二进制数	十六进制数
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F
8 4 2 1	

课前练习分析

先将十六进制数转换为二进制数

```
#define  BIT5  0x20
#define  BIT2  0x04
char dataA=0xb6, dataB=0xd9, dataC=0x5a;
   dataA &=    BIT5;        //第一条
   dataB |= BIT5;           //第二条
   dataC ^= BIT5;          //第三条
```

BIT5 = 0x20 = 0010 0000B
BIT2 = 0x04 = 0000 0100B
dataA = 0xb6 = 1011 0110 B
dataB = 0xd9 = 1101 1001B
dataC = 0x5a = 0101 1010 B

十六进制的数不区分大小写

dataA = 0xb6 = 0xB6
dataB = 0xd9 = 0xD9
dataC = 0x5a =0x5A

注意：关于位的编号

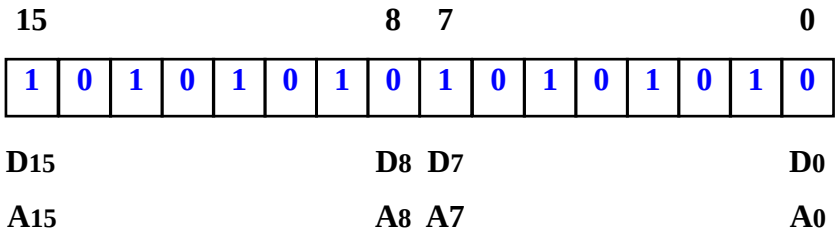
为便于描述，对二进制数中的各位进行编号。

从低位开始，从右到左依次为 0、1、2．．．

	7	6	5	4	3	2	1	0	← 编号
字节	1	0	1	0	0	0	1	0	
	D7	D6	D5	D4	D3	D2	D1	D0	描述数据Data
	A7	A6	A5	A4	A3	A2	A1	A0	描述地址Address

注意：是从 0 开始编号！！！

字的编号为15 ~ 0



双字的编号依此类推，为31 ~ 0

dataA &= ~BIT5 //第一条语句 (非完后与)

dataA = dataA & (~BIT5)

= 0xB6 & (~0x20)

= 1011 0110 &(~0010 0000)

= 1011 0110 & 1101 1111

= 1001 0110

= 0x 96

使D5位为0
其他位保持不变

1011 0110
& 1101 1111
1001 0110

注意：一个字节的8位从高位到低位编号依次为D7~D0，
一个字的16位编号依次是D15~D0
一个双字的32位编号依次是D31~D0

dataB |= BIT5

dataB = dataB | BIT5

= 0xd9 | 0x20

= 1101 1001| 0010 0000

= 1111 1001

= 0x F9

```
  1101 1001
| 0010 0000
-----------
  1111 1001
```

使D5位为1
其他位保持不变

dataC ^= BIT5

dataC = dataC ^ BIT5

= 0x5a ^ 0x20

= 0101 1010 ^ 0010 0000

= 0111 1010

= 0x 7a

```
  0101 1010
^ 0010 0000
-----------
  0111 1010
```

使D5位求反
其他位保持不变

保持其他位的不变，也很重要

```
#define BIT5 0x20
#define BIT2 0x04
char dataA=0xb6, dataB=0xd9,
dataC=0x5a;
  dataA &=  BIT5;      //第一条
  dataB |= BIT5;       //第二条
  dataC ^= BIT5;       //第三条
```

D5

BIT5 = 0x20 = 0010 0000B

D2

BIT2 = 0x04 = 0000 0100B

回答：2) 宏定义 BIT5、BIT2的特点？

BIT5和BIT2是符号定义的常数，BIT5只有D5位为1，其余位为0

BIT2只有D2位为1，其余位为0

3) 三条语句的本质作用？

实现对指定的某一位 置0、置1、求反 的功能，并保持其他位的值不变

4) 把语句中的BIT5改为BIT2，执行的结果有何不同？

原来是对变量中的D5位进行位置0、置1、求反

改为BIT2后，是对变量中的D2位进行位置0、置1、求反

5) 把BIT5改为(BIT5+BIT2)呢？

完成对变量中的D5和D2两位均进行位置0、置1、求反

可以用类似的方法实现对变量中的多位进行位置0、置1、求反

msp430g2553.h 头文件中

```
#define BIT0 (0x0001)
#define BIT1 (0x0002)
#define BIT2 (0x0004)
#define BIT3 (0x0008)
#define BIT4 (0x0010)
#define BIT5 (0x0020)
#define BIT6 (0x0040)
#define BIT7 (0x0080)
#define BIT8 (0x0100)
#define BIT9 (0x0200)
#define BITA (0x0400)
#define BITB (0x0800)
#define BITC (0x1000)
#define BITD (0x2000)
#define BITE (0x4000)
#define BITF (0x8000)
```

对字中各位的值用符号定义，
使只有指定的位为1，其他位均为0，方便编程

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0004
					…	…	…	…	…							
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0020
					…	…	…	…	…							
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000

使用符号定义的位值，可提高程序的可读性。

```
dataA &= ~BIT5;   //置dataA的D5位为0
dataA |= BIT2;    //置dataA的D2位为1
dataA^= (BIT5+BIT2); //置dataA的D5和D2两位求反
```

这样的符号定义，在编程中常用到

如何编程检测变量中的某位是0，还是1？？

if ((dataA & BIT2)== 0) ,可以判断dataA中的哪位为0？

```
          ****  *D2**   dataA
      &   0000  0 1  00   BIT2
  如果D2=0  0000  0 0 00
    D2=1    0000  0 1 00
```

结论

所以，反推：如果(dataA & BIT2) == 0　　D2 = 0
(dataA & BIT2) = BIT2 != 0　　D2 = 1 != 0

6) if ((dataA & BIT2)== 0) ,可以判断dataA中的D2位为0

7) if ((dataA & BIT2)!= 0) ,可以判断dataA中的D2位为1

8) 如果 写成if ((dataA & BIT2)==1)

因 (dataA & BIT2) 要么等于0，要么= BIT2 = 0x02 ，没有等于1的情况

所以从 (dataA & BIT2)==1 ，判断不了 dataA中的D2位 是0，还是1

9) 如果 写成 if (dataA & BIT2 ==0)， 不能判断 dataA中的D2位

因逻辑关系符 == 的优先级高于 逻辑运算符 &

所以先 判断 BIT2==1 成立否，再由该结果和dataA 做与运算

而 BIT2==1 是不成立的，是false

导致 dataA & BIT2 ==0 不成立，

这样 if (dataA & BIT2 ==0) 语言后面的代码不会被执行

10) 如果 把 if ((dataA & BIT2) ==0) 语句中的BIT2该为 BIT5

可以判断dataA 的D5位是否为0

依次类推，掌握C语言对一个变量中的某一位置0、置1、求反，和测试的编程方法。

请大家务必消化上面的内容，

单片机实验的编程大量用到这样的语句，操作底层的硬件。

二. 完成下面二进制与十六进制的转换

10111010B = BA H　　11010111B = D7 H

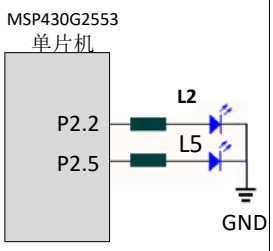
5EH = 01011111 B　　C3H= 11000011 B

三. 十进制、二进制和十六进制的运算

255	1111 1111 B	FF H
+ 109	+ 0110 1101 B	+ 6D H
364	10110 1100 B	16C H

现在我们来看一下在安装CCS、搭建430单片机实验平台时，使用的测试程序test_g2553.c

```
#include "msp430.h"
unsigned int i;          //定义全局变量
int main ( void )
{  //unsigned int i;                  //定义局部变量
   //register unsigned int i;         //定义寄存器型局部变量
   WDTCTL = WDTPW + WDTHOLD;  //关闭看门狗
   P2SEL &=~(BIT2+BIT5);              //设置引脚P2.2和P2.5为基本输入输出功能
   P2SEL2 &=~(BIT2+BIT5);
   P2OUT |=BIT2+BIT5;                 //设置引脚P2.2和P2.5输出的初值为1
   P2DIR |=BIT2+BIT5;                 //设置端口P2.2和P2.5为输出方向
   for (;;)                    //主循环
   {  P2OUT ^=(BIT2+BIT5);            //将P2.2和P2.5的值取反后输出
      for (i=0xFFFF; i>0; i--);       //延时
   };
}
```



稍读一下程序，就会发现，程序主要就是用非、与、或、异或等逻辑操作对变量P2SEL、P2SEL2、P2OUT、P2DIR 的利用宏定义BIT2、BIT5对变量中的D2位、D5位进行了**置0、置1、求反的操作**，同时不影响变量中的其他位。这些变量和宏定义在msp430.h文件定义，课堂上会对这些变量和宏定义做介绍。

程序并不复杂，但是它却可以控制与单片机连接的两个LED的闪烁，如果稍微改写一下程序，就可以实现各种不同的LED亮灭变化。

这是为什么呢？这是因为变量**P2SEL、P2SEL2、P2OUT、P2DIR**是与单片机底层硬件有关系的变量，是特殊的变量。它们与一般的变量不同，如程序中的变量i就是一般的变量。计硬课程中，会重点讲解这些特殊变量与单片机底层硬件有什么关系，以及如何通过对这些变量的操作，实现对底层硬件的控制。

“计算机硬件技术基础”课程，就会介绍单片机里有哪些这样的特殊变量，它们的作用是什么？，然后我们通过编写程序，设置和改变这些变量中某些位的值，就是练习中给的这种置0、置1方式，实现对与单片机连接的那些设备的控制，比如大家在搭建单片机实验平台时，控制的两个发光二级管闪烁。