EMBEDDED SYSTEMS CMPE-453

Department of Computer Engineering

Bitwise operations

Bitwise Logic Operators

		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	

SOME EXAMPLES

1. 0x35 & 0x0F = 0x05 /* ANDing */
2. 0x04 | 0x68 = 0x6C /* ORing */
3. 0x54 ^ 0x78 = 0x2C /* XORing */
4. ~0x55 = 0xAA /* Inverting 55H */

EX-1

Write an AVR C program to toggle only bit 4 of Port B continuously without disturbing the rest of the pins of Port B.

Solution:

```
#include <avr/io.h>                    //standard AVR header

int main(void)
{
   DDRB = 0xFF;                        //PORTB is output

   while(1)
   {
      PORTB = PORTB | 0b00010000;      //set bit 4 (5th bit) of PORTB
      PORTB = PORTB & 0b11101111;      //clear bit 4 (5th bit) of PORTB
   }

   return 0;
}
```

?

```
PORTB.4 = 0;

PORTB.4 = 1;
```

EX-2

A door sensor is connected to bit 1 of Port B, and an LED is connected to bit 7 of Port C. Write an AVR C program to monitor the door sensor and, when it opens, turn on the LED.

Solution:

```
#include <avr/io.h>                    //standard AVR header

int main(void)
{
  DDRB = DDRB & 0b11111101;            //pin 1 of Port B is input
  DDRC = DDRC | 0b10000000;            //pin 7 of Port C is output

  while(1)
  {
    if (PINB & 0b00000010)             //check pin 1 (2nd pin) of PINB
      PORTC = PORTC | 0b10000000;      //set pin 7 (8th pin) of PORTC
    else
      PORTC = PORTC & 0b01111111;      //clear pin 7 (8th pin) of PORTC
  }
  return 0;
}
```

EX-3

Write an AVR C program to monitor bit 7 of Port B. If it is 1, make bit 4 of Port B input; otherwise, change pin 4 of Port B to output.

Solution:

```
#include <avr/io.h>                          //standard AVR header

int main(void)
{
  DDRB = DDRB & 0b01111111;                  //bit 7 of Port B is input

  while (1)
  {
    if(PINB & 10000000)
      DDRB = DDRB & 0b11101111;              //bit 4 of Port B is input
    else
      DDRB = DDRB | 0b00010000;              //bit 4 of Port B is output
  }

  return 0;
}
```

EX-4

Write an AVR C program to get the status of bit 5 of Port B and send it to bit 7 of port C continuously.

Solution:

```
#include <avr/io.h>                    //standard AVR header

int main(void)
{
  DDRB = DDRB & 0b11011111;            //bit 5 of Port B is input
  DDRC = DDRC | 0b10000000;            //bit 7 of Port C is output

  while (1)
  {
    if(PINB & 0b00100000 )
      PORTC = PORTC | 0b10000000;      //set bit 7 of Port C to 1
    else
      PORTC = PORTC & 0b01111111;      //clear bit 7 of Port C to 0
  }
  return 0;
}
```

EX-5

Write an AVR C program to toggle all the pins of Port B continuously.

(a) Use the inverting operator.

(b) Use the EX-OR operator.

(a)

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
      DDRB = 0xFF;                     //Port B is output
      PORTB = 0xAA;
      while (1)
            PORTB = ~ PORTB;           //toggle PORTB
      return 0;
}
```

(b)

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
      DDRB = 0xFF;                     //Port B is output
      PORTB = 0xAA;
      while (1)
            PORTB = PORTB ^ 0xFF;
      return 0;
}
```

Compound assignment Operators

Table 7-3: Compound Assignment Operator in C

Operation	Abbreviated Expression	Equal C Expression
And assignment	a &= b	a = a & b
OR assignment	a |= b	a = a | b

EX-6

Write an AVR C program to monitor bit 5 of Port B. If it is 1, make bit 4 of Port B input; otherwise, change pin 4 of Port B to output.

```
#include <avr/io.h>              //standard AVR header
int main(void)
{
   DDRB = DDRB & 0b11011111;  //bit 5 of Port B is input
   while (1)
   {
      if(PINB & 0b00100000)
         DDRB &= 0b11101111;     //bit 4 of Port B is input
      else
         DDRB |= 0b00010000;     //bit 4 of Port B is output
   }
   return 0;
}
```

EX-7

Write an AVR C program to get the status of bit 5 of Port B and send it to bit 7 of port C continuously.

```
#include <avr/io.h>                //standard AVR header
int main(void)
{
   DDRB &= 0b11011111;             //bit 5 of Port B is input
   DDRC |= 0b10000000;             //bit 7 of Port C is output

   while (1)
   {
      if(PINB & 0b00100000)
         PORTC |=  0b10000000;     //set bit 7 of Port C to 1
      else
         PORTC &= 0b01111111;      //clear bit 7 of Port C to 0
   }
   return 0;
}
```

Bitwise Shift Operation

Table 7-4: Bit-wise Shift Operators for C

Operation	Symbol	Format of Shift Operation
Shift right	>>	data >> number of bits to be shifted right
Shift left	<<	data << number of bits to be shifted left

The following shows some examples of shift operators in C:

1. 0b00010000 >> 3 = 0b00000010 /* shifting right 3 times */
2. 0b00010000 << 3 = 0b10000000 /* shifting left 3 times */
3. 1 << 3 = 0b00001000 /* shifting left 3 times */

EX-8

Write code to generate the following numbers:

(a) A number that has only a one in position D7
(b) A number that has only a one in position D2
(c) A number that has only a one in position D4
(d) A number that has only a zero in position D5
(e) A number that has only a zero in position D3
(f) A number that has only a zero in position D1

Solution:

(a) `(1<<7)`
(b) `(1<<2)`
(c) `(1<<4)`
(d) `~(1<<5)`
(e) `~(1<<3)`
(f) `~(1<<1)`

EX-9

Write an AVR C program to get the status of bit 5 of Port B and send it to bit 7 of port C continuously.

Solution:

```
#include <avr/io.h>                    //standard AVR header

int main(void)
{
  DDRB = DDRB & ~(1<<5);               //bit 5 of Port B is input
  DDRC = DDRC | (1<<7);                //bit 7 of Port C is output

  while (1)
  {
    if(PINB & (1<<5) )
      PORTC = PORTC | (1<<7);          //set bit 7 of Port C to 1
    else
      PORTC = PORTC & ~(1<<7);         //clear bit 7 of Port C to 0
  }
  return 0;
}
```

LED FLOW FROM L2R AND R2L

```
//Include files
#include <avr/io.h>
#include <util/delay.h>

//macros
#define DELAY 400

//method definations
void blinkLEDs(uint8_t byte)
        { PORTD=byte;
        _delay_ms(DELAY);
        }
//main method
int main (void)
        {
        //Initialization
        DDRD = 0xFF;
        //Event loop
```

```
while(1) {

        blinkLEDs(0b10000000);
        blinkLEDs(0b01000000);
        blinkLEDs(0b00100000);
        blinkLEDs(0b00010000);
        blinkLEDs(0b00001000);
        blinkLEDs(0b00000100);
        blinkLEDs(0b00000010);
        blinkLEDs(0b00000001);

        blinkLEDs(0b00000010);
        blinkLEDs(0b00000100);
        blinkLEDs(0b00001000);
        blinkLEDs(0b00010000);
        blinkLEDs(0b00100000);
        blinkLEDs(0b01000000);
        blinkLEDs(0b10000000);

        }

return 0;

}
```

Too much code
Difficult to read
Prone to errors

How to improve?

Bit shifting
To set the nth bit to 1,
shift 1 to left n times.

IMPROVED EXTENDED VERSION

```
//Include files
#include <avr/io.h>
#include <util/delay.h>

//macros
#define DELAY 400

//method definations
void blinkLEDs(uint8_t byte)
        { PORTD=byte;
        _delay_ms(DELAY);
        }
//main method
int main (void)
        {
        //Initialization
        DDRD = 0xFF;
        uint8_t i=0;
        //Event loop
while(1)
        {
        //right-to-left
        for (i=0;i<7;i++)
                {
                blinkLEDs(1 << i);
                }
        //left-to-right
        for (;i>0;i--)
                {
                blinkLEDs(1<<i);
                }
        }
        return 0;
}
```

IMPROVED EXTENDED VERSION

```
//Include files
#include <avr/io.h>
#include <util/delay.h>

//macros
#define DELAY 400

//method definations
void blinkLEDs(uint8_t byte)
        { PORTD=byte;
        _delay_ms(DELAY);
        }
//main method
int main (void)
        {
        //Initialization
        DDRD = 0xFF;
        uint8_t i=0;
        //Event loop
while(1)
        {
        //left to right
        for (i=7;i>0;i--)
                {
                blinkLEDs(1 << i);
                }
        //right to left
        for (i=0;i<7;i++)
                {
                blinkLEDs(1<<i);
                }
        }
        return 0;
}
```