Architecture of a Modern Microcontroller.

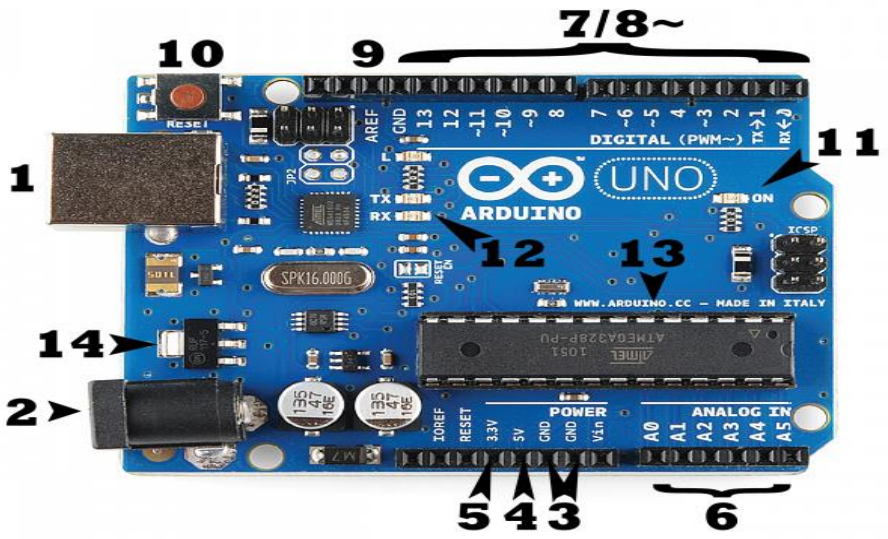
Prepared By- Nadia Nowshin

Basics features of Arduino Uno

- Consists of both a programmable microcontroller (ATMega328) and a piece of software, or IDE

- It uses a computer to write and upload code (written in the easier version of C++) to the microcontroller board and doesn't need programmer/burner.

- It uses a computer to write and upload code (written in the easier version of C++) to the microcontroller board and doesn't need programmer/burner.
- Operating voltage- 5Volt (7-12 Volt is recommended).
- Digital pin- 14 (of which 6 provides PWM output).
- DC Current per I/O pin= 20mA
- DC Current 3.3V pin= 50mA



Arduino Uno pin configuration

1. Power (USB/Barrel Jack)
Every Arduino board needs a power source through a USB cable coming from your computer **(1)** or a wall power supply that is terminated in a barrel jack **(2)**.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF):

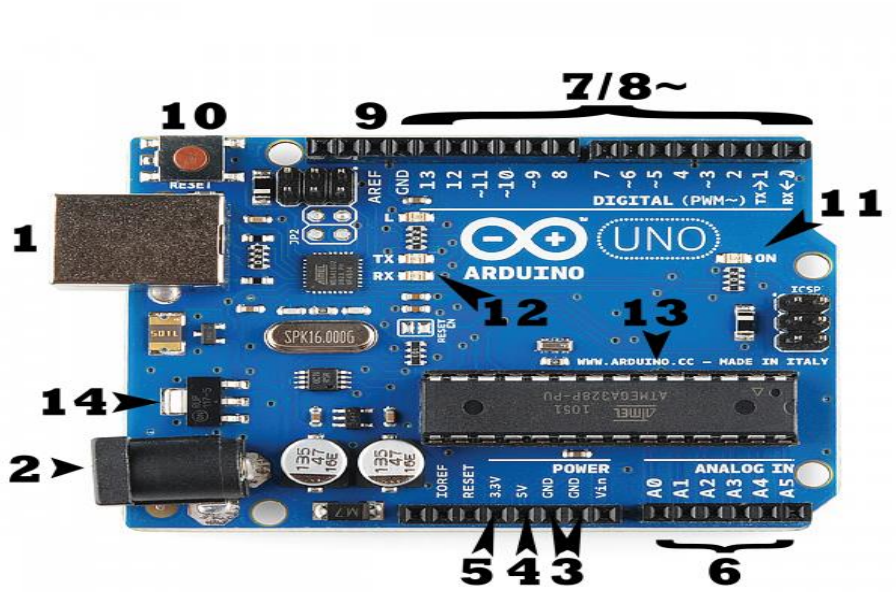
GND (3): Short for 'Ground' and **5V (4) & 3.3V (5)**: used as supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power.

Analog (6): The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog Input pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

Digital (7): Across from the analog pins are the digital pins (0 through 13 on the UNO).

PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as Pulse-Width Modulation (PWM, think of these pins as being able to simulate analog output (like fading an LED in and out).

AREF (9): Most of the time you can leave this pin alone.



Arduino Uno pin configuration

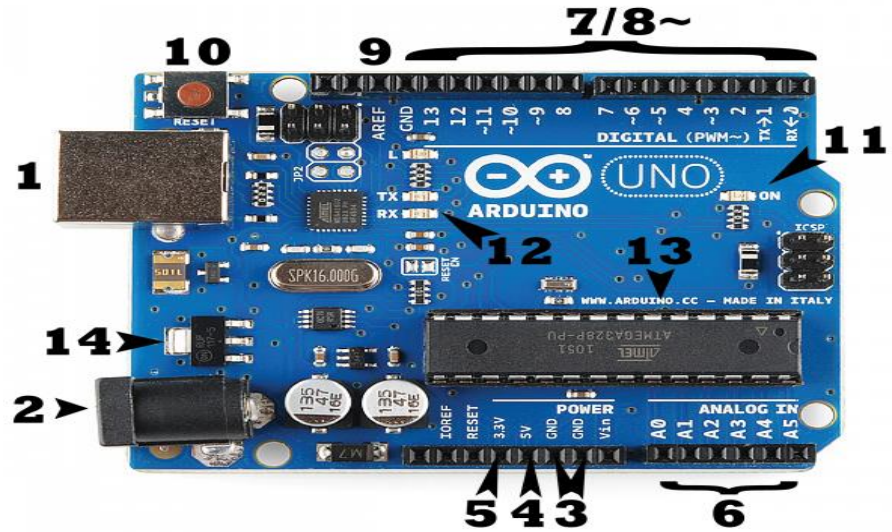
Reset Button:
Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino.

Power LED indicator (11) : If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit.

TX RX LEDs(12) : These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

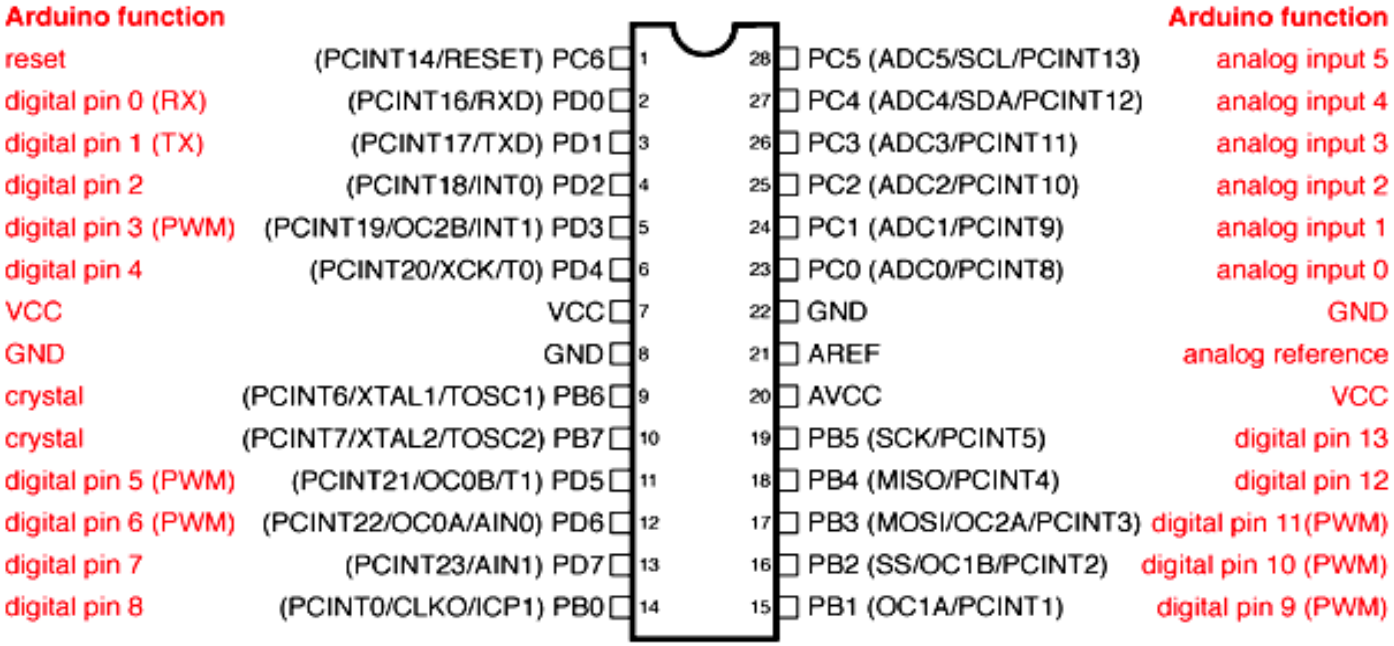
Main IC (Integrated Circuit) **(13) :** The main IC on the Arduino is the ATmega328 microcontroller of the ATMEL company., where we have to upload the code.

Voltage Regulator:
The voltage regulator **(14)** is used to control the amount of voltage that is let into the Arduino board. But don't hook up your Arduino to anything greater than 20 volts.



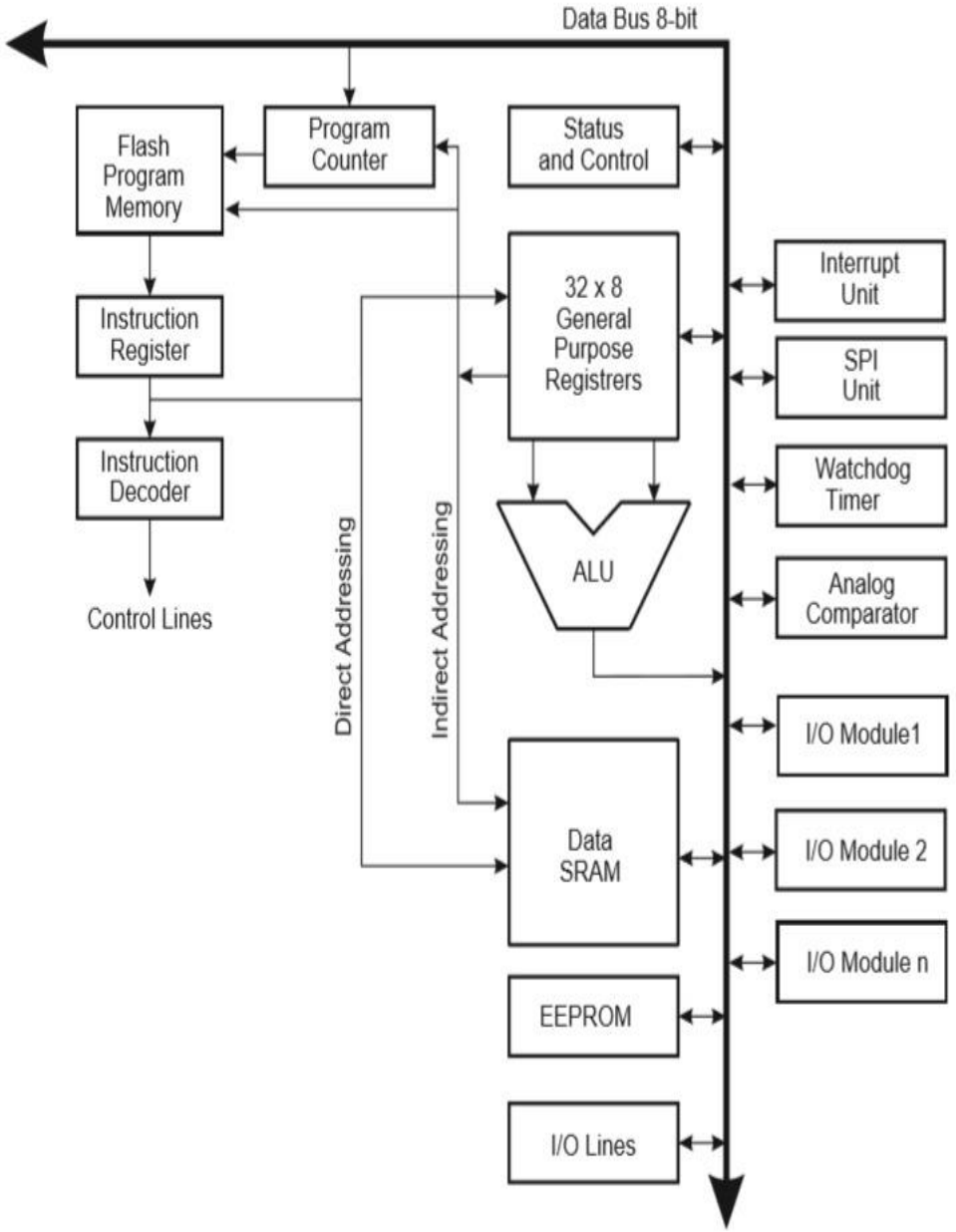
Arduino Uno and ATmega328P chip.

ATMega328P and Arduino Uno Pin Mapping



- **Pins- 28 pins, CPU- 8-bit AVR,**
- **Communication Interface-** Master/Slave SPI Serial Interface(17,18,19 PINS) [Can be used for programming this controller]
- **CPU Speed**- 1MHz or 1MIPS
- **RAM**-2Kbytes Internal SRAM
- **Program Memory or Flash memory**- 32Kbytes[10000 write/erase cycles]

Internal Architecture of AtMega328



1. The ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC (reduced instruction set computer) architecture.

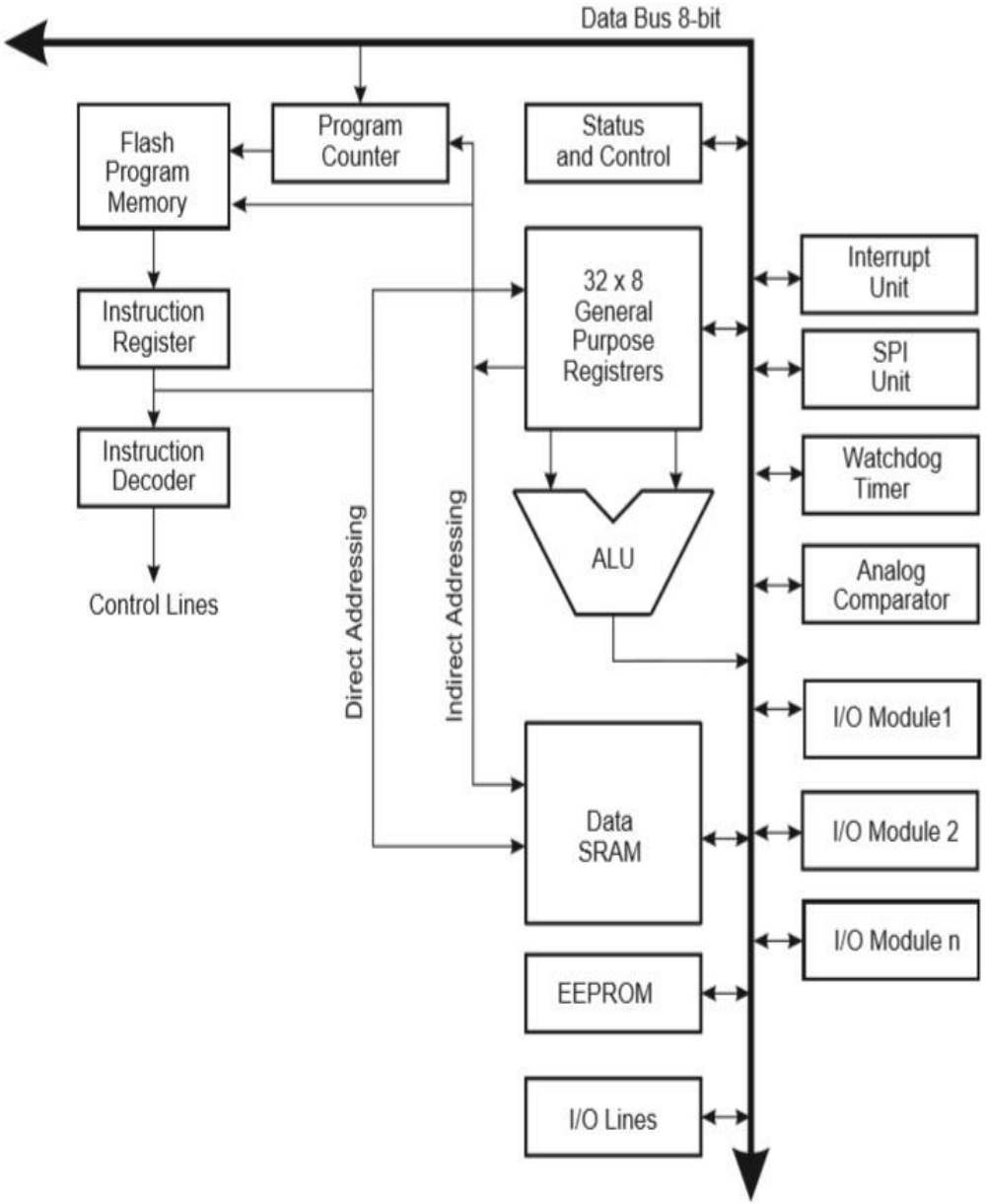
2. The clock is controlled by an external 16MHz Crystal Oscillator.

3. The data is uploaded in serial via the port.

4. Instructions are sent to instruction register and it decodes the instructions on the same clock pulse.

5. In general purpose registers the registers are of 8-bit (used to store data for normal calculations and results) but there are 3 16-bit registers also (used to store data of timer counter).

Internal Architecture of AtMega328



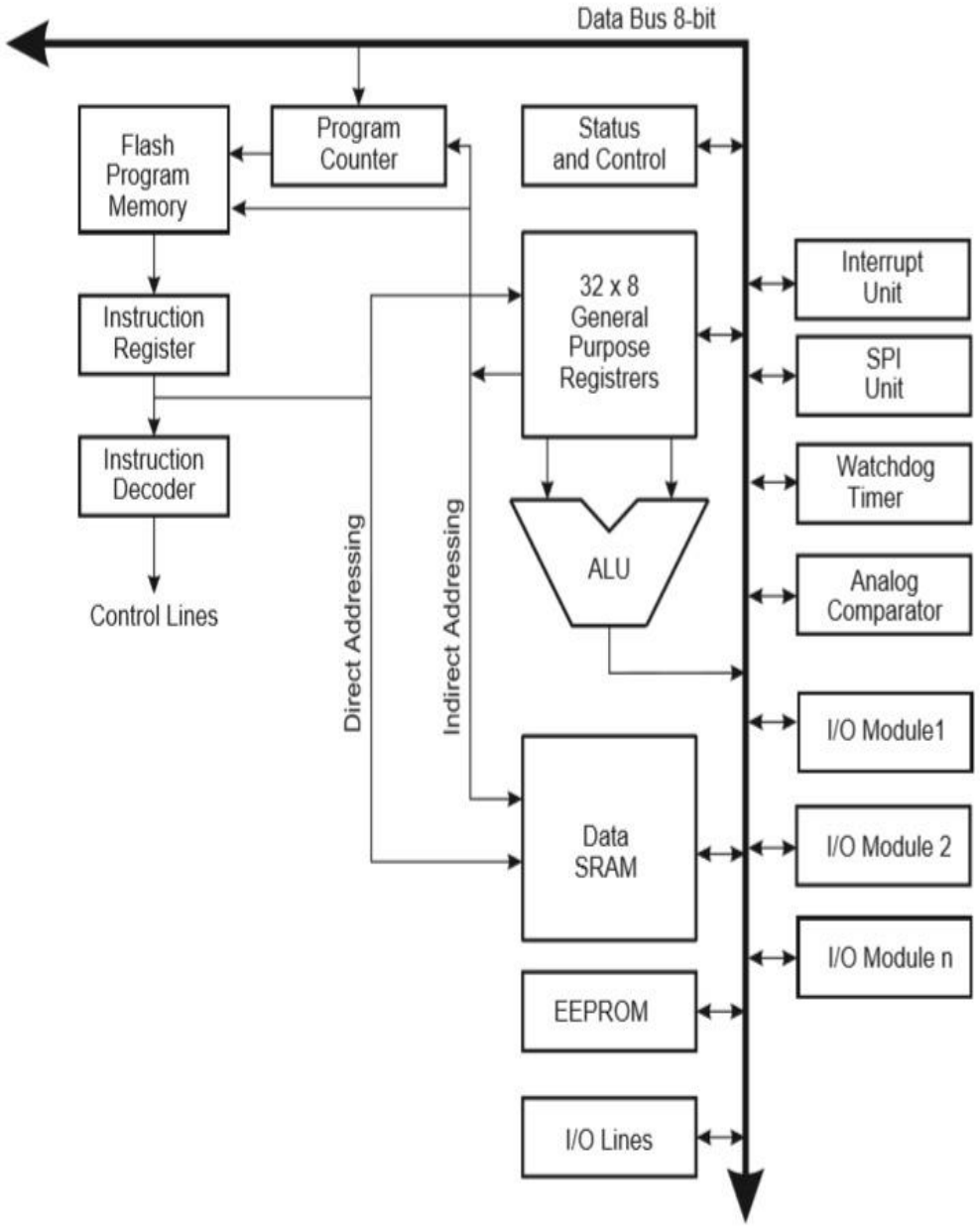
6. EEPROM stores data permanently even if the power is cut out.

7. Interrupt Unit checks whether there is an interrupt for the execution of instruction to be executed in ISR (Interrupt Service Routine).

8. Serial Peripheral Interface (SPI) is used to send data between microcontrollers and small peripherals such as Camera, Display, SD cards, etc. It uses separate clock and data lines, along with a select line to choose the device you wish to talk to.

9. Watchdog timer is used to detect and recover from MCU malfunctioning..

Internal Architecture of AtMega328

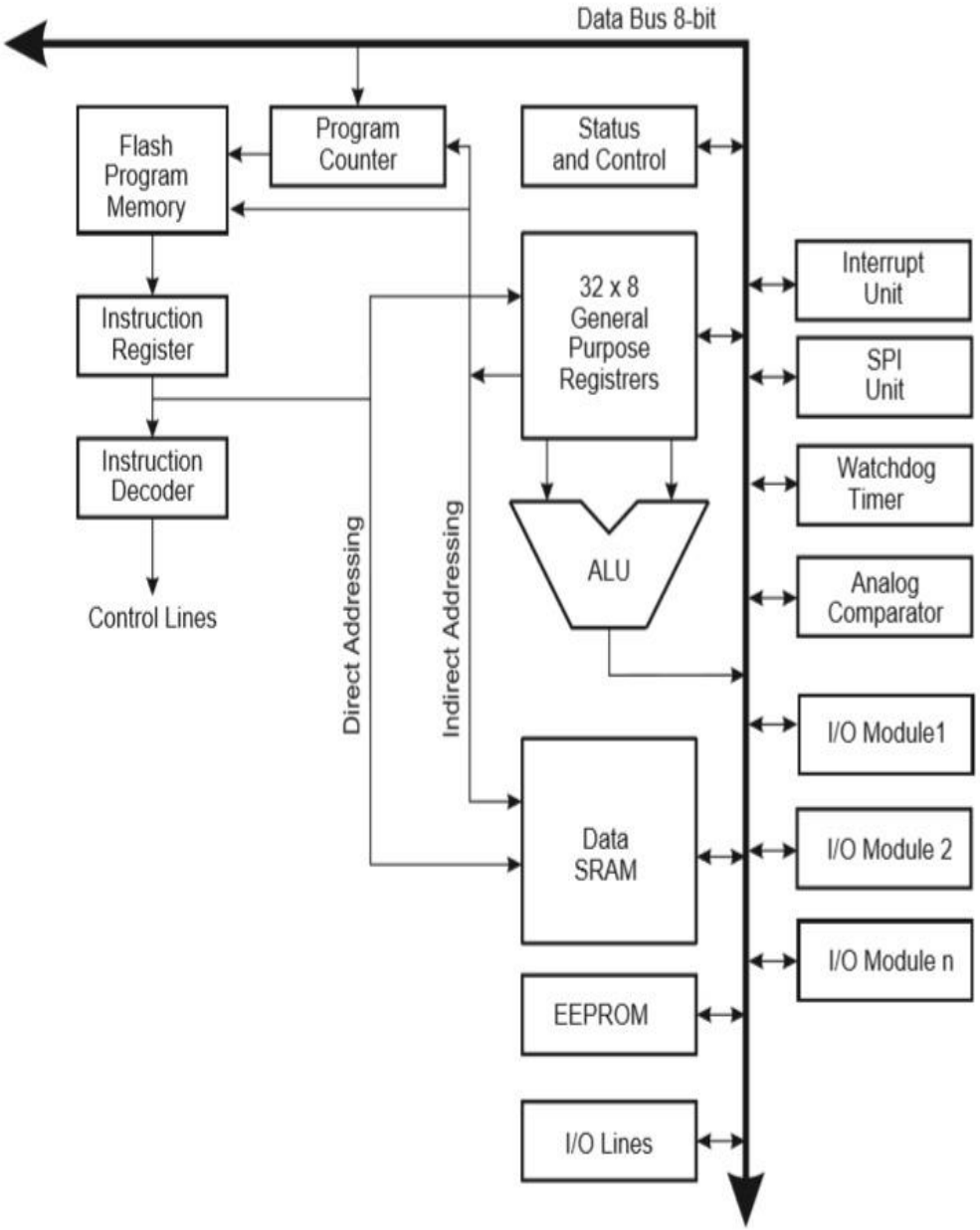


10. Analog comparator compares the input values on the positive and negative pin, when the value of positive pin is higher the output is set.

11. Status and control is used to control the flow of execution of commands by checking other blocks inside the CPU at regular intervals.

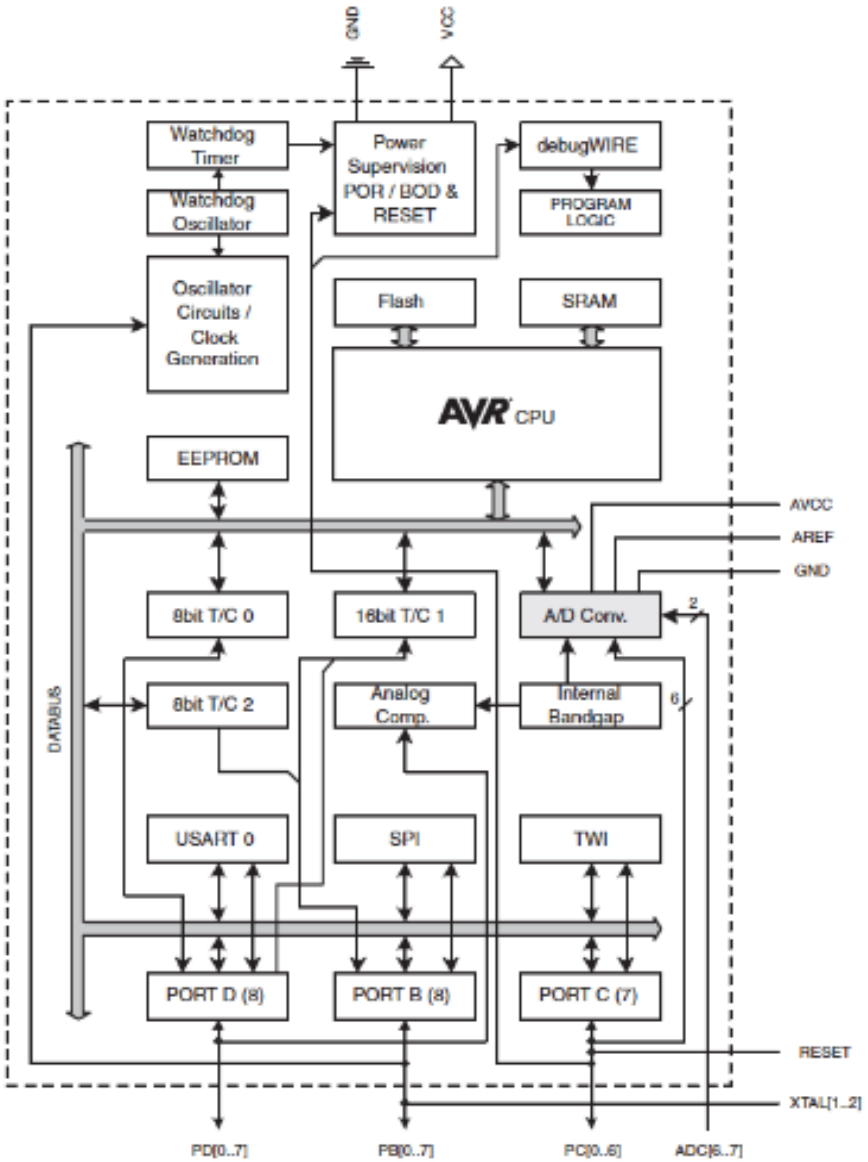
12. ALU (Arithmetic and Logical unit)The high performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations b/w general purpose registers are executed. The ALU operations are divided into 3 main categories – arithmetic, logical and bit-function.

Internal Architecture of AtMega328



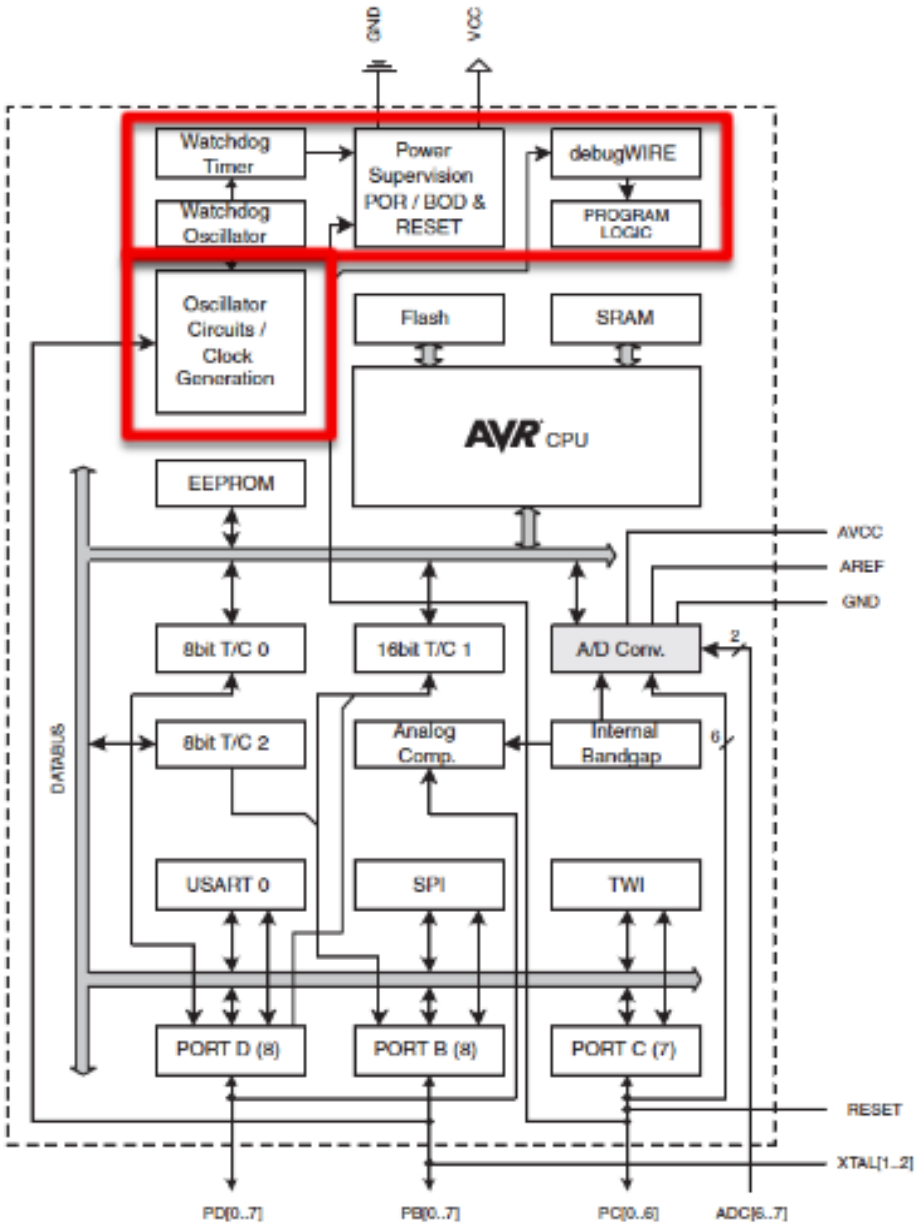
13. I/O pins The digital inputs and outputs (digital I/O) on the Arduino are what allow you to connect the Arduino sensors, actuators, and other ICs. Learning how to use them will allow you to use the Arduino to do some really useful things, such as reading switch inputs, lighting indicators, and controlling relay outputs..

AVR Architecture



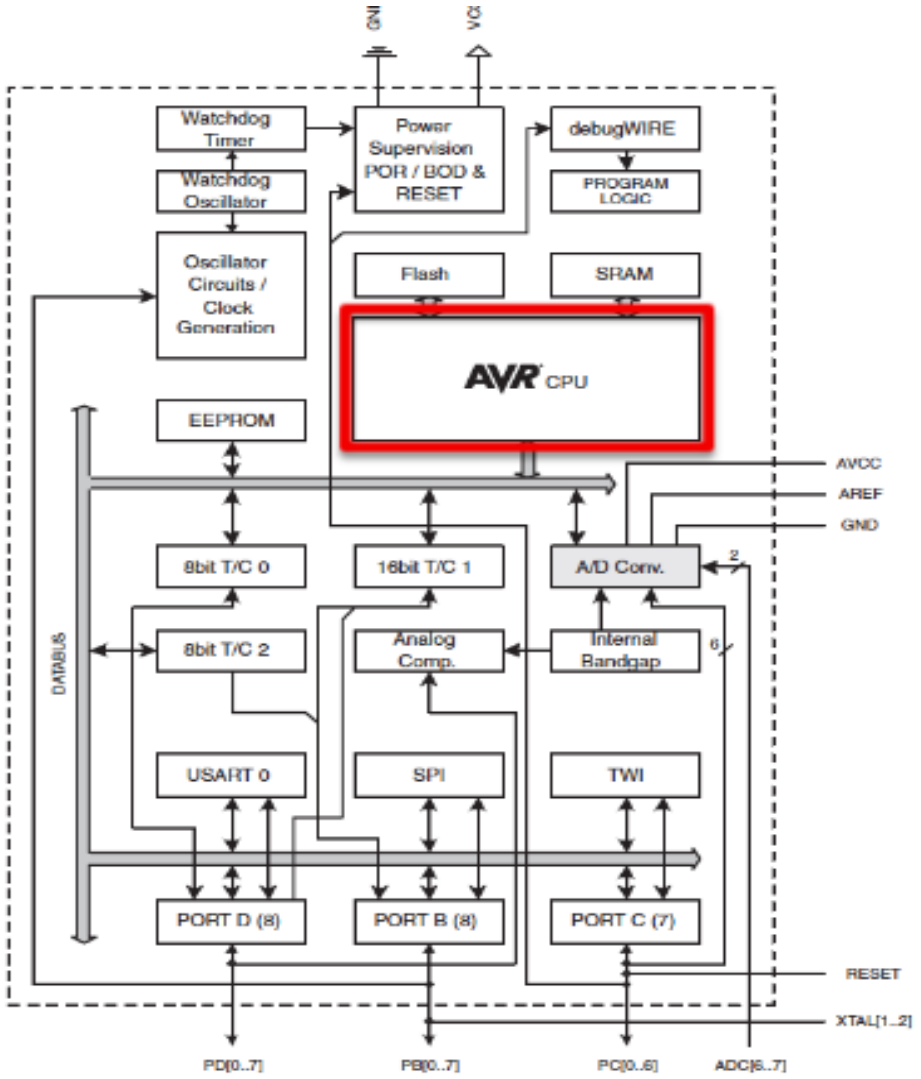
AVR Architecture

- Clocks and Power



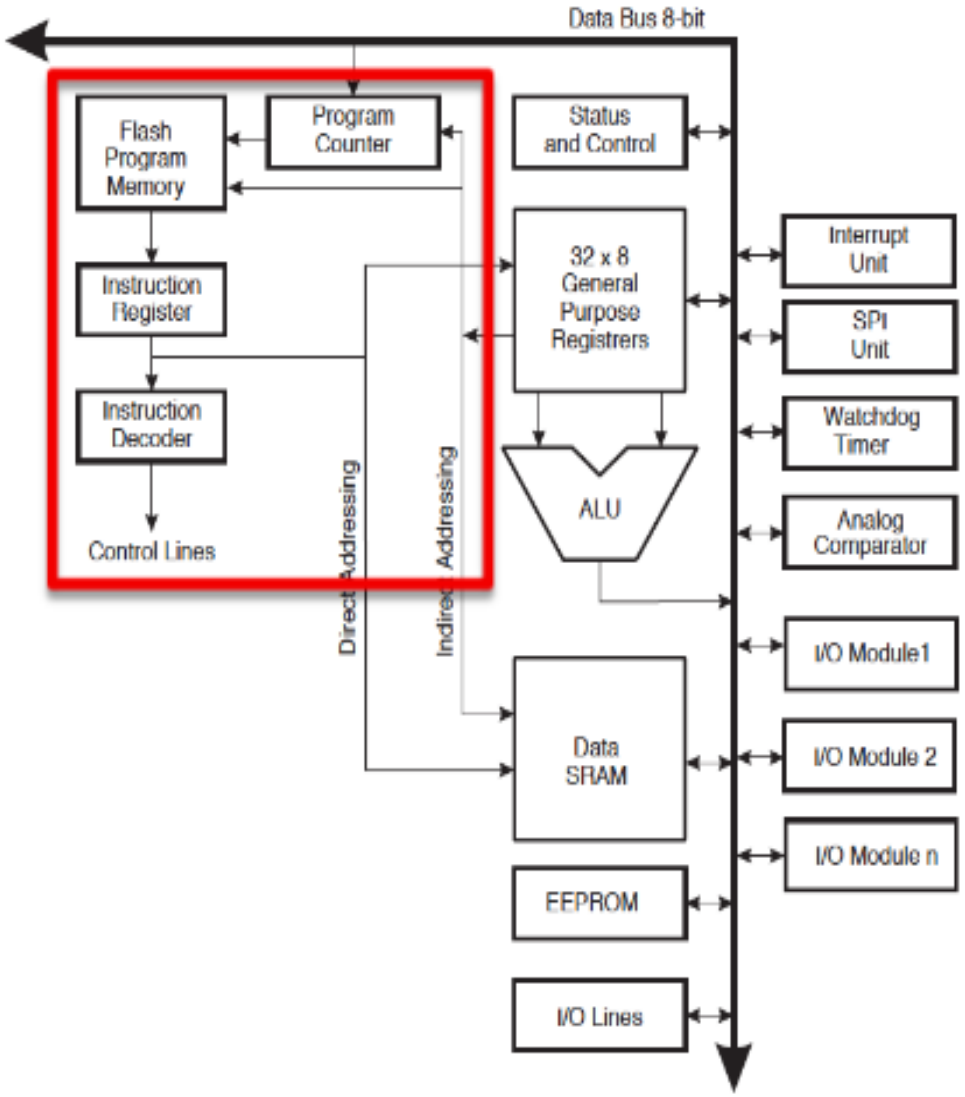
AVR Architecture

- CPU



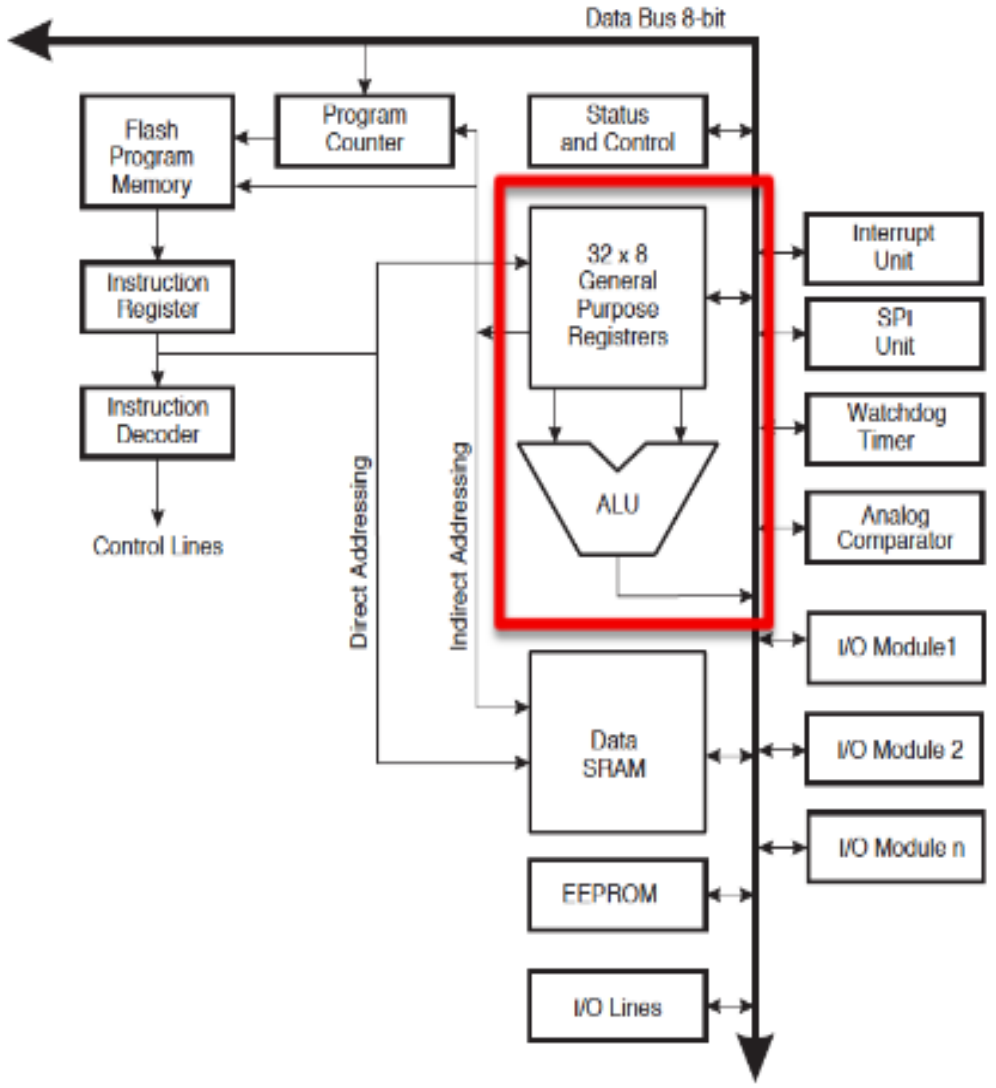
AVR CPU

- Instruction Fetch and Decode



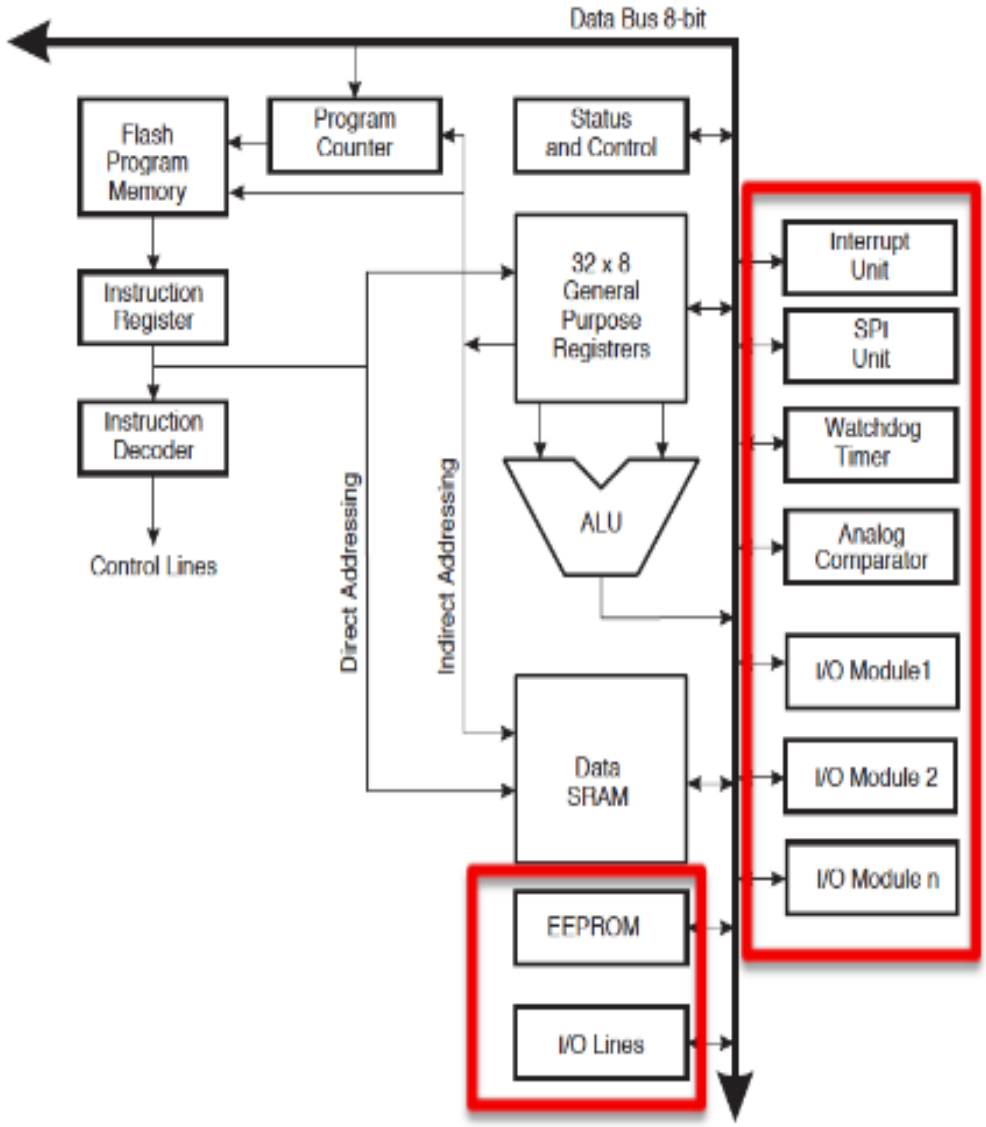
AVR CPU

- ALU Instructions



AVR CPU

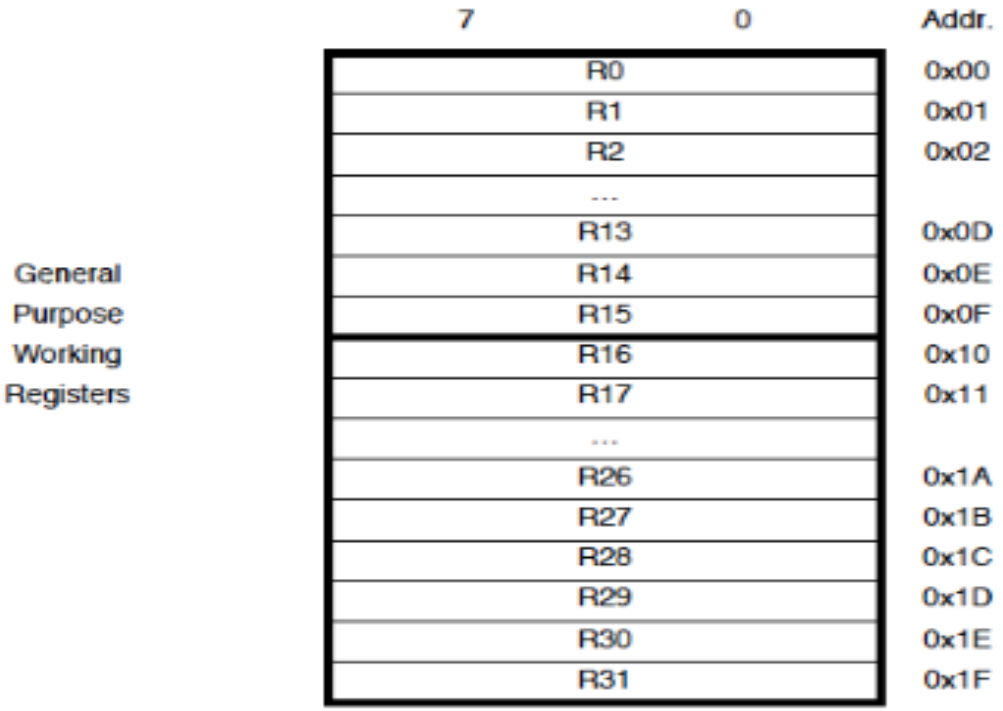
- I/O and special functions



AVR Register File

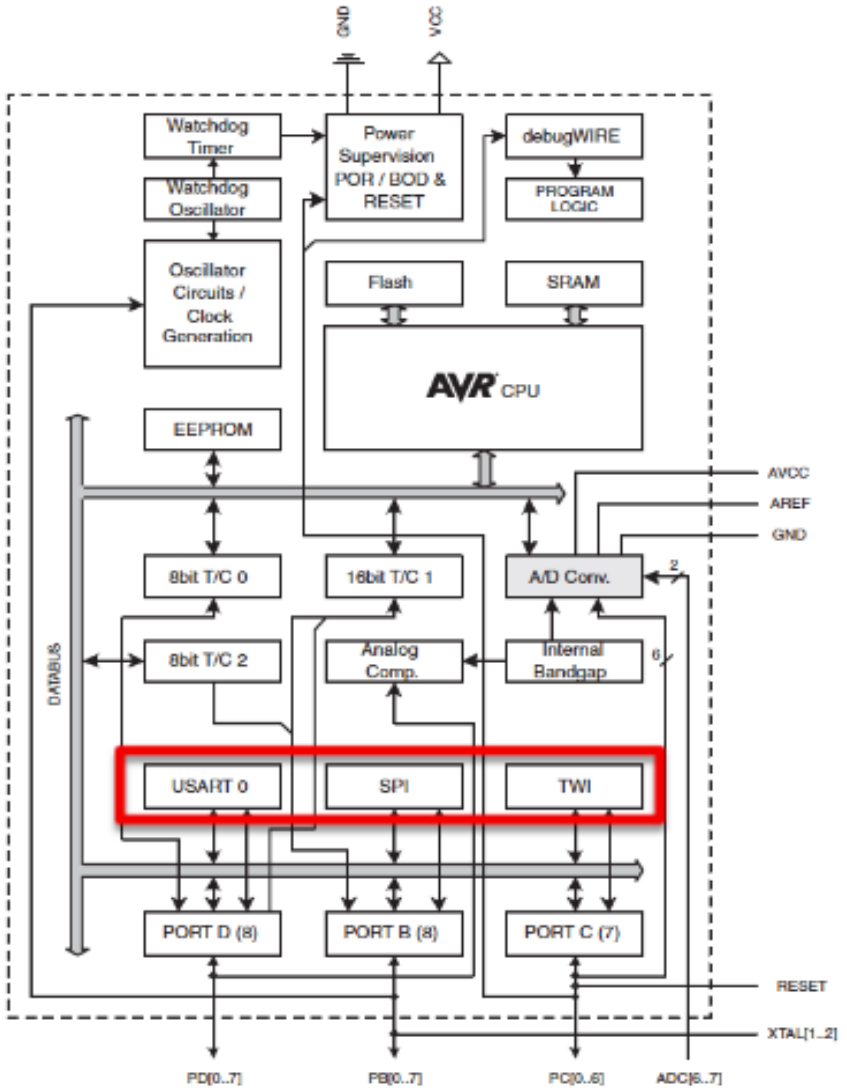
- 32 8-bit GP registers
- Part of SRAM memory space

Figure 6-2. AVR CPU General Purpose Working Registers



AVR Architecture

- Special I/O support
  - Serial protocols
- Uses special pins
- Uses timers
- Beyond scope of this course



Arduino C Programs

- Arduino calls these "sketches"
  - Basically C with libraries
- Program structure
  - Header: declarations, includes, etc.
  - setup()
  - loop()
- Setup is like Verilog initial
  - executes once when program starts
- loop() is like Verilog always
  - continuously re-executed when the end is reached

Blink Program

```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off
  delay(1000);                  // wait for a second
}
```

References:

A. Arduino.cc
B. Wikipedia.com
C. Datasheet of ATmega328/p