Institute of Enginnering , Central Campus,Pulchowk

Embedded system

LAB #1

Familiarization with 8051/8052 Microcontroller

Submitted BY:
Amrit Prasad Phuyal
Roll: PULL074BEX004

Submitted To:
Department of Electronics
and Computer Engineering

October 9, 2020

Contents

List of Figures

1 Introduction

1.1 Microcontroller

A microcontroller is an integrated circuit (IC), usually via an MPU, memory and certain peripherals, to control other parts of an electronic system . These devices are optimized for embed-in applications that require agile and agile processing, digital, analog or electromechanical interactions.

1.2 8051 Microcontroller

In 1981, Intel introduced an 8-bit microcontroller called the 8051. It was referred as system on a chip because it had 128 bytes of RAM, 4K byte of on-chip ROM, two timers, one serial port, and 4 ports (8-bit wide), all on a single chip.

The different features of the 8051 microcontroller include:

- 4KB bytes on-chip program memory (ROM)
- 128 bytes on-chip data memory (RAM)
- Four register banks
- 128 user defined software flags
- 8-bit bidirectional data bus
- 16-bit unidirectional address bus
- 32 general purpose registers each of 8-bit
- 16 bit Timers (usually 2, but may have more or less)
- Three internal and two external Interrupts
- Four 8-bit ports,(short model have two 8-bit ports)
- 16-bit program counter and data pointer
- 8051 may also have a number of special features such as UARTs, ADC, Op-amp, etc.

1.2.1 Memory Architecture

Internal RAM, Program Memory, External Data Memory, and Special Function Registers are Four different typre of memeory available in 8051 microcontroller.The Internal RAM, or generally referred to as the IRAM has an 8-bit address space taking up the addressess from 0x00 to 0xFF.Program memory, referred as PMEM is up to 64 KB of read-only memory, starting at address 0 in a separate address space.XRAM is a third address space memory space starting at address 0 with 16-bit address space. SFR are located at the same address as IRAM i.e. at 0x80 to 0xFF and accessed just as lower half of IRAM.

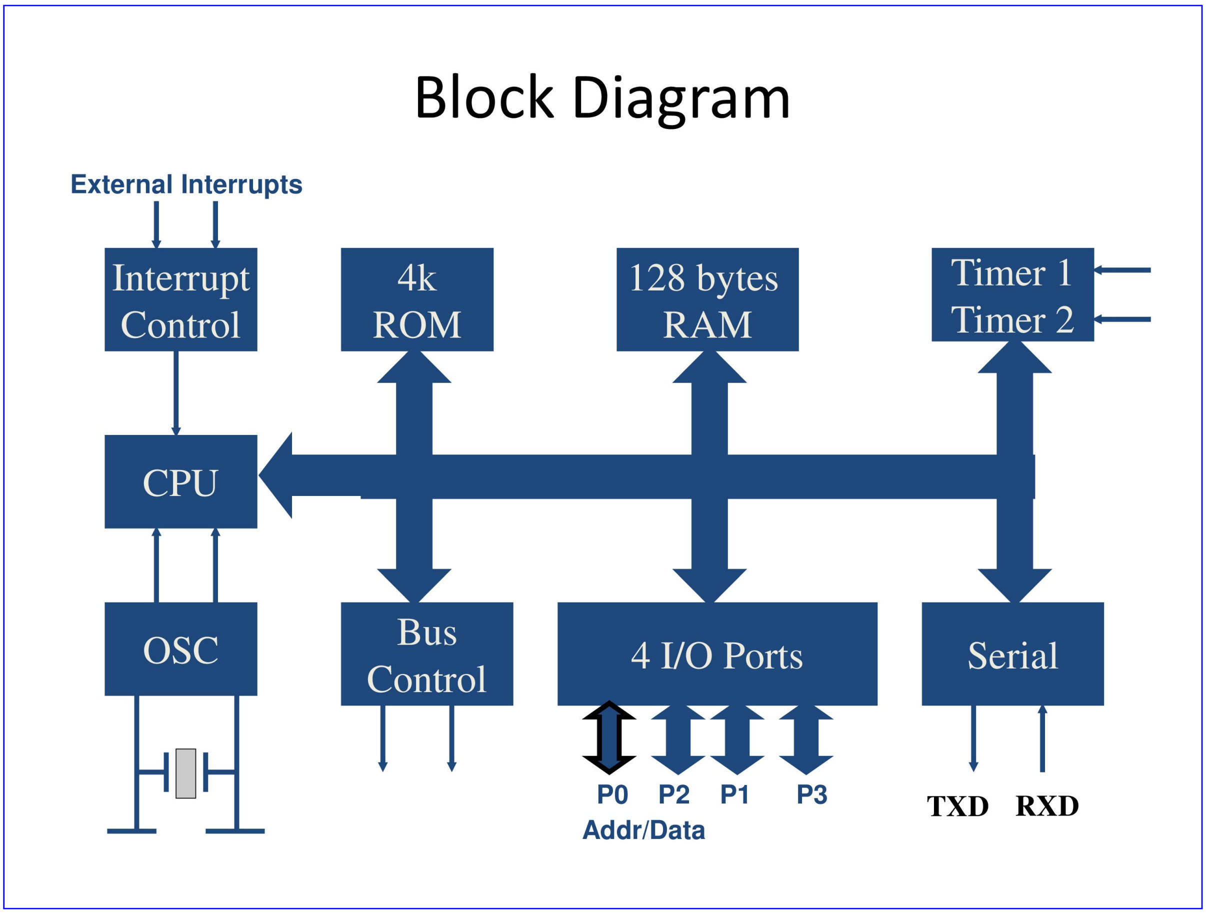


Figure 1: Block diagram of 8051 microcontroller

1.2.2 Programming

8051 can be programmed using both assembly language or embedded C language . In assembly language mnemonics along with hex codes is used , has faster execution and has more control over the memory than in high level language like C, which is more like human readable English language.

1.2.3 Applications of 8051 Microcontroller

Even with the development of many advanced and superior Microcontrollers, 8051 Microcontroller is still being used in many embedded system and applications.

Some of the applications of 8051 Microcontroller are mentioned below:

- Consumer Appliances (TV Tuners, Remote controls, Computers, Sewing Machines, etc.)
- Home Applications (TVs, VCR, Video Games, Camcorder, Music Instruments, Home Security Systems, Garage Door Openers, etc.)
- Communication Systems (Mobile Phones, Intercoms, Answering Machines, Paging Devices, etc.)
- Office (Fax Machines, Printers, Copiers, Laser Printers, etc.)

- Automobiles (Air Bags, ABS, Engine Control, Transmission Control, Temperature Control, Keyless Entry, etc)
- Aeronautical and Space
- Medical Equipment
- Defense Systems
- Robotics
- Industrial Process and Flow Control
- Radio and Networking Equipment
- Remote Sensing

2 Objectives of Lab- 1

Familiarization with the 8051/8052 microcontroller will enable us to write assembly language code for the 8051/8052 microcontroller capble of:

- Data manipulation
- Looping and branching techniques
- Arthimetic and logical operations
- Subroutine calls

3 Lab Experiment Environment

The lab experiments will be performed virtually via various simulation software. The fundamental use of these tools allows the different functional units of the 8051 micro controller to be visualized and defined to do simple logical and arthemetic work. For this lab Proteus design suite for simlulation and KEIl IDE are used .

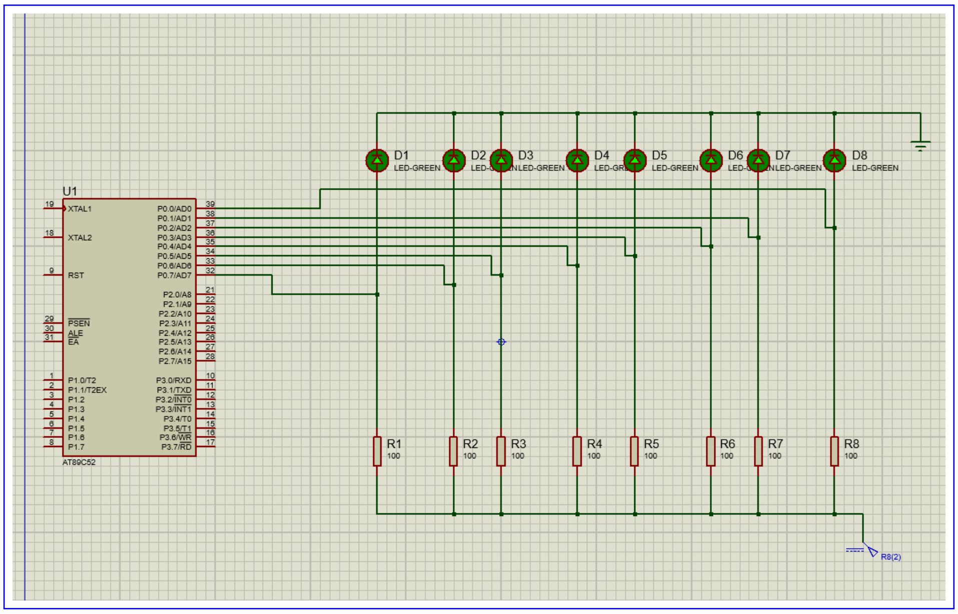


Figure 2: Proteus simulation

Lab Problems

4 Question -1

Write code to add the numbers 897F9AH and 34BC48H and save the result in internal RAM starting at 40H. The result should be displayed continuously on the LEDs of the development board starting from least significant byte with an appropriate timing interval between each byte. Use port zero (P0) of the micro-controller to interface with LEDs.

Assembly

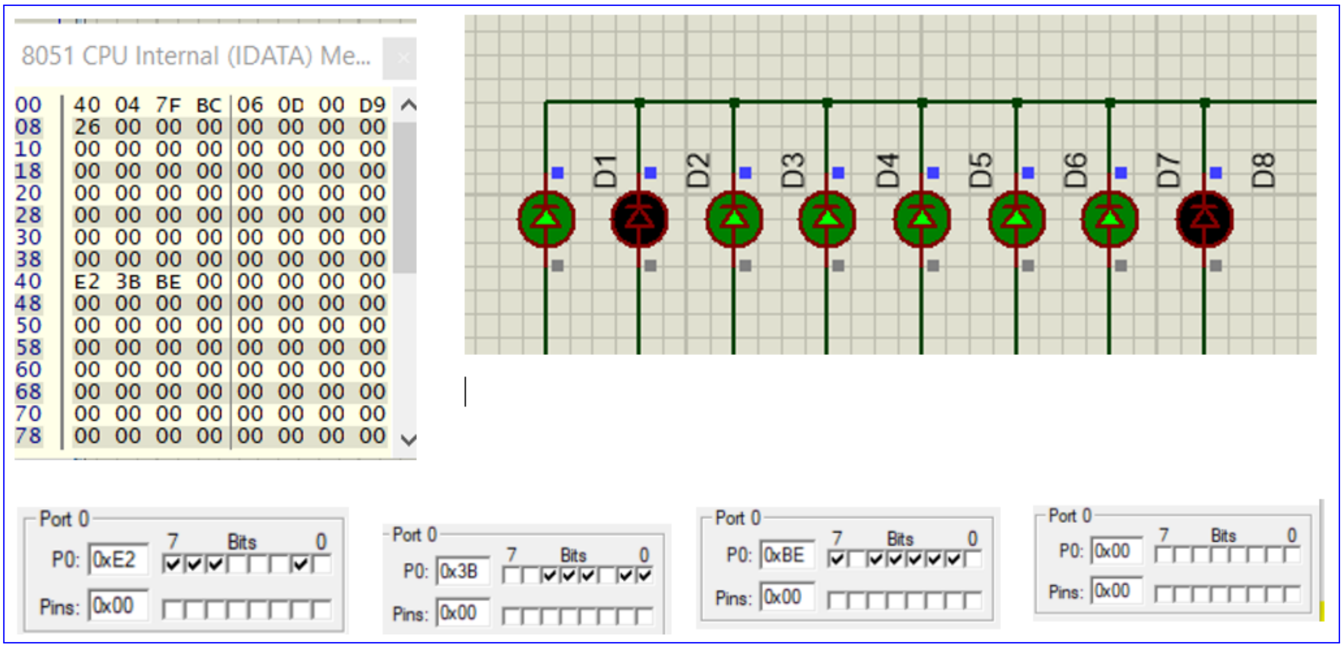
```
        ORG 00H

        MOV R0,#9AH
        MOV R1,#48H
        MOV R2,#7FH
        MOV R3,#0BCH
        MOV R4,#89H
        MOV R5,#34H

        MOV A,R0
        ADD A,R1
        MOV 40H,A
        MOV A,R2
        ADDC A,R3
        MOV 41H,A
        MOV A,R4
        ADDC A,R5
        MOV 42H,A
        MOV A,#0H
        ADDC A,#0H
        MOV 43H,A

REPEAT: MOV R1,#04H
        MOV R0,#40H

NEXT:   MOV P0,@R0
        ACALL DELAY
        INC R0
        DJNZ R1,NEXT
        AJMP REPEAT

DELAY:  MOV R4,#7
POS1:   MOV R5,#255
POS2:   MOV R7,#255
POS3:   DJNZ R7,POS3
        DJNZ R5,POS2
        DJNZ R4,POS1
        RET
        END
```

C language

```
#include <reg51.h>
char data d[4] _at_ 0x40;

void delay(int time)
{
   unsigned int i,j;
   for (i=0; i<time; i++)
      for (j=0; j<125; j++);
}

void main(void)
{
   unsigned long a = 0x897f9a;
   unsigned long b = 0x34bc48;
   unsigned long c = a + b;
   unsigned int i;

   for(i=0; i<4; i++)
   {
      d[i] = c%0x100;
      c >>= 8;
   }

   while(1)
      for(i=0; i<4; i++)
      {
         P0 = d[i];
         delay(1000);
      }
}
```

OUTPUT :

For all output port 0 values are snapshot from keil ide using breakpoint feature. For this particular problem, additional IRAM and snapshot of proteus are included. The addition of 897F9AH and 34BC48H gives 00BE3BE2H which is continuously displayed on Port 0 and stored at 40H starting from LSB, which can be viewed in IRAM table.



8051 CPU Internal (IDATA) Me...

```
00 | 40 04 7F BC | 06 0D 00 D9
08 | 26 00 00 00 | 00 00 00 00
10 | 00 00 00 00 | 00 00 00 00
18 | 00 00 00 00 | 00 00 00 00
20 | 00 00 00 00 | 00 00 00 00
28 | 00 00 00 00 | 00 00 00 00
30 | 00 00 00 00 | 00 00 00 00
38 | 00 00 00 00 | 00 00 00 00
40 | E2 3B BE 00 | 00 00 00 00
48 | 00 00 00 00 | 00 00 00 00
50 | 00 00 00 00 | 00 00 00 00
58 | 00 00 00 00 | 00 00 00 00
60 | 00 00 00 00 | 00 00 00 00
68 | 00 00 00 00 | 00 00 00 00
70 | 00 00 00 00 | 00 00 00 00
78 | 00 00 00 00 | 00 00 00 00
```

Port 0
P0: 0xE2 7 Bits 0 ☑☑☑□□□☑□
Pins: 0x00 □□□□□□□□

Port 0
P0: 0x3B 7 Bits 0 □□☑☑☑□☑☑
Pins: 0x00 □□□□□□□□

Port 0
P0: 0xBE 7 Bits 0 ☑□☑☑☑☑☑□
Pins: 0x00 □□□□□□□□

Port 0
P0: 0x00 7 Bits 0 □□□□□□□□
Pins: 0x00 □□□□□□□□

Figure 3: Addition of two hexadeciaml no.

5 Question -2

Implement a subroutine that replaces the SWAP instruction using rotate right instructions. Test your program on the contents of the accumulator when it contains the number 6BH.

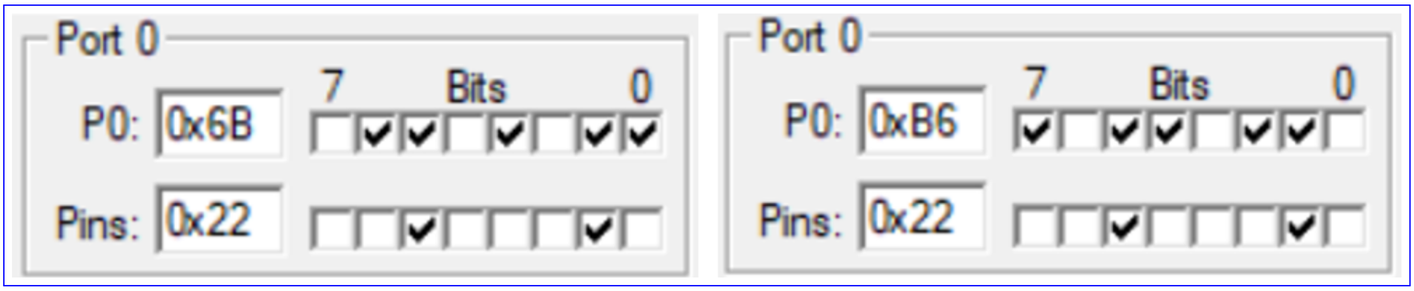
Assembly

```
        ORG 00H

AGAIN:  MOV A,#6BH
        MOV P0,A
        ACALL DELAY
        ACALL SWAP_RR
        MOV P0,A
        ACALL DELAY
        AJMP AGAIN

SWAP_RR:RR A
        RR A
        RR A
        RR A
        RET

DELAY:  MOV R4,#7
HERE1:  MOV R5,#255
HERE2:  MOV R7,#255
HERE3:  DJNZ R7,HERE3
        DJNZ R5,HERE2
        DJNZ R4,HERE1
        RET

        END
```

C language

```c
#include<reg51.h>

void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

void main()
{
   unsigned char value = 0xb6;
   unsigned char ivalue;
   unsigned char a,b;
   a=value/0x10;
   b=value%0x10;
   ivalue = b*(0x10) + a;

   while(1)
   {
      P0 = value;
      delay(1000);
      P0 = ivalue;
      delay(1000);
   }
}
```

OUTPUT :
The upper and lower nibbles of accumulator are swaped without using the SWAP instruction. Hence, 6B H becomes B6 H once the swap is performed.



Port 0
P0: 0x6B 7 Bits 0 □ ☑ ☑ □ ☑ □ ☑ ☑
Pins: 0x22 □ □ ☑ □ □ □ ☑ □

Port 0
P0: 0xB6 7 Bits 0 ☑ □ ☑ ☑ □ ☑ ☑ □
Pins: 0x22 □ □ ☑ □ □ □ ☑ □

Figure 4: Swaping using rotate right

6 Question -3

Multiply, by using looping and successive addition technique, the data in RAM location 22H by the data in RAM location 15H and put the result in RAM locations 19H (low byte) and 1AH (high byte). Data in 22H should be FFH and data in 15H should be DEH.
Assembly

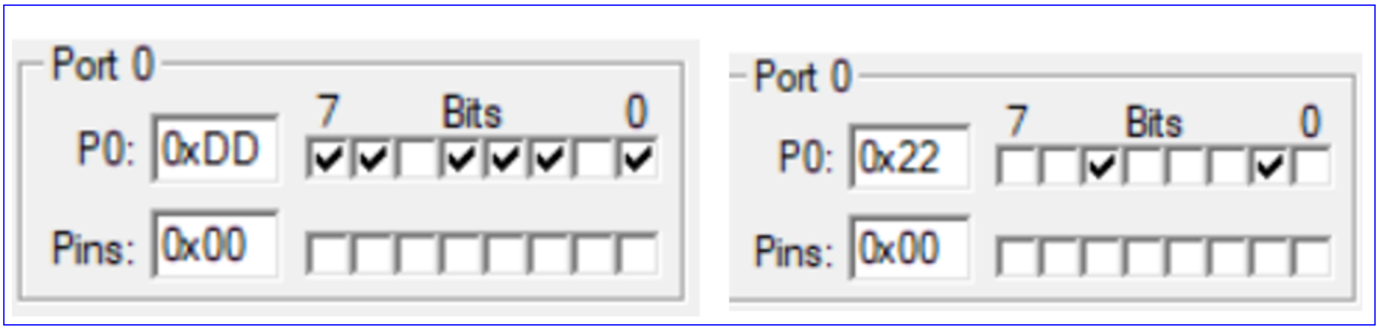
```
        ORG 00H

        MOV 22H,#0FFH
        MOV 15H,#0DEH

        MOV A,#0H
        MOV R1,#0H

        MOV R0,22H
AGAIN:  ADD A,15H
        JNC SKIP
        INC R1
SKIP:   DJNZ R0,AGAIN

        MOV 19H,A
        MOV 1AH,R1

LOOP:   MOV P0,A
        ACALL DELAY
        MOV P0,R1
        ACALL DELAY
        AJMP LOOP

DELAY:  MOV R4,#7
HERE1:  MOV R5,#255
HERE2:  MOV R7,#255
HERE3:  DJNZ R7,HERE3
        DJNZ R5,HERE2
        DJNZ R4,HERE1
        RET

        END
```

C language

```
#include <reg51.h>
unsigned char data multiplicand _at_ 0x22;
unsigned char data multiplier _at_ 0x15;
unsigned char data answer[2] _at_ 0x19;

void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

void main(void)
{
   unsigned int result = 0x0;
   unsigned char i;

   multiplicand = 0xff;
   multiplier = 0xde;

   for(i=0x0;i<multiplier;i++)
      result += multiplicand;

   answer[0] = result%0x100;
   result >>= 8;
   answer[1] = result%0x100;

   while(1)
   {
      P0 = answer[0];
      delay(1000);
      P0 = answer[1];
      delay(1000);
   }
}
```

OUTPUT :

Multiplication of FF H and DE H is DD22 H



Port 0
P0: 0xDD
7 Bits 0
☑ ☑ □ ☑ ☑ ☑ □ ☑
Pins: 0x00
□ □ □ □ □ □ □ □

Port 0
P0: 0x22
7 Bits 0
□ □ ☑ □ □ □ ☑ □
Pins: 0x00
□ □ □ □ □ □ □ □

Figure 5: Multiplication using Addition

7 Question -4

Divide, by using looping and successive subtraction technique, the data in RAM location 3EH by the number 12H; put the quotient in R4 and remainder in R5. Data in 3EH should be AFH.

Assembly

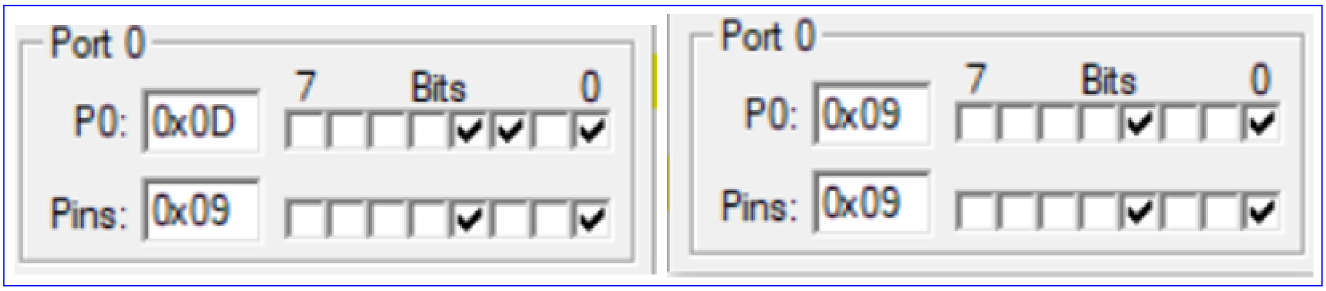
```
        ORG 00H

        MOV 3EH,#0AFH

        MOV A,3EH
        MOV R4,#0H

AGAIN:  SUBB A,#12H
        JC DONE
        INC R4
        AJMP AGAIN
DONE:   ADD A,#12H
        MOV R5,A

LOOP:   MOV P0,R4
        ACALL DELAY
        MOV P0,R5
        ACALL DELAY
        AJMP LOOP

DELAY:  MOV R1,#7
HERE1:  MOV R2,#255
HERE2:  MOV R3,#255
HERE3:  DJNZ R3,HERE3
        DJNZ R2,HERE2
        DJNZ R1,HERE1
        RET

        END
```

C language

```c
#include <reg51.h>
int data dividend _at_ 0x3e;
unsigned char data reg4 _at_ 0x04;
unsigned char data reg5 _at_ 0x05;

void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

void main(void)
{
   unsigned char divisor = 0x12;
   unsigned char quotient = 0x00, remainder;

   dividend = 0x00af;

   while(1)
   {
      dividend -= divisor;
      if(dividend < 0x0)
         break;
      quotient += 0x1;
   }
   remainder = dividend + divisor;

   reg4 = quotient;
   reg5 = remainder;

   while(1)
   {
      P0 = quotient;
      delay(1000);
      P0 = remainder;
      delay(1000);
   }
}
```

OUTPUT :

Dividing AF H by 12 H gives quotient = 9 H and remainder = D H,which are stored in R4 and R5 register.



Port 0
P0: 0x0D
7 Bits 0
□□□□☑☑□☑
Pins: 0x09
□□□□☑□□☑

Port 0
P0: 0x09
7 Bits 0
□□□□☑□□☑
Pins: 0x09
□□□□☑□□☑

Figure 6: Division using Subtraction

8 Question -5

Store ten hexadecimal numbers in internal RAM starting from memory location 50H. The list of numbers to be used is: D6H, F2H, E4H, A8H, CEH, B9H, FAH, AEH, BAH, CCH. Implement a subroutine that extracts both the smallest and largest numbers from the stored numbers.

Assembly

```
        ORG 00H

        MOV 50H,#0D6H
        MOV 51H,#0F2H
        MOV 52H,#0E4H
        MOV 53H,#0A8H
        MOV 54H,#0CEH
        MOV 55H,#0B9H
        MOV 56H,#0FAH
        MOV 57H,#0AEH
        MOV 58H,#0BAH
        MOV 59H,#0CCH

        MOV R0,#50H

        MOV A,@R0
        MOV R7,A      ;SMALLEST
        MOV R1,A      ;LARGEST

        MOV R2,#09H

NEXT:   INC R0
        MOV A,R7
        SUBB A,@R0
        JNC NO_SMALL
        MOV A,@R0
        MOV R7,A
NO_SMALL:   MOV A,R1
        SUBB A,@R0
        JC NO_BIG
        MOV A,@R0
        MOV R1,A
NO_BIG: DJNZ R2,NEXT

LOOP:   MOV P0,R7
        ACALL DELAY
        MOV P0,R1
        ACALL DELAY
        AJMP LOOP

DELAY:  MOV R3,#7
HERE1:  MOV R4,#255
HERE2:  MOV R5,#255
HERE3:  DJNZ R5,HERE3
        DJNZ R4,HERE2
        DJNZ R3,HERE1
        RET

        END
```

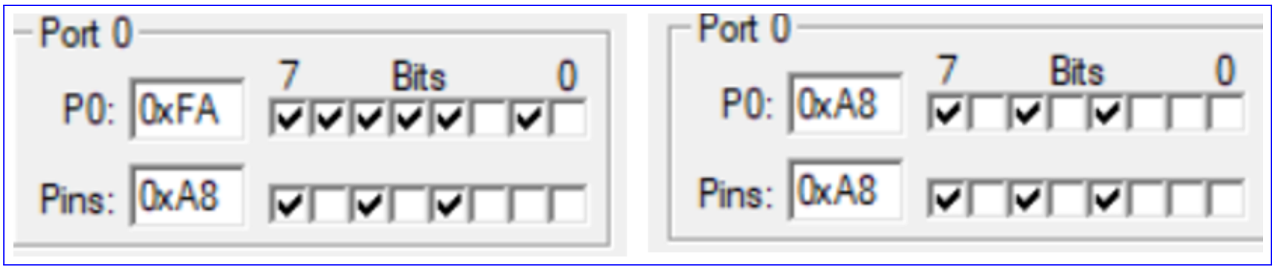
C language

```
#include <reg51.h>
unsigned char data d[10] _at_ 0x50;

void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

void main(void)
{
```

```c
    unsigned char smallest, largest;
    unsigned char i;

    d[0] = 0xd6; d[1] = 0xf2; d[2] = 0xe4;
    d[3] = 0xa8; d[4] = 0xce; d[5] = 0xb9;
    d[6] = 0xfa; d[7] = 0xae; d[8] = 0xba;
    d[9] = 0xcc;

    smallest = largest = d[0];
    for(i=1;i<10;i++)
    {
        if(d[i] < smallest)
            smallest = d[i];
        if(d[i] > largest)
            largest = d[i];
    }

    while(1)
    {
        P0 = smallest;
        delay(1000);
        P0 = largest;
        delay(1000);
    }
}
```

OUTPUT :

Among 10 stored Numbers Largest number = FA H and smallest number = A8 H.



Port 0

P0: 0xFA 7 Bits 0 ☑ ☑ ☑ ☑ ☑ □ ☑ □

Pins: 0xA8 ☑ □ ☑ □ ☑ □ □ □

Port 0

P0: 0xA8 7 Bits 0 ☑ □ ☑ □ ☑ □ □ □

Pins: 0xA8 ☑ □ ☑ □ ☑ □ □ □

Figure 7: Finding largest and smallest number

9 Question -6

Store ten hexadecimal numbers in internal RAM starting from memory location 60H. The list of numbers to be used is: A5H, FDH, 67H, 42H, DFH, 9AH, 84H, 1BH, C7H, 31H. Implement a subroutine that orders the numbers in ascending order using bubble or any other sort algorithm and implement s subroutine that order the numbers in descending order using selection sort algorithm.

Bubble sort

Assembly

```
        ORG 00H

        MOV 60H,#0A5H
        MOV 61H,#0FDH
        MOV 62H,#67H
        MOV 63H,#42H
        MOV 64H,#0DFH
        MOV 65H,#9AH
        MOV 66H,#84H
        MOV 67H,#1BH
        MOV 68H,#0C7H
        MOV 69H,#31H

        MOV R1,#09H
AGN2:   MOV A,R1
        MOV R2,A

        MOV R0,#60H
        MOV A,@R0

AGN1:   INC R0
        MOV R3,A
        MOV A,@R0
        MOV R4,A

        MOV A,R3
        SUBB A,R4
        JC SKIP

        MOV A,R3
        MOV @R0,A
        MOV A,R4
        DEC R0
        MOV @R0,A
        INC R0

SKIP:   MOV A,@R0
        DJNZ R2,AGN1
        DJNZ R1,AGN2

REP:    MOV R1,#0AH
        MOV R0,#60H
LOOP:   MOV A,@R0
        MOV P0,A
        ACALL DELAY
        INC R0
        DJNZ R1,LOOP
        AJMP REP

DELAY:  MOV R3,#7
HERE1:  MOV R4,#255
HERE2:  MOV R5,#255
HERE3:  DJNZ R5,HERE3
        DJNZ R4,HERE2
        DJNZ R3,HERE1
        RET

        END
```

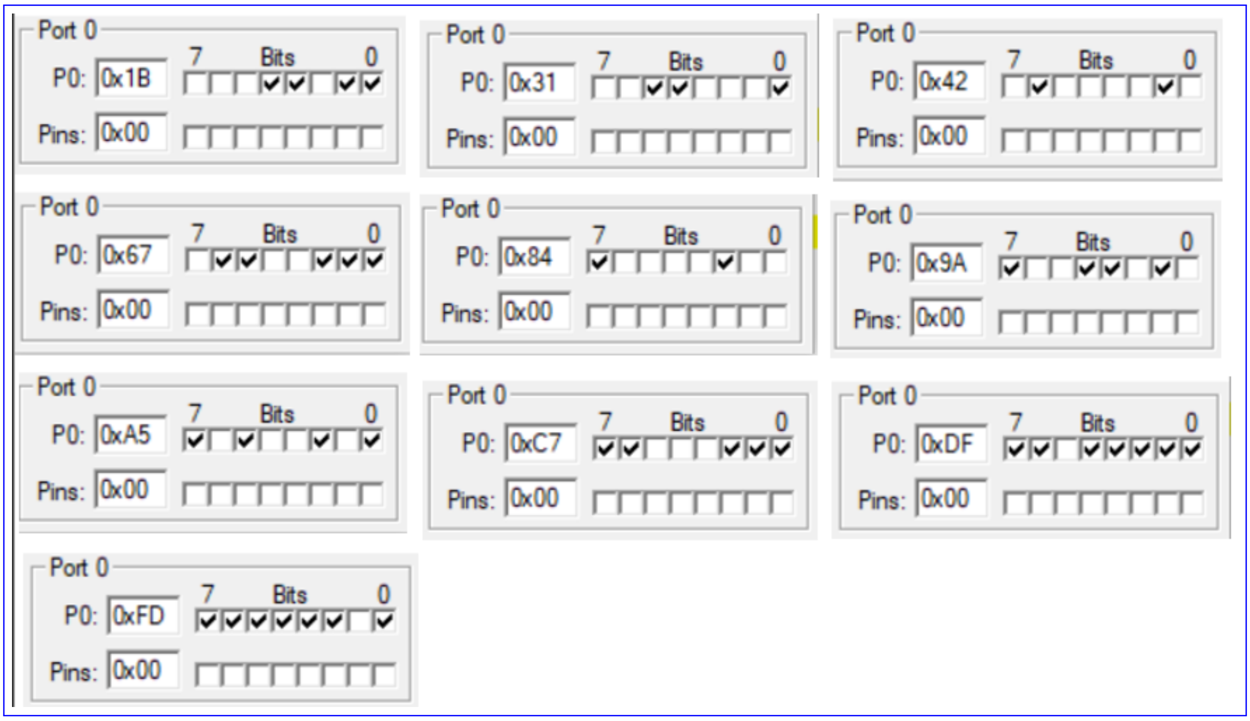
C language

```
#include <reg51.h>
unsigned char data a[10] _at_ 0x60;
void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

void main(void)
{
   unsigned char i, j, temp;
   a[0] = 0xa5; a[1] = 0xfd; a[2] =
   0x67;
   a[3] = 0x42; a[4] = 0xdf; a[5] =
   0x9a;
```

```
    a[6] = 0x84; a[7] = 0x1b; a[8] = 0xc7;
    a[9] = 0x31;

    for(i=0;i<10;i++)
        for(j=0;j<i;j++)
            if(a[j] > a[i])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }

    while(1)
    {
        for( i = 0;i<10;i++)
        {
            P0 = a[i];
            delay(1000);
        }
    }
}
```

OUTPUT :

10 hexadecimal numbers sorted in ascending order using bubble sort algorithm



Port 0
P0: 0x1B 7 Bits 0 ☐☐☐☑☑☐☑☑
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0x31 7 Bits 0 ☐☐☑☑☐☐☐☑
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0x42 7 Bits 0 ☐☑☐☐☐☐☑☐
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0x67 7 Bits 0 ☐☑☑☐☐☑☑☑
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0x84 7 Bits 0 ☑☐☐☐☐☑☐☐
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0x9A 7 Bits 0 ☑☐☐☑☑☐☑☐
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0xA5 7 Bits 0 ☑☐☑☐☐☑☐☑
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0xC7 7 Bits 0 ☑☑☐☐☐☑☑☑
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0xDF 7 Bits 0 ☑☑☐☑☑☑☑☑
Pins: 0x00 ☐☐☐☐☐☐☐☐

Port 0
P0: 0xFD 7 Bits 0 ☑☑☑☑☑☑☐☑
Pins: 0x00 ☐☐☐☐☐☐☐☐

Figure 8: Sorting in Ascending order using bubble sort

Selection Sort

Assembly

```
        ORG 00H

        MOV 60H,#0A5H
        MOV 61H,#0FDH
        MOV 62H,#67H
        MOV 63H,#42H
        MOV 64H,#0DFH
        MOV 65H,#9AH
        MOV 66H,#84H
        MOV 67H,#1BH
        MOV 68H,#0C7H
        MOV 69H,#31H

        MOV R0,#60H
        MOV R6,#09H
AGN:    ACALL F_LARGE
        MOV @R0,A
        INC R0
        DJNZ R6,AGN

AGAIN:  MOV R1,#0AH
        MOV R0,#60H
LOOP:   MOV A,@R0
        MOV P0,A
        ACALL DELAY
        INC R0
        DJNZ R1,LOOP
        AJMP AGAIN

F_LARGE:MOV B,R0
        MOV A,R6 ;COUNTER MAIN
        MOV R2,A ;COUNTER 2

        MOV A,@R0
        MOV R1,A

NEXT:   INC R0
        MOV R4,A ;save A
        SUBB A,@R0
        JNC SKIP

        MOV A,@R0 ;ACC=LARGEST NOW
        MOV R1,A;R1=LARGEST NOW
        MOV A,R4
        MOV @R0,A;XCHG A AND @R0

SKIP:   MOV A,R1;ACC- LARGEST
        DJNZ R2,NEXT
        MOV R0,B
        RET

DELAY:  MOV R3,#7
HERE1:  MOV R4,#255
HERE2:  MOV R5,#255
HERE3:  DJNZ R5,HERE3
        DJNZ R4,HERE2
        DJNZ R3,HERE1
        RET

        END
```

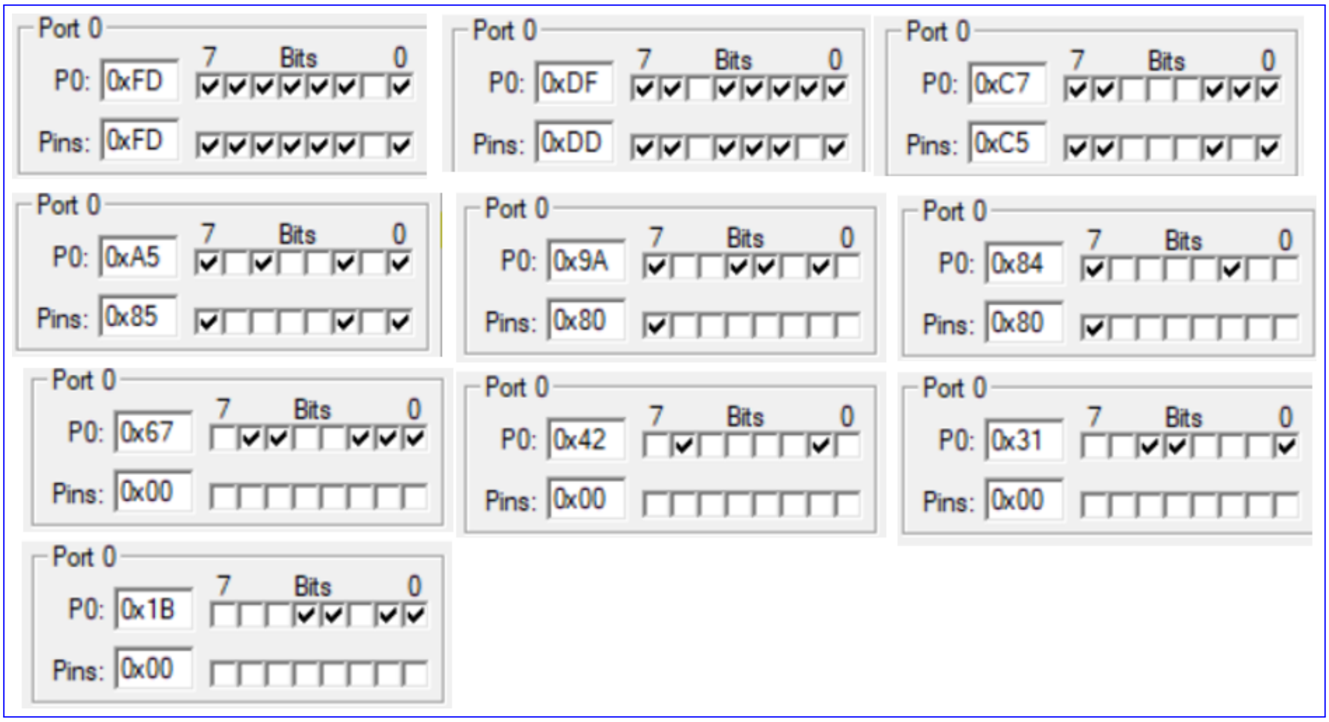
C language

```
#include <reg51.h>
unsigned char data a[10] _at_ 0x60;

void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

void main(void)
{
   unsigned char i, j, temp;
   unsigned char largest = a[0];

   a[0] = 0xa5; a[1] = 0xfd; a[2] = 0x67;
   a[3] = 0x42; a[4] = 0xdf; a[5] = 0x9a;
   a[6] = 0x84; a[7] = 0x1b; a[8] = 0xc7;
   a[9] = 0x31;

   for(i=0;i<10;i++)
   {
      for(j=i;j<10;j++)
         if(a[j] > a[i])
         {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
         }

   }

   while(1)
   {
      for( i = 0;i<10;i++)
```

```
        {
            P0 = a[i];
            delay(1000);
        }
    }
}
```

OUTPUT :

10 hexadecimal numbers sorted in descending order using selection sort algorithm.



Port 0
P0: 0xFD 7 Bits 0 ☑☑☑☑☑☑□☑
Pins: 0xFD ☑☑☑☑☑☑□☑

Port 0
P0: 0xDF 7 Bits 0 ☑☑□☑☑☑☑☑
Pins: 0xDD ☑☑□☑☑☑□☑

Port 0
P0: 0xC7 7 Bits 0 ☑☑□□□☑☑☑
Pins: 0xC5 ☑☑□□□☑□☑

Port 0
P0: 0xA5 7 Bits 0 ☑□☑□□☑□☑
Pins: 0x85 ☑□□□□☑□☑

Port 0
P0: 0x9A 7 Bits 0 ☑□□☑☑□☑□
Pins: 0x80 ☑□□□□□□□

Port 0
P0: 0x84 7 Bits 0 ☑□□□□☑□□
Pins: 0x80 ☑□□□□□□□

Port 0
P0: 0x67 7 Bits 0 □☑☑□□☑☑☑
Pins: 0x00 □□□□□□□□

Port 0
P0: 0x42 7 Bits 0 □☑□□□□☑□
Pins: 0x00 □□□□□□□□

Port 0
P0: 0x31 7 Bits 0 □□☑☑□□□☑
Pins: 0x00 □□□□□□□□

Port 0
P0: 0x1B 7 Bits 0 □□□☑☑□☑☑
Pins: 0x00 □□□□□□□□

Figure 9: Sorting in Decending order using Selection sort

10 Question -7

Store ten hexadecimal numbers in internal RAM starting from memory location 60H. The list of numbers to be used is: A5H, FDH, 67H, 42H, DFH, 9AH, 84H, 1BH, C7H, 31H. Implement a subroutine that orders the numbers in ascending order using bubble or any other sort algorithm and implement subroutine that order the numbers in descending order using selection sort algorithm.

Assembly

```
        ORG 00H

        MOV R0,#40H
        MOV A,#00H
AGAIN:  MOV @R0,A
        INC A
        INC R0
        MOV R1,A
        SUBB A,#20H

        JZ DONE2
        MOV A,R1
        AJMP AGAIN



DONE2:  MOV A,42H
        MOV P0,A
        ACALL DELAY
        MOV A,43H
        MOV P0,A
        ACALL DELAY

        MOV R0,#44H
        MOV R1,#1DH ;
NEXT:   ACALL PRIME
        INC R0
        DJNZ R1,NEXT
        AJMP DONE2

PRIME:  MOV A,@R0
        MOV R4,A ; SAVE A

        MOV R2,#02H
INC_B:  MOV A,R4
        MOV B,R2
        DIV AB

        MOV A,B

        JNZ N_RET
        RET
N_RET:  INC R2
        MOV A,R2
        SUBB A,@R0
        JNZ INC_B
        MOV A,R4
        MOV P0,A
        ACALL DELAY
        RET

DELAY:  MOV R7,#7
HERE1:  MOV R6,#255
HERE2:  MOV R5,#255
HERE3:  DJNZ R5,HERE3
        DJNZ R6,HERE2
        DJNZ R7,HERE1
        RET

        END
```

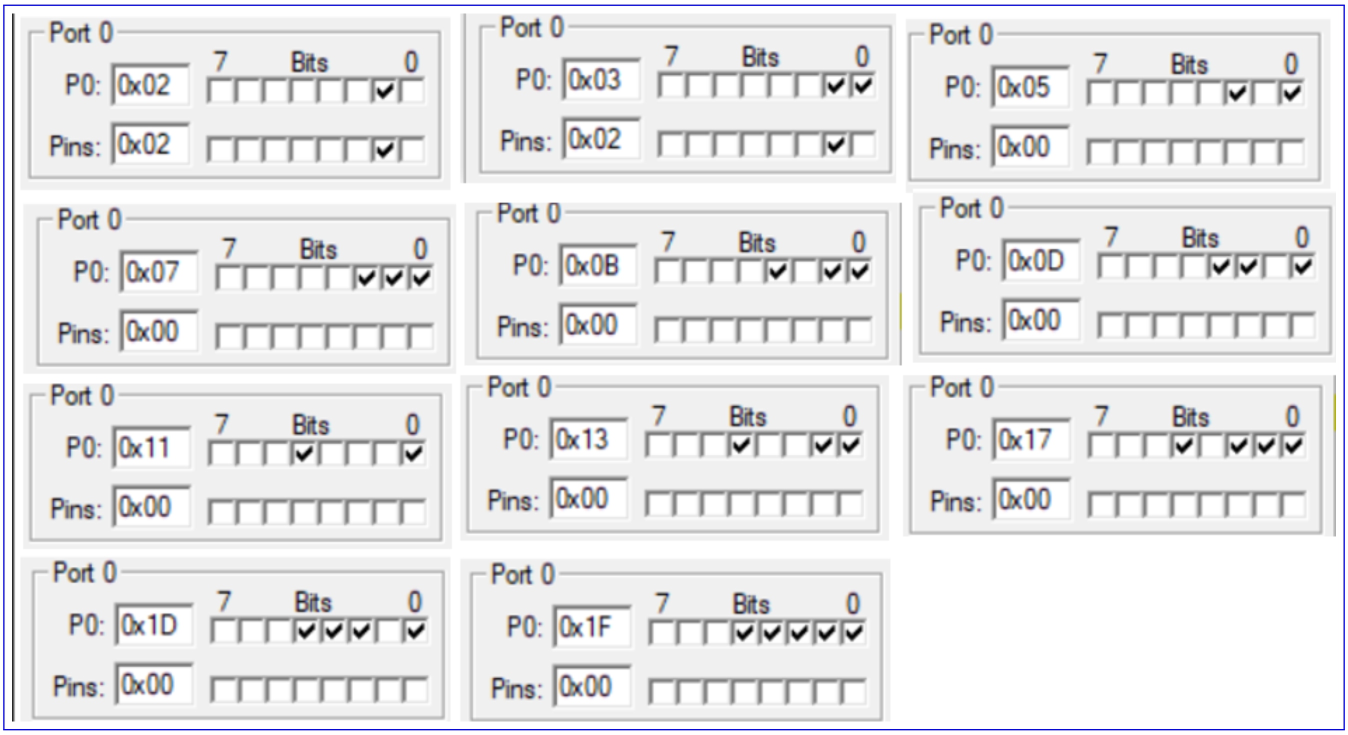
C language

```
#include <reg51.h>
unsigned char data d[21] _at_ 0x40;

void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

int isprime(unsigned char val)
{
   unsigned char j;
   for(j=0x2;j<val;j++)
      if(val % j == 0x0)
            break;
   if(j==val)
         return 1;
   return 0;
}

```

```
void main(void)
{
    unsigned char a[20];
    unsigned char i, count=0;
    for(i = 0x0; i<0x21; i++)
        d[i] = i;

    a[count++] = 0x2;

    for(i=0x3;i<0x21;i++)
    {
        if(isprime(d[i]))
            a[count++] = d[i];
    }

    while(1)
    {
        for(i = 0;i<count;i++)
        {
            P0 = a[i];
            delay(1000);
        }
    }
}
```

OUTPUT :

Only the prime numbers among 00 H to 20 H stored in memory location starting from 40H were to be shown.



Port 0 — P0: 0x02 □□□□□□☑□ — Pins: 0x02 □□□□□□☑□

Port 0 — P0: 0x03 □□□□□□☑☑ — Pins: 0x02 □□□□□□☑□

Port 0 — P0: 0x05 □□□□□☑□☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x07 □□□□□☑☑☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x0B □□□□☑□☑☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x0D □□□□☑☑□☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x11 □□□☑□□□☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x13 □□□☑□□☑☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x17 □□□☑□☑☑☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x1D □□□☑☑☑□☑ — Pins: 0x00 □□□□□□□□

Port 0 — P0: 0x1F □□□☑☑☑☑☑ — Pins: 0x00 □□□□□□□□

Figure 10: Extracting Prime numbers

11 Question -8

Find the factorial of a number stored in R3. The value in R3 could be any number in the range from 00H to 05H. Implement a subroutine that calculates the factorial. The factorial needs to be represented in both hexadecimal and decimal formats.

Assembly

```
        ORG 00H

        MOV R3,#05H

        MOV B,R3
        MOV R1,B

        ACALL FACTO

        MOV R1,A
AGAIN:  MOV A,R1
        MOV P0,A
        ACALL DELAY

        ACALL HTOD
        MOV P0,A
        ACALL DELAY

        MOV A,B
        MOV P0,A
        ACALL DELAY
        SJMP AGAIN

HTOD:   MOV R4,#00H
        MOV B,#0AH
        DIV AB
        MOV R2,A
        SUBB A,#0AH
        JC SKIP
        MOV A,R2
        MOV R3,B
        MOV B,#0AH
        DIV AB
        MOV R4,A
        MOV P0,A
        MOV A,B
        MOV B,R3
        MOV R2,A
SKIP:   MOV A,R2
        SWAP A
        ADD A,B
        MOV B,R4
        RET

DELAY:  MOV R7,#7
HERE1:  MOV R6,#255
HERE2:  MOV R5,#255
HERE3:  DJNZ R5,HERE3
        DJNZ R6,HERE2
        DJNZ R7,HERE1
        RET

FACTO:  MOV A,#01H
LOOP:   MOV B,R1
        MUL AB
        DJNZ R1,LOOP
        RET
        END
```

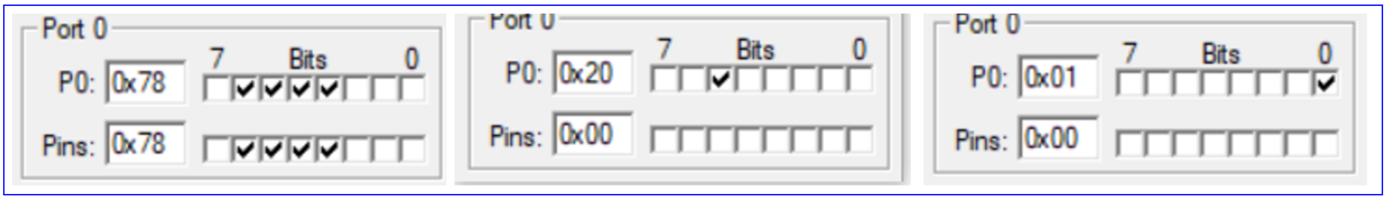
C language

```
#include<reg51.h>

void delay(int time)
{
   unsigned int i,j;
   for (i=0;i<time;i++)
      for (j=0;j<125;j++);
}

void main()
{
   unsigned int a = 0x5;
   unsigned int fact = 0x1;
   unsigned char i;
   unsigned char x, d1, d2, d3;

   for(i = 0x1;i<=a;i++)
      fact *=i;

   x = fact / 0xa;
   d1 = fact % 0xa;
   d2 = x % 0xa;
   d3 = x / 0xa;
   while(1)
   {
      P0 = fact;
      delay(1000);
      P0 = d1;
      delay(1000);
      P0 = d2;
```

```
        delay(1000);
        P0 = d3;
        delay(1000);
    }
}
```

OUTPUT :

Factorial of 5 is 78 H or 120 D.



Port 0
P0: 0x78 7 Bits 0 □☑☑☑☑□□□
Pins: 0x78 □☑☑☑☑□□□

Port 0
P0: 0x20 7 Bits 0 □□☑□□□□□
Pins: 0x00 □□□□□□□□

Port 0
P0: 0x01 7 Bits 0 □□□□□□□☑
Pins: 0x00 □□□□□□□□

Figure 11: Finding Factorial of a number

12 Discussion & Conclusion

In this Lab we perform Addition, substraction, rotation, multiplication, division, additional data manipulation, various logical oper ations based on flags and subroutine calls to be familiar with the 8051/52 microcontroller and basic programming approaches to 8051/52 MCUs.Keil IDE and Proteus Simulation Software were used to verify the result. Schematic diagram made in Proteus is included . Codes of both language Assembly an embedded C is included int his lab report.