VUT FEKT

[BPA-DE2] Digital Electronics 2

GitHub: *https://github.com/ShalaKreshnik*

Name and Surname: *Kreshnik Shala*

Person ID: *226108*

Date: Monday, September 28, 2020

Compound assignment operators

1) Bitwise **AND** operator (**&** Symbol)

Use:
Table of true: If any bits in the operation is equal to 0, the corresponding result of that bit will be 0. Useful when we need to disable bits in a register, force it to become zero or disable output register pins.

For example, if we need to filter or eliminate 4 Most significant bits of a 8 bit register, we can do it as follows:
8bit_reg = 8bit_reg & 0b00001111 Result: 8bit_reg = 0b0000xxxx

Where "x" Could be any value from 0 to 1 depending of the previous value of the 8bit_reg

Example: Reg = 0b0011 & 0b0101
Result: Reg = 0b0001

In truth Table:

A	B	Output (X)
0	0	0
0	1	0
1	0	0
1	1	1

2) Bitwise **OR** operator (**|** Symbol)

Use:
Table of true: If any bits in the operation is equal to 1, the corresponding result of that bit will be 1. Useful when we need to enable bits in a register, force it to become one or enable output register pins.

For example, if we need to set (Force to be logic one) the 4 Most significant bits of a 8 bit register, we can do it as follows: 8bit_reg = 8bit_reg | 0b00001111

Result: 8bit_reg = 0bxxxx1111

Where “x” Could be any value from 0 to 1 depending of the previous value of the 8bit_reg

Example: Reg = 0b0011 & 0b0101
Result: Reg = 0b0111

In truth Table:

A	B	Output (X)
0	0	0
0	1	1
1	0	1
1	1	1

3) Bitwise **XOR** operator (**^** Symbol)

Use:
Table of true: If any bits in the operation is equal to 1, the corresponding result of that bit will be inverted or switched. Useful when we need to switch an specific bit (from 0 to 1 or 1 to 0).

For example, if we need to switch 4 Least significant bits of a 8 bit register, we can do it as follows:
8bit_reg = 0b10101111
8bit_reg = 8bit_reg ^ 0b00001111

Result: 8bit_reg = 0b10100000

Example: Reg = 0b0011 ^ 0b0101
Result: Reg = 0b0110

A	B	Output (X)
0	0	0
0	1	1
1	0	1
1	1	0

4) Bitwise **Left Shift** operator (**<<** Symbol)

Use:
Useful when we need to move or displace a bit or specific bit to the left. For example, if we need to displace to the left 2 bits of a 8 bit register, we can do it as follows:

```
8bit_reg = 0b00100011
8bit_reg = 8bit_reg << 2
```

Result: 8bit_reg = 0b10001100

5) Bitwise **Right Shift** operator (**>>** Symbol)

Use:
Works in the same way as the << symbol and its Useful when we need to move or displace a bit or specific bits to the right.

For example, if we need to displace to the left 2 bits of a 8 bit register, we can do it as follows:

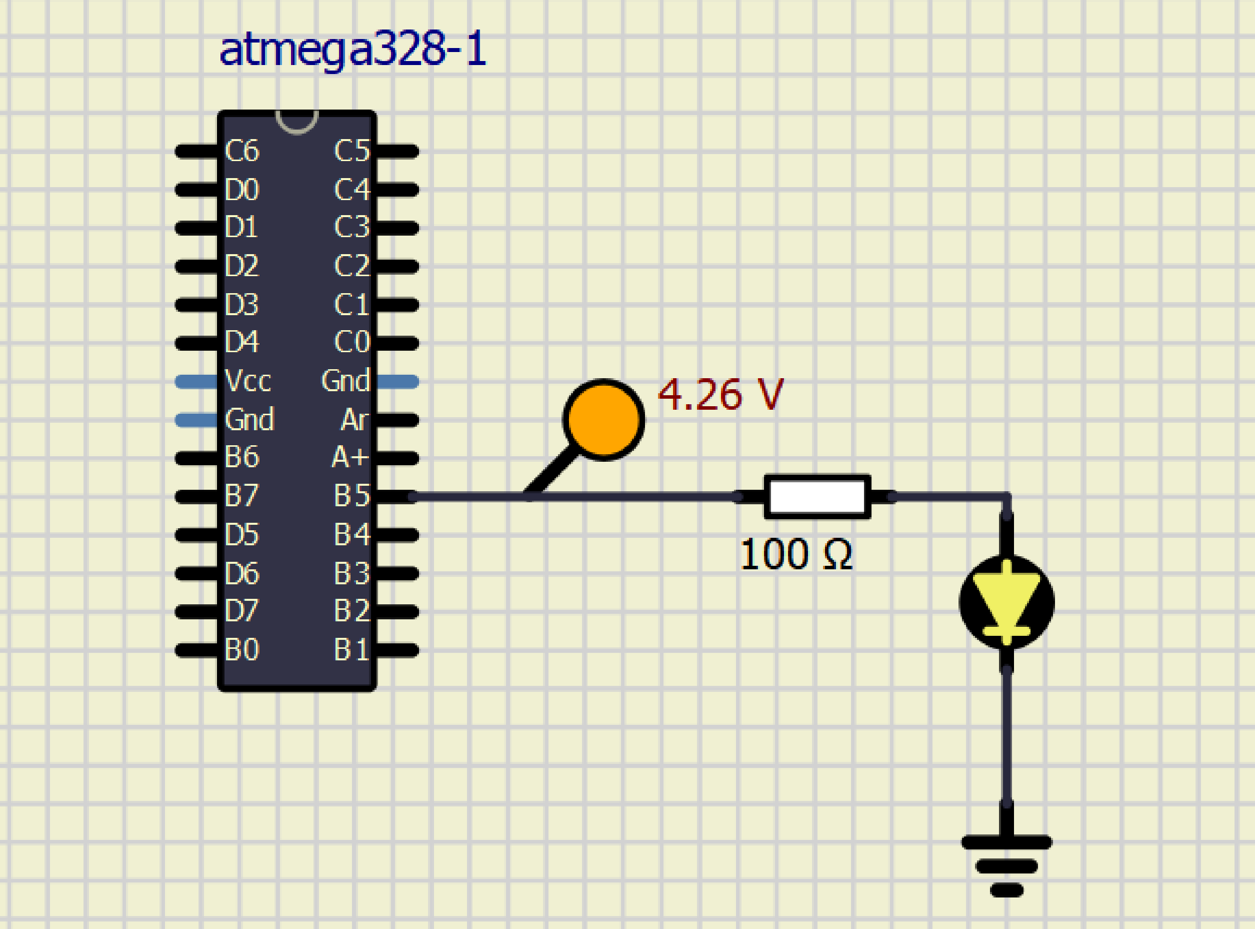
```
8bit_reg = 0b10001100
8bit_reg = 8bit_reg >> 2
```

Result: 8bit_reg = 0b00100011

```
/*////////////////////////////////////////////////////////////////////////////////////////
///                                                                                    ///
/// VUT FEKT                                        Name and Surname: Kreshnik Shala   ///
/// [BPA-DE2] Digital Electronics 2                 Person ID: 226108                  ///
/// Date:  Monday, September 28, 2020                                                  ///
/// GitHub: https://github.com/ShalaKreshnik                                           ///
///                                                                                    ///
*/////////////////////////////////////////////////////////////////////////////////////////


#include <avr/io.h>         // AVR Basic I/O Header


/* We can just use #define F_CPU 16000000UL
But In order to avoid duplicate definition, we have defined as follows. If F_CPU is not
defined previously, define here.
*/
#ifndef F_CPU
#define F_CPU 16000000
#endif
#include <util/delay.h>     // AVR Delay Header (For _delay_ms function)


#define LED_GREEN PB5
#define ONE_UNIT_DELAY 400
#define THREE_UNITS_DELAY 1200
#define SEVEN_UNITS_DELAY 2100




void dot() // Function whenever it will be called in the main it will perform these three
lines of code
{
        PORTB = PORTB | (1<<LED_GREEN); // Turn ON LED PB5
        _delay_ms(ONE_UNIT_DELAY); // length of dot is one unit
        PORTB = PORTB & ~(1<<LED_GREEN); // Turn OFF LED PB5
}

void dash()
{
        PORTB = PORTB | (1<<LED_GREEN); // Turn ON LED PB5
        _delay_ms(THREE_UNITS_DELAY); // length of dash is three units
        PORTB = PORTB & ~(1<<LED_GREEN); // Turn OFF LED PB5
}

void space()
{
        PORTB = PORTB & ~(1<<LED_GREEN); // Turn OFF LED PB5 (SPACE BETWEEN TWO WORDS)
        _delay_ms(SEVEN_UNITS_DELAY); // Length between words is SEVEN units
}



int main(void)
```

```
{
    // DDRB = DDRB or 0010 0000
    DDRB = DDRB | (1<<LED_GREEN); // Making pin5 of PORTB as an output pin

    // PORTB = PORTB and 1101 1111
    PORTB = PORTB & ~(1<<LED_GREEN); // Initially storing 0 on PB5



    // Infinite loop
    while (1)
    {

        /*^^^^^^^(DE2 = -.. . ..---)^^^^^^^^*/


        /*1. Letter "D" -> (-..)*/
        dash();
        _delay_ms(ONE_UNIT_DELAY); // length between different parts is one unit
        dot();
        _delay_ms(ONE_UNIT_DELAY); // length between different parts is one unit
        dot();

        // SHORT GAP (between letters)
        _delay_ms(THREE_UNITS_DELAY);
        /* Alphabet “D” completed */


        /*2. Letter "E" -> (.)*/
        dot();

        // SHORT GAP (between letters)
        _delay_ms(THREE_UNITS_DELAY);
        /* Alphabet “E” completed */


        /*2. Number "2" -> (..---)*/
        dot();
        _delay_ms(ONE_UNIT_DELAY); // length between different parts is one unit
        dot();
        _delay_ms(ONE_UNIT_DELAY); // length between different parts is one unit
        dash();
        _delay_ms(ONE_UNIT_DELAY); // length between different parts is one unit
        dash();
        _delay_ms(ONE_UNIT_DELAY); // length between different parts is one unit
        dash();       // NUMBER 2 completed (..---)

        // SHORT GAP (between letters)
        _delay_ms(THREE_UNITS_DELAY);
        /* NUMBER “2” completed */


        // MEDIUM GAP (between words)
        space();
        }

        return 0;
}
```

In SimulIDE:


atmega328-1
C6
D0
D1
D2
D3
D4
Vcc
Gnd
B6
B7
D5
D6
D7
B0
C5
C4
C3
C2
C1
C0
Gnd
Ar
A+
B5
B4
B3
B2
B1
4.26 V
100 Ω