Chapter 7
8051 Programming in C

Sections

7.1 Data types and time delay in 8051 C

7.2 I/O programming in 8051 C

7.3 Logic operations in 8051 C

7.4 Data conversion programs in 8051 C

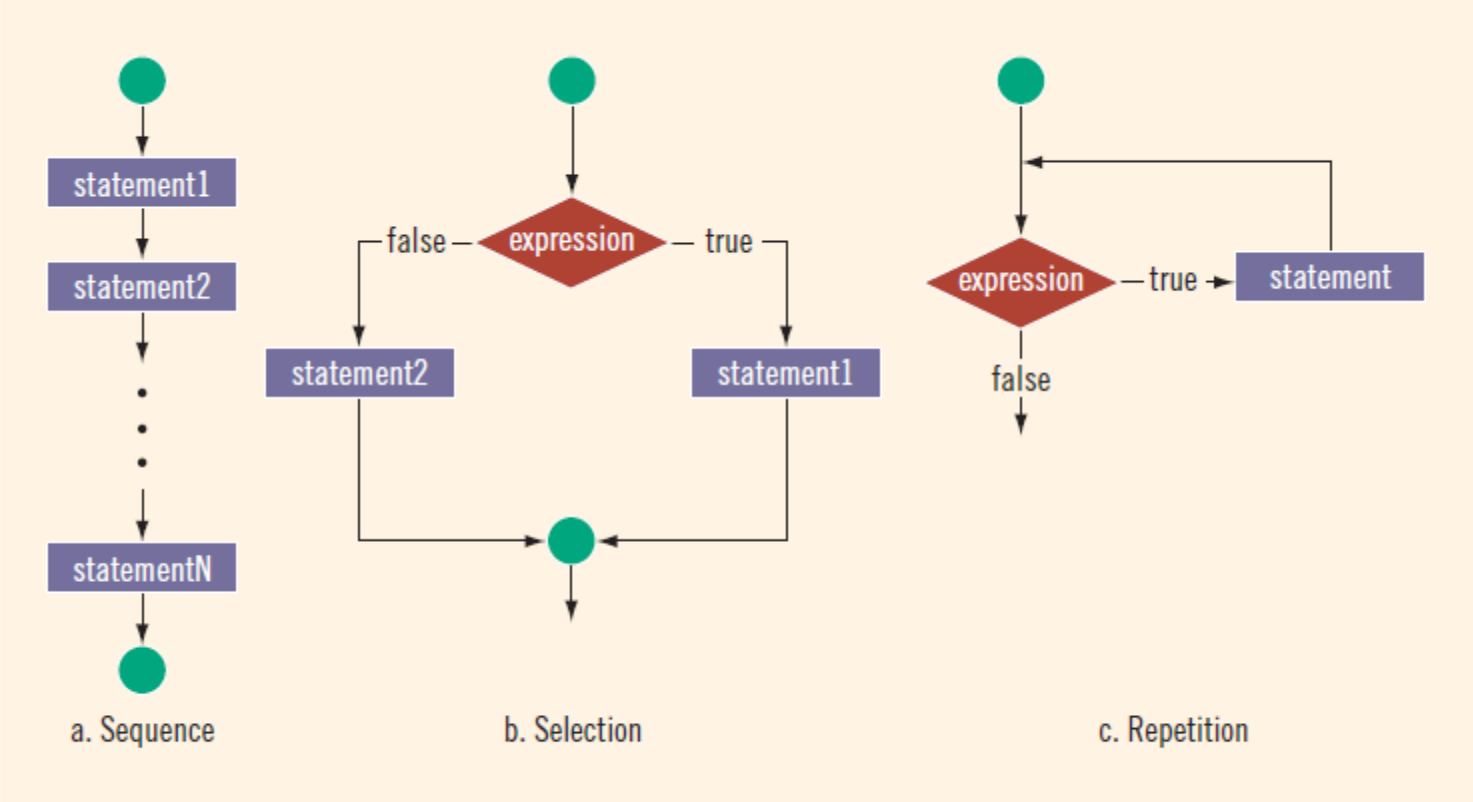
7.5 Accessing code ROM space in 8051 C

7.6 Data serialization using 8051 C

Why Program the 8051 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 microcontrollers with little or no modification.

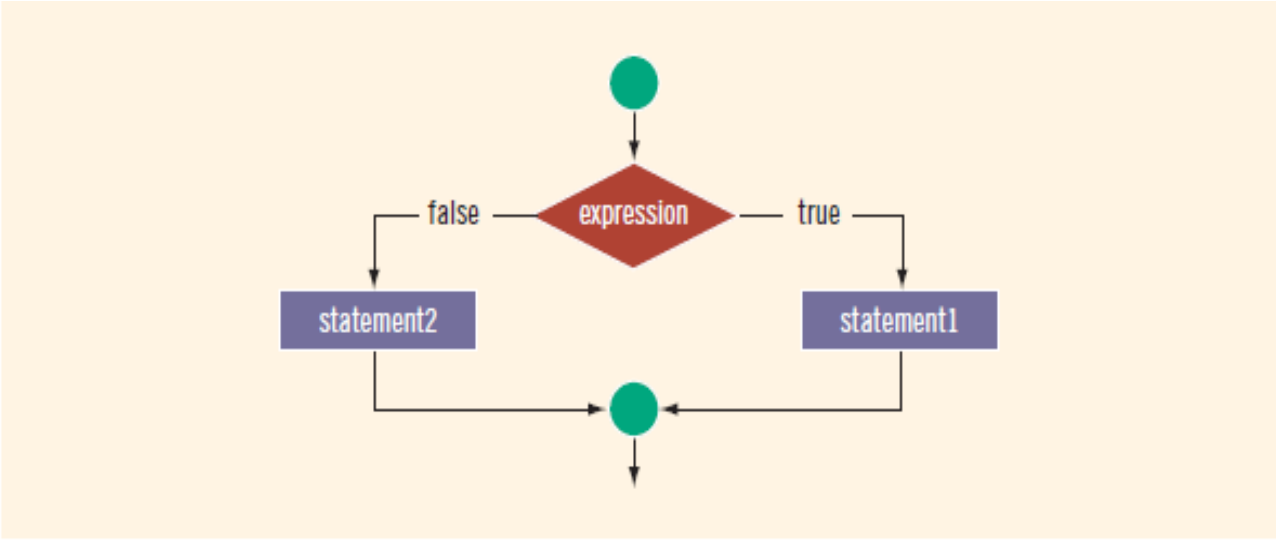
Sequence for any code



Flow of execution

1. if **statement (selection)** Structure

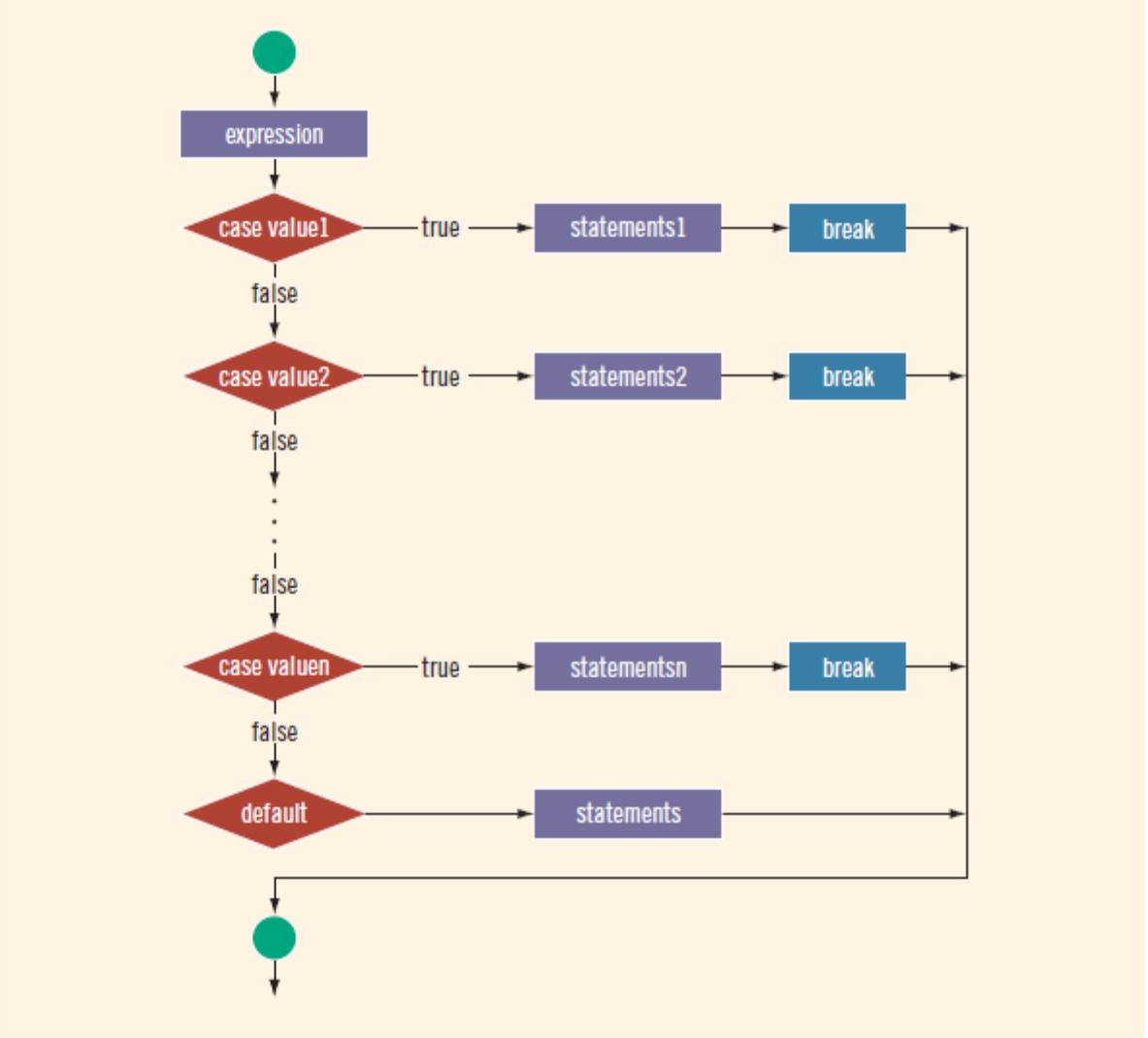
```
if (expression)
     statement1
else
     statement2
```



Two-way selection

2. switch statement (selection) Structure

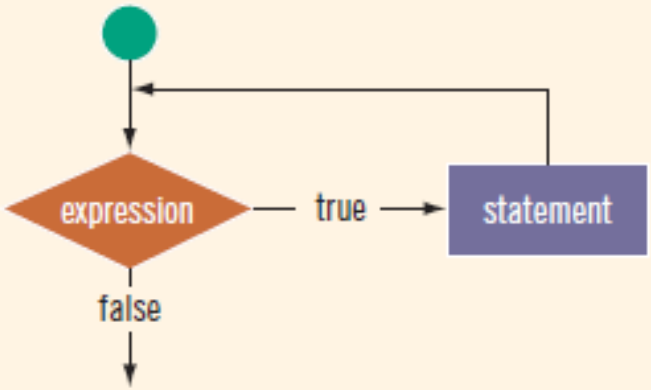
```
switch (expression)
{
case value1:
    statements1
    break;
case value2:
    statements2
    break;
    .
    .
    .
case valuen:
    statementsn
    break;
default:
    statements
}
```



switch statement

3. while Looping (Repetition) Structure

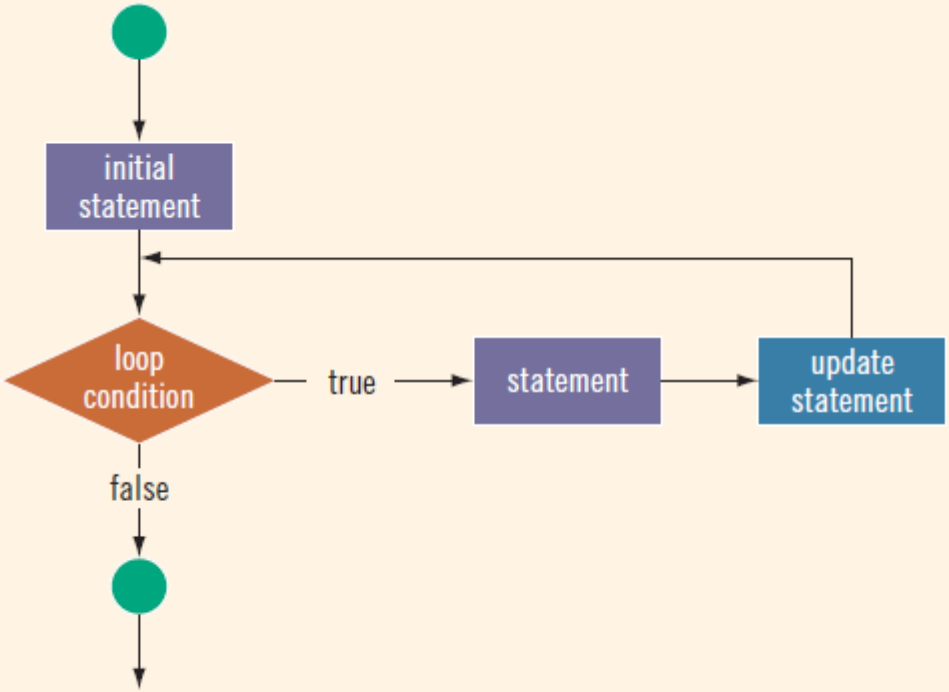
```
while (expression)
    statement
```



while loop

4. for Looping (Repetition) Structure

```
for (initial statement; loop condition; update statement)
    statement
```



for loop

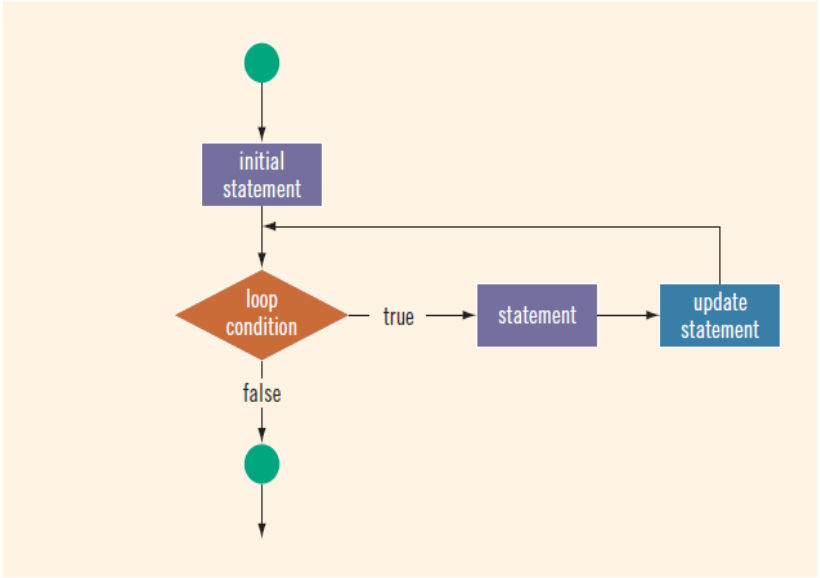
Translation between C and Assembly

- A loop in Assembly

```
      MOV R2,#255
ABC:MOV P1,R2
      DJNZ R2,ABC
```

- A for-loop in C

```
for (int z=255; z>0; z--)
 P1=z;
```



for loop

Section 7.1

Data Types and Time Delay in 8051 C

Re-examine C Data Type

- Understanding the data types of C can help programmers to write an efficient code.
- We will focus on (See Table7-1)
  - unsigned char, signed char
  - unsigned int, signed int,
  - Single bit: sbit (sfr) , bit (bit-addressable RAM)
  - Special function register: sfr
- C compiler will allocate a RAM space for your variables (char, int & bit).

Table 7-1. Some Widely Used Data Types for 8051 C

Data Type	Size in Bits	Data Range/Usage
unsiged char	8-bit	0 to 255
(signed) char	8-bit	–128 to +127
unsigned int	16-bit	0 to 65535
(signd) int	16-bit	–32,768 to +32,767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 - FFH only

Unsigned Char

- The most widely used data types for the 8051
- 8-bit data type
- The range of unsigned char: 0-255 (00-FFH)
- When do you use unsigned char?
  - To set counter value (Example 7-1)
  - The string of ASCII character (Example 7-2)
  - For toggling ports (Example 7-3)

Signed Char

- 8-bit data type
- 2’s complement representation
- The range of unsigned char: -128:+127 (00-FFH)
- When do you use signed char?
  - To present a given quantity such as temperature

Example 7-1 (unsigned char)

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;
}
```

Example 7-2 (unsigned char)

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];
}
```

Note: "**012345ABCD**" is stored in RAM and can be changed.

Example 7-3 (unsigned char)

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

Solution:

```
#include <reg51.h>
void main(void) {
  for ( ; ; ) {    //repeat forever
    P1=0x55;       //0x: in hex (binary)
    P1=0xAA;
  }
}
```

Example 7-4 (signed char)

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

Solution:

```
#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];
}
```

Integer

- 16-bit data type
  - The range of unsigned int: 0-65535
  - The range of signed int: -32,768-32,767
- Since the 8051 is an 8-bit microcontroller and the int data type takes two bytes of RAM, we must not used the int data type unless we have to.
- You should try to use unsigned char instead on int.

Example 7-5 (unsigned int, sbit)

Write an 8051 C program to toggle bit D0 of P1 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;
  }
}
```

Time Delay

- Three factors that can affect the accuracy of the time delay:
  - Crystal frequency of 8051 system
  - 8051 machine cycle timing
  - Compiler used for 8051 C

Example 7-6 (time delay)

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

Solution:

```
#include <reg51.h>
void main(void) {
  unsigned int x;
  for (;;) {
    P1=0x55;
    for (x=0;x<40000;x++); //time delay
    P1=0xAA;
    for (x=0;x<40000;x++);   }}
```

Example 7-7 (1/2)

Write an 8051 C program to toggle bits of P1 continuously with a 250 ms delay. **Solution:**

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void) {
  while(1) {          //repeat forever
    P1=0x55;
    MSDelay(250);  //time delay
    P1=0xAA;
    MSDelay(250)      }}
```

Assume the program is tested for the DS89C420 with XTML=11.0592MHz.

90ns X 1275 = 114750ns = 1ms

```
void MSDelay(unsigned int itime) {
  unsigned int i,j;
  for (i=0; i<itime; i++)
    for (j=0; j<1275; j++); // 1ms delay}
```

Section 7.2
I/O programming in 8051 C

- **Way to access SFR and single bit of SFR**
- **Access Bit-addressable RAM**

Access SFR

- Way to access SFR

Use name of SFR and reg51.h

```
#include <reg51.h>
P1=0x55;   //P1=55H
```

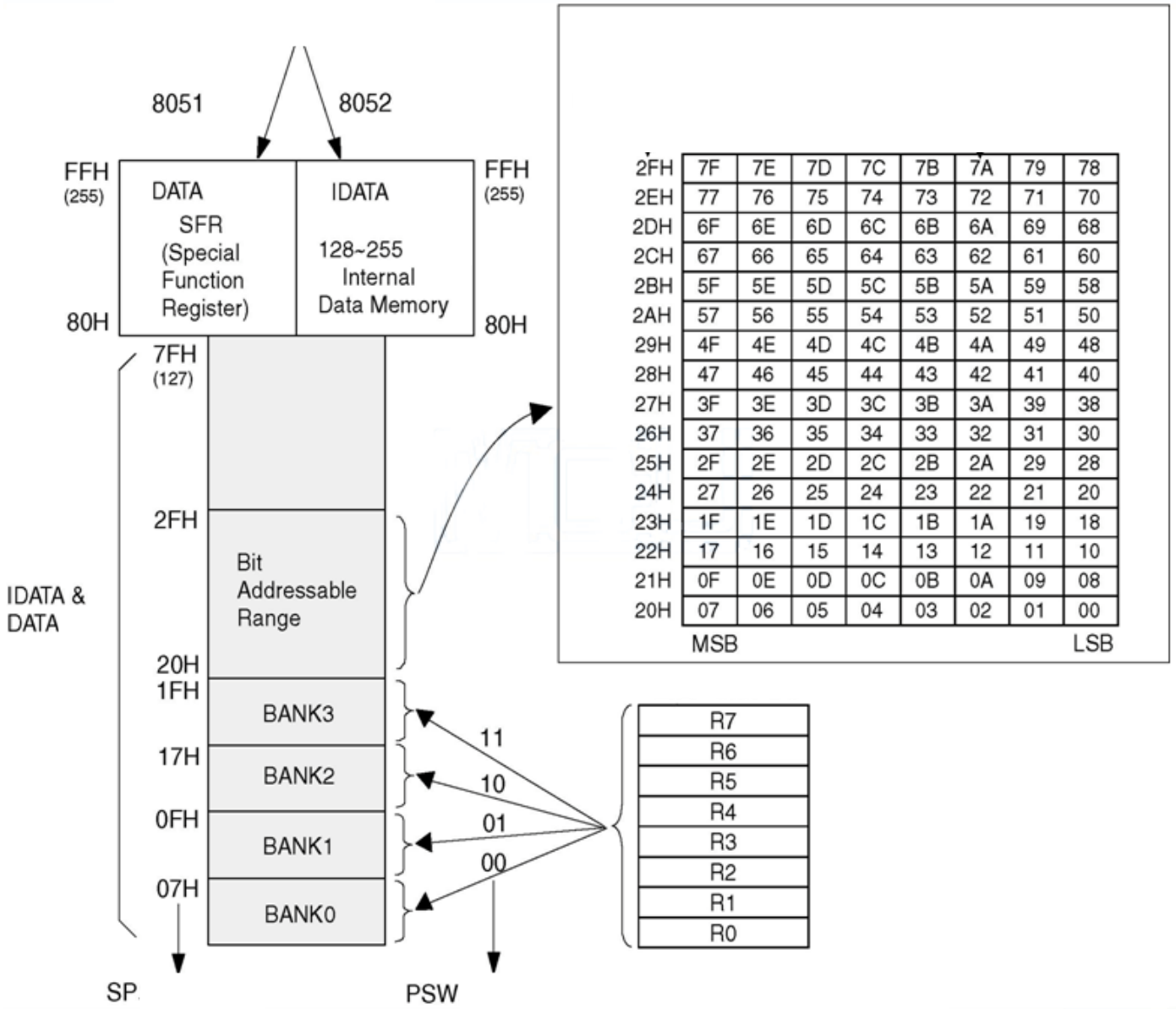
- reg51. h is necessary.
- See Example 7-1.

2. Use the sfr data type and declare by yourself

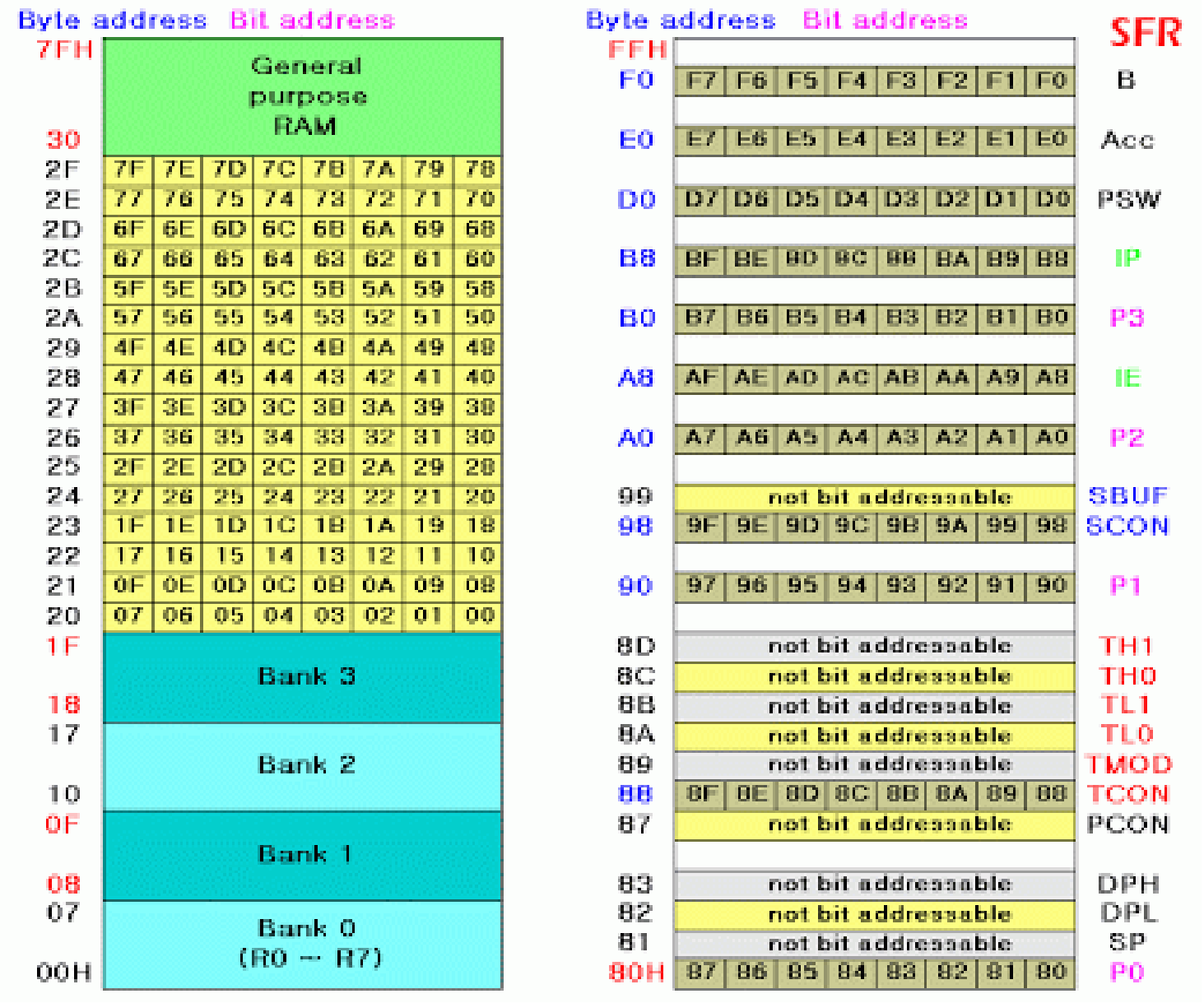
```
sfr P1 = 0x90;
P1=0x55;   //P1=55H
```

- reg51. h is not necessary.

8051 256Byte RAM



8015 256Byte RAM



Access SFR

- Way to access SFR

1. Use name of SFR and reg51.h

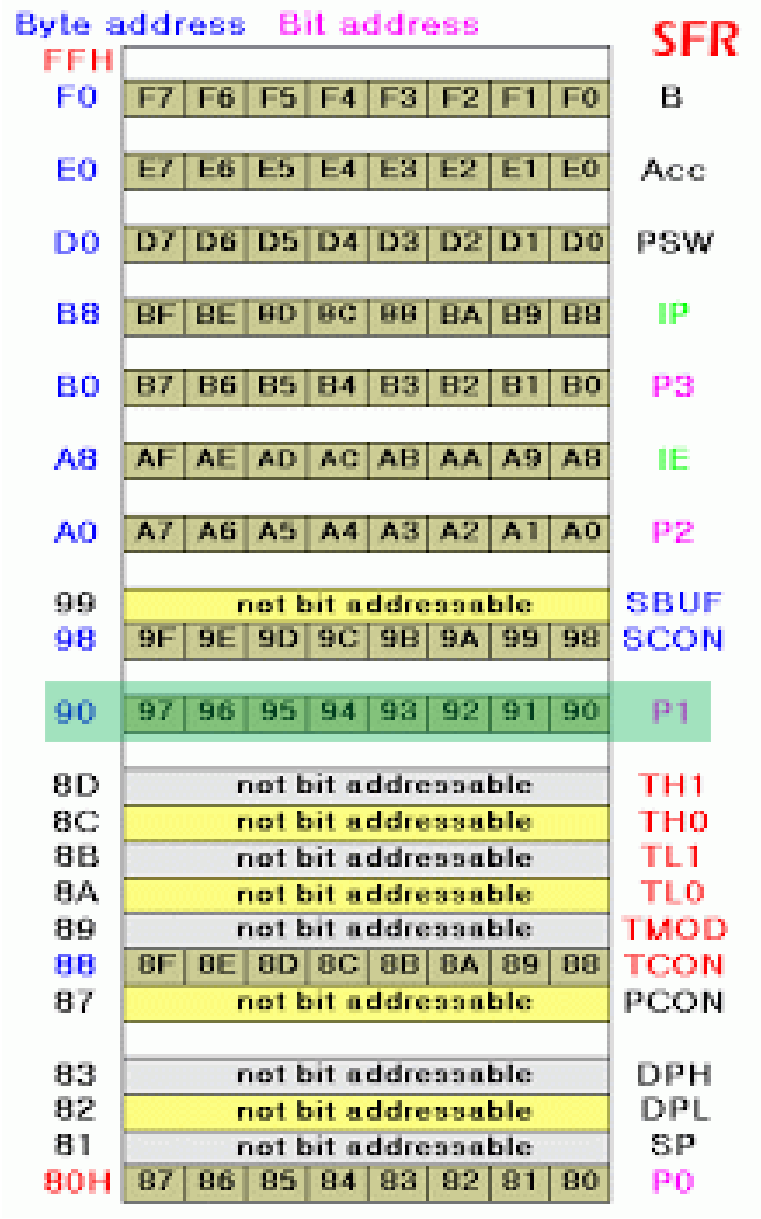
```
#include <reg51.h>
P1=0x55;   //P1=55H
```

   - reg51. h is necessary.
   - See Example 7-1.

2. Use the sfr data type and declare by yourself

```
sfr P1 = 0x90;
P1=0x55;   //P1=55H
```

   - reg51. h is not necessary.

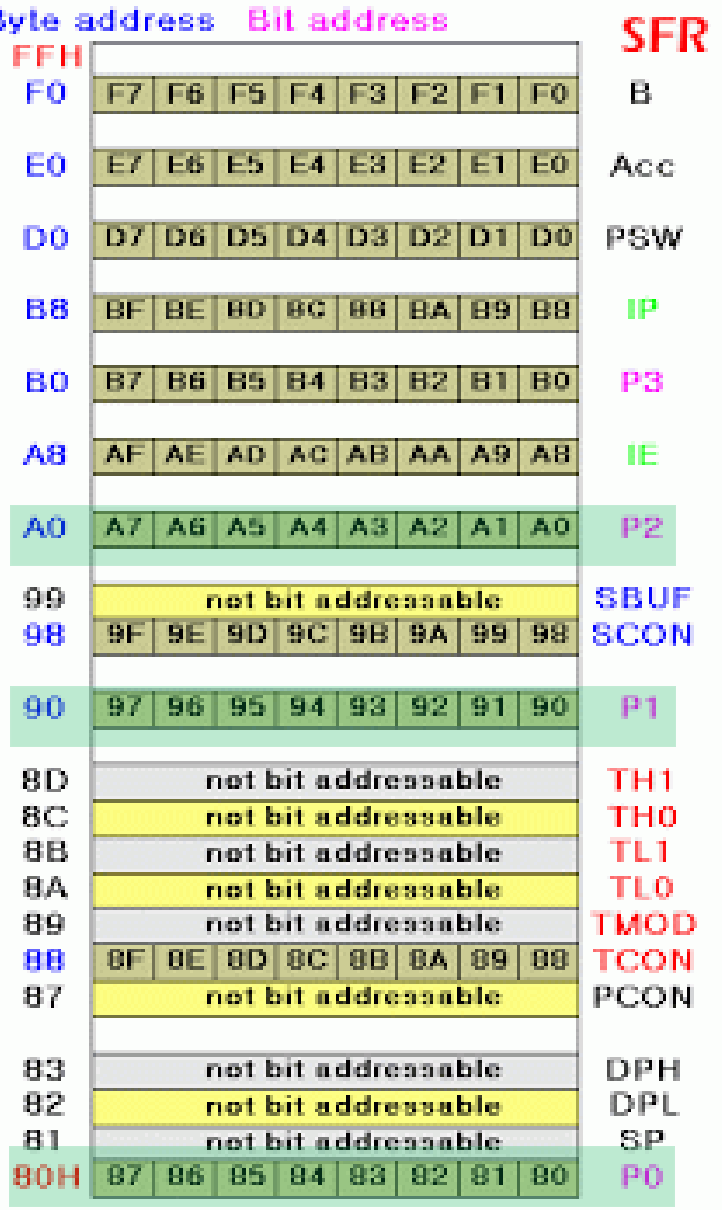


Example 7-16 Use the sfr data type andeclare by yourself

Write an 8051 C program to toggle all the bit of P0, and P2 continuously with a 250 ms time dealy.

Solution (a):

```
sfr P0 = 0x80;  // declare by yourself
sfr P1 = 0x90;  // declare by yourself
sfr P2 = 0xA0;  // declare by yourself
void MSDelay(unsigned int);
 // MSDelay is as same as one in
 // Example7-1. Ignored it here.
void main(void) {
 unsigned int z;
 while(1) {      //repeat forever
  P0=0x55;
  P1=0x55;
  P2=0x55;
  MSDelay(250);  //time delay
  P0=0xAA;
  P1=0xAA;
  P2=0xAA;
  MSDelay(250);  //time delay
} }
```

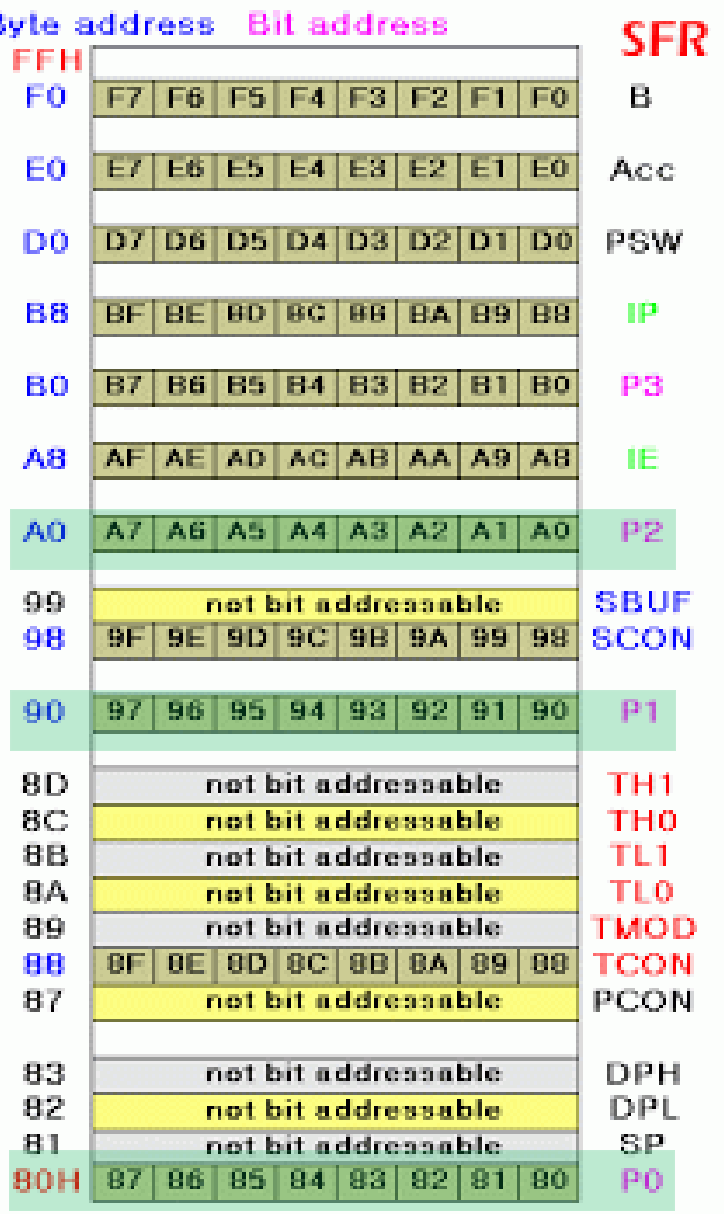


Example 7-16 Use name of SFR and reg51.h

Write an 8051 C program to toggle all the bit of P0, P1 and P2 continuously with a 250 ms time dealy.

Solution (a):

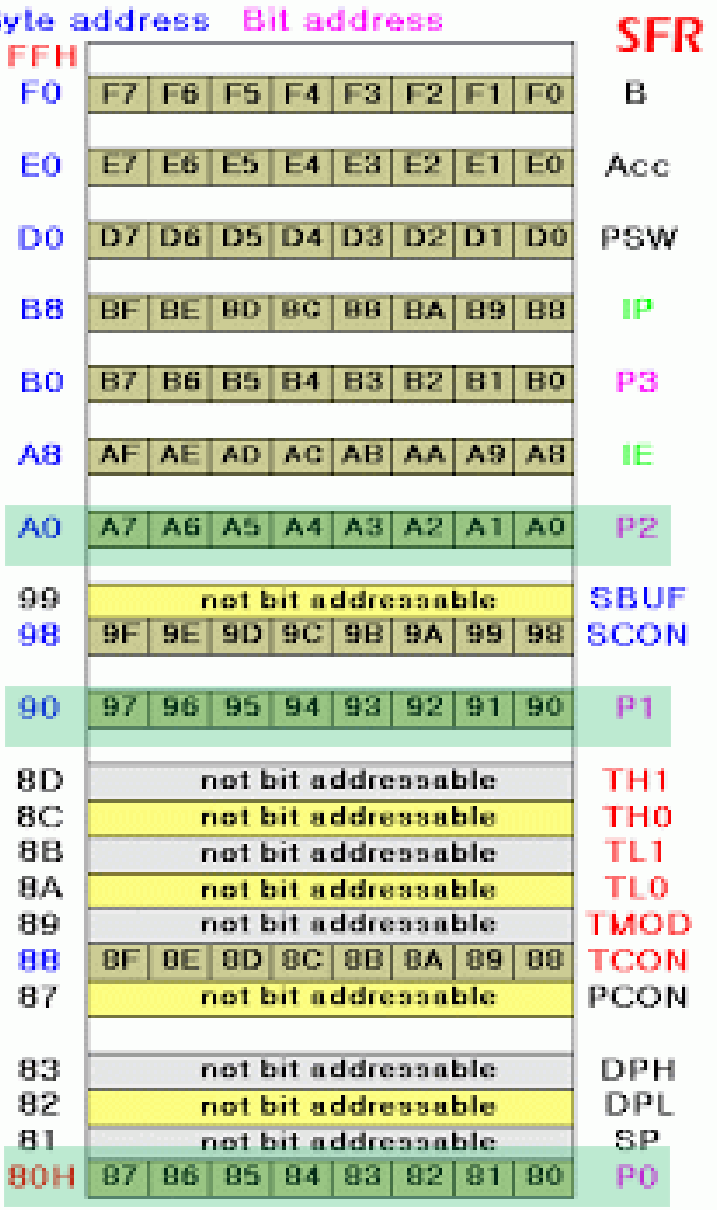
```
sfr P0 = 0x80;  // declare by yourself
sfr P1 = 0x90;  // declare by yourself
sfr P2 = 0xA0;  // declare by yourself
void MSDelay(unsigned int);
 // MSDelay is as same as one in
 // Example7-1. Ignored it here.
void main(void) {
 unsigned int z;
 while(1) {      //repeat forever
  P0=0x55;
  P1=0x55;
  P2=0x55;
  MSDelay(250);  //time delay
  P0=0xAA;
  P1=0xAA;
  P2=0xAA;
  MSDelay(250);  //time delay
} }
```



Example 7-16 Use name of SFR and reg51.h

Write an 8051 C program to toggle all the bit of P0, P1 and P2 continuously with a 250 ms time dealy.

Solution (a):




```
#include <reg51.h>

void MSDelay(unsigned int);
 // MSDelay is as same as one in
 // Example7-1. Ignored it here.
void main(void) {
 unsigned int z;
 while(1) {      //repeat forever
  P0=0x55;
  P1=0x55;
  P2=0x55;
  MSDelay(250);  //time delay
  P0=0xAA;
  P1=0xAA;
  P2=0xAA;
  MSDelay(250);  //time delay
} }
```

```
sfr P0 = 0x80;  // declare by yourself
sfr P1 = 0x90;  // declare by yourself
sfr P2 = 0xA0;  // declare by yourself
```

=

#include ?<reg51.h>

```
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software,
Inc.
All rights reserved.
------------------------------------------------------------------
---------*/

#ifndef __REG51_H__
#define __REG51_H__

/*  BYTE Register  */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;
```

Access Single Bit of SFR

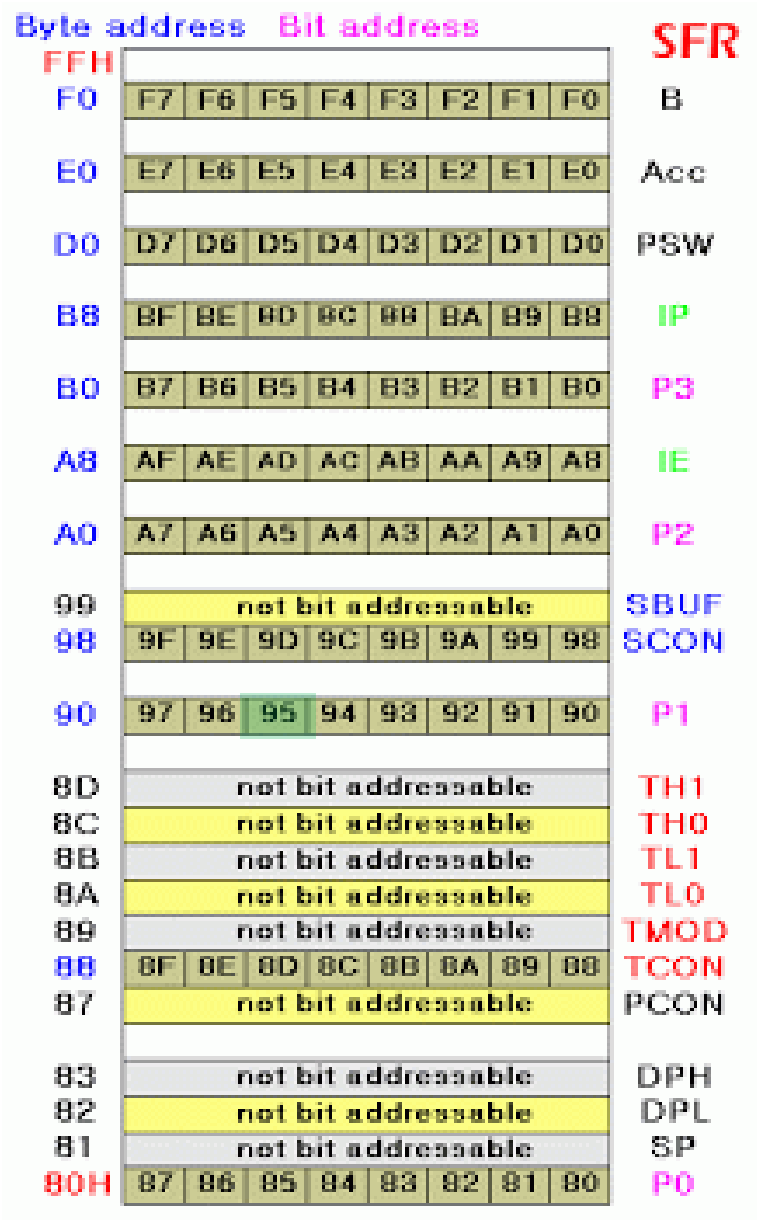
- Way to access a single bit of SFR
  - Use sbit and name of SFR

```
#include <reg51.h>
sbit MYBIT = P1^5;   //D5 of P1
```

    - See Example 7-5.
  - Use sbit to declare the bit of SFR and declare by yourself

```
sbit MYBIT = 0x95;   //D5 of P1
```

    - reg51. h is not necessary.
    - See Example 7-17.



Example 7-17 (sbit)

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

Solution (a):

```
sbit MYBIT = 0x95;  // P1^5
void main(void) {
  unsigned int z;
  for (z=0;z<50000;z++) {
    MYBIT=1;
    MYBIT=0;
} }
```

This program is similar to Example 7-5.

Example 7-17 (sbit)

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

Solution (a):

```
sbit MYBIT = 0x95;   // P1^5
#include <reg51.h>
sbit MYBIT=P1^0;
void main(void) {
  unsigned int z;
  for (z=0;z<50000;z++) {
    MYBIT=1;
    MYBIT=0;
} }
```

This program is similar to Example 7-5.

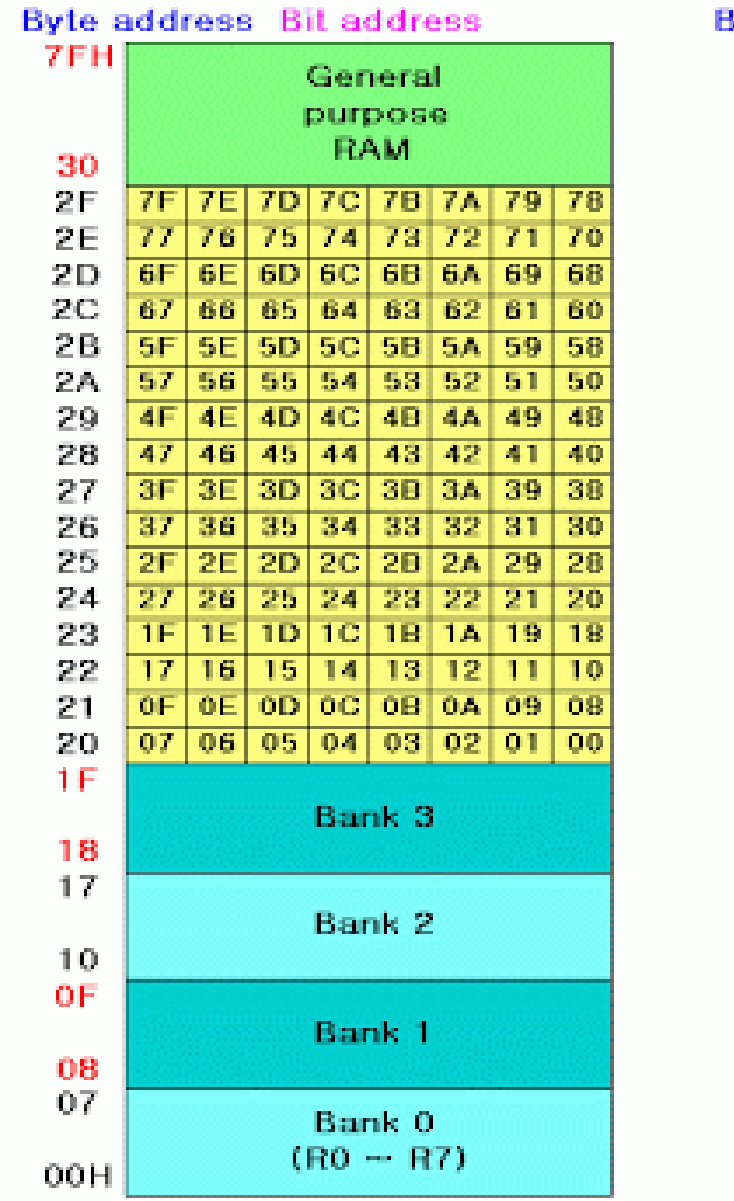
Access Bit-addressable RAM

- You can use bit to access one bit of bit-addressable section of the data RAM space 20H-2FH.

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

bit membit; //C compiler assign a RAM space for mybit
membit = inbit;  //Read P1^0 to RAM
```

  - See Example 7-18.



Example 7-18 (bit)

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;
void main(void) {
  while(1) {          //repeat forever
    membit = inbit;
    outbit = membit    }}
```

Section 7.3

Logic operations in 8051 C

Bit-wise Operations in C

- AND &

  `0x35 & 0x0F = 0x05`

- OR |

  `0x04 | 0x68 = 0x6C`

- Exclusive-OR ^

  `0x54 ^ 0x78 = 0x2C`

- Inverter ~

  `~0x55 = 0xAA`

NOT 0101 0101
0000 0101

0011 0101
AND 0000 1111
0000 0101

0000 0100
OR 0110 1000
0110 1100

0101 0100
XOR 0111 1000
0010 1100

Bit-wise Shift Operation in C

- Shift Right >>

  `0x9A >> 3 = 0x13`

  shift right 3 times

  1001 1010 → 0001
  0011

  `0x77 >> 4 = 0x07`

  shift right 4 times

  0111 0111 → 0000
  0111

- Shift Left <<

  `0x96 << 4 = 0x60`

  shift left 4 times

  1001 0110 ← 0110
  0000

Table 7-3. Bit-wise Logic Operations for C

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

Example 7-19

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

Solution:

```
#include <reg51.h>
void main(void) {
  P0=0x35 & 0x0F;
  P0=0x04 | 0x68;
  P0=0x54 ^ 0x78;
  P0=~0x35;
  P0=0x9A >> 3;
  P0=0x77 >> 4;
  P0=0x96 << 4; }
```

Section 7.4

Data conversion programs in 8051 C

Data Conversion

- Packed BCD to ASCII conversion
  - See Example 7-24
- ASCII to packed BCD conversion
  - See Example 7-25
- Checksum byte in ROM
  - See Example 7-26, 27, 28 (using +)
- Binary to decimal and ASCII conversion in C
  - See Example 7-29(using /, %)

Table 7-4. ASCII Code for Digits 0-9

Key	ASCII (hex)	Binary	BCD (unpacked)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

Packed BCD to ASCII conversion
Example 7-24

Write an 8051 C program to convert packed BCD 0x29 to ASCII and display the bytes on P1 and P2.

Solution:

```
#include <reg51.h>
void main(void)
{unsigned char x,y,z;
unsigned char mybyte=0x29;
x=mybyte&0x0F;
P1=x|0x30;
y=mybyte&0xF0;
y=y>>4;
P2=y|0x30;}
```

ASCII to packed BCD conversion

See Example 7-25

Write an 8051 C program to convert ASCII digits of ‘4’ and ‘7’ to packed BCD and display them on P1.

Solution:

```
#include <reg51.h>
void main(void)
{ unsigned char bcdbyte;
unsigned char w=‘4’;
unsigned char z=‘7’;
w=w&0x0F;
w=w<<4;
z=z&0x0F;
bcdbyte=w|z;
P1=bcdbyte;}
```

Binary to decimal and ASCII conversion in C

See Example 7-29(using /, %)

Write an 8051 C program to convert 11111101 (FD hex) to decimaland display the digits on P0, P1 and P2. **Solution:**

```
#include <reg51.h>
void main(void){
unsigned char x,binbyte,d1,d2,d3;
binbyte=0xFD;
x=binbyte/10;
d1=binbyte%10;
d2=x%10;
d3=x/10;
P0=d1;
P1=d2;
P2=d3;}
```

Section 7.5

Accessing code ROM space in 8051 C

To Store Data

- We have three spaces to store data:
  1. the 128 bytes RAM space with address range 00-7FH
     - If you declare variables (ex: char) to store data, C compiler will allocate a RAM space for these variable.
  2. Use code space
     - External code memory (64K) + on-chip ROM (64K)
     - Data is embedded to code or is separated as a data section
  3. External data memory for data
     - RAM or ROM
     - see Chapter 14

Data Stored in RAM

- Where are the variables assigned by C compiler?
- 8051 C compiler allocates RAM locations for variables in the following order:
  - Bank 0: addresses 0-7
  - Individual variables: addresses 08H and beyond
  - Array elements: addresses right after variables
    - Array size is limited in 128 bytes RAM.
  - Stack: address right after array elements
- Check your simulation tool

8052 RAM Space

- 8052 has 256 bytes RAM.
- Based on 8052 architecture, you should
  - Use the reg52.h header file.
  - Choose the 8052 option when compiling the program.

Using ROM to Store Data

- To make C compiler use the code space (on-chip ROM) instead of RAM space, we can put the keyword "code" in front of the variable declaration.

  unsigned char mydata[] = "HELLO"

  - HELLO is saved in RAM.

  code unsigned char mydata[] = "HELLO"

  - HELLO is saved in ROM.

- See Example 7-33

Example 7-33 (1/3)

Compare and contrast the following programs and discuss the advantages and disadvantages of each one.

Solution (a):

```
#include <reg51.h>
void main(void) {
   P1="H";     P1="E";
   P1="L";     P1="L";    P1="O";
}
```

Data is embedded into code.

Simple, short, not flexible.

Example 7-33 (2/3)

Solution (b):

```
#include <reg51.h>
void main(void) {
  unsigned char mydata[]="HELLO";
  unsigned char z;
  for (z=0; z<5; z++)
    P1 = mydata[z];
}
```

Data is stored in RAM and does not occupied ROM.

Example 7-33 (3/3)

Solution (c):

```
#include <reg51.h>
void main(void) {
   code unsigned char mydata[]="HELLO";
   unsigned char z;
   for (z=0; z<5; z++)
      P1 = mydata[z];
}
```

Data is stored in ROM. However, data and code are separate.

Section 7.6

Data serialization using 8051 C

Serializing Data

- Two ways to transfer a byte of data:
  1. Using the serial port.
     - For PC. See Chapter 10.
  2. To transfer data one bit a time and control the sequence of data and spaces in between them.
     - For LCD, ADC, ROM
     - Example 7-34

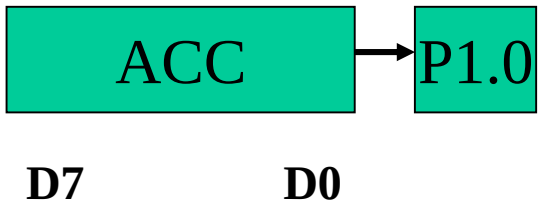
Example 7-34

Write a C program to send out the value 44H serially one bit at a time via P1.0. The LSB should go out first.

Solution:

```
#include <reg51.h>
sbit P1b0 = P1^0;
sbit regALSB = ACC^0;

void main(void) {
   unsigned char conbyte = 0x44;
   unsigned char x;
   ACC = conbyte;
   for (x=0; x<8; x++) {
      P1b0 = regALSB;
      ACC = ACC >> 1; }
}
```



Thank you