Embedded Systems CSEN701

Dr. Catherine Elias

Eng. Abdalla Mohamed

Office: C1.211
Mail : abdalla.abdalla@guc.edu.eg

Eng. Mohamed Elshafie

Office: C1.211
Mail : Mohamed.el-shafei@guc.edu.eg

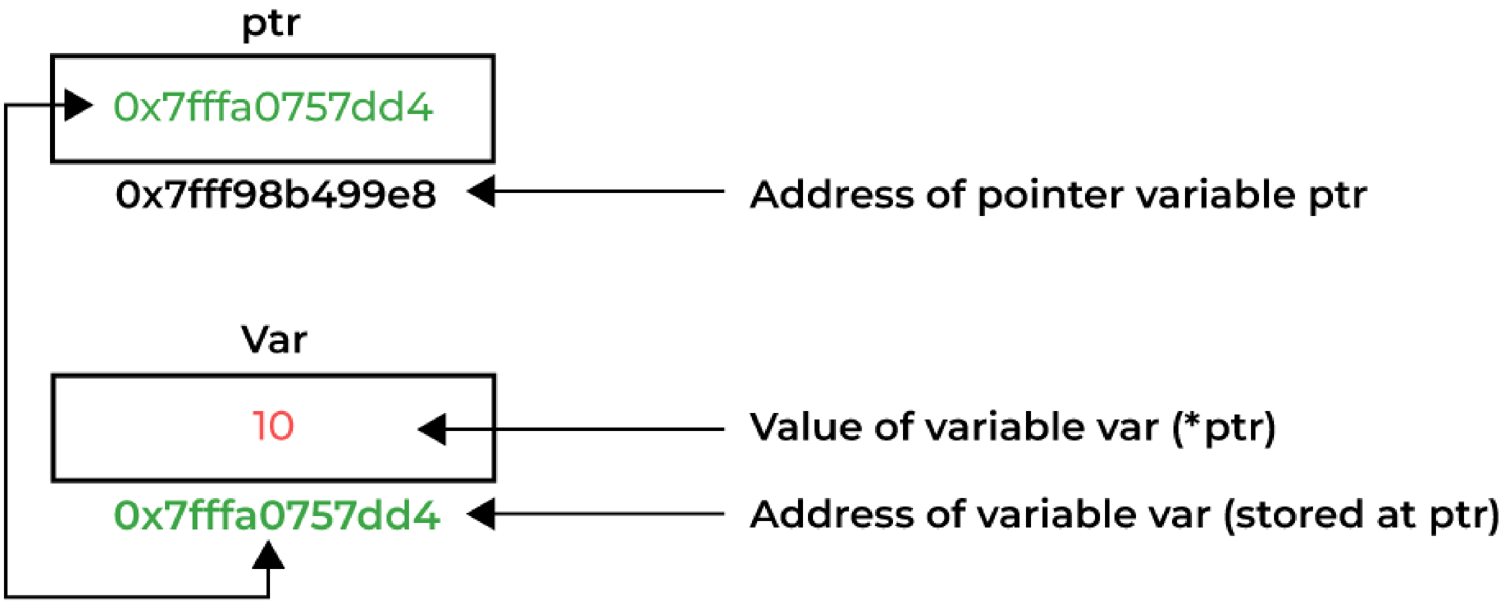
Eng. Maysarah El Tamalawy

Office: C7.201
Mail : Maysarah.mohamed@guc.edu.eg

Outline :

- **Recap.**
- Arduino Components.
- Harvard Architecture vs Von Neuman Architecture.
- AVR architecture.
- PINs and Ports in AVR.
- Bitwise operations
- Examples

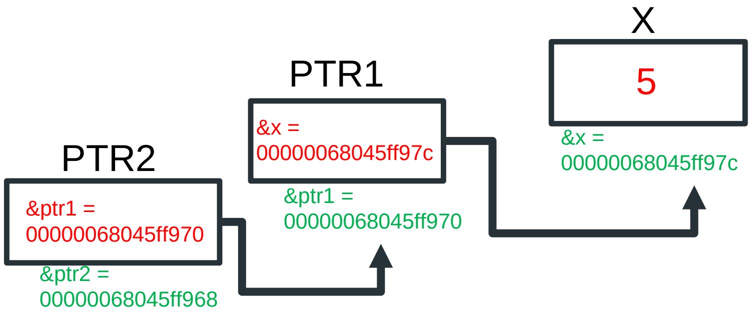
Pointers in C.



Pointers in C.

```c
int main ( void ){
    int x = 5 ;
    int *ptr1= &x ;
    int **ptr2 = &ptr1  ;
printf( " value of X : =  %d \n" , x ) ;
printf( " value of address X = %p \n "  , &x ) ;
printf( " value of the pointer 1  = %p  \n" , ptr1 ) ;
printf( " value of address pointer 1  = %p  \n" , &ptr1 ) ;
printf( " value of Asterisk POINTER 1 = %d \n " , *ptr1 ) ;
printf( " value of ASTERISK address of pointer 1  = %p  \n  " , *&ptr1 ) ;
printf( " value of pointer 2 = %p   \n " , ptr2 ) ;
printf( " value of asterisk pointer 2 = %p   \n " , *ptr2 ) ;
printf( " value of double asterisk pointer 2 = %d   \n " , **ptr2 ) ;
printf( " value of address  pointer 2 = %p   \n " , &ptr2 ) ;
printf( " value of  asterisk address  pointer 2 = %p   \n " , *&ptr2 ) ;
}
```

Pointer to pointer to integer



value of X : = 5
value of address X = 00000068045ff97c
value of the pointer 1 = 00000068045ff97c
value of address pointer 1 = 00000068045ff970
value of Asterisk POINTER 1 = 5
value of ASTERISK address of pointer 1 = 00000068045ff97c
value of pointer 2 = 00000068045ff970
value of asterisk pointer 2 = 00000068045ff97c
value of double asterisk pointer 2 = 5
value of address pointer 2 = 00000068045ff968
value of asterisk address pointer 2 = 00000068045ff970

Pass By Value VS Pass By address

Pass By Value

```c
void increment( int a){
    a = a+5 ; // a new Local variable is created
    printf( " the value of 'a' inside increment  is %d   " , a)  ;
    printf( " the address of 'a' is %p \n " , &a) ;
} // local variable value is lost once we get out of the function
int main( void) {
    int x  = 5  ;
    printf( " the value of 'x' before increment  is %d  " , x)  ;
    printf( " the address of 'x' is %p \n " , &x) ;
    increment(x) ;
    printf( " the value of 'x' after increment  is %d  " , x) ;
// X is passed by value . the value of x is copied to a local variable a
    // real value of x in not altered !!!!
    }
```

Console

```
<terminated> (exit value: 0) tut_2_CSEN701.exe [C/C++ Application] C:\Users\Abdalla\eclipse-workspace\tut_2_CSEN701\Debug\tut_2_CSEN701.exe (10/5/23
 the value of 'x' before increment  is 5   the address of 'x' is 00000060383ffc0c
 the value of 'a' inside increment  is 10    the address of 'a' is 00000060383ffbe0
 the value of 'x' after increment  is 5
```

Pass By address

```c
void increment( int * a){
    *(a) = *(a)+5 ; // pointer to integer is introduced to carry the address of x
    printf( " the value of 'a' inside increment  is %d   " , *a)  ;
    printf( " the address of 'a' is %p \n " , a) ;
}
int main( void) {
    int x  = 5  ;
    printf( " the value of 'x' before increment  is %d  " , x)  ;
    printf( " the address of 'x' is %p \n " , &x) ;
    increment(&x) ;
    printf( " the value of 'x' after increment  is %d  " , x) ;
// X is passed by address . the address of x is sent to the function
    // real value of x is edited !!!!
    }
```

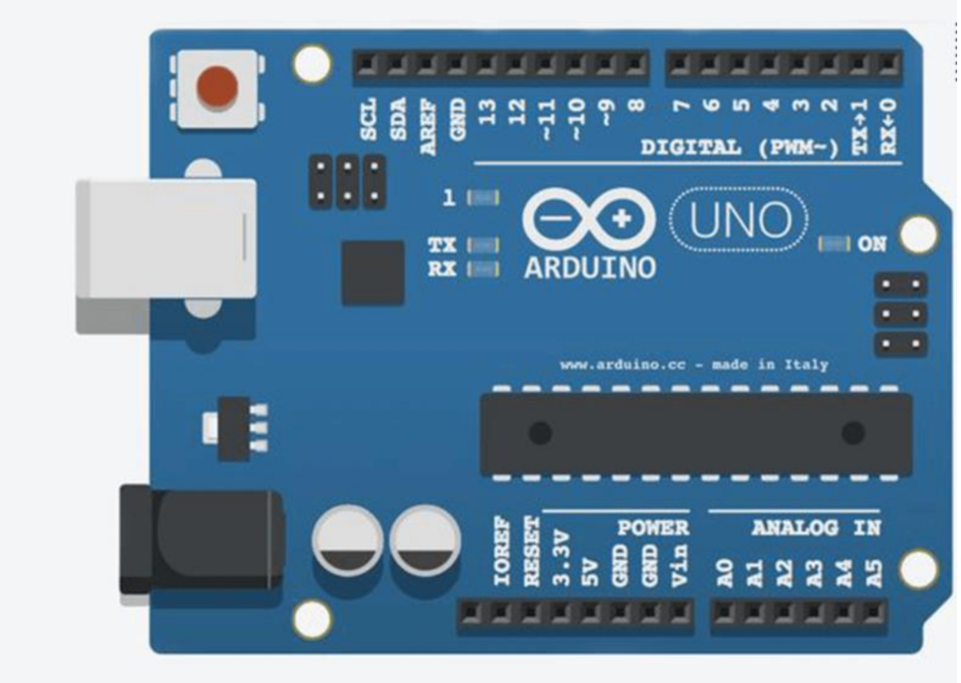
Console

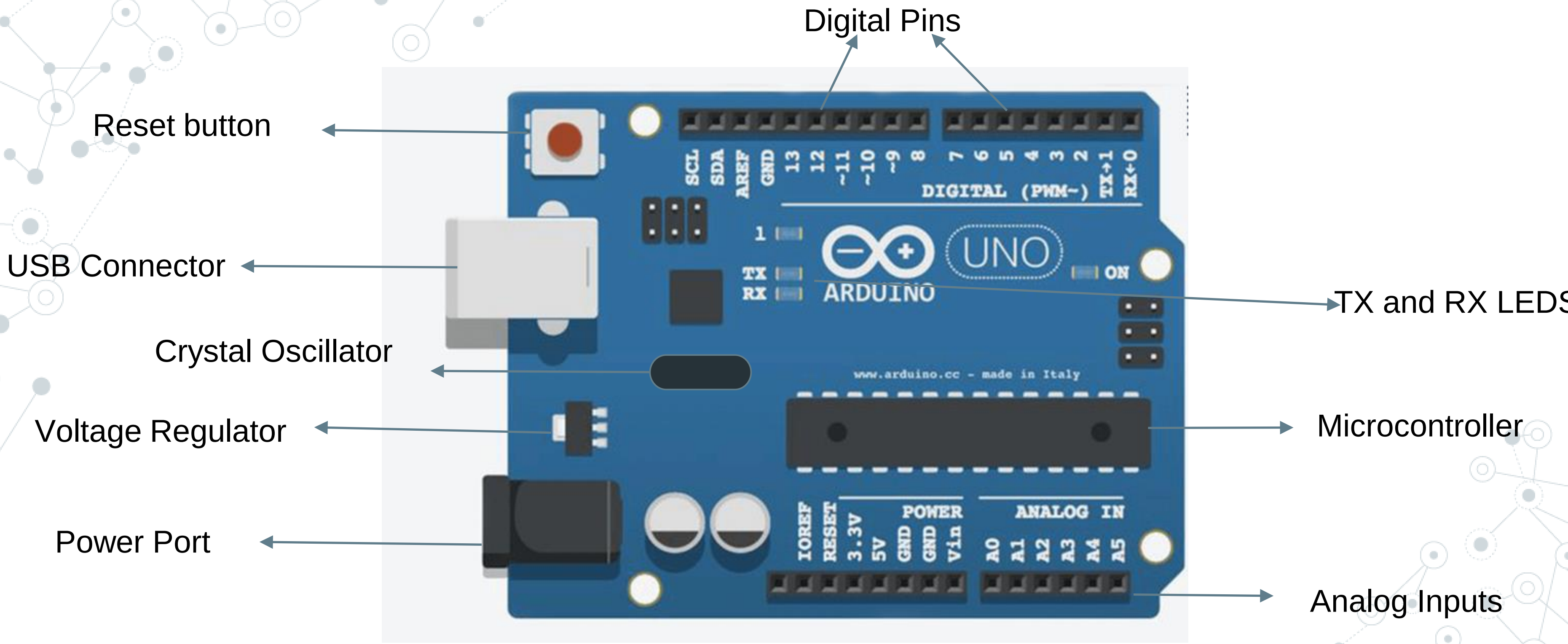
```
terminated> (exit value: 0) tut_2_CSEN701.exe [C/C++ Application] C:\Users\Abdalla\eclipse-workspace\tut_2_CSEN701\Debug\tut_2_CSEN701.exe (10/5/23, 6:11 PM)
the value of 'x' before increment  is 5   the address of 'x' is 0000003f23fff94c
 the value of 'a' inside increment  is 10    the address of 'a' is 0000003f23fff94c
 the value of 'x' after increment  is 10
```

Outline :

- Recap.
- **Arduino Components.**
- Harvard Architecture vs Von Neuman Architecture.
- AVR architecture.
- PINs and Ports in AVR.
- Bitwise operations
- Examples

Can you name the components?

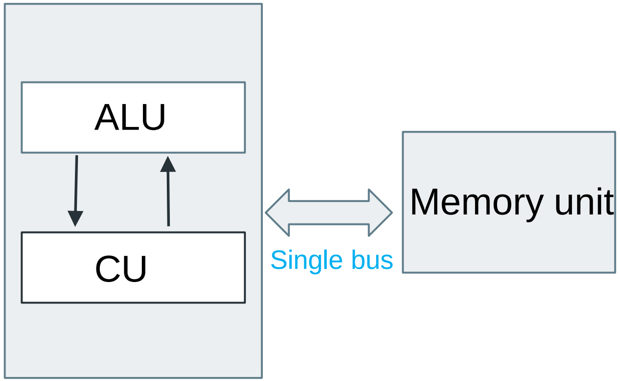



Digital Pins
Reset button
USB Connector
Crystal Oscillator
Voltage Regulator
Power Port
TX and RX LEDS
Microcontroller
Analog Inputs
SCL SDA AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX→1 RX←0
DIGITAL (PWM~)
UNO
ARDUINO
ON
TX
RX
www.arduino.cc - made in Italy
POWER
ANALOG IN
IOREF RESET 3.3V 5V GND GND Vin
A0 A1 A2 A3 A4 A5

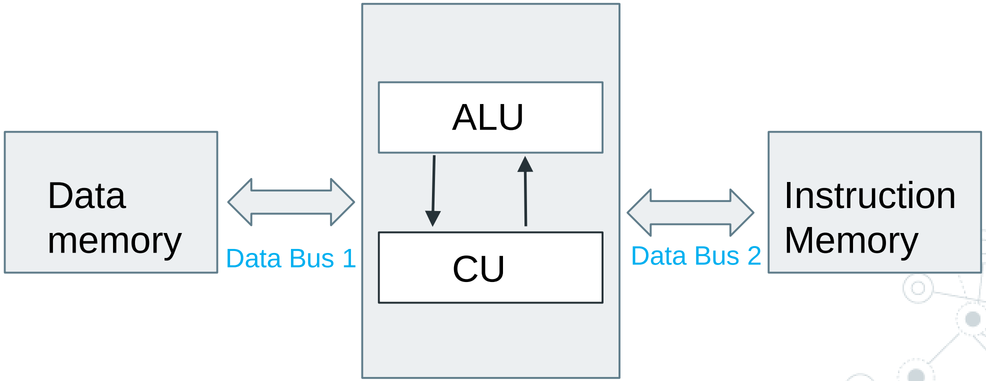
Outline :

- Recap.
- Arduino Components.
- **Harvard Architecture vs Von Neuman Architecture.**
- AVR architecture.
- PINs and Ports in AVR.
- Bitwise operations
- Examples

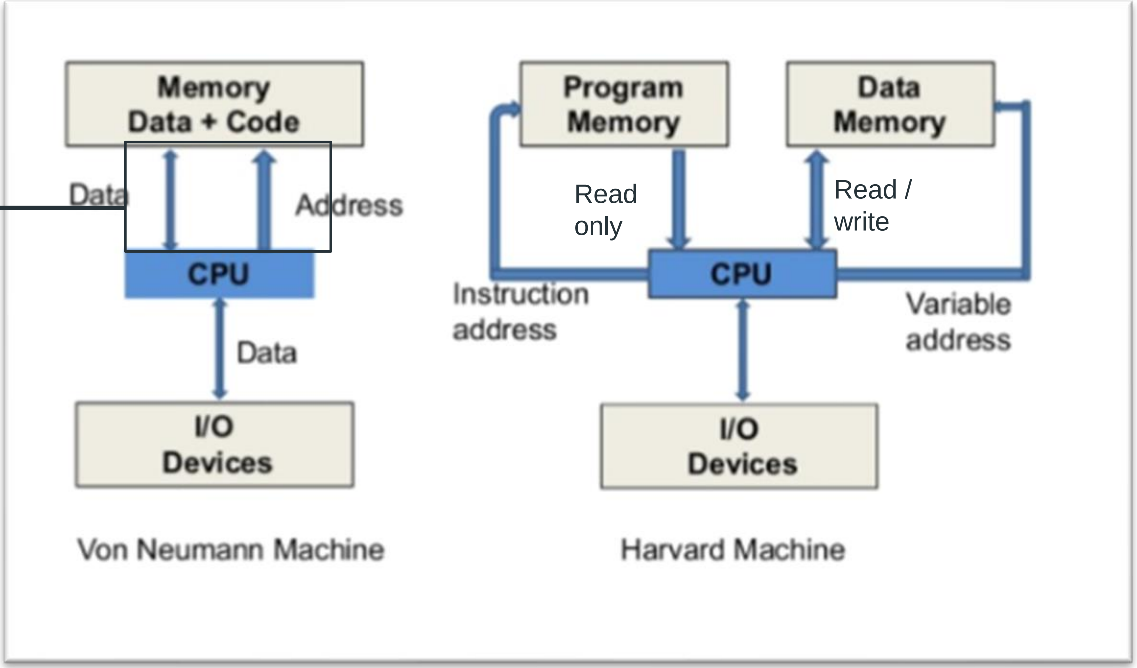
Von Neuman



Harvard



Single data bus



Von Neuman

- Instruction fetch and data operation cannot happen at the time.
- The instructions and data are in the same place so they use a common bus.

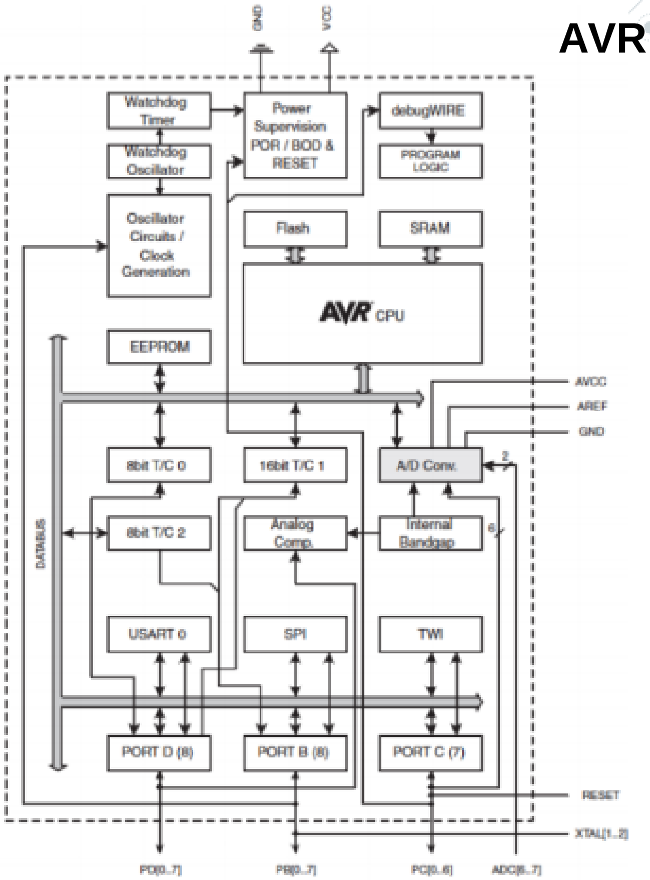
Harvard

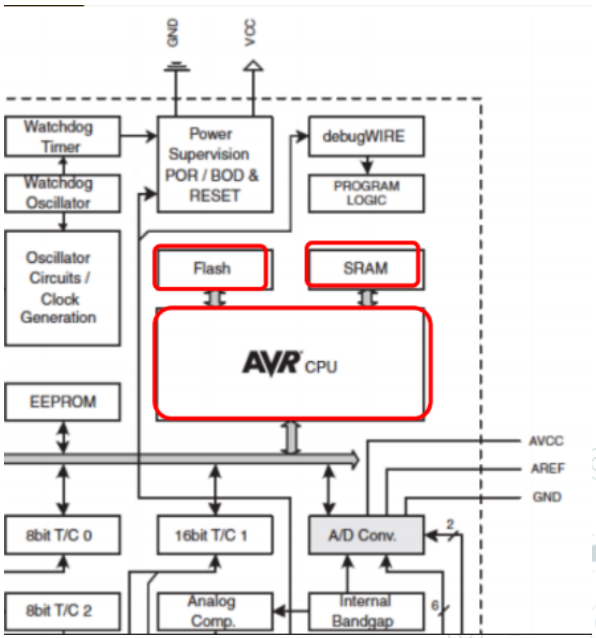
- The instructions are in separate memory.
- There are two buses one between control unit and instruction memory and one between data memory and control unit.

Outline :

- Recap.
- Arduino Components.
- Harvard Architecture vs Von Neuman Architecture.
- **AVR architecture.**
- PINs and Ports in AVR.
- Bitwise operations
- Examples

AVR architecture from Data Sheet





AVR architecture

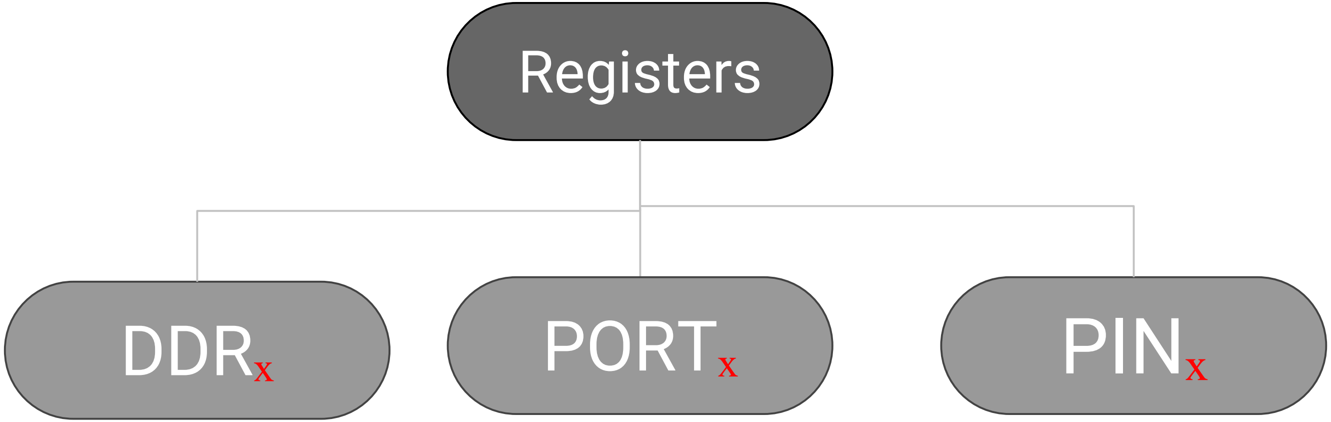
- **Simple Instruction Set:** AVR uses a small, well-defined set of instructions, each of which performs a single operation. This simplicity leads to efficient and fast execution so it's based on RISC architecture.
- **Separate Program and Data Memory:** AVR architecture features distinct memory spaces for program instructions (Flash memory) and data (SRAM). This separation allows for simultaneous access to program and data, improving performance.
- **Separate Buses:** It employs separate buses for program memory and data memory, enabling parallel fetching of instructions and data.

Outline :

- Recap.
- Arduino Components.
- Harvard Architecture vs Von Neuman Architecture.
- AVR architecture.
- **PINs and Ports in AVR.**
- Bitwise operations
- Examples

Ports in Arduino AVR

- A port on the Arduino is a group of pins, each consists of 3 types of registers that control the functionality of this port. These registers determine the setup of the pins.



* x is the name of the port (A ,B, C or D) *

Dr.Catherine Elias
Eng. Abdalla Mohamed Eng. Mohamed Elshafie Eng. Maysarah El Tamalawy

DDR register

- DDR register is the register the determine the data direction for a group of pins .
- You can select whether a certain pin is input or output by changing the value of the corresponding bit in the DDR register .
- Ex:- if we want to set pin 5 in port B as input , then this bit is set to 0 , if we want to set it to output then it is set to 1
- So if all pins are set to input and only pin 5 is set to output in port B, the value of the DDR_B is 00100000 = 0x20

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

PORT register

- PORT registers have 2 functionalities :
  - If the bit is set to output in DDR register :
    - If a bit in the register is set to 1 , then the corresponding pin is driven HIGH
    - If a bit in the register is set to 0 , then the corresponding pin is driven LOW

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PORT register

- PORT registers have 2 functionalities :
  - If the bit is set to input in DDR register :
    - If a bit in the PORT register is set to 1 , then the internal pull up resistor is activated.
    - If a bit in the PORT register is set to 0 , then the pin is tri-stated (default input pin).

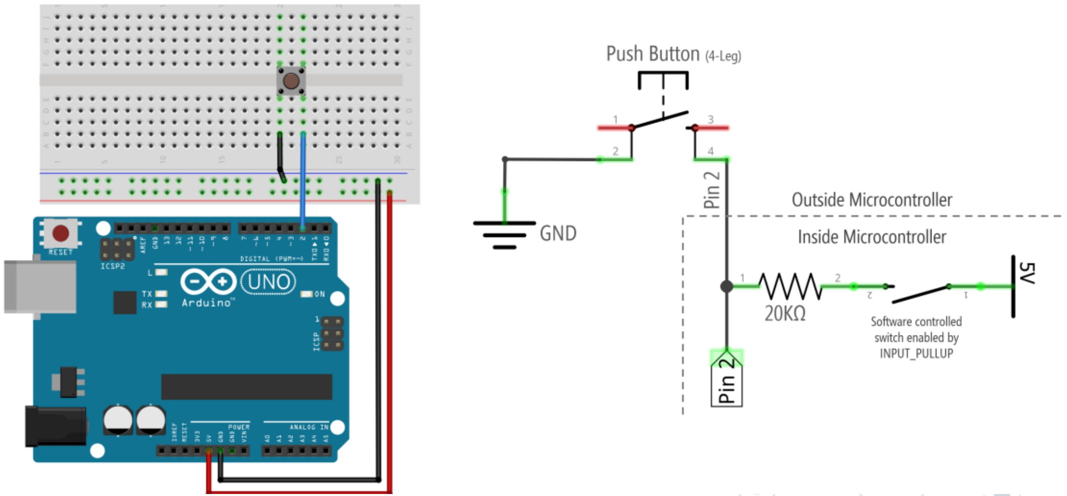
Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PORT register

- If the bit is set to input 0 in DDR register and the same bit in the PORT register is set to 1 , then the internal pull up resistor is activated at the corresponding pin .

- The Initial Value of this pin will be High in case of no input signal .

INTERNAL PULL-UP RESISTOR CONFIGURATION



PIN register

- PIN registers are used to read the input data from a port pin
- When the pin is set as input in the DDR, and the pull-up resistor is enabled (in the PORT register) then the bit will indicate the state of the signal at the pin.

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Outline :

- Recap.
- Arduino Components.
- Harvard Architecture vs Von Neuman Architecture.
- AVR architecture.
- PINs and Ports in AVR.
- **Bitwise operations**
- Examples

Bitwise operations in C

- Bitwise operations are used to directly manipulate registers in embedded C .
- Bitwise operations are used to :
  - Set bits (LOGIC HIGH)
  - Clear bits (LOGIC LOW)
  - Toggle bits (XORING)
  - Shift bits

Operator	Description
&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
<<	shift left
>>	shift right
~	one's complement

SET BITS

DDRD = 0b00000000 ; DDRD = 0x00 ; (hexadecimal)

Set bits 0 and 2 as outputs :

DDRD = 0b00000101 ; DDRD = 5 ; (PORT Assignment)

DDRD |= 5 ; / (*DDRD = DDRD | 0b00000101*)

DDRD 0 0 0 0 0 0 0 0

OR |

5 0 0 0 0 0 1 0 1

Hint 1 : any bit ORED | with 0 is unchanged

Hint 2 : any bit ORED | with 1 is SET

DDRD 0 0 0 0 0 1 0 1 (bit assignment)

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

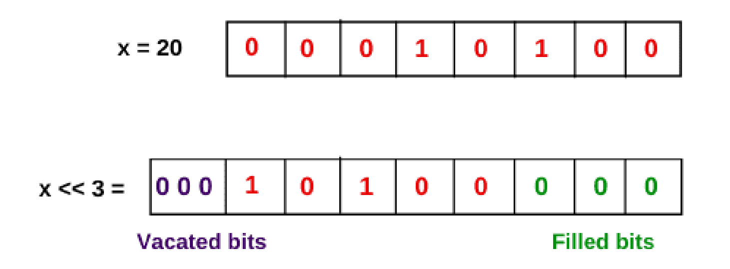


Fig: Shifting bits towards left 3 times

bit 1	bit 2	&	||	^	~ bit 1	~ bit 2
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

SET BITS

DDRD = 0b00000000 ; DDRD = 0x00 ; (hexadecimal)

Set bits 0 and 2 as outputs :

DDRD |= (1<<0) |(1<<2) ; (1<<bit number)

DDRD	OR	0 0 0 0 0 0 0 0
(1<<0)	|	0 0 0 0 0 0 0 1
(1<<2)	|	0 0 0 0 0 1 0 0

DDRD = 0 0 0 0 0 1 0 1

Hint : any bit ORED | with 0 is unchanged

Hint 2 : any bit ORED | with 1 is SET

only targeted bits are assigned and set

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

DDRD = 0b00000101 ;

DDRD |=5 ;

DDRD |= 0x05 ;

DDRD |= (1<<0)|(1<<2);

CLEAR BITS

DDRD = 0b00000101 ; DDRD = 0x05 ; (hexadecimal)

Clear bit 2 to change it to input pin :

DDRD = 0b00000001 ; DDRD = 1 ;   (PORT Assignment)
DDRD & = 0b11111011 ;

0 0 0 0 0 1 0 1
AND &
1 1 1 1 1 0 1 1

DDRD   0 0 0 0 0 0 0 1 (bit assignment)

Hint 1 : any bit ANDED | with 1 is unchanged

Hint 2 : any bit ANDED | with 0 is cleared

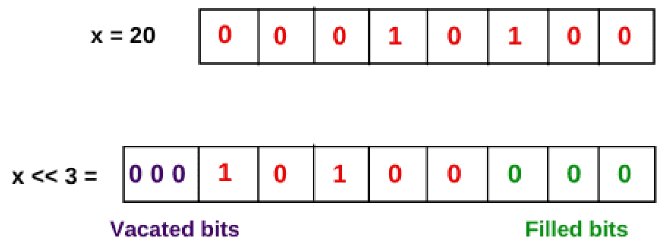


Fig: Shifting bits towards left 3 times

bit 1	bit 2	&	||	^	~ bit 1	~ bit 2
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

CLEAR BITS

DDRD = 0b00000101 ; DDRD = 0x05 ; (hexadecimal)

Clear bit 2 to change it to input pin :

DDRD & = 0b11111011 ; 0b11111011 == ~(00000100)

DDRD &= ~(0b00000100)

DDRD &= ~(1<<2) ; (1<<bit number)

Hint 1 : any bit ANDED | with 1 is unchanged

Hint 2 : any bit ANDED | with 0 is cleared

DDRD		0 0 0 0 0 1 0 1
AND	&	
~(1<<2)		1 1 1 1 1 0 1 1
DDRD		0 0 0 0 0 0 0 1

only targeted bits are assigned and cleared

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

DDRD = 0b00000001 ;

DDRD &=0b11111011 ;

DDRD &= ~(0b00000100) ;

DDRD &= ~(1<<2);

Bitwise operations in a Nutshell

- SET BIT : REG | = (1<<bit_number) | (1<<bit_number) …..
- CLEAR BIT : REG &= ~(1<<bit_number) & ~(1<<bit_number)….
- TOGGLE BIT : REG ^=(1<<bit_number)^(1<<bit_number)…..

HINTS :

- ✓ **any bit ORED | with 0 is unchanged**
- ✓ **any bit ORED | with 1 is SET**
- ✓ **any bit ANDED | with 1 is unchanged**
- ✓ **any bit ANDED | with 0 is cleared**
- ✓ **any bit XORED | with 1 is Toggled**

Outline :

- Recap.
- Arduino Components.
- Harvard Architecture vs Von Neuman Architecture.
- AVR architecture.
- PINs and Ports in AVR.
- Bitwise operations
- **Examples**

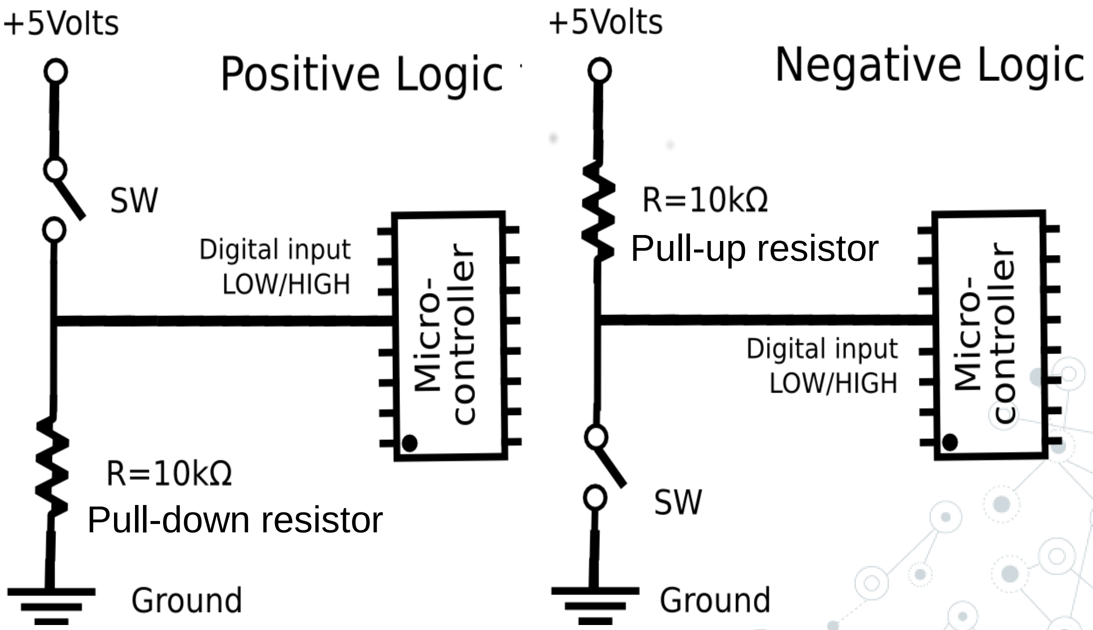
EX1

Implement an embedded C code to :

1. Connect push button A to pin 5 in PORT C (Positive Logic)
2. Connect push button B to pin 3 in PORT B (Negative Logic)
3. Apply the Internal pullup resistor to pin3 PORT B
4. Configure PIN 2 in PORTD as output
5. Connect Pin 2 to RED LED Pin5-- RED (Hardware step)
6. Turn on Red LED when A is pressed
7. Turn off the RED LED when B is pressed .

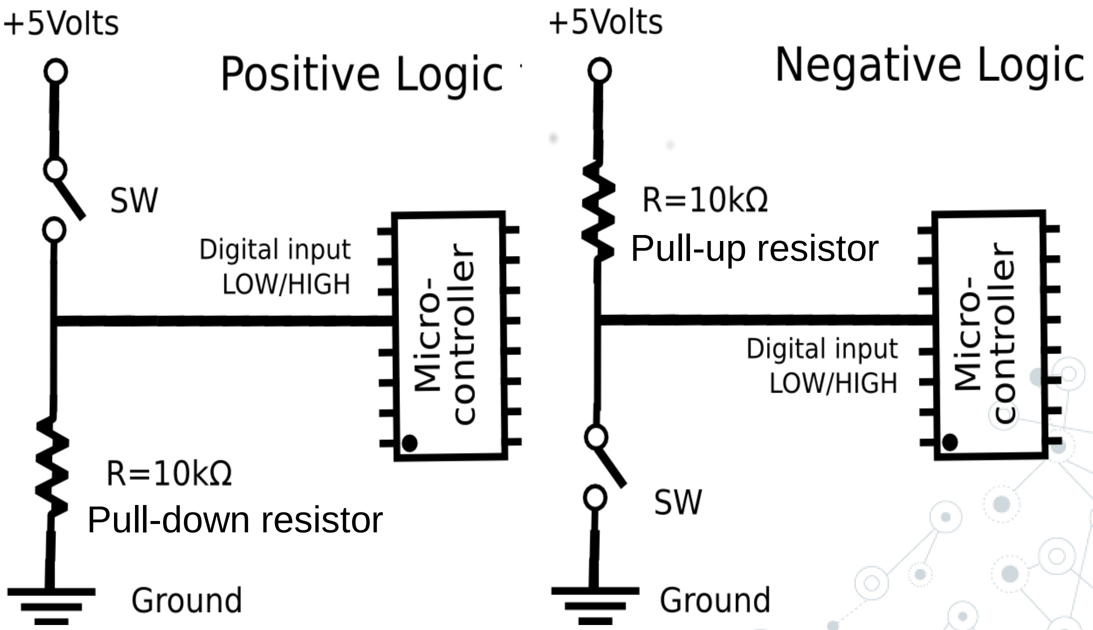
But first we have to understand different digital logics …

- Positive logic is the **default logic** , the input is initially low until the button is ON /activated so it deliver High voltage (5v) to the Microcontroller/LED when pressed.
- Sensors/Pins working with positive logic are called Active-HIGH.
- Pull-down resistors are associated with positive logic to **pull** the initial pin value **down** to GND thus preventing the floating of the input value.
- Button Is OFF/open -- Input = GND (0V)
- Button is ON/Closed -- Input = VCC (5V)



But first we have to understand different digital logics …

- Negative Logic connection operates in an opposite manner, the input is initially high until the button is On/activated it delivers GND (0V) to the Microcontroller/LED.
- Sensors/Pins working with negative logic are called Active-Low.
- Pull-up resistors are associated with negative logic to **pull** the initial pin value **up** to High voltage (5V) thus preventing the floating of the input value .
- Button Is OFF/open -- Input = VCC (5V)
- Button is ON/Closed -- Input = GND (0V)



EX1

```
#include <avr/io.h>
int main (void){
DDRB = 0x00 ; DDRD = 0x00 ; DDRC = 0x00 ; PORTC = 0x00 ; PORTB=0x00; PORTD=0x00 ;    // initialize the registers
DDRC &=~(1<<5)  ;  // configure pin5 as input  in PORTA ( pushbutton A is connected to PIN 5 in positive Logic )
DDRB &=~(1<<3)  ;  // configure pin3 as input in PORTB ( pushbutton B is connected to PIN 3 in negative Logic )
PORTB |= (1<<3) ; // set bit 3 to HIGH to activate the internal pull-up resistor at pin 3
DDRD   |= (1<<2) ; // configure pin 2 as an output pin  at PORTD
while (1 ) {
if ( PINC & ( 1<<5) ) {   //  HINT : (PINC & 0b00100000) is only true when bit 5 at PINC is 1 (pushbutton A is pressed +ve L)
PORTD |= (1<<2) ;  // set the output to HIGH  to TURN ON the LED
}
if ( !(PINB & (1<<3) ) {   // (PINB & (1<<3)) is true when Bit 3 is ON ( not pressed ) so it will be false (!) if  pressed ( -ve L)
    PORTD &= ~(1<<2)  ; // set the output to LOW  by clearing bit 2        // pushbutton B is connected in negative logic
  } }                                    }
```