(https://github.com/Nadir011995/Digital-electronics-2.git)

Binary operators &,|, ^ , ~ and <<

& (AND Operator)

Truth Table

X	Y	Output
0	0	0
0	1	0
1	0	0
1	1	1

Example in C code

If PORTB initially is 00000000 (PIN0 TO PIN7 are low)

Then PORTB = PORTB & (00000001) this will store 00000000 in PORTB because 0&1 is 0

If PORTB initially is 00000100 (PIN0 TO PIN7 are low except PIN2 (which is high))

Then PORTB = PORTB & (00000101) this will store 00000100 in PORTB because 0&1 is 0 and 1&1 is 1

| (OR Operator)

Truth Table

X	Y	Output
0	0	0
0	1	1
1	0	1
1	1	1

Example in C code

DDRB or" 0010 0000"

DDRB = DDRB | (1<<LED_GREEN);

DDRB ="0010 0000";

^ (XOR Operator) Output is 1 when the Inputs are different values and is 0 when both are same.

Truth Table

X	Y	Output
0	0	0
0	1	1
1	0	1
1	1	0

Example in C code (ACCORDING TO OUR TASK)

LED_GREEN means Pin 5 of PORT B (PB5)

If PORTB initially is 00000000 (PIN0 TO PIN7 are low)

Then PORTB = PORTB ^ (1<<LED_GREEN) this will store 00100000 in PORTB because 0^1 is 1

Now PORTB = 00100000, when we execute the same line of code (PORTB = PORTB ^ (1<<LED_GREEN))

this time it will store 00000000 because 1^1 is 0

~ (Tilde) We mostly use this to make pin as input pin OR to turn off the led

Used to reverse the bit

For example

PORTB = PORTB & ~ (1<<LED_GREEN)

We are doing AND operation of PORTB and 0 (because 1<<LED_GREEN IS 1 and ~(1<<LED_GREEN) will be zero)

DDRB = DDRB & ~ (1<<LED_GREEN)

This line will make pin 5 of PORTB as input

<< (Shift Operator)

This operator is used to shift the value either it is 1 or 0

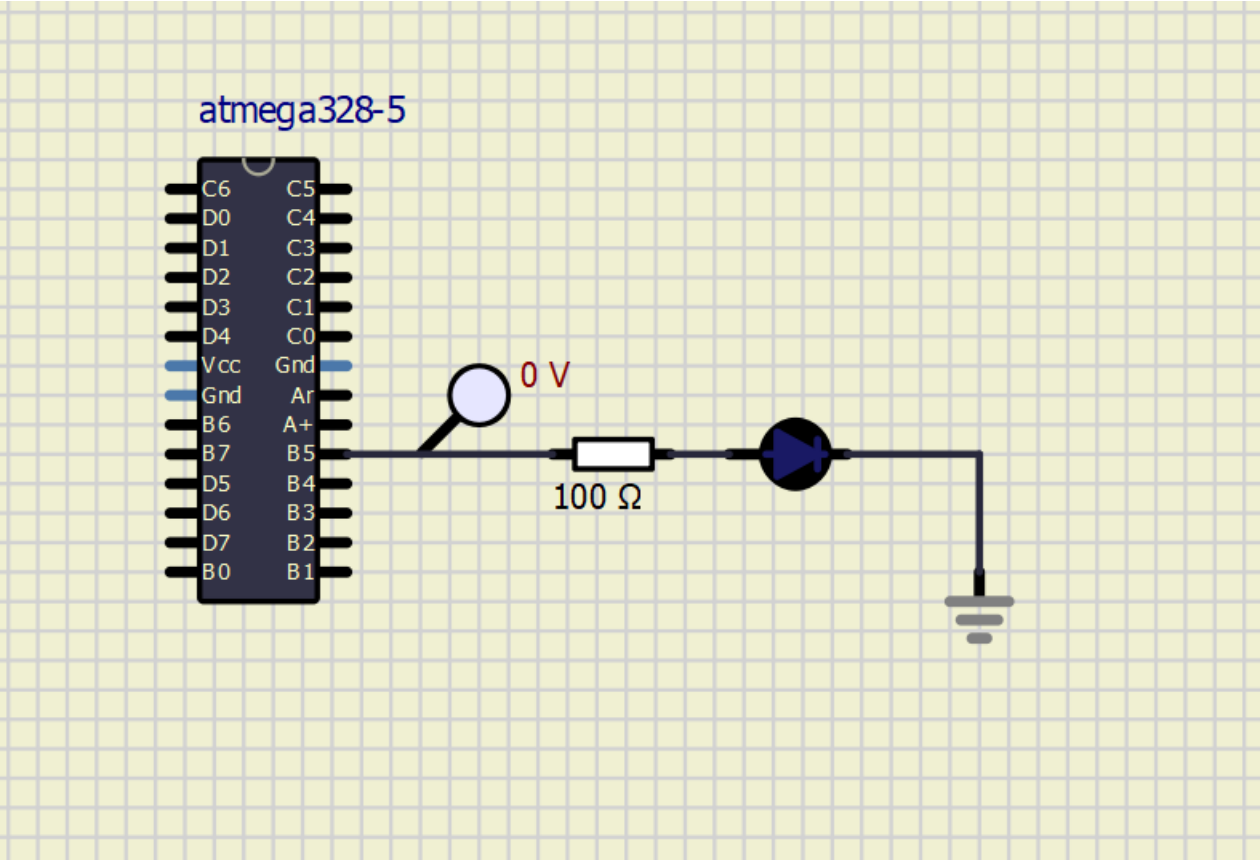
For example

PORTB = PORTB|(1<<LED_GREEN) it will move 1 to pin 5 of port B

DDRB = DDRB | (1<<LED_GREEN) As we are using DDRB here then it will make the pin 5 of PORTB as output pin because we have moved or shifted 1 to pin 5

If we write

DDRB = DDRB & ~(1<<LED_GREEN) It means we are shifting 0 to pin 5 making it input pin



```
/*
                    Author : Nadir Osman Al-Wattar
*/
#define LED_GREEN PB5    // AVR pin where green LED is connected
#define SHORT_DELAY 1000  // Delay in milliseconds
#define LONG_DELAY 3000
#define SPACE 3000
#ifndef  F_CPU
#define F_CPU 16000000   // CPU frequency in Hz required for delay
#endif
#include <util/delay.h>  // Functions for busy-wait delay loops
#include <avr/io.h>


void dot()
{
       PORTB = PORTB | (1<<LED_GREEN); // Turn ON LED PB5

       _delay_ms(SHORT_DELAY);
       PORTB = PORTB & ~(1<<LED_GREEN); //LED off
       //_delay_ms(SHORT_DELAY); // Delay between two symbols
}

void dash()
{
       PORTB = PORTB | (1<<LED_GREEN); // LED on
       _delay_ms(LONG_DELAY);
       PORTB = PORTB & ~(1<<LED_GREEN); // LED off
       //_delay_ms(SHORT_DELAY);
}

void space()// Delay between two letters
{
       PORTB = PORTB & ~(1<<LED_GREEN); //LED off
       _delay_ms(SPACE);
}

void D()
{
       dash();
       _delay_ms(SHORT_DELAY);
       dot();
       _delay_ms(SHORT_DELAY);
       dot();

}

void E()
{
       dot();

}

void two()
{
       dot();
       _delay_ms(SHORT_DELAY);// Delay between two symbols
       dot();
       _delay_ms(SHORT_DELAY);// Delay between two symbols
       dash();
       _delay_ms(SHORT_DELAY);// Delay between two symbols
       dash();
       _delay_ms(SHORT_DELAY);// Delay between two symbols
       dash();
```

```
}

int main(void)
{
       // DDRB = DDRB or 0010 0000
    // DDRB = 0b00100000 ; // set the pin 5 '1'
        DDRB = DDRB | (1<<LED_GREEN); //Set pin as output in Data Direction Register

       // PORTB = PORTB and 1101 1111
       PORTB = PORTB & ~(1<<LED_GREEN); //LED off

              while (1)
              {
                     D(),space(),E(),space(),two(),_delay_ms(7000);  //(DE2 = -.. . ..-
--       )
        }
       return 0;  // Will never reach this
}
```