf('1  0  0  0  0  0  0  0 \n');
printf('A7,A6=10    DACK DREQ High  \n');
printf('A5=0         Late write \n');
printf('A4=0         Fixed priority \n');
printf('A3=0         Normal time \n');
printf('A2=0        DMA enable \n')
printf('A0=0         Disable mem to mem \n \n');
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

```
// page no 622
// example no A.1
// BINARY INTO HEX AND OCTAL
clc;
printf('Binary no= 10011010 \n \n');
str='10011010';
h=bin2dec(str);
H=dec2hex(h);
printf('Hex Equivalent= ');
disp(H);
O=dec2oct(h);
printf('\n Octal Equivalent= ')
disp(O);
```

Scilab code Exa 19.2 SUBTRACTION OF TWO NUMBERS

```
// page no 623
```

Figure 19.1: BINARY INTO HEX AND OCTAL

```
// example no A.2
// SUBTRACTION OF TWO NUMBERS
clc;
printf('Minuend: 52 \n');
printf('Subtrahend: 23 \n \n')
printf('BORROW METHOD \n \n');

m=5*10+2; // minuend
s=2*10+3; // subtrahend
// to subtract 3 from 2, 10 must be borrowed from the second place of the minuend.

m=4*10+12;

sub=m-s;
printf('Subtraction= ')
disp(sub);

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

// 9's complement of 23 is

n=99-23;
// add 1 to the 9's complement to find the 10's complement
t=n+1;
```

Figure 19.2: SUBTRACTION OF TWO NUMBERS

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

Scilab code Exa 19.3 SUBTRACTION OF TWO NUMBERS

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

```
printf('BORROW METHOD \n \n');

m=2*10+3; // minuend
s=5*10+2; // subtrahend
// subtraction of the digits in the first place results in
a=3-2;
// to subtract the digits in the second place a borrow is required from the third place. assuming 1 at third place.

x=12-5; // with a borrowed 1 from the third place

sub=10*x+a;
printf('Subtraction= ')
disp(sub);
printf('this is negative 29, expressed in 10s complement. \n negative sign is verified by the borrowed 1 from the third place.');

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

// 9's complement of 52 is

n=99-52;
// add 1 to the 9's complement to find the 10's complement
t=n+1;
// add the 10's complement of the subtrahend(23) to minuend(52) to subtract 23 from 52
a=m+t;

printf('Subtraction= ');
disp(a);
printf('this is negative 29, expressed in 10s complement');
```

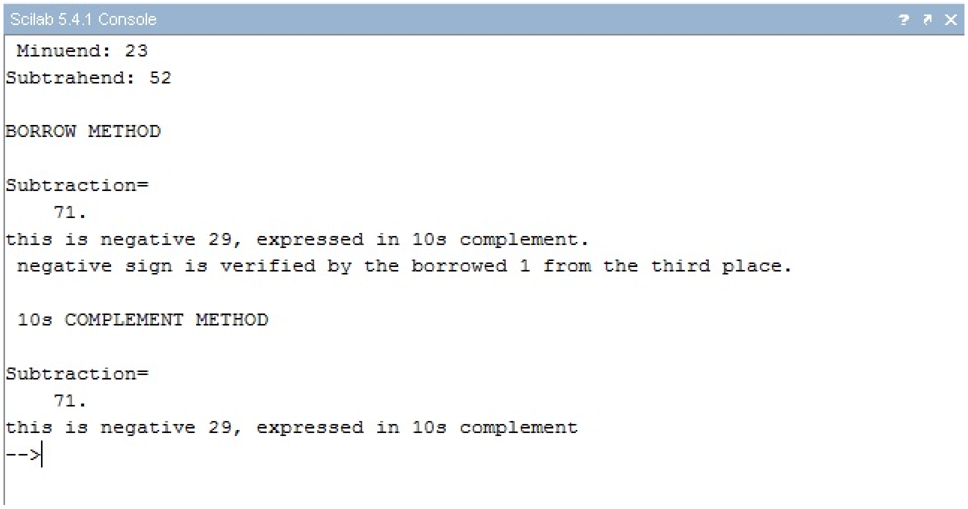


Figure 19.3: SUBTRACTION OF TWO NUMBERS

Scilab code Exa 19.4 2s COMPLIMENT OF BINARY NUMBER

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

Figure 19.4: 2s COMPLIMENT OF BINARY NUMBER

Scilab code Exa 19.5 SUBTRACTION OF TWO NUMBERS

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

Figure 19.5: SUBTRACTION OF TWO NUMBERS

Scilab code Exa 19.6 SUBTRACTION OF TWO NUMBERS

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

Scilab code Exa 19.7 SUBTRACTION OF UNSIGNED NUMBERS

```
// page no 628
```

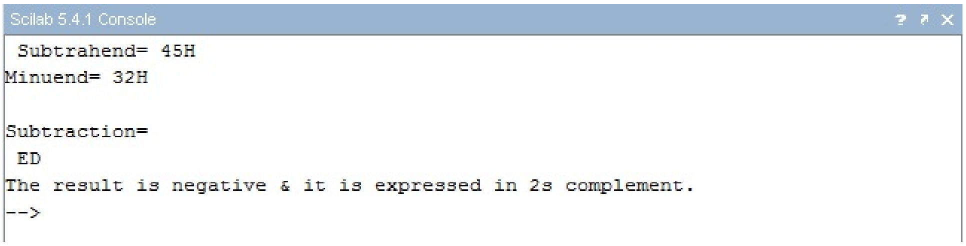


Figure 19.6: SUBTRACTION OF TWO NUMBERS

```
// example no A.7
// SUBTRACTION OF UNSIGNED NUMBERS
clc;
printf('Part a \n \n')
printf('Subtrahend= 62H \n');
printf('Minuend= FAH \n \n');
// finding 2's complement of subtrahend (62H);
m=hex2dec(['FA']);
x=hex2dec(['62']);
y=bitcmp(x,8); // 1's compliment of 62H
z=y+1; // 2's compliment of 62H
s=m+z;
f=s-256; // to compensate the effect of 2's
    compliment
e=dec2hex(f);
printf('Subtraction= ');
disp(e);
printf('This result is positive \n \n');


printf('Part b \n \n')
printf('Subtrahend= FAH \n');
printf('Minuend= 62H \n \n');
// finding 2's complement of subtrahend (FAH);
m=hex2dec(['62']);
x=hex2dec(['FA']);
y=bitcmp(x,8); // 1's compliment of FAH
z=y+1; // 2's compliment of FAH
```

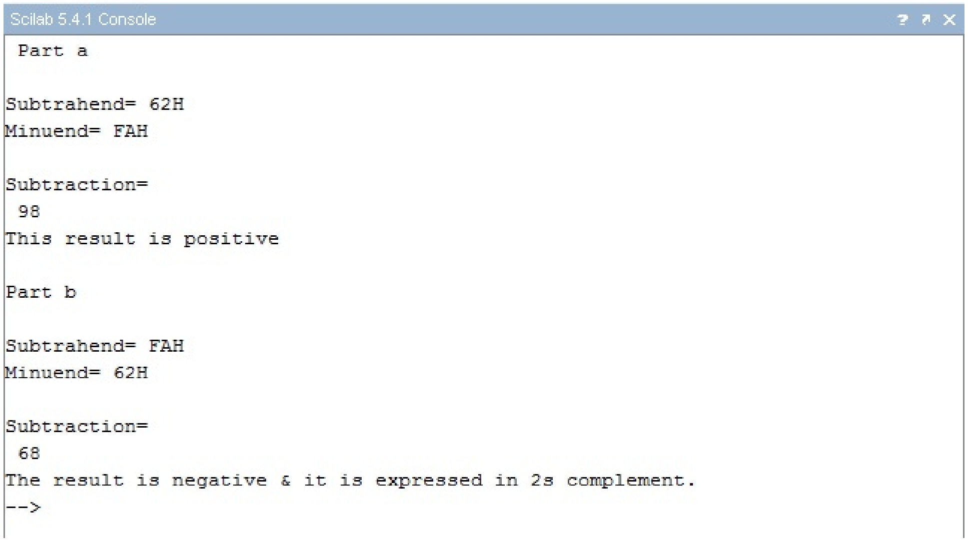


Figure 19.7: SUBTRACTION OF UNSIGNED NUMBERS

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

Scilab code Exa 19.8 SUBTRACTION OF SIGNED NUMBERS

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

```
this must be represented in \n2s compliment form.
\n');
// finding 2's complement of subtrahend (FAH);
m=hex2dec(['FA']);
x=hex2dec(['62']);
y=bitcmp(m,8); // 1's compliment of FAH
z=y+1; // 2's compliment of FAH
printf('2s compliment of minuend is= ');
disp(z);

printf('\n \n Subtrahend= 62H \n');
printf('It is a positive no since D7= 0 for 62H. \n'
);
// subtraction can be represented as
// FAH-62H= (-06H)-(+62H)
s=-x-z;
a=-s;
d=dec2hex(a);
printf('Subtraction= ');
disp(s);
disp(d);
printf('in hexadecimal with a negative sign \n \n');
g=bitcmp(a,8); // 1's compliment of result
q=g+1; // 2's compliment of result
e=dec2hex(q);
printf('2s compliment of result would be= ');
disp(e);
```

Scilab code Exa 19.9 ADDITION OF TWO POSITIVE NUMBERS

```
//page no 629
//example no A.9
// ADDITION OF TWO POSITIVE NUMBERS
```

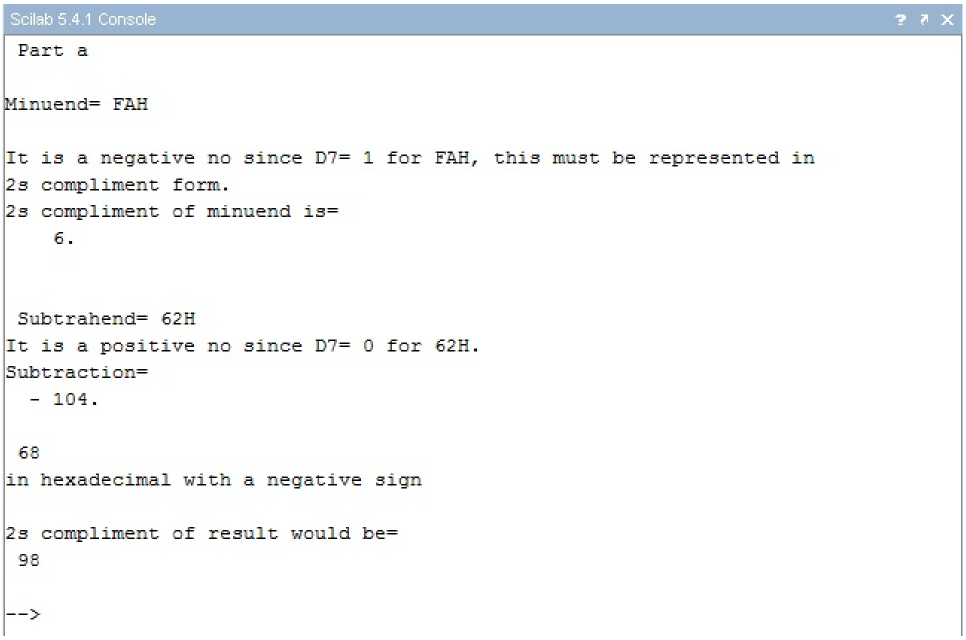


Figure 19.8: SUBTRACTION OF SIGNED NUMBERS

```
clc;
//the given numbers are 41H & 54H.
A=hex2dec(['41']);
B=hex2dec(['54'])
Y=A+B;
X=dec2hex(Y);
printf('Sum =')
disp(X);
if Y>255 then                    // checking the carry flag
   .
    printf('CY=1 \n \n')
else
    printf('CY=0 \n \n')
end
if Y>127 then                    // checking the sign flag.
    printf('S=1 \n \n')
else
    printf('S=0 \n \n')
```

Figure 19.9: ADDITION OF TWO POSITIVE NUMBERS

```
end
if Y>0 then                         // checking the zero flag.
    printf('Z=0 \n \n')
else
    printf('Z=1 \n \n')
end
```