Exercise-3: Handling Parallel I/O Ports

- P0 / P1 / P2 / P3 – Detailed pin and port internal descriptions and functionalities

Sample Hands-on exercises for IO ports:

- Send 8b hex number to port 0
- Send 8b binary numbers (0 – 9) to port 2 repetitively
- Send ASCII characters ‘A’ ‘@’, ‘!’, ‘*’ to P0, p1, p 2, p3

- Toggle 8b of port 1
- Toggle Alternate bits of port1
- Toggle MSB bit of Port1
- Toggle LSB bit of Port 1

- Left shift data at port 1 repetitively

- Receive 8b data from P0 and p1. AND data and send result to port 2
- Receive 1b data from P0.0 and P1.3. AND the bits and send result to P2.0
- Receive 1b data from P0.0 and P1.2. AND the bits and send result to P2.7

- (Rx and check condition) Rx data from port0; if 20H send FFH to port2 else send 00H to port3

Bubble Sort

- Pseudocode
- Debugging and output in keil
- Hands On
    - Debugging of bubble sort in keil
    - Watch Window
    - Memory Window
    - Stepping through the code
    - Call Stack
    - Disassembly
    - Performance Analyzer
    - Logic Analyzer
    - Output view on ports and memory window

Generate a Square wave on a port pin

```
#include <REGx51.h> //header file for 89C51
void main(){
	//main function starts
	unsigned int i;
	//Initializing Port1 pin1
	P1_1 = 0; //Make Pin1 o/p
	while(1){
		//Infinite loop main application
		//comes here
		for(i=0;i<1000;i++)
			; //delay loop
		P1_1 = ~P1_1;
		//complement Port1.1
		//this will blink LED connected on Port1.1
	}
}
```

Write a program to make 8 to 1 multiplexer with enable signal

```
#include<reg51.h>

sbit D0 = P0^0;                  // set P0.0-P0.7 (D0-D7) as input pins of mux
sbit D1 = P0^1;
sbit D2 = P0^2;
sbit D3 = P0^3;
sbit D4 = P0^4;
sbit D5 = P0^5;
sbit D6 = P0^6;
sbit D7 = P0^7;
sbit S0 = P1^0;               // set P1.0-P1.2(S0-S2) as select pins of mux
sbit S1 = P1^1;
sbit S2 = P1^2;
sbit EN = P1^7;               // set P1.7 as enable pin
sbit pin = P3^5;              // set P3.5 as output

main()
{
       P3=0x00;P0=0xFF;      // clear op and set all ips
       TMOD = 0x01;           // set timer 0 for 16 bit timer
 next:TL0 = 0xAF;             // load count
       TH0 = 0x3C;
       while(EN==1){}         // wait till enable pin becomes 0
  if((S0==0)&&(S1==0)&&(S2==0)) pin = D0;              // if select inputs are 000 op is
first ip
  else if((S0==1)&&(S1==0)&&(S2==0)) pin = D1;        // if select inputs are 001 op is
second ip
  else if((S0==0)&&(S1==1)&&(S2==0)) pin = D2;        // same as above
  else if((S0==1)&&(S1==1)&&(S2==0)) pin = D3;
  else if((S0==0)&&(S1==0)&&(S2==1)) pin = D4;
  else if((S0==1)&&(S1==0)&&(S2==1)) pin = D5;
  else if((S0==0)&&(S1==1)&&(S2==1)) pin = D6;
  else if((S0==1)&&(S1==1)&&(S2==1)) pin = D7;
  else pin = 0;                                       // by default op is cleared
  TR0 = 1;                                            // start timer 0
  while(TF0==0){}                                      // wait until overflow means
50ms
  TR0 = 0;                                            // stop timer
  goto next;                                          // go for next input
}
```

Take parallel input from port P1 convert it into serial and send it via P0.0

```
#include<reg51.h>
sbit sout = P0^0;                    // serial out on P0.0
sbit D0 = P1^0;                      // parallal input from P1 (D0-D7)
sbit D1 = P1^1;
sbit D2 = P1^2;
sbit D3 = P1^3;
sbit D4 = P1^4;
sbit D5 = P1^5;
sbit D6 = P1^6;
sbit D7 = P1^7;
int i;
void delay(void);                    //  1 ms delay

main()
  {
    for(i=0;i<8;i++)                 // rotate loop for 8 times
      {
        sout = D0;                   // first bit out
        D0 = D1;                     // shift all bits in sequence
        D1 = D2;
        D2 = D3;
        D3 = D4;
        D4 = D5;
        D5 = D6;
        D6 = D7;
        delay();                     // generate 1 ms delay after each bit shifted
      }
   }
void delay()
  {
    int k
    for(k=0;k<1000;k++);
  }
```

Write a program to transmit and receive data via ports as well as control a load by reading switch input

```
#include <REGX51.H>

sbit DIPswitch = P1^4;              // DIP switch input on Port 1 bit 4
sbit greenLED = P1^5;               // green LED output on Port 1 bit 5

void main (void)
{
 unsigned char inval;

 inval = 0;                    // initial value for inval
 while (1)
 {
  if (DIPswitch == 1)
        {           // check if input P1.4 is high
   inval = P1 & 0x0F;              // read bit 0 .. 3 from P1
   greenLED = 0;        // set output P1.5 to low
  }
  else
        {                     // if input P1.4 is low
   greenLED = 1;        // set output P1.5 to high
  }
  P3 = (P3 & 0xF0) | inval;  // output inval to P3.0 .. P3.3
 }
}
```

Detect an external interrupt (by means of inputs) on P3.2 and P3.3. Indicate on LEDs connected on P0.0 & P0.1 (assume switches connected to P3 pins and LED connected to P0 pins) (Try it based on above code)

Take input from P0. Count no of 1 in input – Parity Checker

```
#include <reg51.h>
sbit D0 = B^0; // define D0-D7 as bits of reg B
sbit D1 = B^1;
sbit D2 = B^2;
sbit D3 = B^3;
sbit D4 = B^4;
sbit D5 = B^5;
sbit D6 = B^6;
sbit D7 = B^7;
unsigned bdata reg; // define reg as bit addressable variable
sbit b = reg^0; // define b as bit 0 of reg
void main()
    {
      unsigned int i,c=0; // initialize counter
      B=P0; // take input from P0 to B
      b=0;
      for(i=0;i<8;i++) // rotate bits of reg B
          { // 8 times
              b=D7;
              D7=D6;
              D6=D5;
              D5=D4;
              D4=D3;
              D3=D2;
              D2=D1;
              D1=D0;
              if(b==1)c++; // check every bit if 1
          }
        while(1); // loop continue
      }
```

Assume a key is connected with P3.2. Count no of key press and display it on other port