# PROGRAMMING IN C

Dr. Qamaruddin

*Department of Software Engineering*
*Bahria University, Karachi Campus*

# WHY PROGRAM 8051 IN C

- Compilers produce hex files that is downloaded to ROM of microcontroller
  - The size of hex file is the main concern
    - Microcontrollers have limited on-chip ROM
    - Code space for 8051 is limited to 64K bytes
- C programming is less time consuming, but has larger hex file size
- The reasons for writing programs in C
  - It is easier and less time consuming to write in C than Assembly
  - C is easier to modify and update
  - You can use code available in function libraries
  - C code is portable to other microcontroller with little of no modification

## DATA TYPES

- A good understanding of C data types for 8051 can help programmers to create smaller hex files
  - Unsigned char
  - Signed char
  - Unsigned int
  - Signed int
  - Sbit (single bit)
  - Bit and sfr

## DATA TYPES

## Unsigned char

- The character data type is the most natural choice
  - 8051 is an 8-bit microcontroller
- Unsigned char is an 8-bit data type in the range of 0 – 255 (00 – FFH)
  - One of the most widely used data types for the 8051
    - Counter value
    - ASCII characters
- C compilers use the signed char as the default if we do not put the keyword unsigned

# DATA TYPES

## Unsigned char (cont')

Write an 8051 C program to send values 00 – FF to port P1.

**Solution:**

```
#include <reg51.h>
void main(void)
  {
    unsigned char z;
    for (z=0;z<=255;z++)
       P1=z;
  }
```

1. Pay careful attention to the size of the data
2. Try to use unsigned *char* instead of *int* if possible

Write an 8051 C program to send hex values for ASCII characters of 0, 1, 2, 3, 4, 5, A, B, C, and D to port P1.

**Solution:**

```
#include <reg51.h>
void main(void)
  {
    unsigned char mynum[]="012345ABCD";
    unsigned char z;
    for (z=0;z<=10;z++)
       P1=mynum[z];
  }
```

# DATA TYPES

## Unsigned char (cont')

Write an 8051 C program to toggle all the bits of P1 continuously.

**Solution:**

```
//Toggle P1 forever
#include <reg51.h>
void main(void)
  {
    for (;;)
      {
         p1=0x55;
         p1=0xAA;
       }
  }
```

## DATA TYPES

### Signed char

- The signed char is an 8-bit data type
  - Use the MSB D7 to represent – or +
  - Give us values from –128 to +127
- We should stick with the unsigned char unless the data needs to be represented as signed numbers
  - temperature

Write an 8051 C program to send values of –4 to +4 to port P1.

**Solution:**

```
//Singed numbers
#include <reg51.h>
void main(void)
  {
    char mynum[]={+1,-1,+2,-2,+3,-3,+4,-4};
    unsigned char z;
    for (z=0;z<=8;z++)
       P1=mynum[z];
  }
```

DATA TYPES

Unsigned and Signed int

- The unsigned int is a 16-bit data type
  - Takes a value in the range of 0 to 65535 (0000 – FFFFH)
  - Define 16-bit variables such as memory addresses
  - Set counter values of more than 256
  - Since registers and memory accesses are in 8-bit chunks, the misuse of int variables will result in a larger hex file
- Signed int is a 16-bit data type
  - Use the MSB D15 to represent – or +
  - We have 15 bits for the magnitude of the number from –32768 to +32767

# DATA TYPES

## Single Bit (cont')

Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50,000 times.

**Solution:**

```
#include <reg51.h>
sbit MYBIT=P1^0;

void main(void)
  {
    unsigned int z;
    for (z=0;z<=50000;z++)
       {
         MYBIT=0;
         MYBIT=1;
       }
  }
```

*sbit* keyword allows access to the single bits of the SFR registers

# DATA TYPES

## Bit and sfr

- The bit data type allows access to single bits of bit-addressable memory spaces 20 – 2FH
- To access the byte-size SFR registers, we use the sfr data type

| Data Type | Size in Bits | Data Range/Usage |
|---|---|---|
| unsigned char | 8-bit | 0 to 255 |
| (signed) char | 8-bit | -128 to +127 |
| unsigned int | 16-bit | 0 to 65535 |
| (signed) int | 16-bit | -32768 to +32767 |
| sbit | 1-bit | SFR bit-addressable only |
| bit | 1-bit | RAM bit-addressable only |
| sfr | 8-bit | RAM addresses 80 – FFH only |

## TIME DELAY

- There are two way s to create a time delay in 8051 C
  - Using the 8051 timer
    - The crystal frequency connected to the X1 – X2 input pins
  - Using a simple for loop
    - Compiler choice
      - C compiler converts the C statements and functions to Assembly language instructions
      - Different compilers produce different code

# TIME DELAY (cont')

Write an 8051 C program to toggle bits of P1 continuously forever with some delay.

**Solution:**

```
//Toggle P1 forever with some delay in between
//"on" and "off"
#include <reg51.h>
void main(void)
  {
    unsigned int x;
    for (;;)                    //repeat forever
      {
        p1=0x55;
        for (x=0;x<40000;x++); //delay size
                               //unknown
        p1=0xAA;
        for (x=0;x<40000;x++);
       }
  }
```

We must use the oscilloscope to measure the exact duration

# TIME DELAY (cont')

Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms.

**Solution:**

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
  {
    while (1)                  //repeat forever
       {
         p1=0x55;
         MSDelay(250);
         p1=0xAA;
         MSDelay(250);
       }
  }

void MSDelay(unsigned int itime)
  {
    unsigned int i,j;
    for (i=0;i<itime;i++)
       for (j=0;j<1275;j++);
  }
```

# I/O PROGRAMMING

## Byte Size I/O

LEDs are connected to bits P1 and P2. Write an 8051 C program that shows the count from 0 to FFH (0000 0000 to 1111 1111 in binary) on the LEDs.

**Solution:**

```
#include <reg51.h>
#defind LED P2;

void main(void)
  {
    P1=00;              //clear P1
    LED=0;              //clear P2
    for (;;)            //repeat forever
       {
         P1++;          //increment P1
         LED++;         //increment P2
       }
  }
```

Ports P0 – P3 are byte-accessable and we use the P0 – P3 labels as defined in the 8051/52 header file.

# I/O PROGRAMMING

## Byte Size I/O (cont')

Write an 8051 C program to get a byte of data form P1, wait 1/2 second, and then send it to P2.

**Solution:**

```
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
  {
    unsigned char mybyte;
    P1=0xFF;                    //make P1 input port
    while (1)
       {
         mybyte=P1;             //get a byte from P1
         MSDelay(500);
         P2=mybyte;             //send it to P2
       }
  }
```

# I/O PROGRAMMING

## Byte Size I/O (cont')

Write an 8051 C program to get a byte of data form P0. If it is less than 100, send it to P1; otherwise, send it to P2.

**Solution:**

```
#include <reg51.h>

void main(void)
  {
    unsigned char mybyte;
    P0=0xFF;                    //make P0 input port
    while (1)
       {
         mybyte=P0;             //get a byte from P0
         if (mybyte<100)
            P1=mybyte;          //send it to P1
         else
            P2=mybyte;          //send it to P2
       }
  }
```

# I/O PROGRAMMING

## Bit-addressable I/O

Write an 8051 C program to toggle only bit P2.4 continuously without disturbing the rest of the bits of P2.

**Solution:**

```
//Toggling an individual bit
#include <reg51.h>
sbit mybit=P2^4;

void main(void)
  {
    while (1)
       {
         mybit=1;          //turn on P2.4
         mybit=0;          //turn off P2.4
       }
  }
```

Ports P0 – P3 are bit-addressable and we use *sbit* data type to access a single bit of P0 - P3

Use the Px^y format, where x is the port 0, 1, 2, or 3 and y is the bit 0 – 7 of that port

# I/O PROGRAMMING

## Bit-addressable I/O (cont')

Write an 8051 C program to monitor bit P1.5. If it is high, send 55H to P0; otherwise, send AAH to P2.

**Solution:**

```
#include <reg51.h>
sbit mybit=P1^5;

void main(void)
  {
    mybit=1;                    //make mybit an input
    while (1)
       {
         if (mybit==1)
            P0=0x55;
         else
            P2=0xAA;
       }
  }
```

# I/O PROGRAMMING

## Bit-addressable I/O (cont')

A door sensor is connected to the P1.1 pin, and a buzzer is connected to P1.7. Write an 8051 C program to monitor the door sensor, and when it opens, sound the buzzer. You can sound the buzzer by sending a square wave of a few hundred Hz.

**Solution:**

```
#include <reg51.h>
void MSDelay(unsigned int);
sbit Dsensor=P1^1;
sbit Buzzer=P1^7;

void main(void)
  {
    Dsensor=1;                  //make P1.1 an input
    while (1)
       {
         while (Dsensor==1)//while it opens
           {
              Buzzer=0;
              MSDelay(200);
              Buzzer=1;
              MSDelay(200);
           }
       }
  }
```

# I/O PROGRAMMING

## Bit-addressable I/O (cont')

The data pins of an LCD are connected to P1. The information is latched into the LCD whenever its Enable pin goes from high to low. Write an 8051 C program to send "The Earth is but One Country" to this LCD.

**Solution:**

```
#include <reg51.h>
#define LCDData P1   //LCDData declaration
sbit En=P2^0;        //the enable pin

void main(void)
  {
    unsigned char message[]
              ="The Earth is but One Country";
    unsigned char z;
    for (z=0;z<28;z++) //send 28 characters
       {
         LCDData=message[z];
         En=1;     //a high-
         En=0;     //-to-low pulse to latch data
       }
  }
```

# I/O PROGRAMMING

## Accessing SFR Addresses 80 - FFH

Write an 8051 C program to toggle all the bits of P0, P1, and P2 continuously with a 250 ms delay. Use the sfr keyword to declare the port addresses.

Another way to access the SFR RAM space 80 – FFH is to use the *sfr* data type

**Solution:**

```
//Accessing Ports as SFRs using sfr data type
sfr P0=0x80;
sfr P1=0x90;
sfr P2=0xA0;
void MSDelay(unsigned int);

void main(void)
  {
    while (1)
       {
         P0=0x55;
         P1=0x55;
         P2=0x55;
         MSDelay(250);
         P0=0xAA;
         P1=0xAA;
         P2=0xAA;
         MSDelay(250);
       }
  }
```

# I/O PROGRAMMING

## Accessing SFR Addresses 80 - FFH (cont')

Write an 8051 C program to turn bit P1.5 on and off 50,000 times.

**Solution:**

```
sbit MYBIT=0x95;

void main(void)
  {
    unsigned int z;
    for (z=0;z<50000;z++)
       {
         MYBIT=1;
         MYBIT=0;
       }
  }
```

We can access a single bit of any SFR if we specify the bit address

Notice that there is no `#include <reg51.h>`. This allows us to access any byte of the SFR RAM space 80 – FFH. This is widely used for the new generation of 8051 microcontrollers.

# I/O PROGRAMMING

## Using bit Data Type for Bit-addressable RAM

Write an 8051 C program to get the status of bit P1.0, save it, and send it to P2.7 continuously.

**Solution:**

```
#include <reg51.h>
sbit inbit=P1^0;
sbit outbit=P2^7;
bit membit;              //use bit to declare
                         //bit- addressable memory

void main(void)
  {
    while (1)
       {
         membit=inbit;     //get a bit from P1.0
         outbit=membit;    //send it to P2.7
       }
  }
```

We use bit data type to access data in a bit-addressable section of the data RAM space 20 – 2FH

# LOGIC OPERATIONS

## Bit-wise Operators in C

- Logical operators
  - AND (&&), OR (||), and NOT (!)
- Bit-wise operators
  - AND (&), OR (|), EX-OR (^), Inverter (~), Shift Right (>>), and Shift Left (<<)
    - These operators are widely used in software engineering for embedded systems and control

Bit-wise Logic Operators for C

| | | AND | OR | EX-OR | Inverter |
|---|---|---|---|---|---|
| A | B | A&B | A\|B | A^B | ~B |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | |
| 1 | 1 | 1 | 1 | 0 | |

# LOGIC OPERATIONS

## Bit-wise Operators in C (cont')

Run the following program on your simulator and examine the results.

**Solution:**

```
#include <reg51.h>

void main(void)
  {
    P0=0x35 & 0x0F;          //ANDing
    P1=0x04 | 0x68;          //ORing
    P2=0x54 ^ 0x78;          //XORing
    P0=~0x55;                //inversing
    P1=0x9A >> 3;            //shifting right 3
    P2=0x77 >> 4;            //shifting right 4
    P0=0x6 << 4;             //shifting left 4
  }
```

# LOGIC OPERATIONS

## Bit-wise Operators in C (cont')

Write an 8051 C program to toggle all the bits of P0 and P2 continuously with a 250 ms delay. Using the inverting and Ex-OR operators, respectively.

**Solution:**

```
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
  {
    P0=0x55;
    P2=0x55;
    while (1)
       {
         P0=~P0;
         P2=P2^0xFF;
         MSDelay(250);
       }
  }
```

# DATA CONVERSION

## Checksum Byte in ROM (cont')

Write an 8051 C program to perform the checksum operation to ensure data integrity. If data is good, send ASCII character 'G' to P0. Otherwise send 'B' to P0.

**Solution:**

```
#include <reg51.h>

void main(void)
  {
    unsigned char mydata[]
                    ={0x25,0x62,0x3F,0x52,0xE8};
    unsigned char shksum=0;
    unsigned char x;
    for (x=0;x<5;x++)
       chksum=chksum+mydata[x];
    if (chksum==0)
       P0='G';
    else
       P0='B';
}
```