# CAB202 Week 6,7,8,9,10,11,12

**Week 6 Notes**

* \*Decimal Representation of Numeric Data\*
* Due to only being able to use four digits the carry hasn’t been stored anywhere. This is overflow.
* Can happen during calculations.
* Programmer must take measurements to ensure it does not happen
* in base 10 we use something called base compliment
* 9's compliment
* 1)Subtract each digit from 9.
* 2)Add 1 to the resulting 4 digit number
* to verify A+ (-A) = 0
* A>>1234 >> 8765 >> +0001 >> 8766 >> -A
* Multiplication by power of 10
* A = 0035, 10xA = 0350

100xA= 3500

1000xA= 5000

* Left shift >> Multiplying by 10 OR (0001 becomes 0010 )
* Division by power of 10
* A= 1234
* A/(10^0)= A/1 = 1234
* A/(10^1)= A/10 = 0123
* A/(10^1)= A/100 = 0012
* Right shift.
* \*Binary Representation of Numerical Data\*
* 4-digit decimal 8-bit Binary
* Min|0 base2 0 base2
* Max| 9999 11111111

BIT >> Binary digit

* Know how to represent the decimal numbers in Binary

The value gets doubled every time the 1 gets let shifted

We use this because of binary

* 2's compliment: 1's compliment + 1
* That is 1000 0000 - 0000 1100 = 1111 0011
* then plus 0000 0001

75(base10) – 12(base10)

75 = 64 + 11

=64 + 8 + 2 + 1

12 = 00001100(base2)

1s comp plus 1 = 11110011 + 00000001 = 11110100

>>-12

Binary of 75 + (-12) => 00111111(base2)

* Multiplication by power of 2
* 7x(2^0) = 111x 1 = 111
* 7x(2^1) = 111x 2 = 1110 i.e 111 + 111 = 1110

Left shift corresponds to multiplication of 2

(7<<0) == 0b00000111

(7<<1) == 0b00001110

(7<<2) == 0b00011100

(7<<3) == 0b00111000

(7<<4) == 0b01110000

((7<<3)>>0)

((7<<3)>>1)

((7<<3)>>2)

((7<<3)>>3)

((7<<3)>>4)

|  |  |
| --- | --- |
| b | 1<<b |
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |

Right Shift corresponds to division of 2

* \*Bitwise operations on bytes\*

Compile function

function compile () {

Gcc $1.c -o $1 -Wall -Werror -std=gnu99

}

* Bitwise operator vs Logical operator

BITWISE;

& identifies positions where bit values are both 1

| identifies where either bit is 1

^ identifies where both bits are different

~ gives the compliment of whatever is present

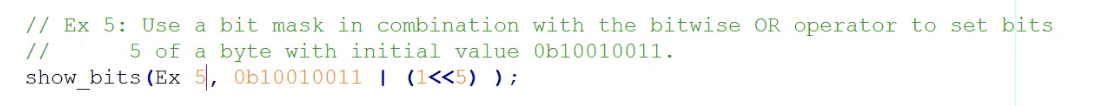
//combing xor with complement finds bits that are the same

LOGICAL;

* \*Bit masks\*

Bitmask is a numeric value in which one or more bits have been set so they can be used to insert or extract specific bits from some other expression

Using the OR with other << and original is setting the bit.

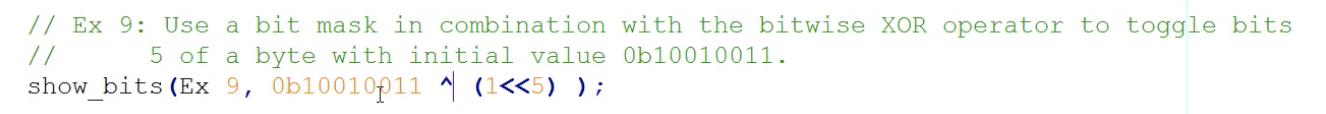


Clearing bit: Using a mask, complement with AND operator.

A screenshot of a cell phone

Description automatically generated

Toggle bit;



Extracting bit;

YouTube Video

* Adding binary then carrying the one.
* Negative integers are stored in 2's compliment
* 1st bit represents the sign
* 0 means Positive
* 1 means Negative
* (1)Convert the number to Binary
* (2)Invert bits
* (3)Add 1 ie 0000 0001
* Example 123>> \_111 1011

invert \_000 0100

add 1 \_000 0101

invert 1st 1000 0101

Left shift

* Logical shift
* 1110 1001 >>>>>> 0111 0100

Arithmetic shift

* Shift including the first bit but also makes the first bit what it was

1110 1001 >>>>>> 1111 0100

**Week 7**

* Integrated chip in an embedded system. “mini computer”
* They do a specific task. Include CPU ROM RAM I/O ports.
* The program must be loaded into the microcontroller
* PWM: Duty cycle >> (amount of time on high) / (total time)
* Ports B, C, D
* DDRx register has 8bits

1 represents an output and 0 for input

* PORTx
* PINx

Unsigned char temp; //temporary variable

Temp = PINB; //read input

* Read data from whole port
* Operations with bits:
  + Left and Right shift
  + NOT operation: Invert the bits. 1>>0 and 0>>1
  + AND operation:
  + OR operation:
* Tinker CAD:
  + Check with the datasheet the output of ports and etc.