Chapter-1

Address Bus is a set of parallel Unidirectional signal lines.

In **Address Bus** CPU sends out the address of the memory location or I/O port

Data Bus is a set of parallel bidirectional signal lines.

Data Bus allows for the transferring of data from one component to another

Microprocessor(µP) is a program-controlled IC which **fetches**, **decodes**, and **executes** instructions.

Note// its use as a CPU in a computer

Bit = It represents either 1 or 0

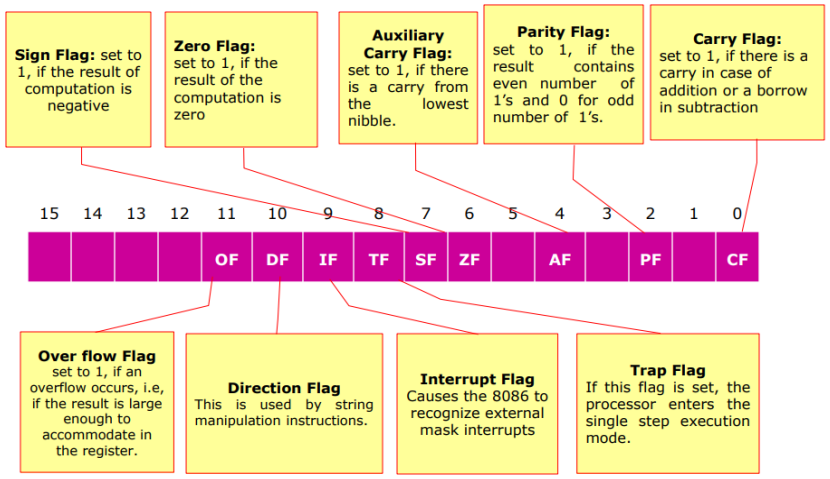
Byte = 8 bits

Word = 8 bits, 16 bits, 32 bits word

Microprocessor Evolution

Generation	First (1971-73)	Second (1973-78)	Third (1978-80)	Fourth (1980..)
processors → pins	• 4 bit→16 pins • 8 and 16 bit → 40 pins	4 / 8/ 16 bit → 40pins	16 bit →40/ 48/ 64 pins	32 bit processors note/no pins
features		• Ability to address large memory spaces and I/O ports • Better interrupt handling capabilities	• Easier to program • Processor has multiply/ divide arithmetic hardware • Max Memory : 1 Mb	*Physical memory space 2^{24} bytes = 16 Mb *Virtual memory space 2^{40} bytes = 1 Tb
Example	Intel 4004	Intel 8085	Intel 8086	Intel 80386

Flags



Instruction queue (Pipelining)

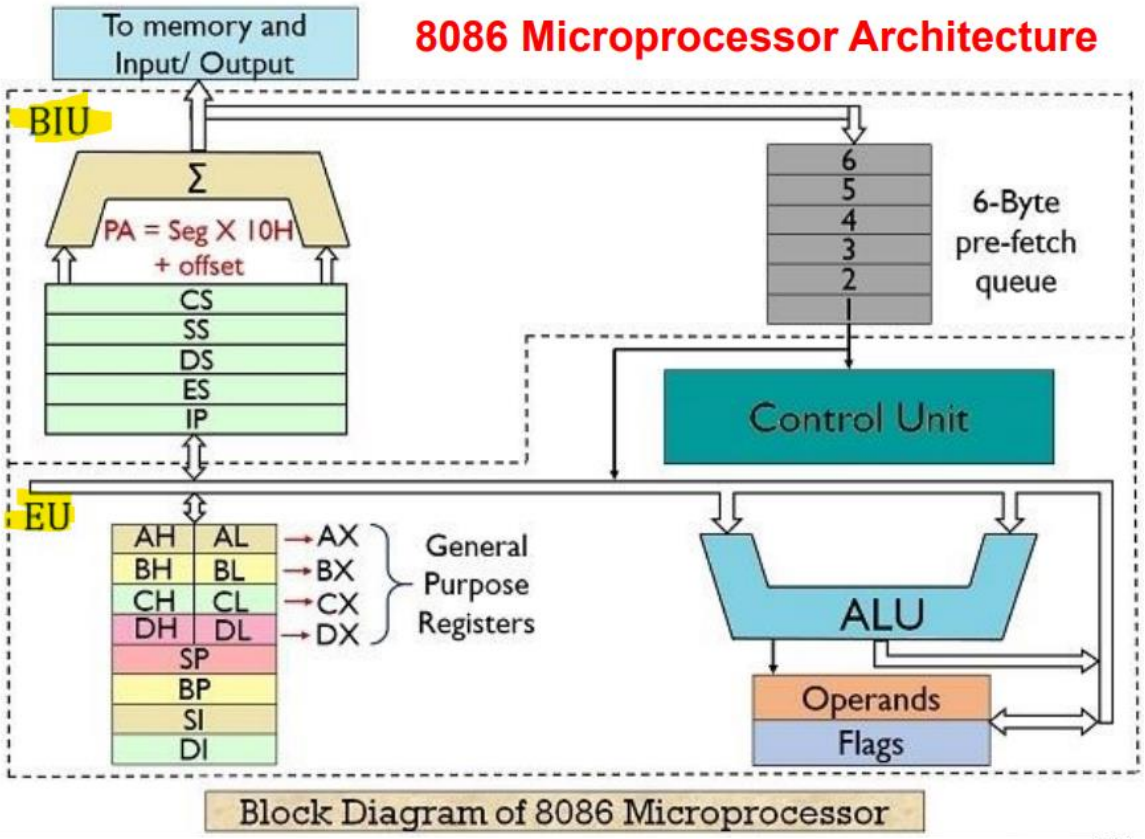
A group of (FIFO) in which up to 6 bytes of instruction code are pre-fetched from the memory ahead of time.

8086 Microprocessor blocks

1. **Execution Unit (EU)** executes instructions that have already been fetched by the BIU.
   Note// BIU and EU functions separately

   EU REGISTERS
   1. Accumulator Register (AX)
   2. Base Register (BX)
   3. Counter Register (CX)
   4. Data Register (DX)
   5. Stack Pointer (SP)
   6. Base Pointer (BP)
   7. Source Index (SI)
   8. Destination Index (DI)
2. **Bus Interface Unit (BIU)** reads/writes data from/to memory and I/O ports.
   Note// BIU fetches instructions

   BIU REGISTERS
   1. Code Segment Register (CS)
   2. Data Segment Register (DS)
   3. Stack Segment Register (SS)
   4. Extra Segment Register (ES)
   5. Instruction Pointer (IP)

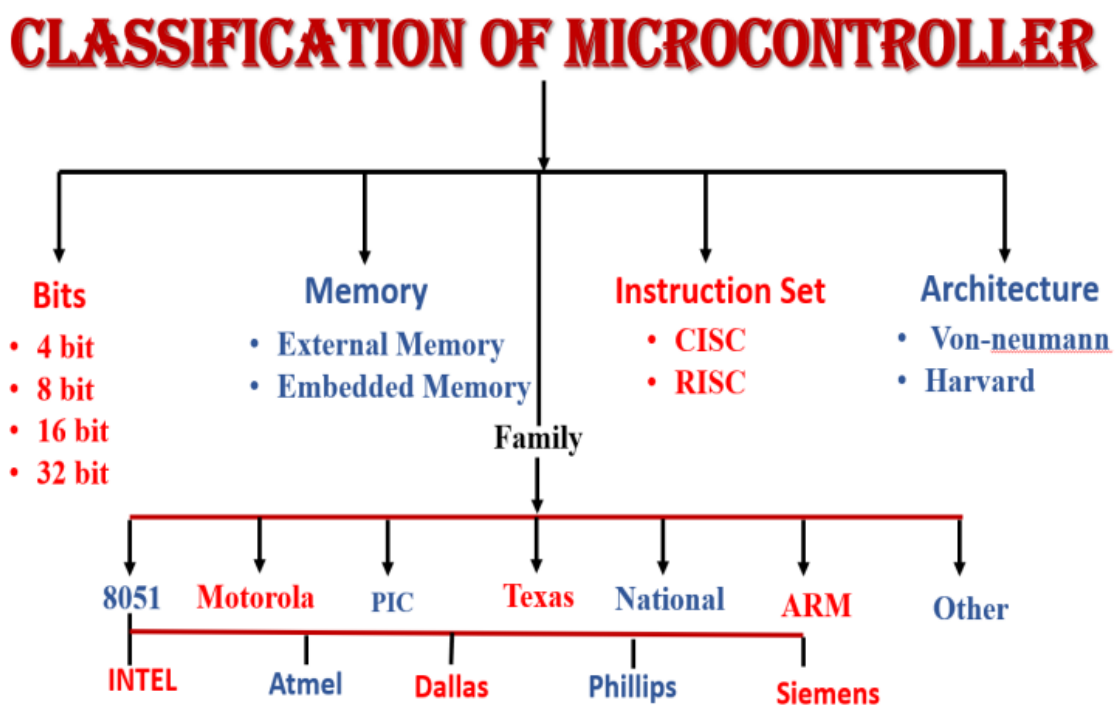


Block Diagram of 8086 Microprocessor

Chapter-2

Microprocessor VS Microcontroller

Microprocessor	Microcontroller
CPU is stand-alone, RAM, ROM, I/O, timer are separate	CPU, RAM, ROM, I/O and timer are all on a single chip
used in Personal Computers (PC)	used in an embedded system
based on Von Neumann model	based on Harvard architecture
expansive	cheap
general-purpose	single-purpose



μC based Embedded Systems

Is a special-purpose computer system usually inside the device it controls

Criteria

- Performance
- Reliability
- Availability
- Safety

Examples

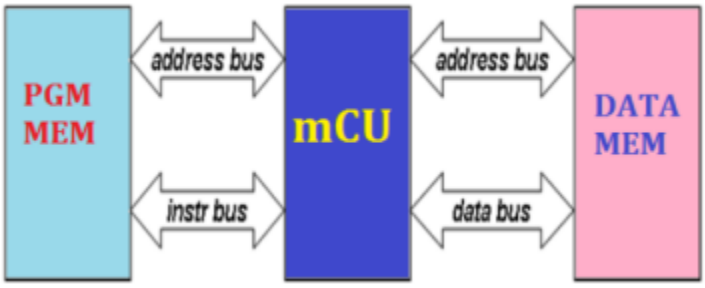
1) Camera 2) Smart Phone 3) Game machine 4) Electronic book , Tablet 4) Car 5) Printer 6) Television 7) Amusement machines 8) Broadcasting , Medical 9) Mobile Phones

Chapter-3

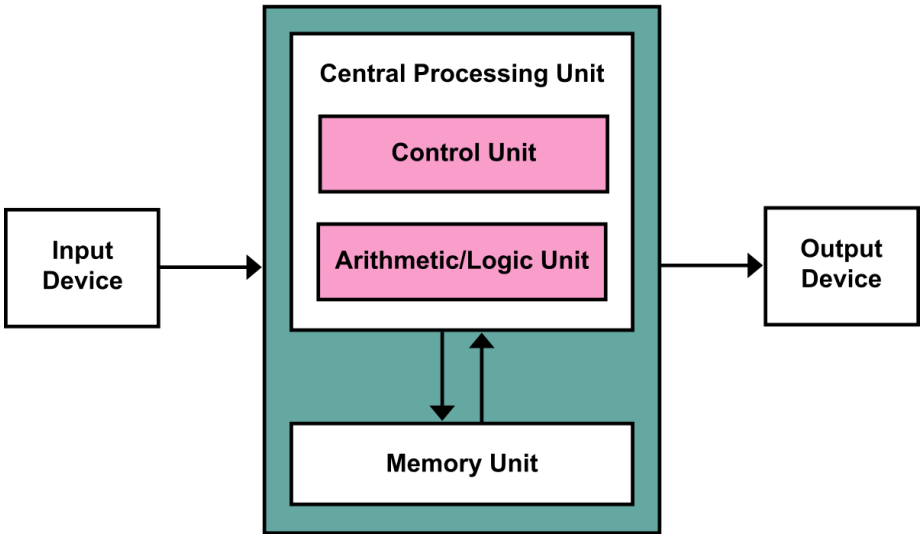
In **Harvard Architecture** the data and instructions are stored in separate memory units each with their own bus.

Advantages:

- Speeding up the data transfer rate,
- Permits the designer to implement different bus widths and word sizes for program and data memory space.



Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory

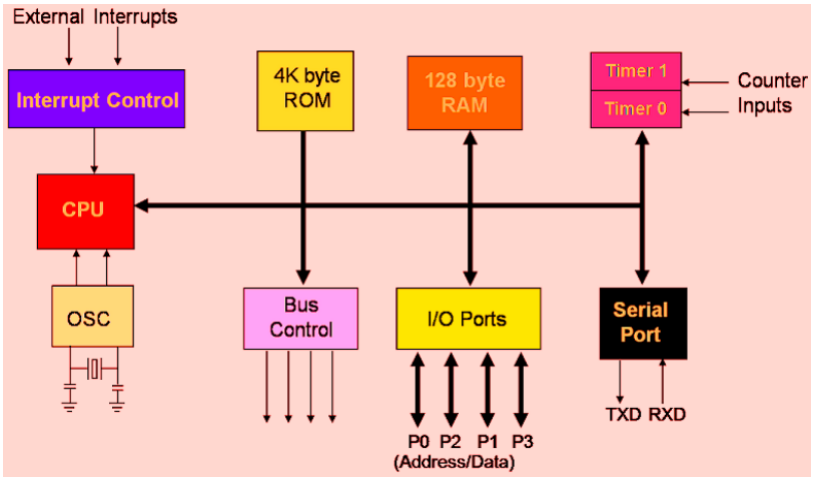


Important Features of 8051

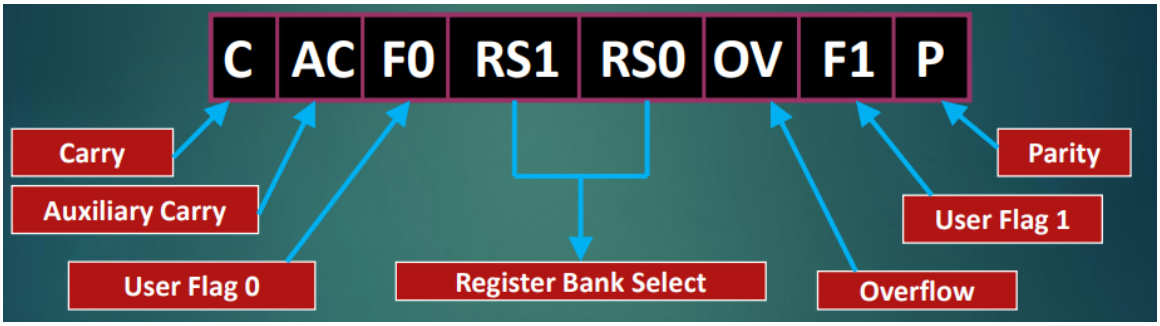
- 4K bytes ROM
- 128 bytes RAM
- Four 8-bit I/O ports
- Two 16-bit timers
- Serial interface
- 64K external code memory space
- 64K data memory space

Note//8051 Microcontroller is an 8-bit processor

Block Diagram of 8051



Program Status Word (PSW) in 8051



كلام الدكتور: "والله حتيقي لك في اختيار من متعدد"^

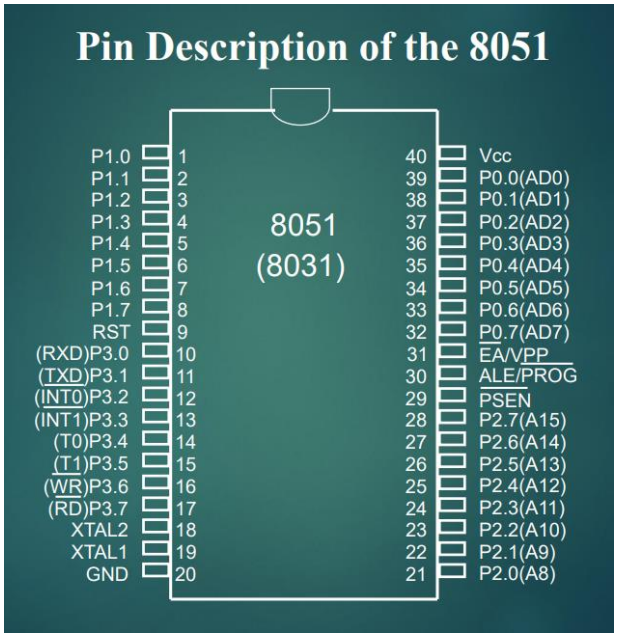
8051 Ports

- **Port 0** - external memory access (low address byte/data)
- **Port 2** - external memory access (high address byte)
- **Port 1** - general purpose I/O (pins 0, 1 for timer/counter 2)
- **Port 3** - Special features ∪ 0 - RxD: serial input
  1 - TxD: serial output
  2 - INT0: external interrupt
  3 - INT1: external interrupt
  4 - T0: timer/counter 0 external input
  5 - T1: timer/counter 1 external input
  6 - WR: external data memory write strobe
  7 - RD: external data memory read strobe

I/O Port Pins

The four 8-bit I/O ports P0, P1, P2 and P3 each uses 8 pins.

All the ports upon RESET are configured as output, ready to be used as input ports by the external device.



Port 0

Port 0 is also designated as AD0-AD7.

When connecting an 8051 to an external memory, port 0 provides both address and data.

Port 1 and Port 2

8051-based systems with no external memory connection:

- Both P1 and P2 are used as simple I/O.

8051-based systems with external memory connections:

- Port 2 must be used along with P0 to provide the 16-bit address for the external memory.
- P0 provides the lower 8 bits via A0 – A7.
- P2 is used for the upper 8 bits of the 16-bit address, designated as A8 – A15, and it cannot be used for I/O.

Port 3

Port 3 can be used as input or output.

Port 3 has the additional function of providing some extremely important signals.

128 Byte RAM

There are 128 bytes of RAM in the 8051.

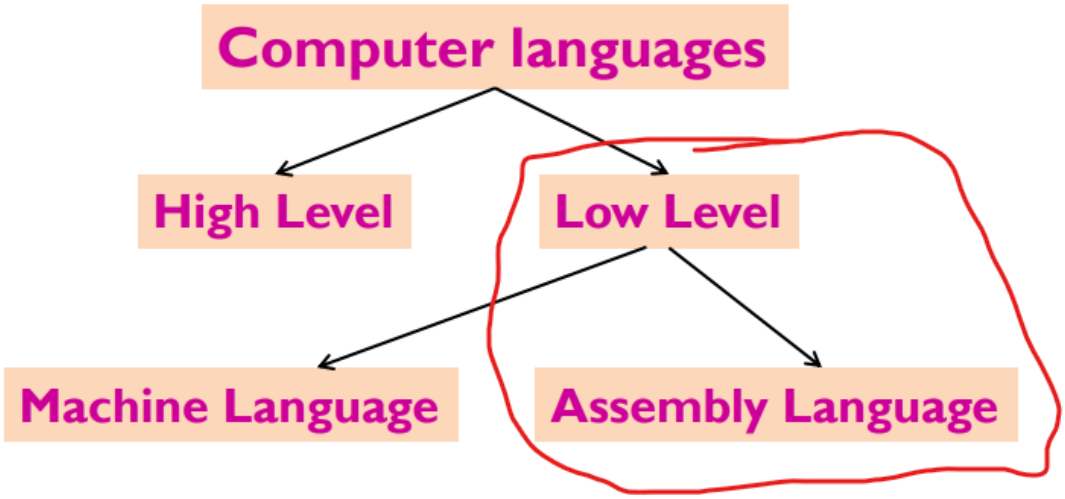
The 128 bytes are divided into 3 different groups

1. A total of **32 bytes** from locations 00 to 1F hex are set aside for **register banks** and the **stack**.
2. A total of **16 bytes** from locations 20H to 2FH are set aside for **bit-addressable** read/write memory.
3. A total of **80 bytes** from locations 30H to 7FH are used for read and write storage, called **scratch pad**

8051 Register Bank Structure

Chapter-4

Classification of Computer Languages



Assembly Language Statement

Example: ADD AX, BX

- ADD is the operation
- BX is called the source operand
- AX is called the destination operand
- The result is AX = AX + BX

Format of a Statement

Label	Instruction	Comment
Start:	MOV AX, BX	; copy BX into AX

Start is a user defined name and you only put in a label in your statement when necessary.

The symbol **:** is used to indicate that it is a label.

EMU8086 is an Emulator for 8086 Microprocessor Assembly Language Programming.

Note// The emu8086 consists of a tutorial and the reference for a complete instruction set.

Addressing Modes

The different ways in which a source operand is denoted in an instruction are known as addressing modes.

1. **Register Addressing**
   The instruction will specify the name of register which holds the data to be operated.
   MOV CL, DH
2. **Immediate Addressing**
   An 8-bit or 16-bit data is specified as part of the instruction.
   MOV DL, 08H
3. **Direct Addressing**
   The effective address is just a 16-bit number, written directly in the instruction.
   MOV BX, [1354H]
4. **Register Indirect Addressing**
   Name of the register which holds the Effective Address (EA) will be specified indirectly in a Register.
   MOV CX, [BX]
5. **Based Addressing**
   BX or BP is used to hold the base value for an effective address and the displacement will be specified.
   MOV AX, [BX + 08H]
6. **Indexed Addressing**
   Index Registers, SI or DI will be used as an operand. Displacement is added to the index value in SI or DI register to obtain the EA.
   MOV CX, [SI + 0A2H]
7. **Based Index Addressing**
   The effective address is computed from a base register (BX or BP), an index register (SI or DI) and the displacement.
   MOV DX, [BX + SI + 0AH]
8. **Relative Addressing**
   Effective address of an instruction is specified relative to the Instruction Pointer (IP) by an 8-bit signed displacement .
   JNZ 09H

Programming Example-2

Program to Print a string and Characters on the Screer

```
include emu8086.inc
PRINT 'Hello World!'
GOTOXY 10, 5
PUTC 65       ; 65 - is an ASCII code for 'A'
PUTC 'B'
RET           ; return to operating system.
```

Programming Example-3

Program Calling a Procedure

```
CALL m1
MOV AX, 2
ADD AX, BX
HLT           ; Halts the emulator.

m1 PROC
MOV BX, 5
RET           ; return to caller.
```

Programming Example-4

Program to demonstrate Unconditional Jump

```
org 100h
mov ax, 5       ; set ax to 5.
mov bx, 2       ; set bx to 2.
jmp calc        ; go to 'calc'.
back:  jmp stop ; go to 'stop'.
calc:  add ax, bx ; add bx to ax.
jmp back        ; go 'back'.
stop:
ret             ; return to operating system.
```

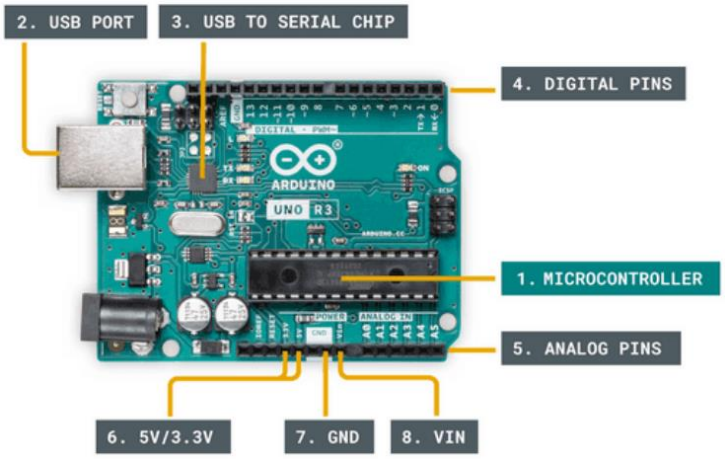
Chapter-5

Arduino

Arduino is an open-source prototyping platform in electronics based on easy-to-use hardware and software.

Note// Arduino boards are generally based on microcontrollers from Atmel Corporation like 8, 16 or 32 bit AVR architecture based microcontrollers.

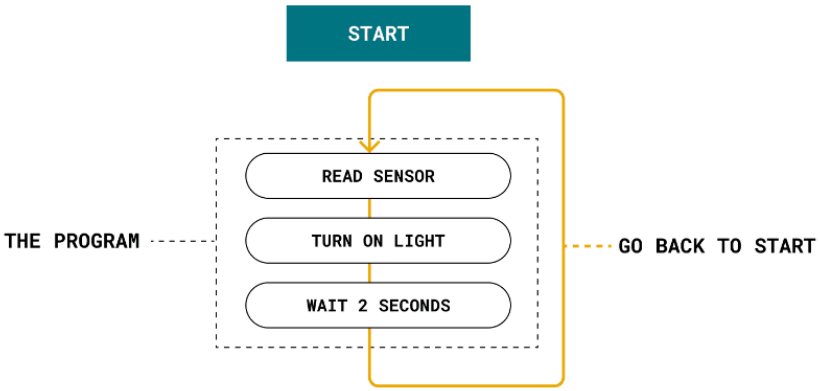
Key components of an Arduino board.

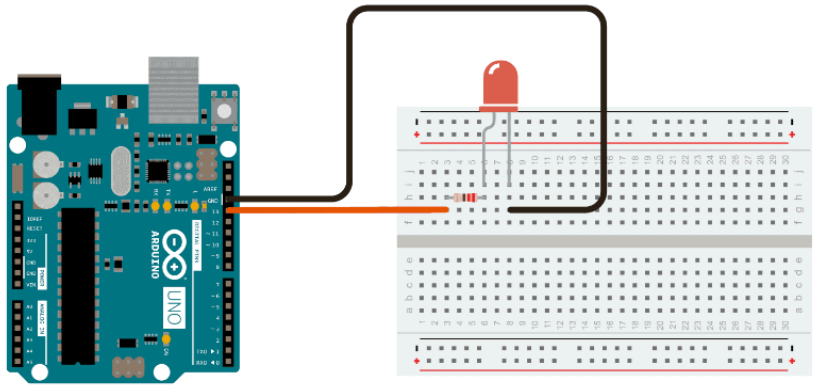


1. **Microcontroller**
   is the brain of an Arduino, and is the component that we load programs into.
2. **USB port**
   used to connect your Arduino board to a computer.
3. **USB to Serial chip**
   it helps translating data that comes from a computer to the on-board microcontroller.
4. **Digital pins**
   pins that use digital logic Commonly used for switches and to turn on/off an LED.
5. **Analog pins**
   pins that can read analog values in a 10 bit resolution (0-1023).
6. **5V / 3.3V pins**
   these pins are used to power external components.
7. **GND**
   also known as ground.
8. **VIN**
   stands for Voltage in, where you can connect external power supplies.

Basic Operation

- Most Arduino boards are designed to have a single program running on the microcontroller.
- The program can be designed to perform one single action, such as blinking an LED.
- It can also be designed to execute hundreds of actions in a cycle.
- The scope varies from one program to another.
- The program that is loaded to the microcontroller will start execution as soon as it is powered.





- When the pin is set to a **HIGH state**, the microcontroller on the Arduino board will allow an electric current to flow through the circuit, which turns on the LED.
- When the pin is set to a **LOW state**, the LED will turn off, as an electric current is not flowing through the circuit.

All communication between electronic components are facilitated by **electronic signals**.

There are two main types of electronic signals: Analog & Digital.

Sensors & Actuators

1. **Sensors** a sensor, in simple terms, is used to sense its environment, meaning it records a physical parameter, for example temperature, and converts it into an electronic signal. referred to as input.
2. **Actuators** an actuator, in simple terms, is used to actuate or change a physical state. referred to as output.

as example when you press the light button (**sensor**) the light (**actuator**) will turn on

Note// Sensors and actuators, are typically referred to as inputs and outputs.

Serial communication protocols

uses the digital signals to send data.

- The most common are **UART, SPI & I2C.**

- **UART**(Universal Asynchronous Receiver/Transmitter) protocol is used to send data between a computer and Arduino board.
- **SPI** : Serial Peripheral interface.
- **I2C** : Inter-Integrated Circuit
- The **SPI** and **I2C** protocols are used for communication between both internal and external components.
- The communication is handled by something called a **serial bus**, which is attached to a specific pin on the Arduino.

Memory

The "standard" Arduino typically has two memories: **SRAM** and **Flash memory**.

SRAM (Static Random-Access Memory) is used to store the value of a variable.

Note// When powered off, this memory resets.

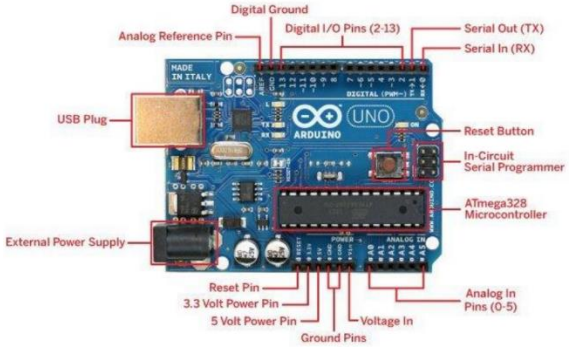
Flash memory is primarily used to store the main program, or the instructions for the microcontroller.

Note// This Flash memory is not erased when powered off so that the instructions for the microcontroller are executed as soon as the board is powered.

Arduino UNO

Arduino UNO is a basic and inexpensive Arduino board and is the most popular of all the Arduino boards.

- Microcontroller used in UNO is ATmega328P, which is an 8-bit microcontroller based on the AVR architecture.
- UNO has 14 digital input – output (I/O) pins
- Out of these 14 pins, 6 pins are capable of producing PWM signal.
- All the digital pins operate at 5V and can output a current of 20mA.



The Digital I/O pins functions

- **Pins 0 and 1** are used for serial communication.
- **Pins 2 and 3** are used for external interrupts.
- Six of the 14 digital I/O **Pins i.e. 3, 5, 6, 9, 10, and 11** can provide 8-bit PWM output.
- **Pins 10, 11, 12 and 13** are used for SPI communication.
- **Pin 13** has a built-in LED connected to it. When the pin is HIGH, the LED is turned on and when the pin is LOW, it is turned off.