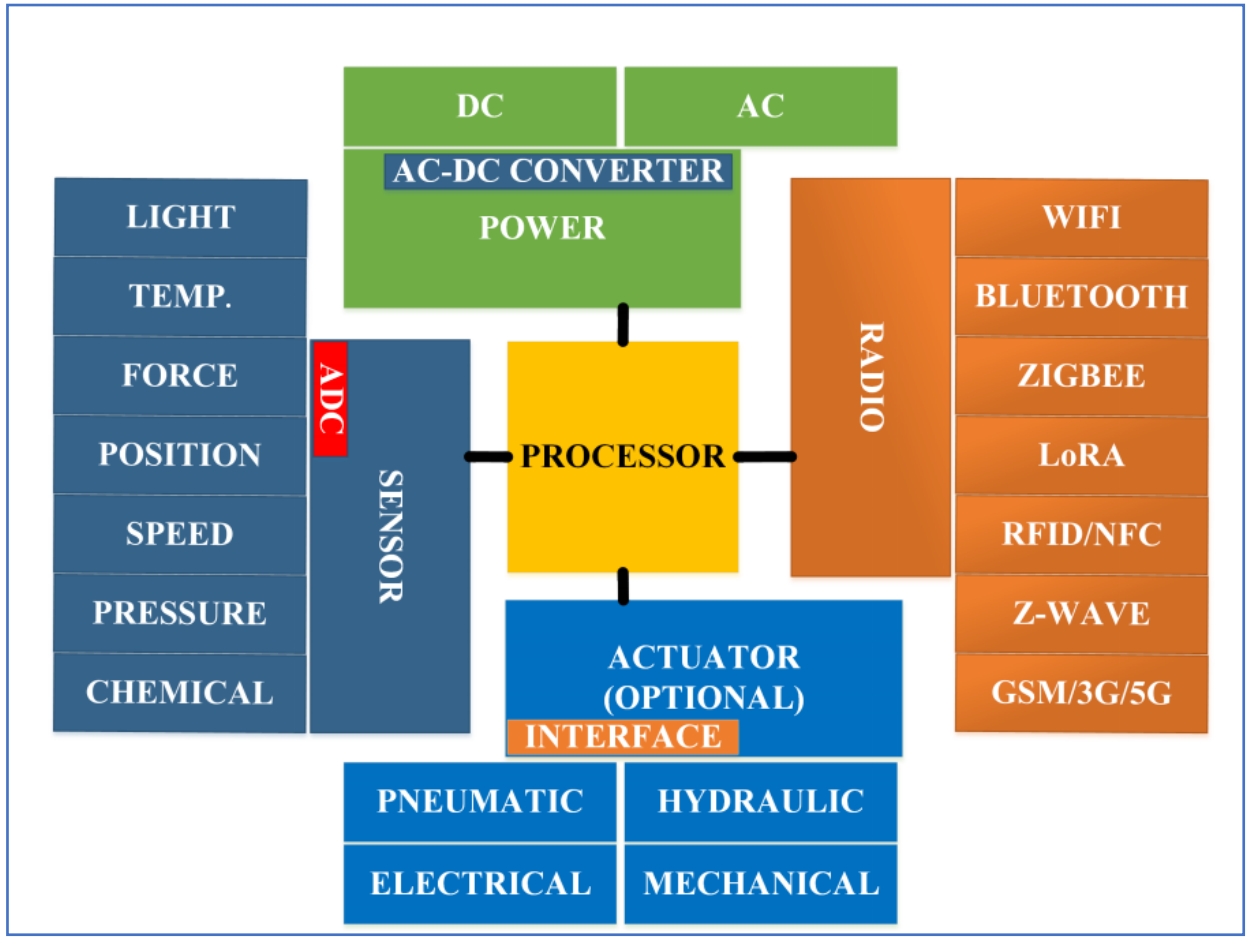
Microcontrollers

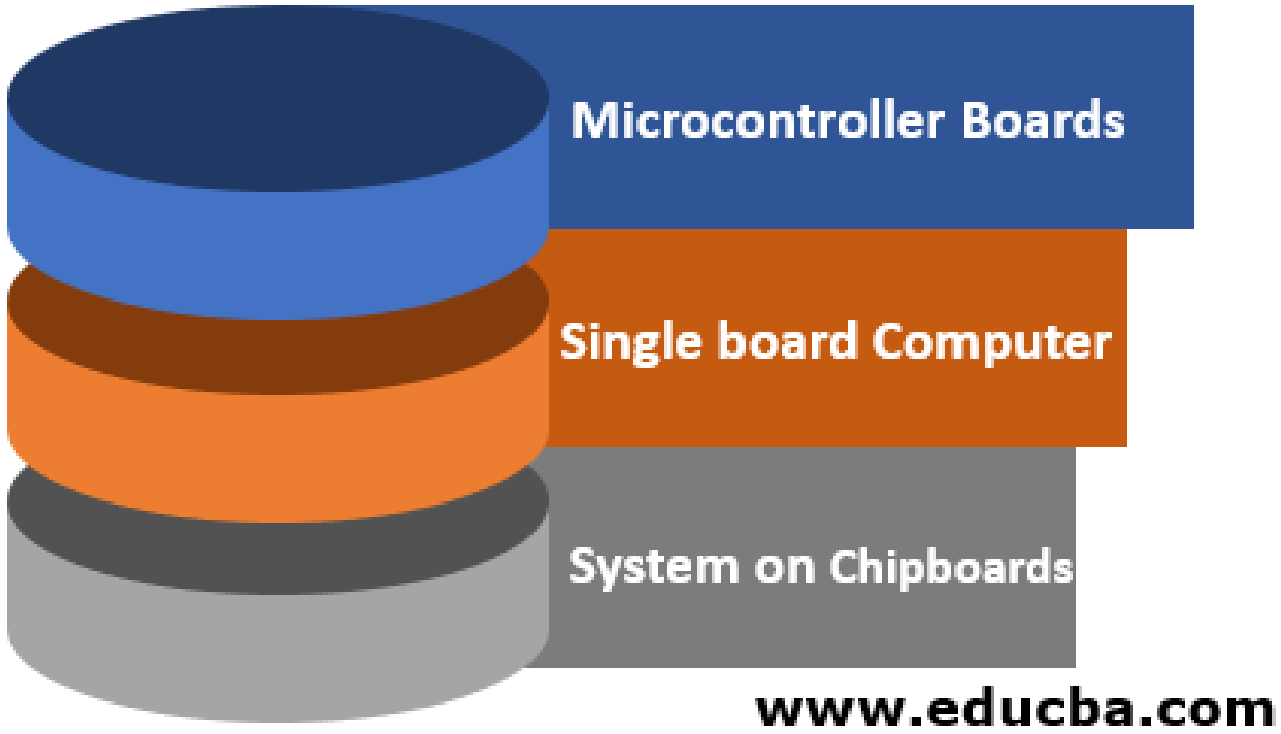
Dr. Viranchi C. Pandya

Nirma University

Typical Sensor Node



Microcontroller Boards



Choosing a Microcontroller

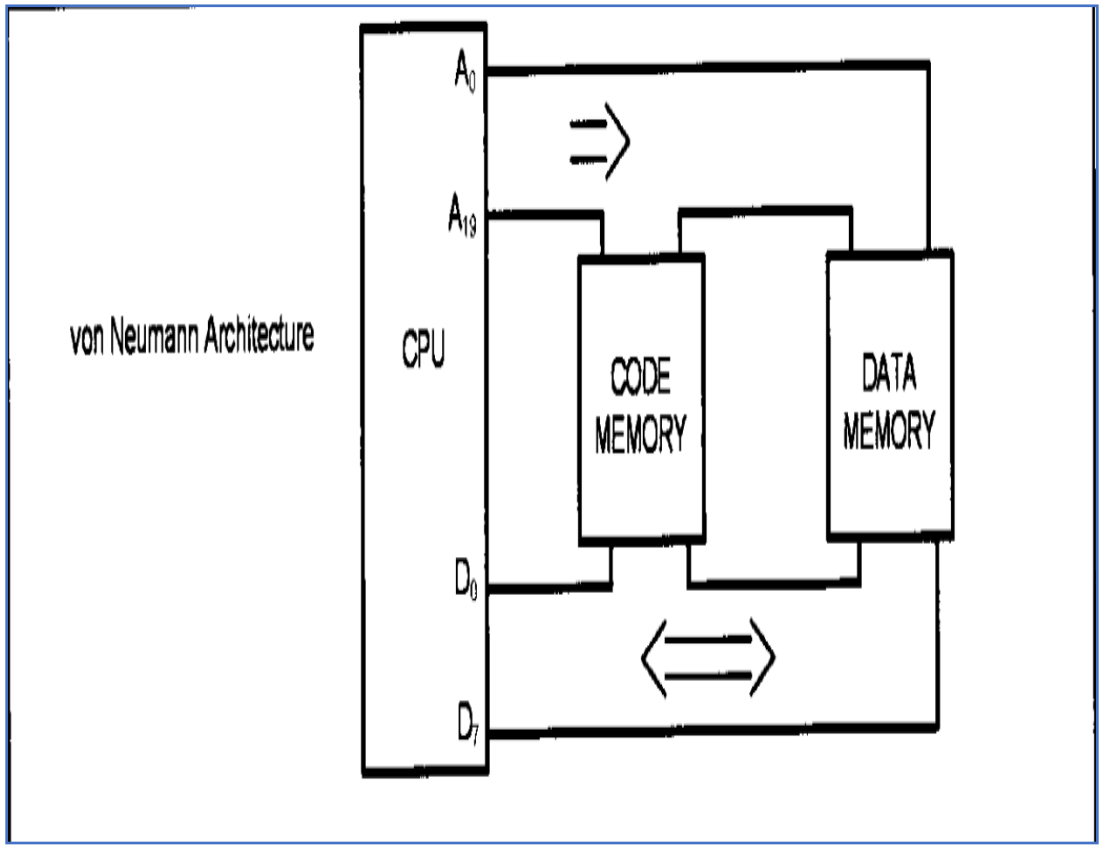
- Speed
- Packaging
- Power Consumption
- On chip RAM and Flash
- GPIO
- Timers
- Other Peripherals
- Easy upgradation
- Cost per unit
- Availability of assembler, debugger, emulator, technical support
- Readily available in market with needed quantities

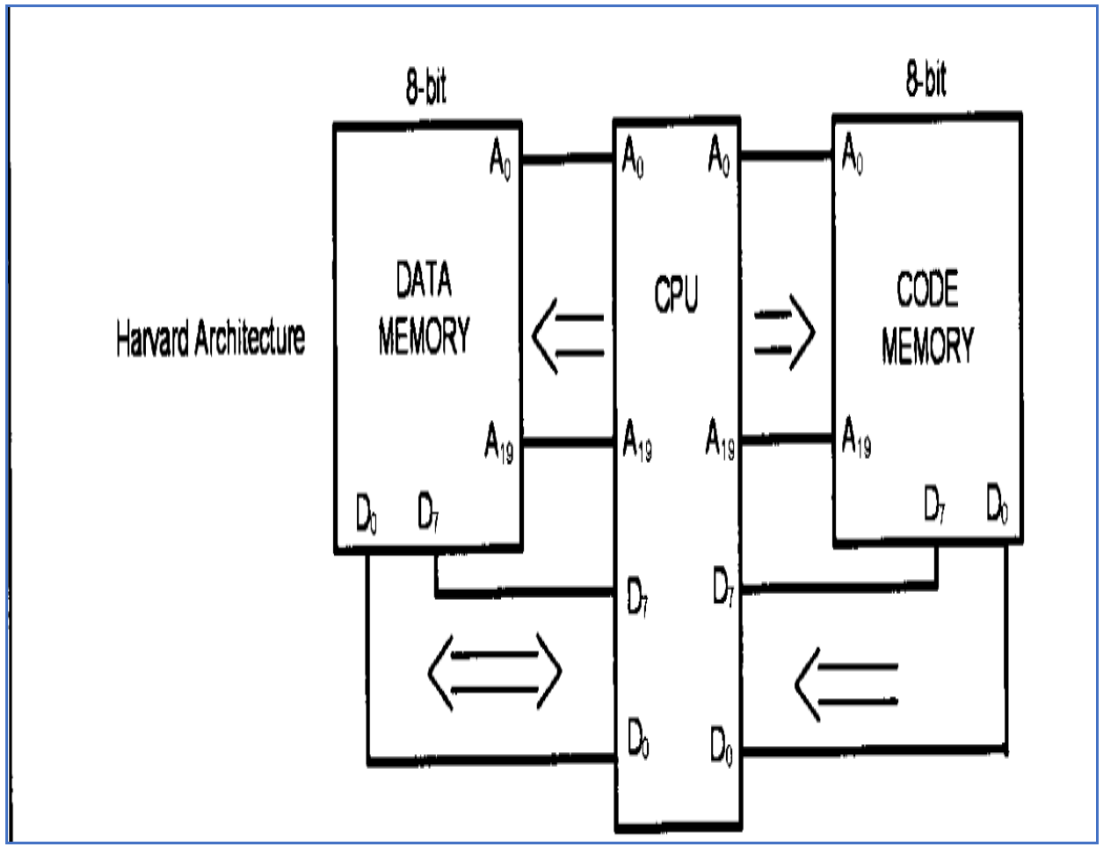
How to select Board?

https://www.nabto.com/best-iot-boards/

- **Determine application's requirements** –
  - kinds of sensors and actuators
  - memory and processing power
  - communication protocols and
  - operating environment (temperature, humidity, etc.).
- **Consider the MCU architecture** –
  - Different types of MCU architectures, such as 8, 16, and 32-bit.
  - Depending on project's complexity and the memory and power
- **Consider the communication protocols –**
  - wireless, Bluetooth, cellular, or Zigbee protocols
- **Look at the available MCUs and other parameters-**
  - Select it based on other parameters like availability, cost, size, resources etc
  - well-established ecosystem of development tools and a strong community of developers

Von Neumann and Harvard Architecture





RISC and CISC Architecture

Reduced Instruction Set Computer (RISC)		**Complex Instruction Set Computer (CISC)**
Small, Simple, Fixed Size (95% are executed in single instruction cycle)	**Instruction Set**	Complex, Variable Size, Multiple Operations in a single Instruction
Fixed size instructions are divided in multiple stages, Fast execution of instruction possible	**Pipelining**	Difficult to implement pipelining because of variable length instructions
Operations are performed mainly from register to register (Initially from Memory to register)	**Load store architecture**	Operations are performed directly from memory
Compiler friendly		Compiler independency
Large number of general Purpose registers		Microcode within instructions for various operations

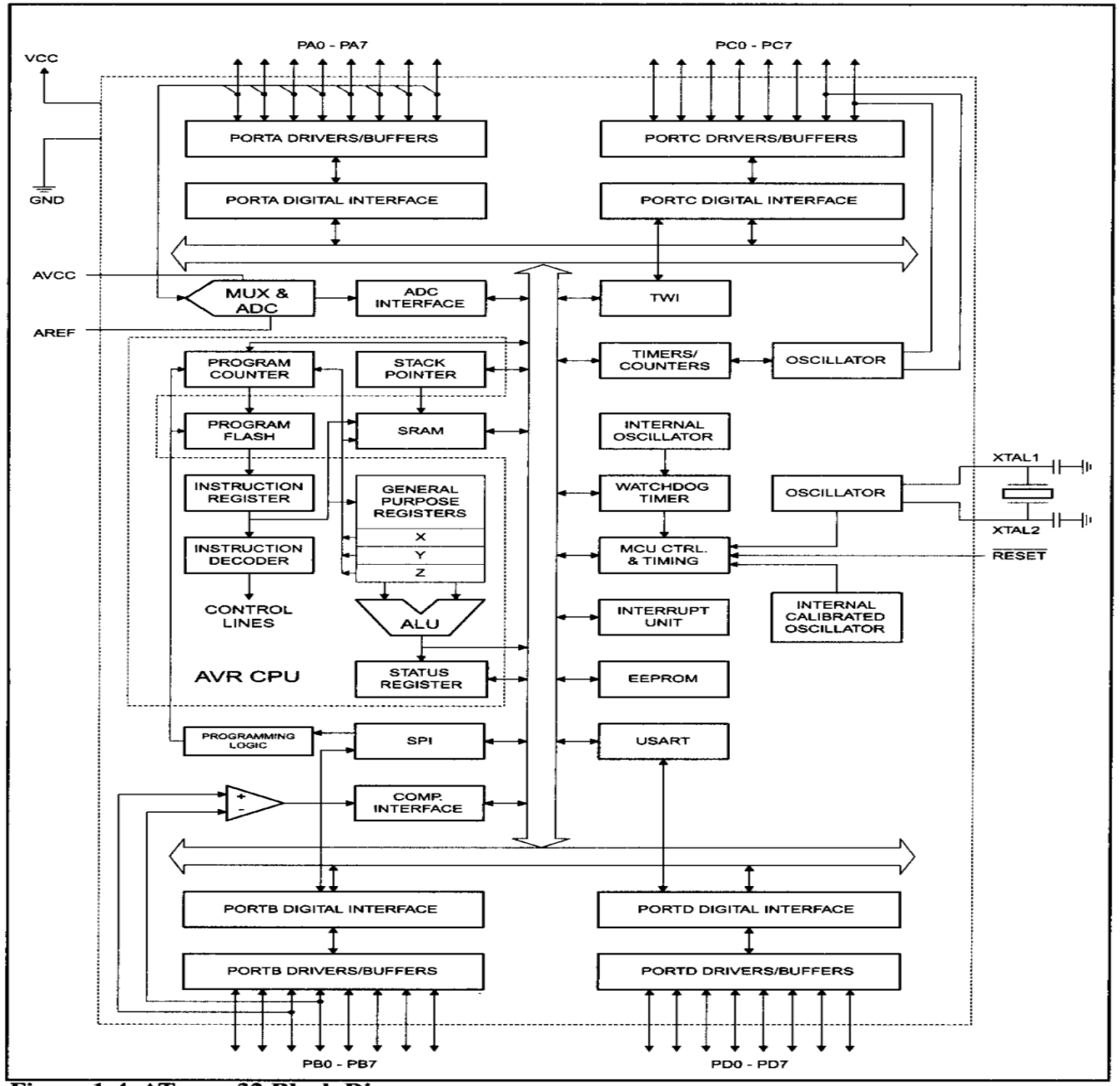
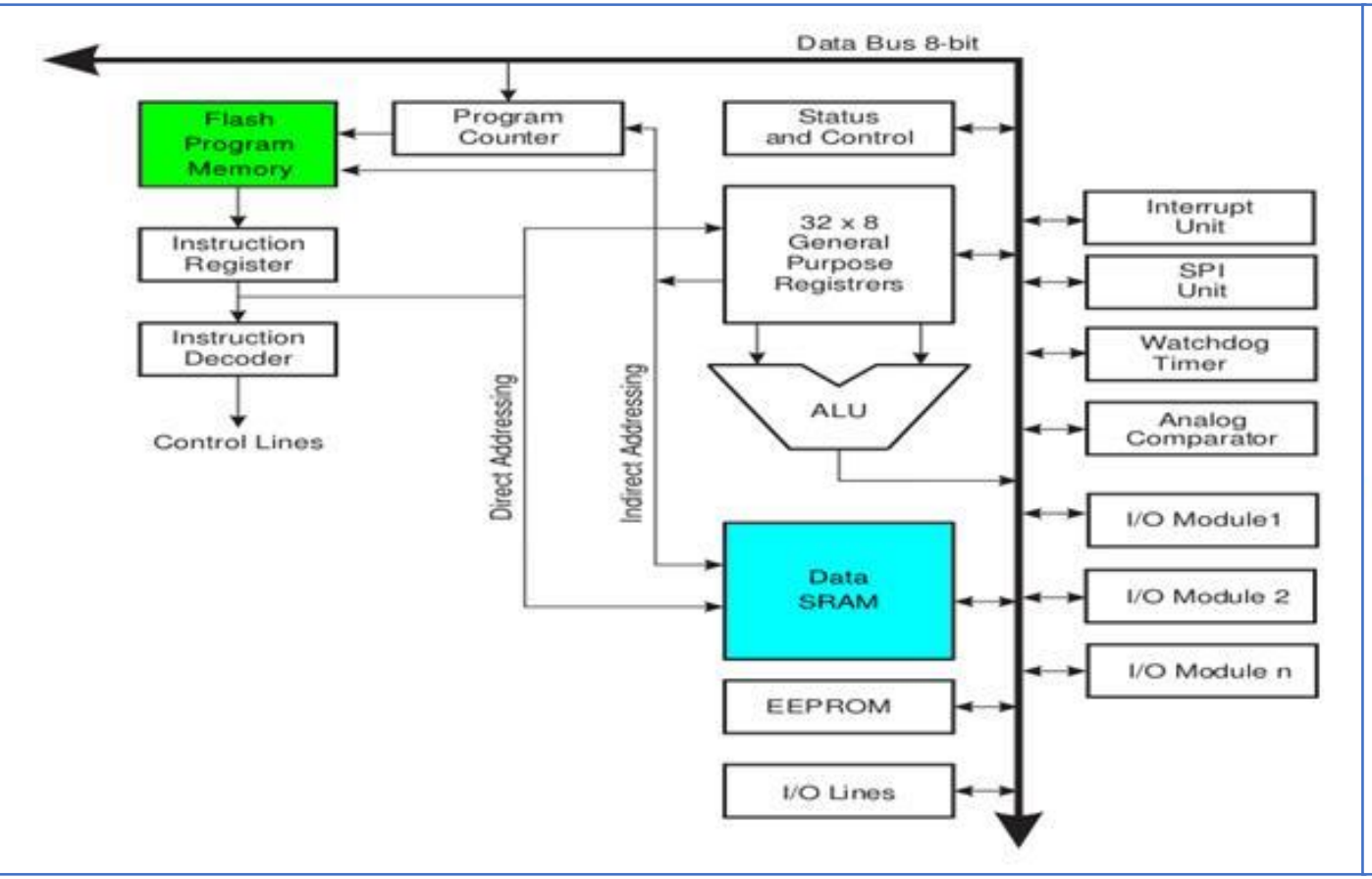
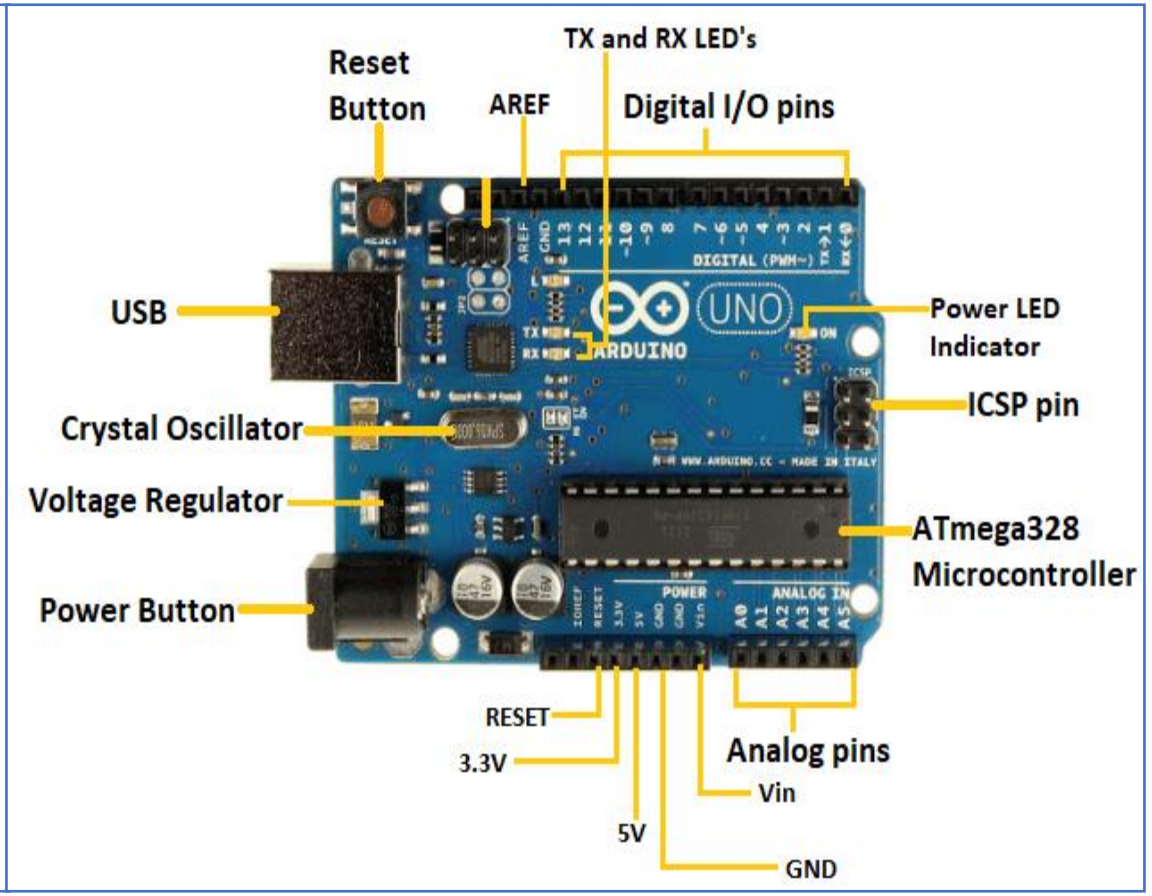


Figure 1-4. ATmega32 Block Diagram

- Ports
- ADC and MUX
- PC
- SP
- Flash
- SRAM
- Instruction decoder
- GP Registers
- ALU
- Status Register
- SPI
- Comparator
- USART
- EEPROM
- Interrupt
- Control and Timing
- Crystal
- WDT
- Timer/Counter
- TWI/I2C

Arduino Uno (ATmega328)



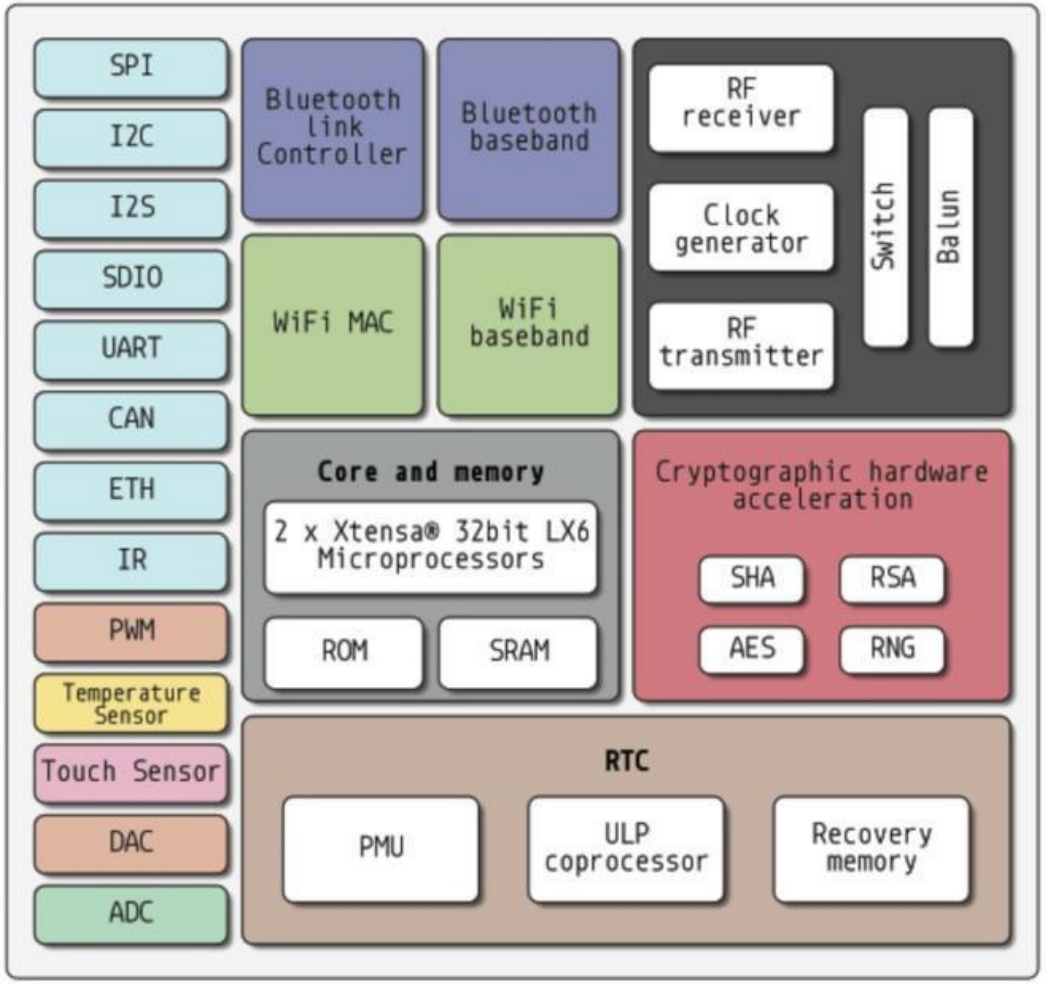


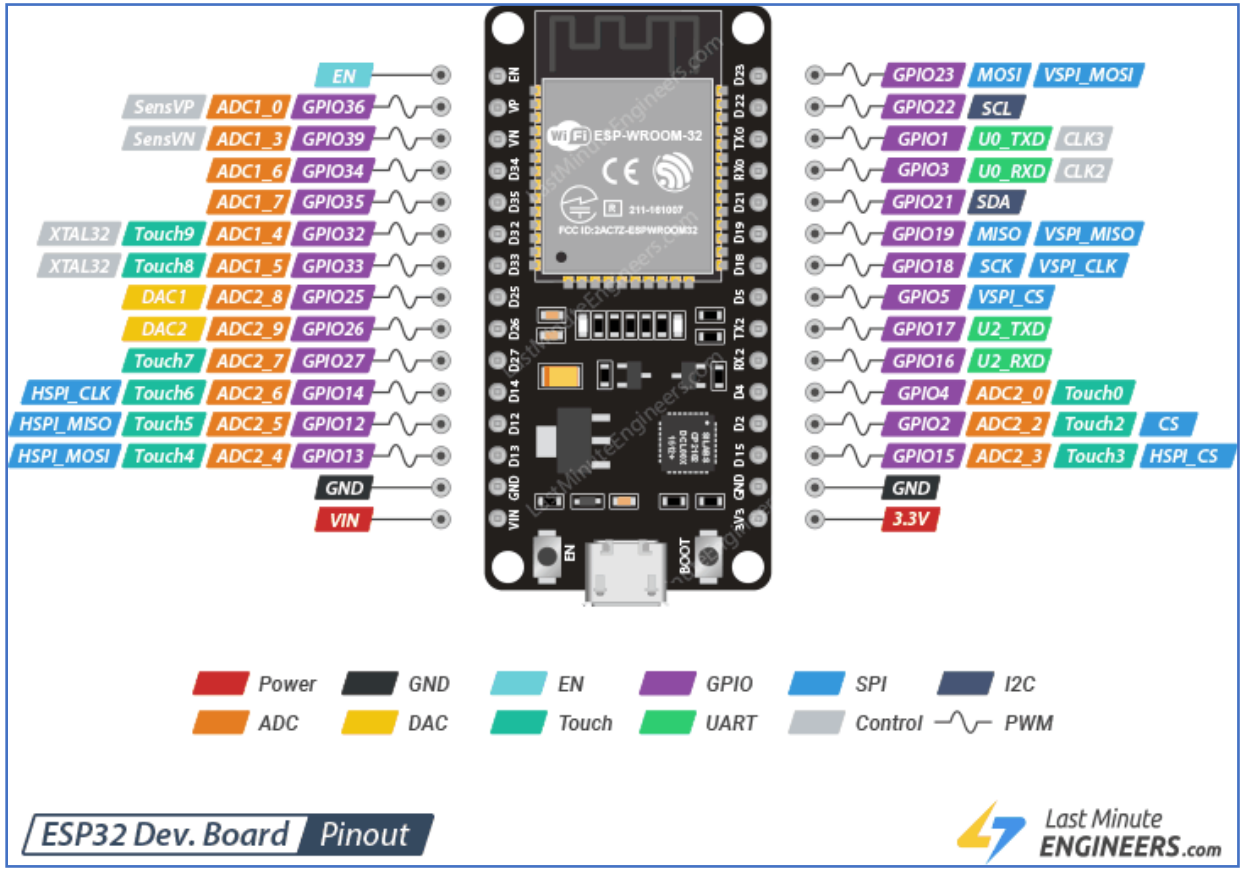
Main Features of Arduino

8 bit RISC Architecture	Brownout Detection: For stability against Power Fluctuations
1 MHz to 20 MHZ Clock Speed	Power Savings Modes for energy savings
2 KB of RAM, 32 KB of Flash Memory	Boot loader support
23 GPIO Pins	3.3 V or 5 V operating voltage support
10 Bits ADC with 8 Multiplexed channels	6 PWM Channels
3 Timers/Counters for various applications	32 General Purpose Registers

- A beginner-friendly board with basic sensor interfacing capabilities

ESP32 Architecture

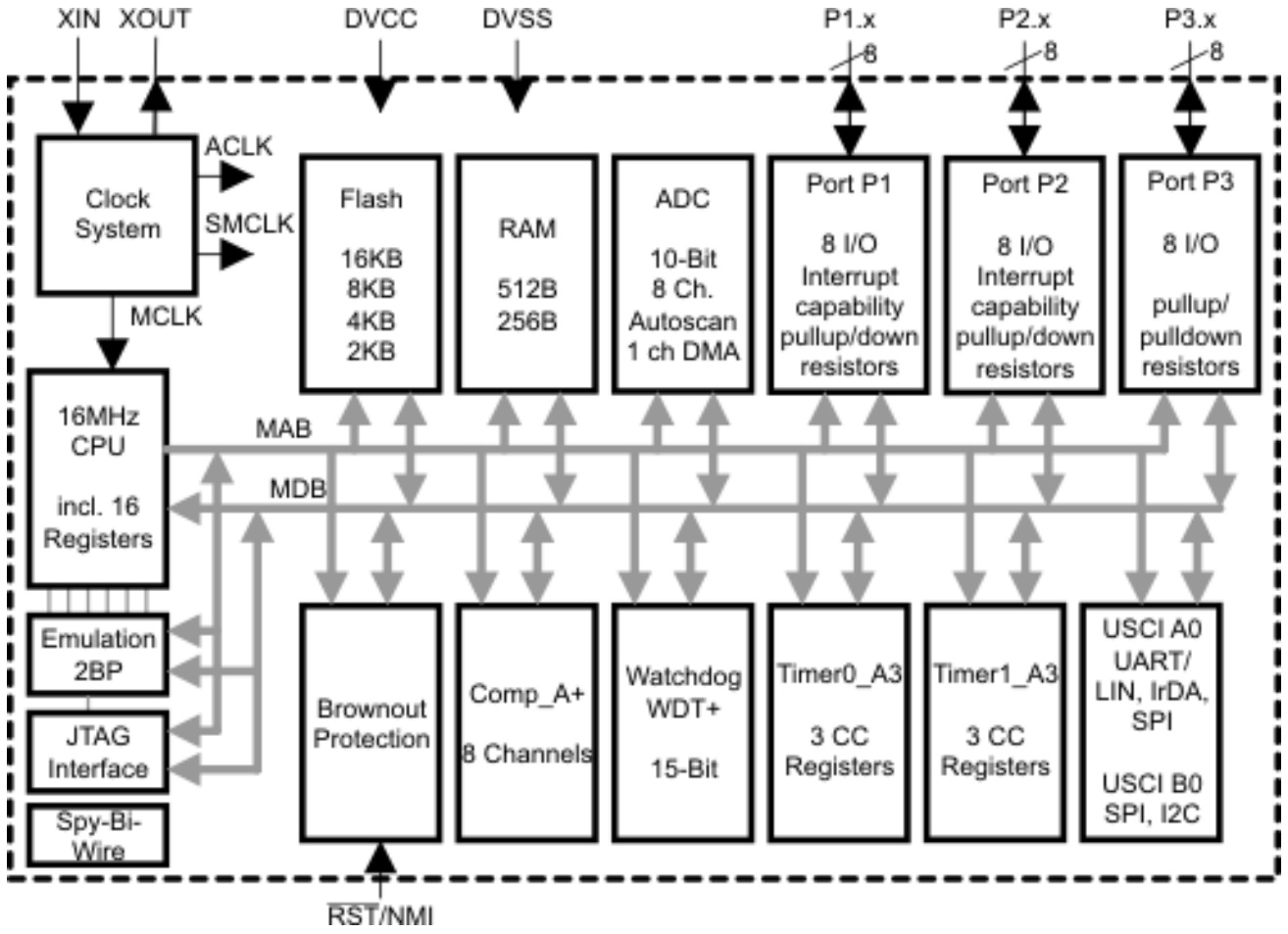


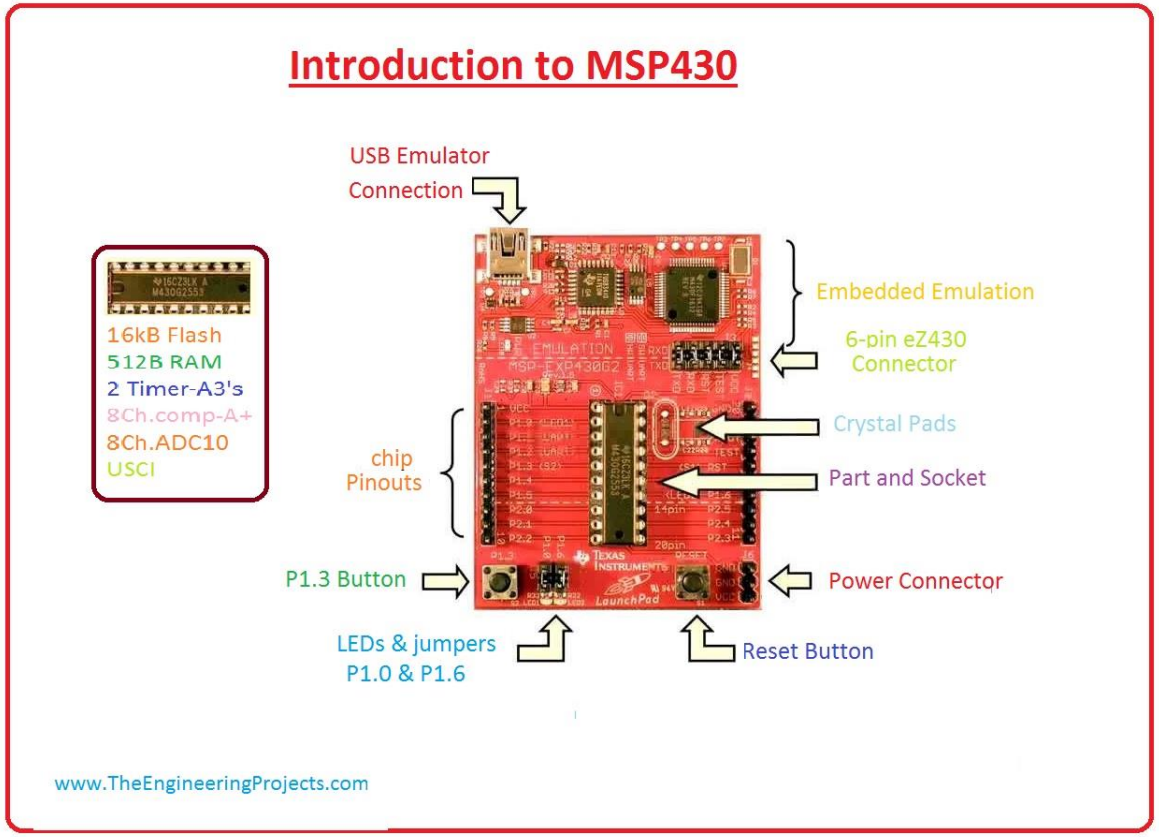


ESP32 Features

Dual core processor	Built in security features
Wireless connectivity through Wi-fi and Bluetooth	Flash memory
Power Down Modes	Integrated Temperature and Humidity sensors
General Purpose Input Output pins	RTOS Support
UART	Development support for Arduino IDE and others
I2C, PWM, ADC, DAC	Over the air (OTA) updates

MSP 430 Architecture





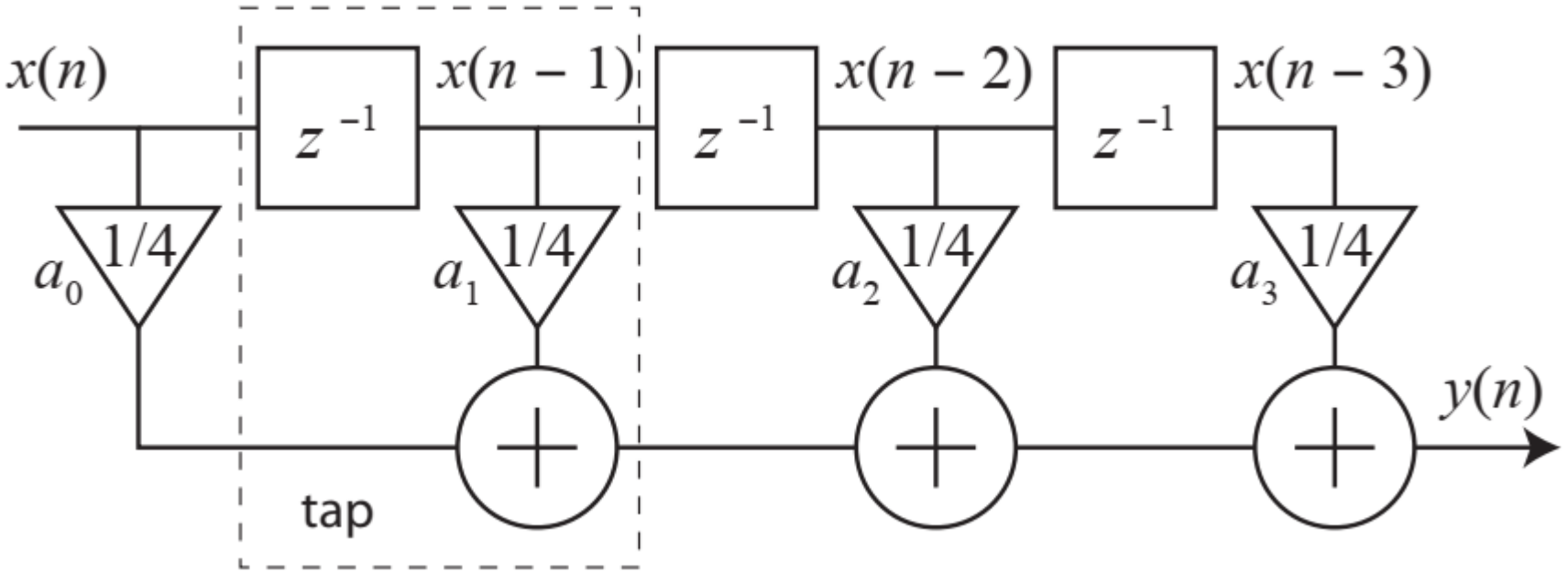
MSP 430 Features

Ultra low power consumption modes (Three power down modes)	Watch dog timers
Mixed signal operations with inbuilt ADCs and DACs	Inbuilt temperature and voltage sensors
UART, I2C, SPI and USB support	RTC support
Timers, GPIO and PWM support	Variable clock support from few KHz to MHz for low power consumption modes

DSP Processors

- Signal processing applications share certain characteristics. First, they deal with large amounts of data.
- The data may represent samples in time of a physical processor (such as samples of a wireless radio signal), samples in space (such as images), or both (such as video and radar).
- Second, they typically perform sophisticated mathematical operations on the data, including filtering, system identification, frequency analysis, machine learning and feature extraction.
- Processors designed specifically to support numerically intensive signal processing applications are called DSP processors, or DSPs (digital signal processors).

Structure of Tapped Delay FIR Filter



The rate at which the input values $x(n)$ are provided and must be processed is called the **sample rate**. If you know the sample rate and N, you can determine the number of arithmetic operations that must be computed per second

Example of FIR Filter

- Suppose that an FIR filter is provided with samples at a rate of 1 MHz (one million samples per second), and that N = 32.
- Then outputs must be computed at a rate of 1 MHz, and each output requires 32 multiplications and 31 additions.
- A processor must be capable of sustaining a computation rate of 63 million arithmetic operations per second to implement this application.
- Of course, to sustain the computation rate, it is necessary not only that the arithmetic hardware be fast enough, but also that the mechanisms for getting data in and out of memory and on and off chip be fast enough.

Television Example

- Analog television is steadily being replaced by digital formats such as **ATSC**, a set of standards developed by the Advanced Television Systems Committee.
- In the US, the vast majority of over-the-air **NTSC** transmissions (National Television System Committee) were replaced with ATSC on June 12, 2009.
- ATSC supports a number of frame rates ranging from just below 24 Hz to 60 Hz and a number of resolutions. High-definition video under the ATSC standard supports, for example, a resolution of 1080 by 1920 pixels at a frame rate of 30 Hz.
- Hence, $H = f0; \cdots ; 1919g$ and $V = f0; \cdots ; 1079g$. This resolution is called 1080p in the industry.
- Professional video equipment today goes up to four times this resolution (4320 by 7680). Frame rates can also be much higher than 30 Hz.
- Very high frame rates are useful for capturing extremely fast phenomena in slow motion.

Graphics Processors

- A **graphics processing unit** (**GPU**) is a specialized processor designed especially to perform the calculations required in graphics rendering.
- Modern GPUs support 3D graphics, shading and digital video. Dominant providers of GPUs today are Intel, NVIDIA and AMD.
- Some embedded applications, particularly games, are a good match for GPUs.
- GPUs are typically quite power hungry and therefore today are not a good match for energy constrained embedded applications.

Parallel v/s Concurrent

- Concurrency is central to embedded systems. A computer program is said to be **concurrent** if different parts of the program *conceptually* execute simultaneously.

- A program is said to be **parallel** if different parts of the program *physically* execute simultaneously on distinct hardware (such as on multicore machines, on servers in a server farm, or on distinct microprocessors)

Example

- Consider the following C statements:
- double pi, piSquared, piCubed;
- pi = 3.14159;
- piSquared = pi * pi ;
- piCubed = pi * pi * pi;

- **The last two assignment statements are independent, and hence can be executed in parallel or in reverse order without changing the behavior of the program.**

- Had we written them as follows, however, they would no longer be independent:
- double pi, piSquared, piCubed;
- pi = 3.14159;
- piSquared = pi * pi ;
- piCubed = piSquared * pi;

- **In this case, the last statement depends on the third statement in the sense that the third statement must complete execution before the last statement starts.**

Superscaler Architecture

- Superscalar architecture is a type of microprocessor design that focuses on improving the performance of a CPU by allowing it to execute multiple instructions in parallel, thereby increasing the overall throughput of the processor.
- The term "superscalar" comes from the idea of going beyond the traditional scalar architecture, which executes one instruction at a time.

Superscaler CPU

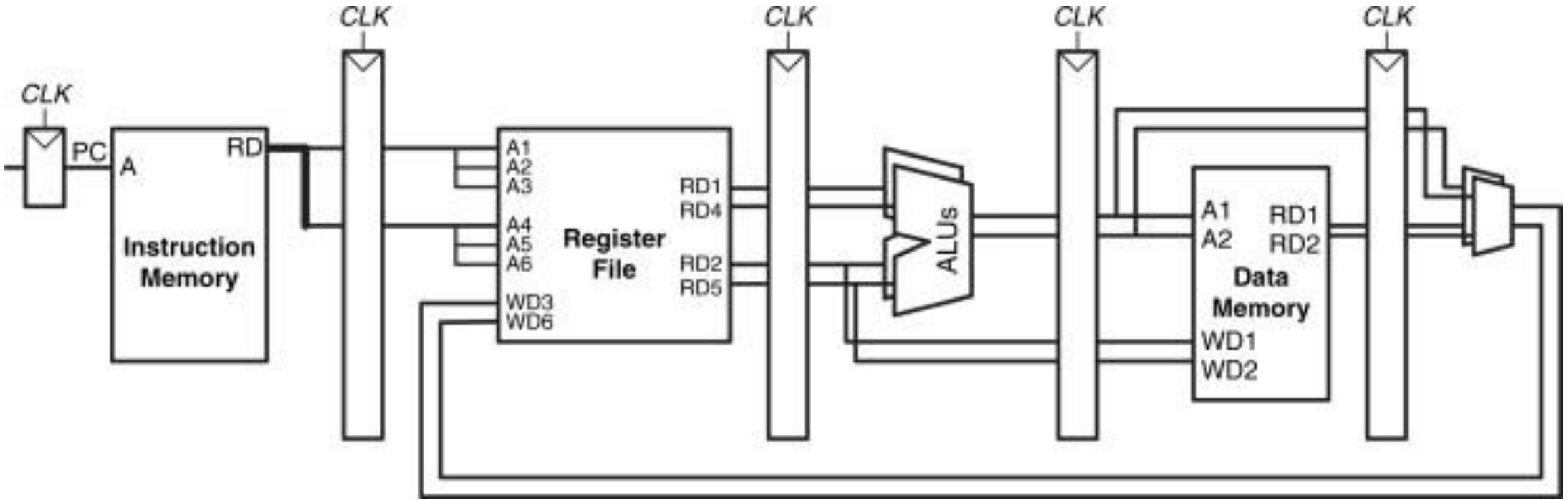
- In a superscalar architecture, the CPU consists of multiple functional units, such as arithmetic logic units (ALUs), floating-point units (FPUs), and memory units.
- These functional units can operate independently and concurrently, allowing the processor to execute multiple instructions simultaneously.

Features of Superscalar Architecture

- Multiple Instructions: A superscalar processor can decode and issue multiple instructions from the instruction stream in a single clock cycle. This is achieved by having multiple instruction decoders and issue units, allowing the processor to identify and execute multiple independent instructions simultaneously.

- Instruction level Parallelism (ILP): Superscalar architectures exploit instruction-level parallelism present in the code. This means that the processor identifies instructions that can be executed concurrently without dependencies on each other. For instance, while one instruction is performing a memory access, another independent instruction can execute an arithmetic operation.

Basic Block Diagram



Features

- Out-of-Order Execution: In superscalar processors, instructions can be executed out of their original order to maximize resource utilization and minimize stalls. The processor's scheduler rearranges the instructions dynamically to execute those that do not depend on stalled or unfinished instructions.
- Register Renaming: Superscalar processors often employ register renaming, which allows multiple instructions that use the same architectural registers to proceed independently by mapping them to physical registers. This helps avoid data hazards and stalls caused by register dependencies.

Features

- Data Forwarding and Hazard Detection: To prevent stalls due to data dependencies, superscalar processors utilize techniques like data forwarding. This allows data produced by one instruction to be forwarded directly to another instruction waiting for that data, bypassing the need to wait for it to be written to a register.
- Speculative Execution: Some superscalar architectures incorporate speculative execution, where the processor predicts the outcome of a branch instruction and starts executing instructions based on that prediction before the actual branch resolution. If the prediction is incorrect, the incorrectly executed instructions are discarded.

Features

- Dynamic Scheduling: Superscalar processors employ dynamic scheduling algorithms to manage the execution of instructions. The scheduler determines which instructions are ready to execute, taking into account dependencies and available resources.

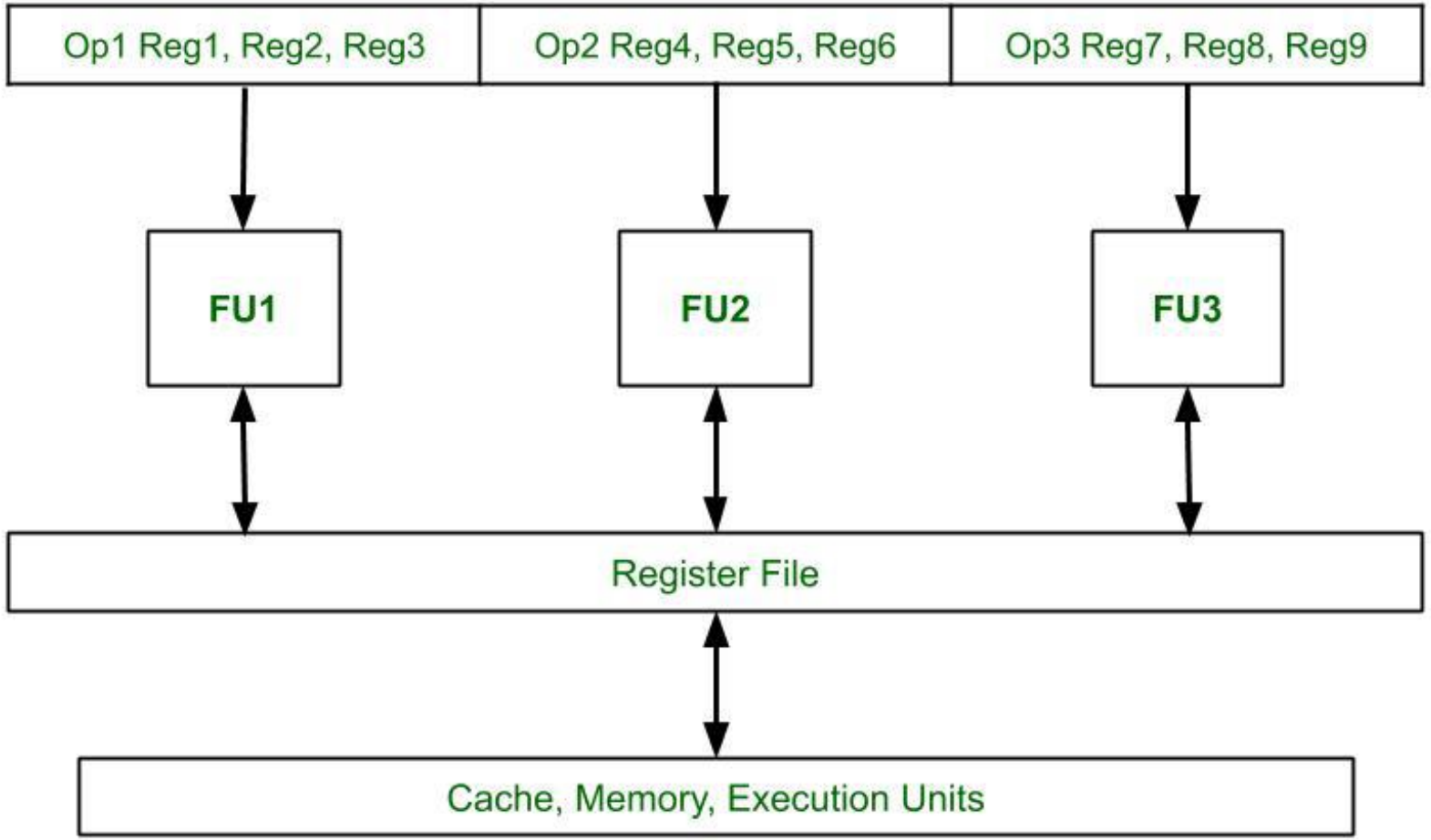
- Overall, superscalar architecture significantly enhances the processing power of modern CPUs by capitalizing on instruction-level parallelism and enabling the concurrent execution of multiple instructions. This results in improved performance and better utilization of available hardware resources, making superscalar processors common in modern high-performance computing systems.

VLIW Architecture

- VLIW stands for Very Long Instruction Word, and it's a type of microprocessor architecture designed to exploit instruction-level parallelism (ILP) in programs. Unlike superscalar architectures, where the hardware dynamically decides which instructions to execute in parallel, VLIW architecture relies on the compiler to explicitly schedule instructions for parallel execution.

- In a VLIW architecture, the compiler organizes instructions into bundles or groups, each containing multiple instructions that can be executed in parallel. These bundles are composed of fixed-length instruction words, typically containing one operation from each functional unit category (like ALU, FPU, etc.), along with any associated operands.

Functional Block Diagram



Features of VLIW

- Fixed Instruction Format: In a VLIW architecture, instructions are grouped together in a fixed-length instruction bundle. Each bundle contains multiple instructions that are intended to be executed in parallel during a single clock cycle.

- Parallel Execution: Unlike superscalar architectures, where the hardware dynamically identifies parallelism at runtime, VLIW processors rely on the compiler to explicitly specify which instructions can run concurrently. The compiler needs to analyse the code and arrange the instructions in a way that minimizes dependencies and maximizes parallelism.