Introduction to Microcontrollers

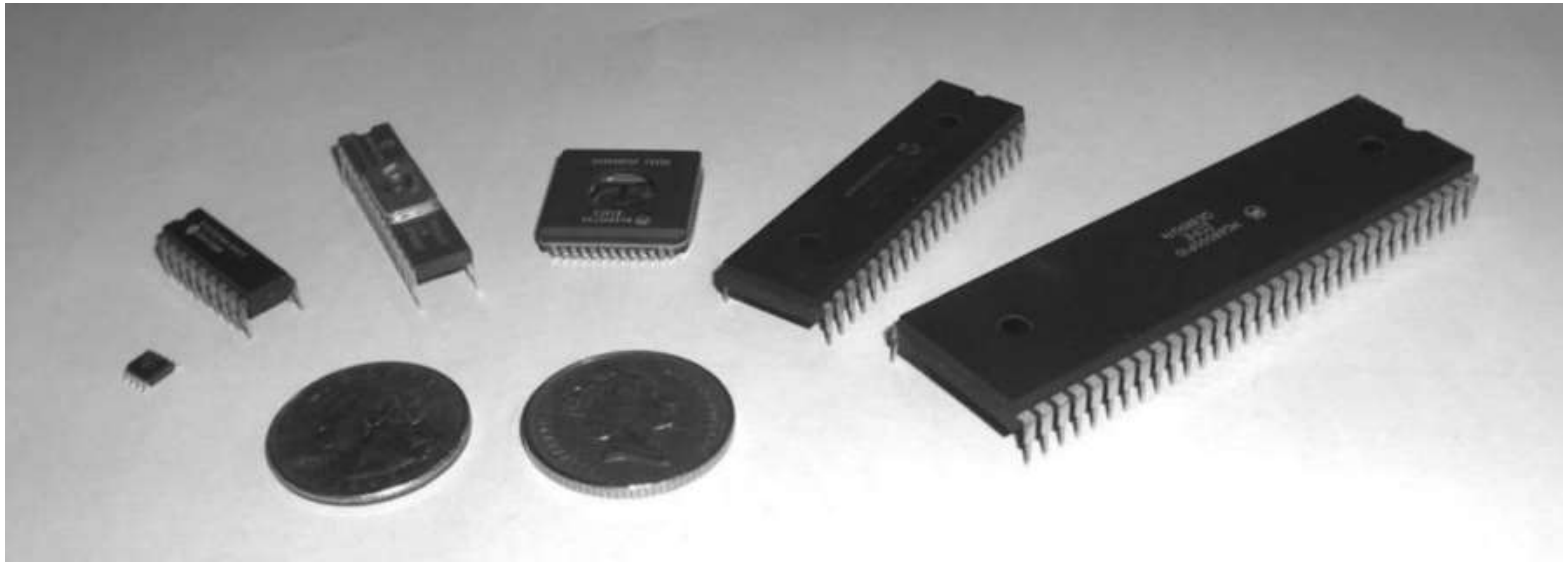
Outline

- Definition of Microcontroller
- Types of Microcontrollers
- Microcontroller Architecture
- Core Microcontroller Components
- Peripheral Microcontroller Components
- Internal and External Operation of Microcontrollers
- Advantages of Microcontrollers
- Application Areas of Microcontrollers
- A Typical Microcontroller Application
- Microcontroller vs. Microprocessor
- Choosing a Microcontroller for a Specific Application

Definition of Microcontroller

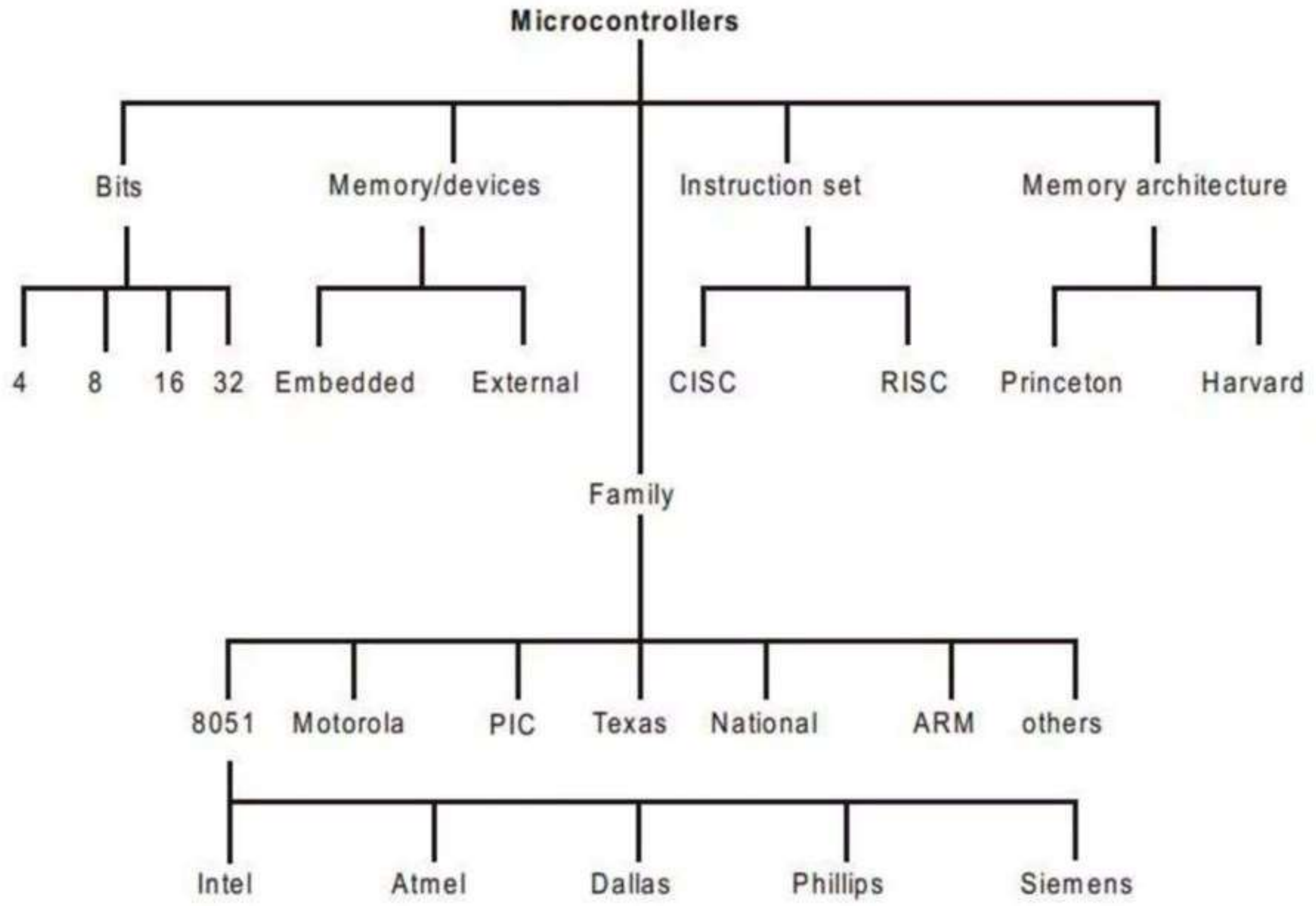
- A microcontroller is an integrated circuit (IC) that can be programmed to perform a set of functions to control a collection of electronic devices.
- A self-contained system in which a processor, support, memory, and input/output (I/O) are all contained in a single package.
- Being programmable is what makes the microcontroller unique.

Definition of Microcontroller (Continued)



From left to right: PIC 12F508, PIC 16F84A, PIC 16C72, Motorola 68HCO05B16, PIC 16F877, Motorola 68000.

Types of Microcontrollers



4-bit Microcontrollers

- ALU performs arithmetic and logical operations on a nibble (4-bits) at an instruction.
- Internal bus width is 4-bit.
- Small size, minimum pin count and low cost controllers.
- Low power consumption and used for low end applications like LED & LCD display drivers, portable battery chargers etc.
- Examples: Renasa M34501 256 and ATAMS862 series from Atmel.

8-bit Microcontrollers

- ALU performs arithmetic and logical operations on a byte (8-bits) at an instruction.
- Internal bus width is 8-bit.
- Examples: Intel 8051 family and Motorola MC68HCI11 family.

16-bit Microcontrollers

- ALU performs arithmetic and logical operations on a word (16-bits) at an instruction.
- Internal bus width of 16-bit microcontroller is 16-bit.
- Enhanced performance, computing capability and precision as compared to the 8-bit microcontrollers.
- Examples: Intel 8096 family, Motorola MC68HCI12 and MC68332 families.

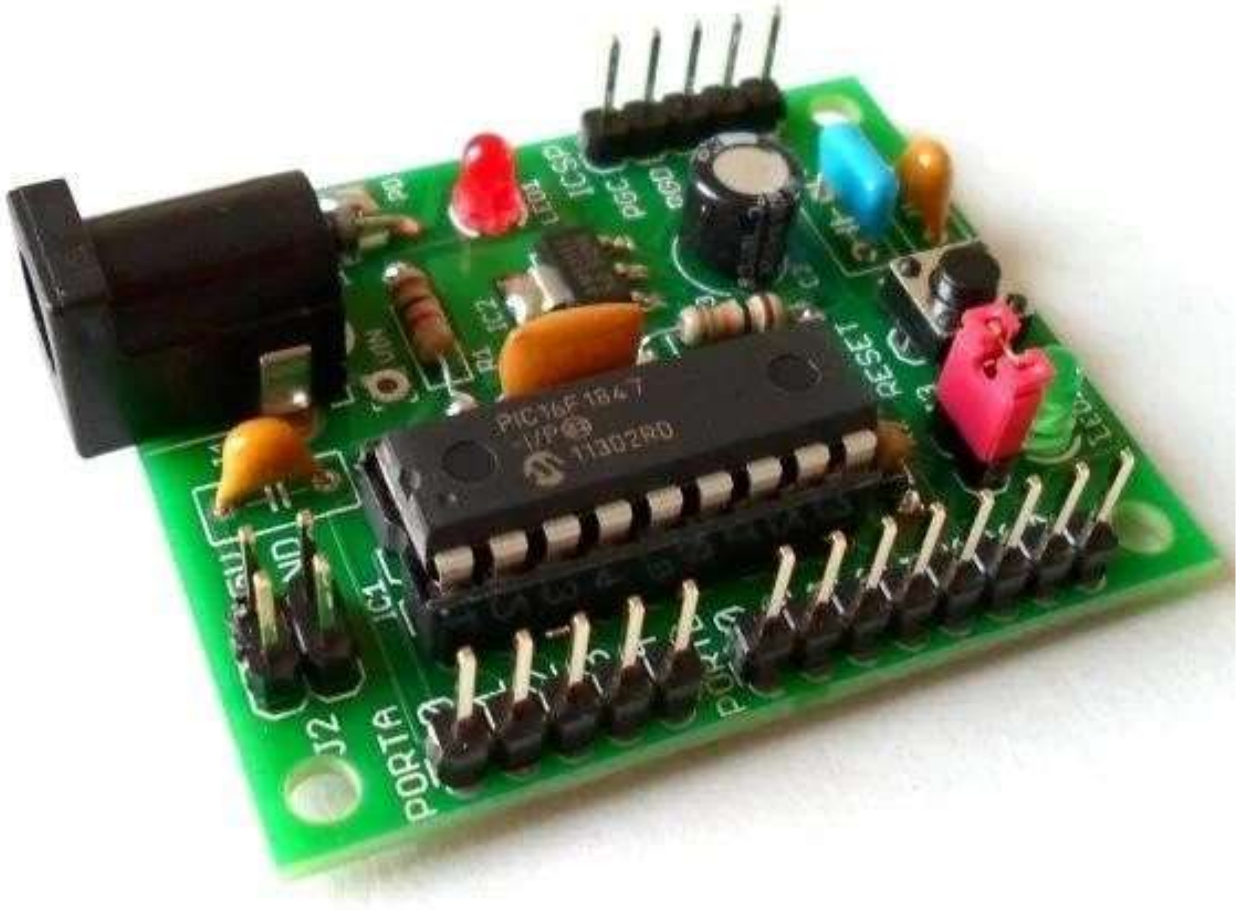
32-bit Microcontrollers

- ALU performs arithmetic and logical operations on a double word (32-bits) per instruction.
- Internal bus width is 32-bit.
- Much more enhanced performance, computing capability with greater precision as compared to 16-bit microcontrollers.
- Examples: Intel 80960 family, Motorola M683xx and Intel/Atmel 251 family.

Embedded Microcontrollers

- An embedded system has a microcontroller unit that has all the functional blocks (including program as well as data memory) available on a the same chip.
- Example: 8051 having Program & Data Memory, I/O Ports, Serial Communication, Counters and Timers and Interrupt Control logic on the chip.

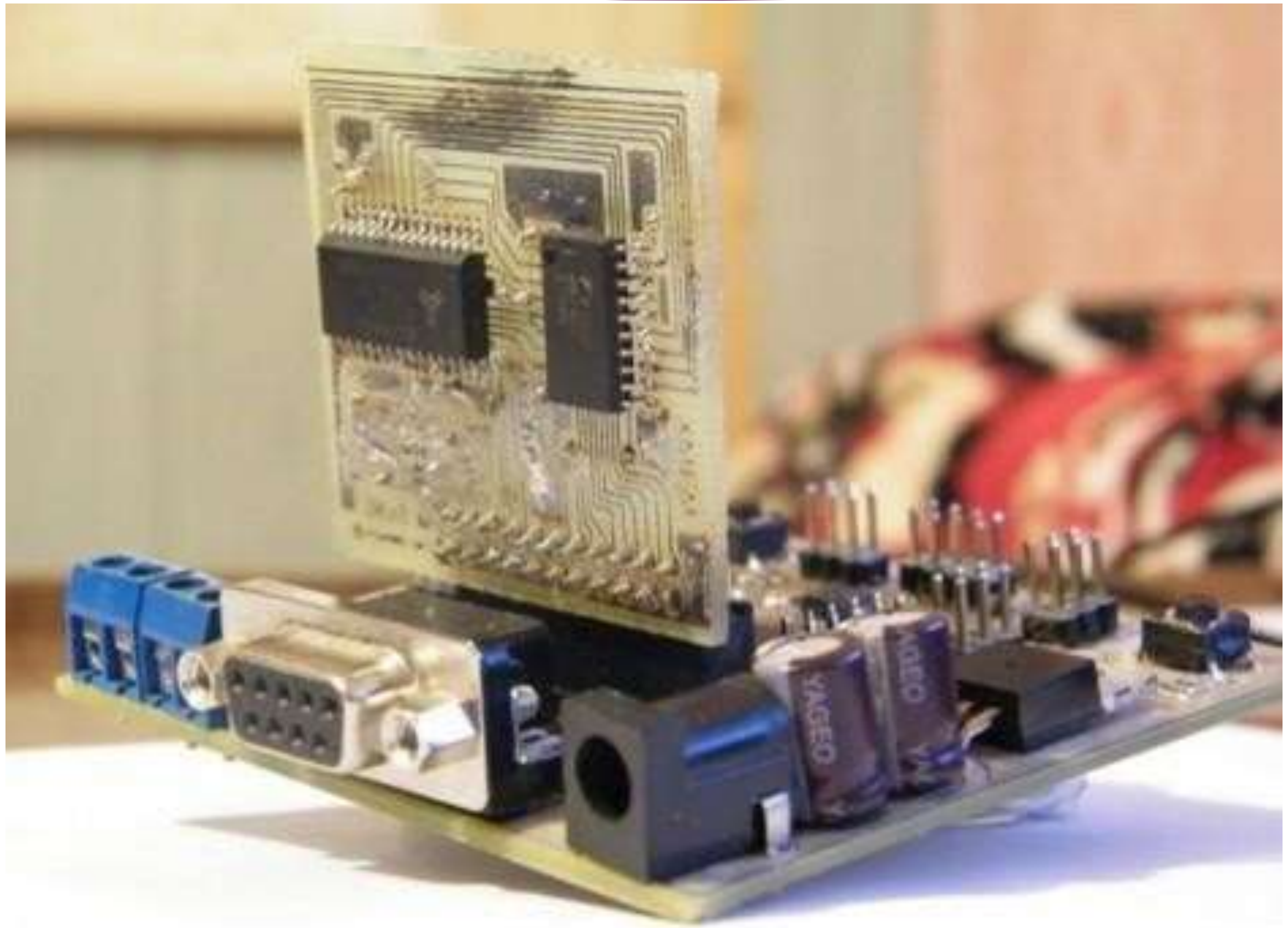
An Embedded PIC Microcontroller



External Memory Microcontrollers

- An external system has a microcontroller unit that does not have all the functional blocks available on a chip.
- All or part of the memory units are externally interfaced using an interfacing circuit called the glue circuit.
- Example: Intel 8031 has no program memory on the chip.

An External Memory Microcontroller



CISC Microcontrollers

- Has an instruction set that supports many addressing modes for the arithmetic and logical instructions, data transfer and memory accesses instructions.
- Many of the instructions are macro like.
- Allows the programmer to use one instruction in place of many simpler instructions.
- Example: Intel 8096 family.

RISC Microcontrollers

- Contains an instruction set that supports fewer addressing modes for the arithmetic and logical instructions and for data transfer instructions.
- Allows simultaneous access of program and data.
- Instruction pipelining increases execution speed.
- Allow each instruction to operate on any register or use any addressing mode.
- Smaller chip and pin count.
- Very low power consumption.

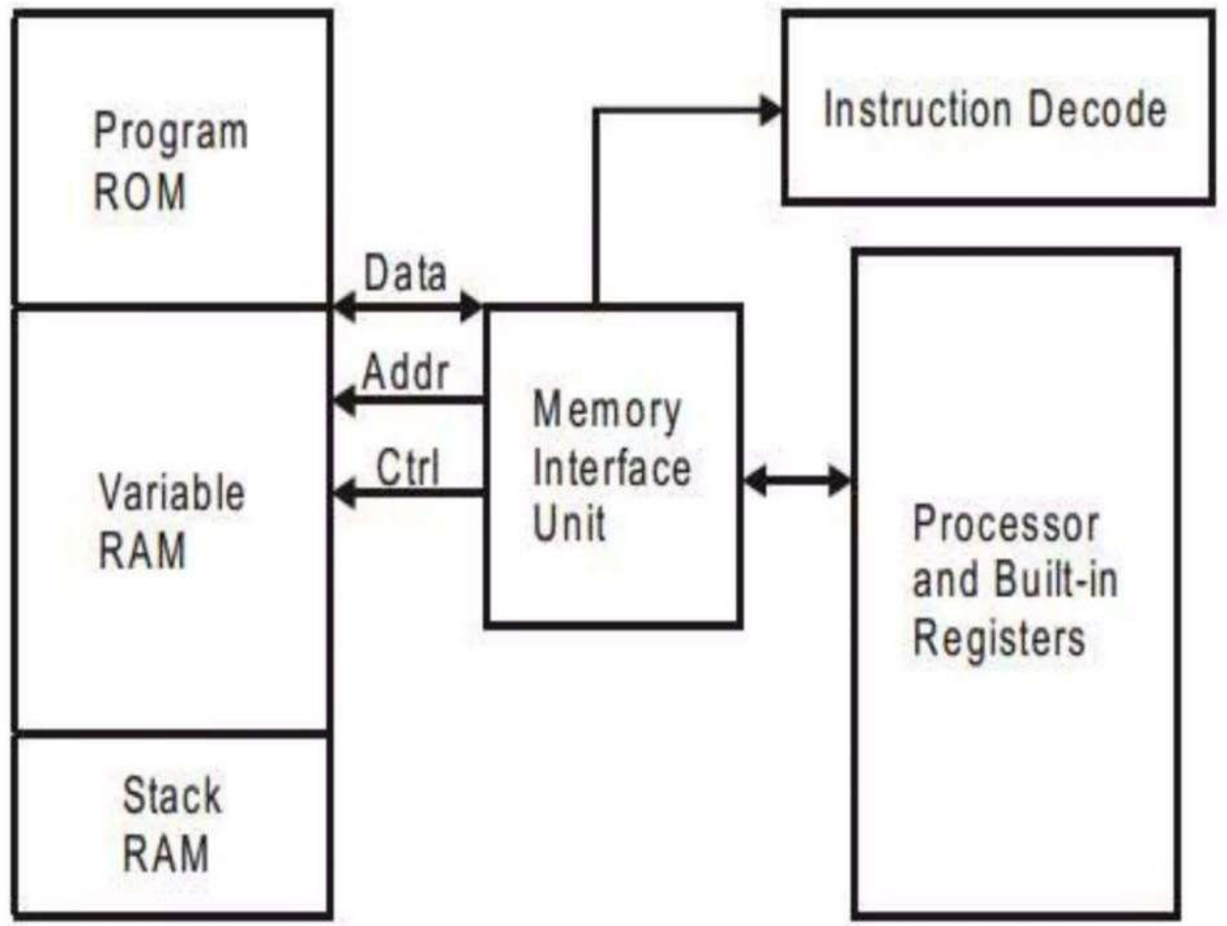
CISC vs RISC Microcontroller

Feature	**CISC (Complex Instruction Set Computer)**	**RISC (Reduced Instruction Set Computer)**
Instruction Set	Large and complex	Small and simple
Instructions per Task	Fewer (single complex instruction can do a lot)	More (simple instructions combined)
Instruction Length	Variable	Fixed
Execution Time	Slow (some instructions take multiple cycles)	Fast (one instruction per cycle)
Hardware Complexity	More complex (needs to decode complex instructions)	Simpler
Code Size	Smaller (fewer instructions needed)	Larger (more instructions required)
Power Consumption	Higher	Lower
Performance	Good for complex tasks	Faster for simpler tasks
Examples	Intel 8051, x86 family	ARM Cortex-M, AVR, MIPS

Von-Neuman Architecture

- Single data bus that is used to fetch both instructions and data.
- Program instructions and data are stored in a common main memory.
- When such a controller addresses main memory, it first fetches an instruction, and then it fetches the data to support the instruction.
- Simplifies the microcontroller design because only one memory is accessed.
- The weakness is that two separate fetches can slow up the controller’s operation.
- Example: Motorola 68HC11.

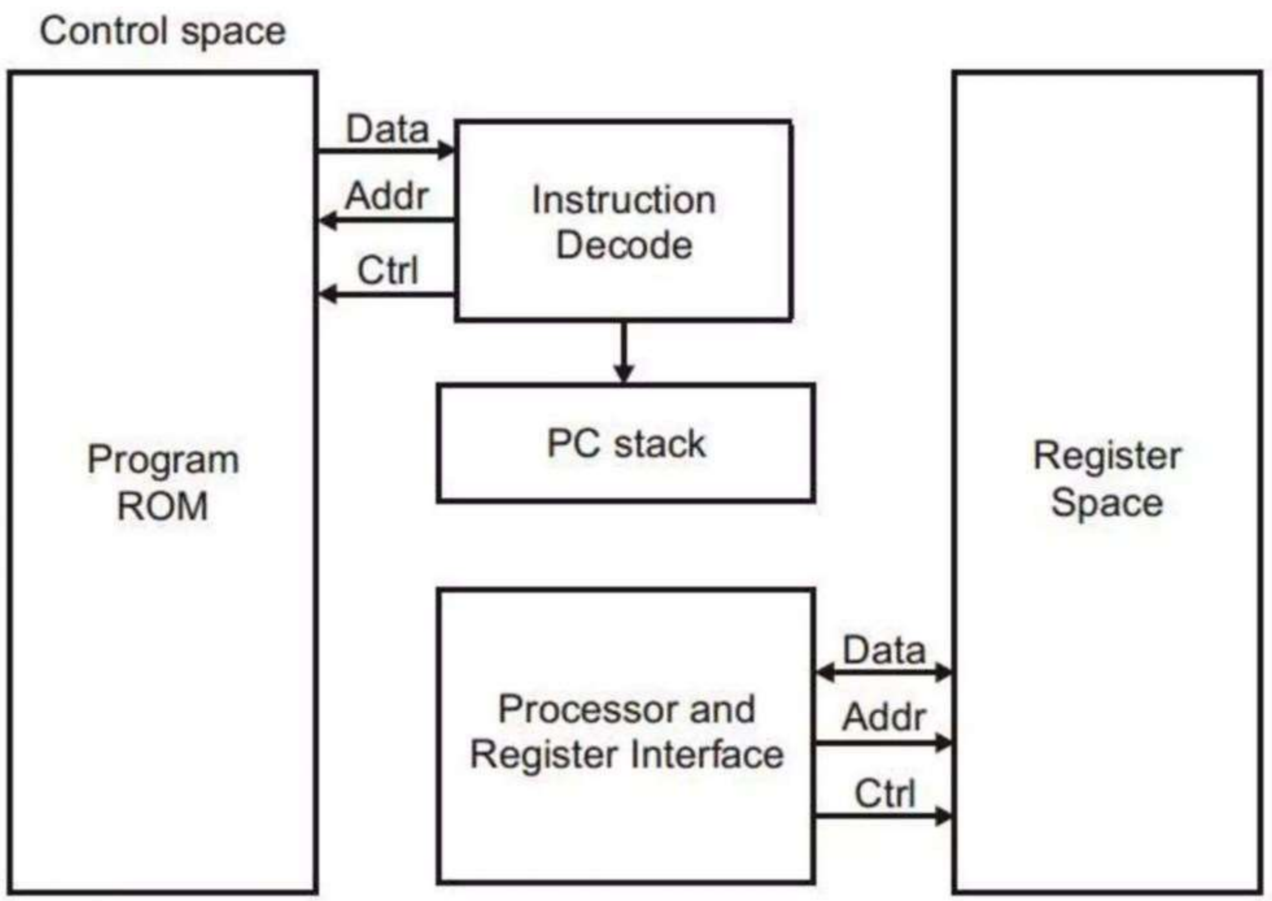
Von-Neuman Architecture Block Diagram



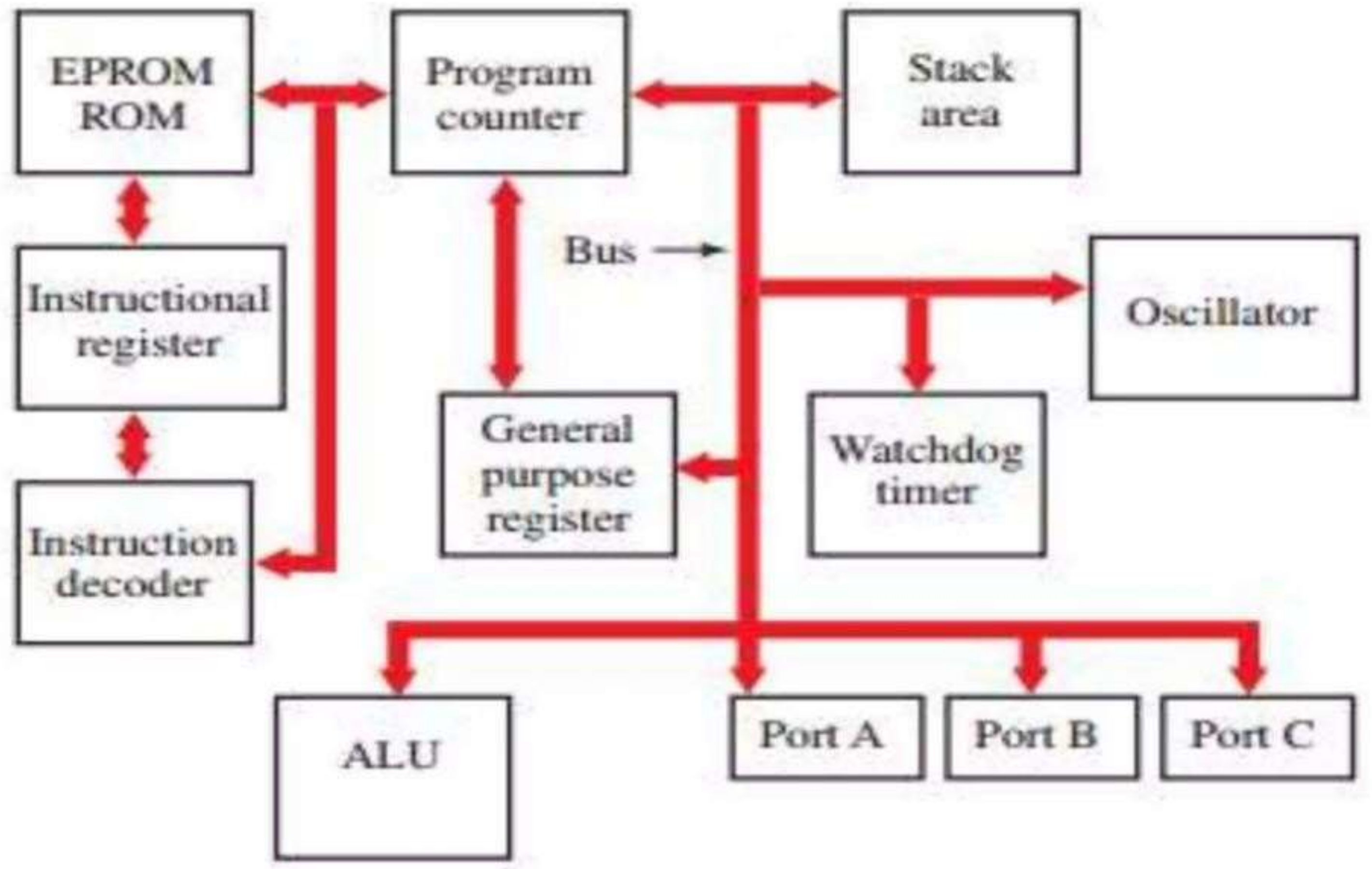
Harvard Architecture

- Separate data bus and an instruction bus.
- Execution occur in parallel.
- Much faster execution than Von-Neuman architecture.
- Design complexity.
- Example: Intel MCS-51 family and PIC microcontrollers.

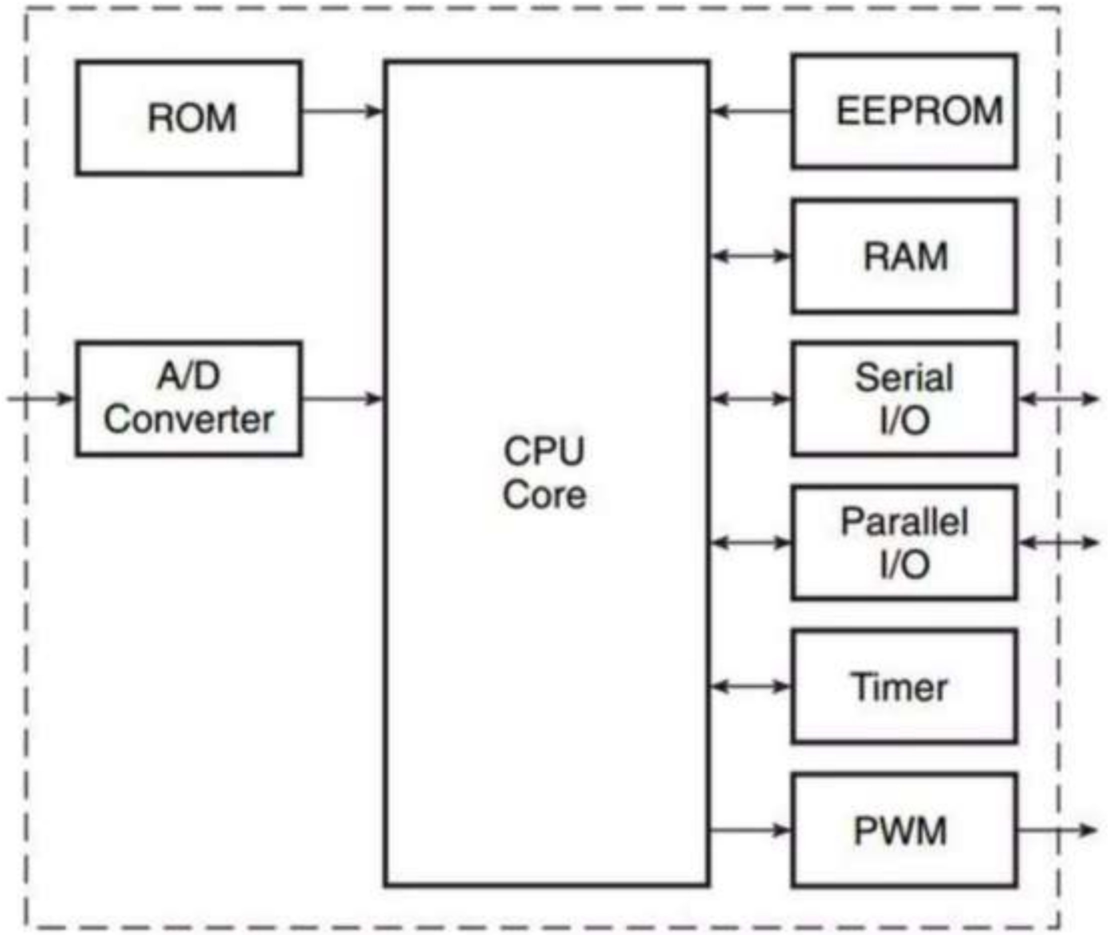
Harvard Architecture Block Diagram



Microcontroller CPU Architecture Block Diagram



Generalized Microcontroller Architecture Block Diagram



Core Microcontroller Components

- **CPU:** The Central Processing Unit (CPU) processes the program. It executes the instructions stored in the program memory pointed to by the program counter in synchronization with the clock signal.
- **ALU:** The Arithmetic/Logic Unit (ALU) performs mathematical and logical operations on data.
- **Oscillator:** A complex digital device that generates steady pulse rate required for timing. All of the separate functions are controlled by one central timing system. The timing pulse provides the basis for proper sequence of all the separate sections of the microcontroller chip.

Core Microcontroller Components (Continued)

Read Only Memory (ROM): ROM holds the program instructions and the constant data. Microcontrollers use one or more of the following memory types for this purpose:

- ROM (mask-programmed ROM)
- PROM (one-time programmable ROM, which is not field programmable)
- EPROM (field programmable and usually UV erasable)
- EEPROM (field programmable, electrically erasable, byte erasable) and flash (similar to EEPROM).
- Microcontrollers can have 4K, 8K and 16K, etc. of ROM

Core Microcontroller Components (Continued)

Random Access Memory (RAM):

- Is used to hold intermediate results and other temporary data during the execution of the program.
- Typically, microcontrollers have a few hundreds of bytes of RAM.

Registers:

Register is used to hold the contents of data being manipulated.

Special-Function Registers:

- Control various functions of a microcontroller.
- These are divided into two groups:
  - Registers Wired into the CPU
    - Do not necessarily form part of addressable memory.
    - Used to control program flow and arithmetic functions.
    - Examples: status register, program counter, stack pointer, etc.
  - Registers Required by Peripheral Components
    - The contents of these registers include set a timer or enable serial communication.
    - Examples: A program counter, stack pointer, RAM address register, program address register, and incrementor register.

Core Microcontroller Components (Continued)

Watchdog timer:

- A specialized program found as part of the microcontroller designed to prevent the microcontroller from halting or "locking up" because of a user-written program since the instructions are processed step-by-step.
- Uses a routine that is based on timing.
- If a program has not been completed or repeated as a loop within a certain amount of time, the watchdog timer issues a reset command.
- A system reset sets all the register values to zero.
- The reset feature allows the microcontroller to recover from a crash.
- It releases the program and sets the MCU to start over again.

Core Microcontroller Components (Continued)

Stack pointer:

- Keeps track of the last stack location used while processor is busy manipulating checking ports, or checking interrupts.

Program counter:

- Is used to hold the address of the instruction to be executed next.

Buses:

- Bus represents a physical connection used to carry a signal from one point to another inside a microcontroller. The signal carried by a bus may represent address, data, control signal, or power.

Peripheral Microcontroller Components

The analog-to-digital converter:

- Provides an interface between the microcontroller and the sensors that produce analogue electrical equivalents of the actual physical parameters to be controlled.

The digital-to-analog converter:

- Provides an interface between the microcontroller and the actuators that provide the control function.

I/O ports:

- Provide an interface between the microcontroller and the peripheral I/O devices such as the keyboard, display, etc.

Counters/timers:

- Are used to keep time and/or measure the time interval between events, count the number of events and generate baud rate for the serial ports.

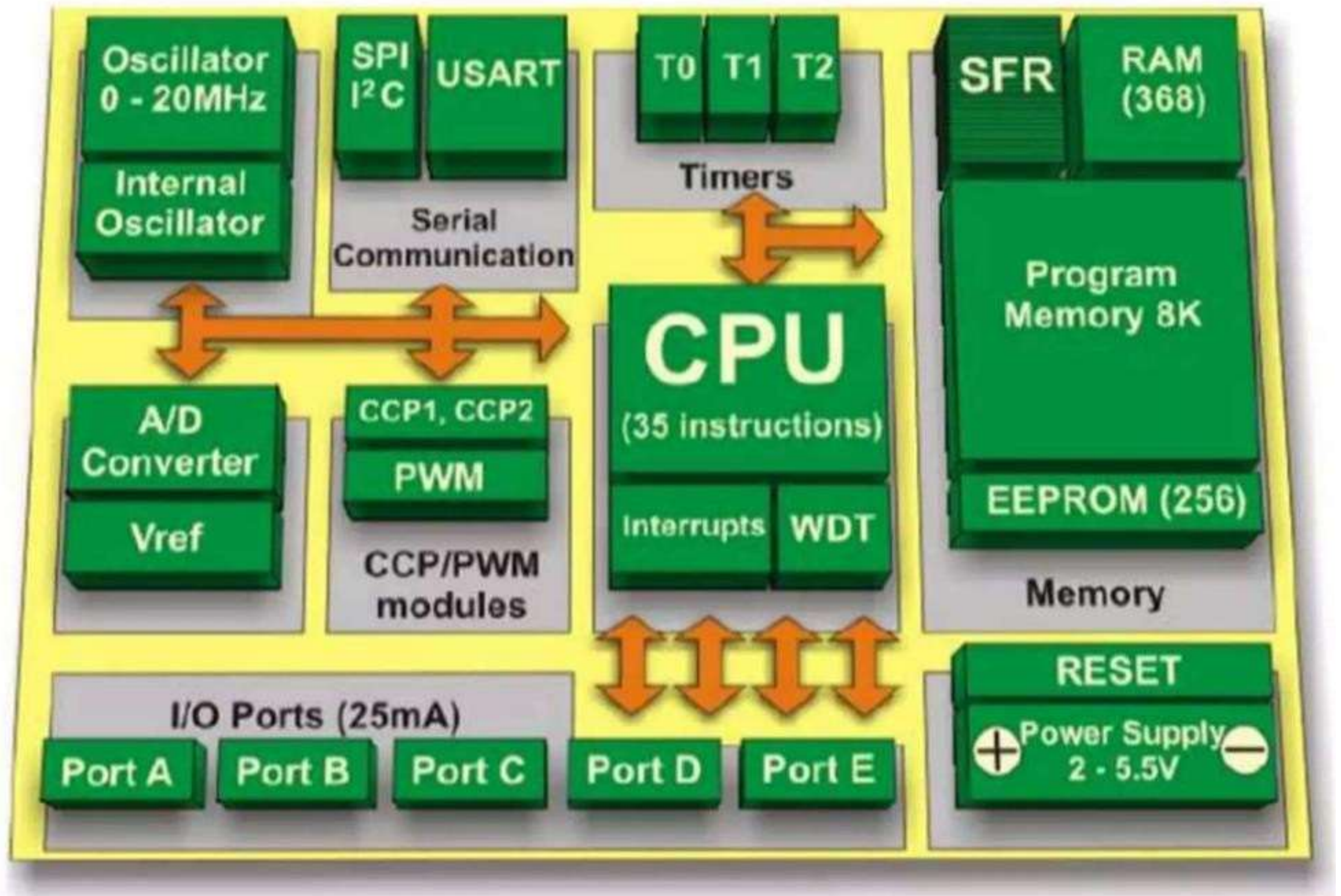
Microcontroller Internal Operation

- The microcontroller consists of thousands of digital circuits that are combined into areas to provide specific functions.
- The CPU components of a microcontroller are used to save data and programs, perform math and logic functions, and generate timing signals.
- The different areas are connected by a bus system. The bus system contains tiny parallel circuits that carry the digital pulse patterns from section to section.
- The ROM stores the program required for the microcontroller to function and controls how the chip components operate and how data and instructions flow through the chip.
- RAM stores programs and data temporarily.
- Ports and registers are special memory locations dedicated to a specific function such as a hardware location or a place to manipulate data.

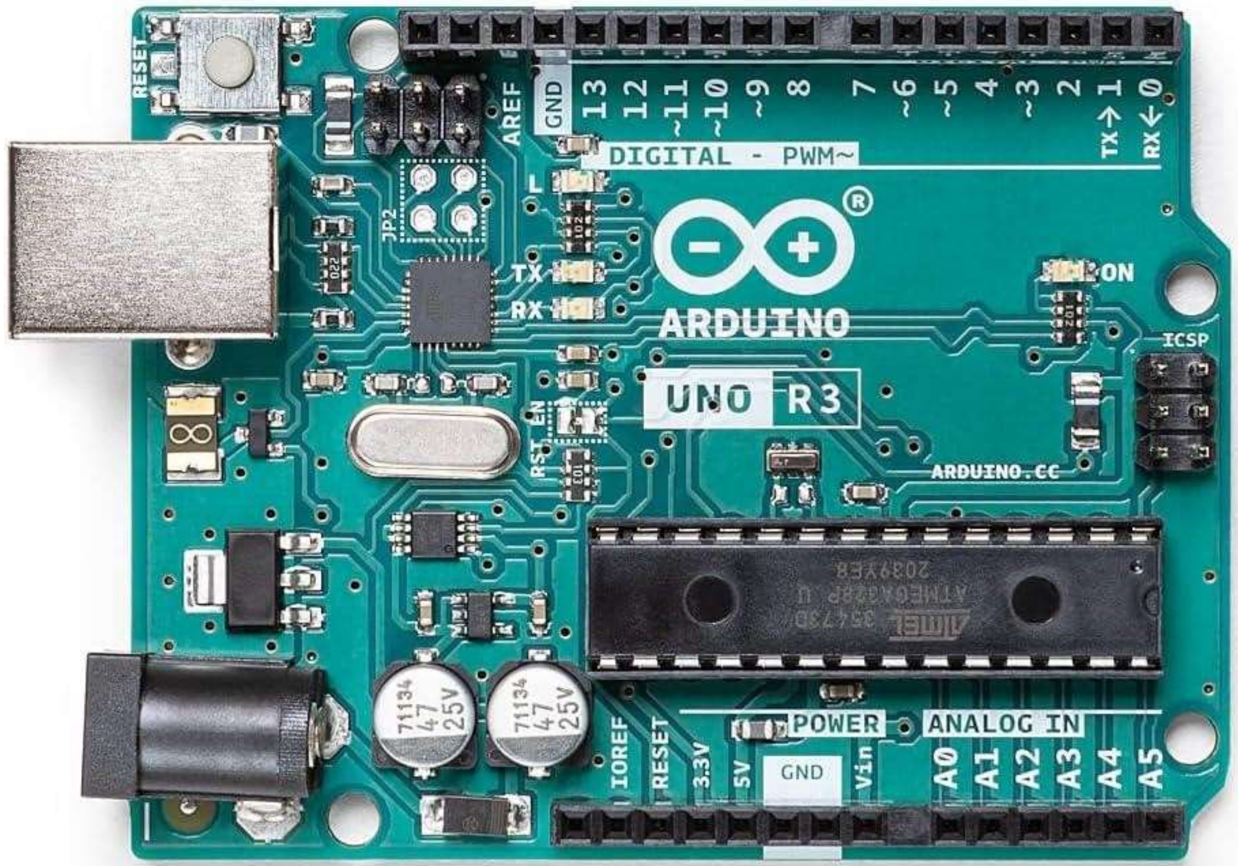
Microcontroller External Operation

- When a microcontroller is mounted on a circuit board with other components, it functions as a single unit, and is referred to as a module or a microcontroller development board.
- A microcontroller module typically consists of a microcontroller, a power source, an interface for connecting to a programming device, I/O ports, and additional memory.
- A power source powers the microcontroller and any accompanying components located on the printed circuit board.
- An interface communicates with the microcontroller.
- A set of input/output (I/O) ports send and receive signals from the devices the microcontroller is designed to control. I/O ports when programmed as an output pin, each pin can output digital signals. When programmed as an input pin, each pin can receive digital signals.
- Digital-to-analog and analog-to-digital converters change the digital pulses into analog signals.

Microcontroller External Operation Block Diagram



Arduino Uno – An Atmel ATMega328P Development Board



Advantages of Microcontrollers

- **Cost-Effective**: Microcontrollers are typically inexpensive, making them ideal for mass-produced devices and applications where cost is a major factor.
- **Low Power Consumption**: Many microcontrollers are designed to operate at low power, making them suitable for battery-operated devices and energy-efficient applications.
- **Compact Size**: Their small size allows them to be embedded directly within devices, even in small or portable electronics.
- **Real-Time Processing**: Microcontrollers can handle real-time tasks, making them ideal for applications where timing is critical, like robotics, automotive controls, and industrial automation.
- **Ease of Programming and Use**: Most MCUs support straightforward programming, with development environments (like Arduino IDE) that make it accessible to both beginners and professionals.
- **Integrated Peripherals**: They often come with built-in peripherals (such as ADCs, timers, UART, I2C, SPI) that simplify interfacing with sensors, actuators, and other components.
- **Single-Chip Solution**: A microcontroller combines CPU, memory, and I/O peripherals on a single chip, reducing the need for additional components and saving space.
- **Reliability**: Due to their simplicity and dedicated functionality, MCUs are reliable and have lower failure rates, which is essential for mission-critical applications.
- **Flexible Development Options**: Microcontrollers support a variety of programming languages and have a wide ecosystem of libraries, frameworks, and community support.
- **Stable and Long-Term Support**: Unlike complex computer processors, MCUs have a stable architecture, which ensures software longevity and reduces the need for frequent hardware updates.

Application Areas of Microcontrollers

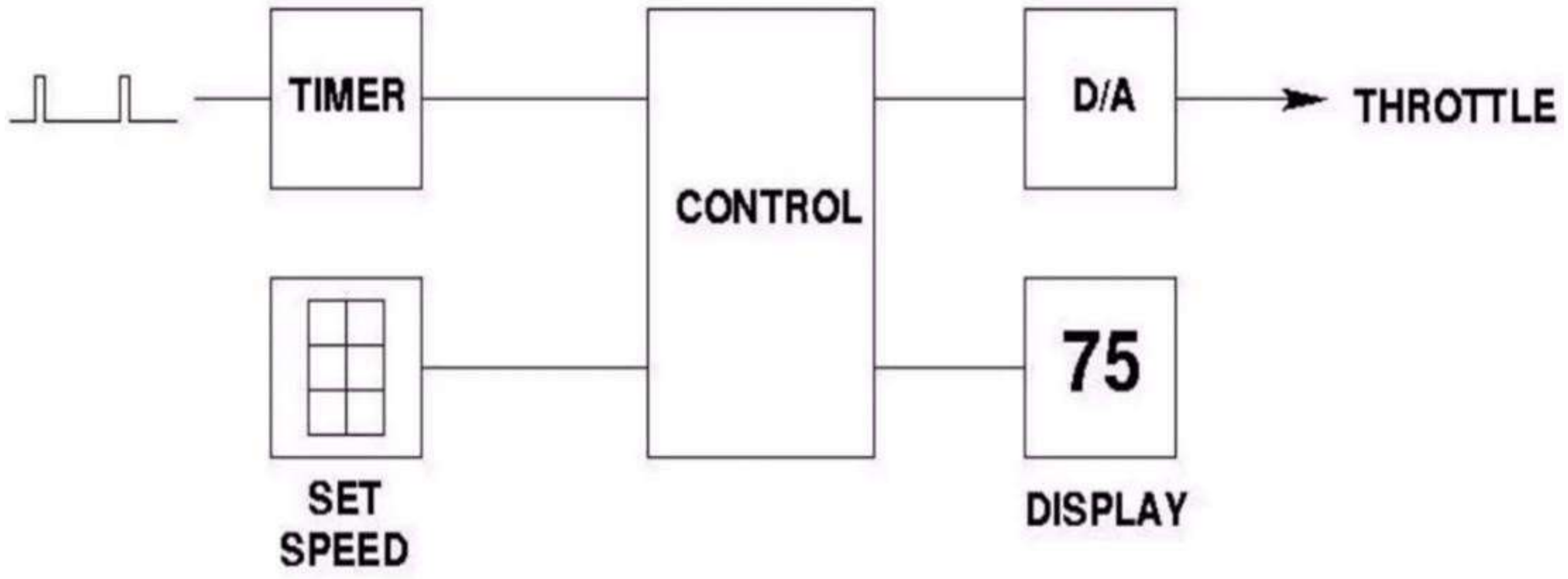
- **Consumer Electronics**
  - **Home Appliances**: Washing machines, microwaves, refrigerators, and air conditioners often use microcontrollers to manage operations.
  - **Smart Home Devices**: Lighting systems, thermostats, security systems, and smart door locks.
  - **Wearable Technology**: Smartwatches, fitness trackers, and health monitors.
- **Automotive Applications**
  - **Engine Control Systems**: Engine management and fuel injection control for optimized performance.
  - **Safety Systems**: Anti-lock braking systems (ABS), airbag control, and stability control.
  - **Infotainment**: Entertainment systems, display controls, and navigation systems.
  - **Telematics**: Communication systems for vehicle data transmission and diagnostics.
- **Industrial Automation**
  - **Process Control**: Monitoring and control of processes in manufacturing, chemical plants, and food production.
  - **Robotics**: Controlling robotic arms, automated guided vehicles (AGVs), and CNC machines.
  - **Sensors and Actuators**: Temperature, pressure, and other sensors for feedback and control systems.
- **Medical Devices**
  - **Portable Medical Equipment**: Blood pressure monitors, glucose meters, and heart rate monitors.
  - **Diagnostic Equipment**: Ultrasound machines, ECG machines, and MRI machines.
  - **Implantable Devices**: Pacemakers and insulin pumps, where low power and reliability are critical.
- **Telecommunications**
  - **Networking Equipment**: Routers, modems, and switches for managing data traffic.
  - **Signal Processing**: Devices for encoding, decoding, and modulation.
  - **Remote Monitoring**: Equipment used in remote cellular towers for monitoring and maintenance.

Application Areas of Microcontrollers (Continued)

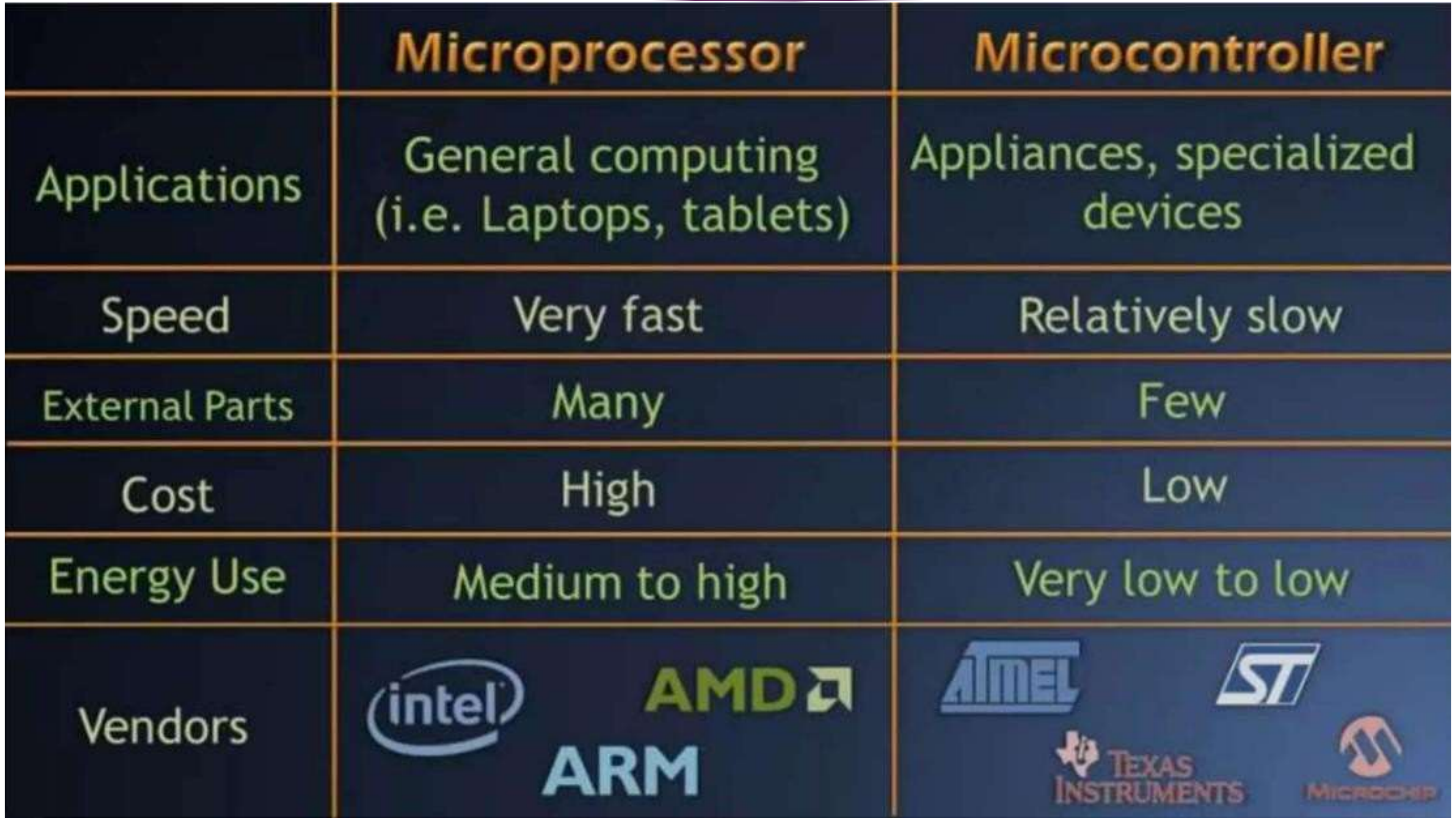
- **Internet of Things (IoT)**
  - **Smart Cities**: Waste management, smart street lighting, and environmental monitoring.
  - **Agriculture**: Soil moisture sensors, weather stations, and crop monitoring systems.
  - **Asset Tracking**: For logistics, tracking, and monitoring the condition of goods in transit.
- **Aerospace and Defense**
  - **Avionics**: Flight control systems, navigation, and autopilot.
  - **Missile Guidance and Control**: For precision in targeting and flight path correction.
  - **Military Communications**: Secure and low-power communication systems for soldiers and equipment.
- **Education and Research**
  - **STEM Education Kits**: Microcontrollers like Arduino and Raspberry Pi are used widely in educational kits to teach programming and electronics.
  - **Prototyping and Testing**: For building and testing new electronic designs in academic research labs and R&D.
- **Environmental Monitoring**
  - **Weather Stations**: Monitoring temperature, humidity, wind speed, and other weather conditions.
  - **Pollution Monitoring**: Air and water quality measurement devices for industrial and environmental monitoring.
  - **Wildlife and Habitat Monitoring**: Remote sensors for tracking animals and monitoring habitats.
- **Renewable Energy**
  - **Solar Power Systems**: Controllers for solar panels, battery management, and tracking systems.
  - **Wind Turbines**: Control systems for optimizing turbine angle and power output.
  - **Energy Meters**: Smart metering devices for monitoring and controlling energy use.

A Typical Microcontroller Application – Automobile Cruise Control

Speed Measurement



Comparison Between Microcontroller and Microprocessor



	Microprocessor	Microcontroller
Applications	General computing (i.e. Laptops, tablets)	Appliances, specialized devices
Speed	Very fast	Relatively slow
External Parts	Many	Few
Cost	High	Low
Energy Use	Medium to high	Very low to low
Vendors	intel AMD ARM	ATMEL ST TEXAS INSTRUMENTS MICROCHIP

Criteria of Choosing a Microcontroller

- **Processing Power (CPU Architecture and Clock Speed)**
  - **CPU Architecture**: ARM, AVR, PIC, RISC-V, etc. Some architectures are better for specific applications, such as ARM for high-performance applications.
  - **Clock Speed**: Determines the speed at which the MCU can process instructions. Applications requiring real-time processing or fast data handling need higher clock speeds.
- **Memory (RAM and Flash)**
  - **Flash Memory**: For storing the program code. Applications with complex code or large libraries need more flash memory.
  - **RAM**: For storing variables and temporary data during runtime. Memory-intensive applications (like those using complex algorithms or handling large datasets) require more RAM.
- **I/O Capabilities**
  - **GPIO Pins**: Number and type of general-purpose I/O pins. Consider whether the MCU has enough pins for the components and sensors you plan to connect.
  - **Analog Inputs/Outputs**: For analog sensors, ensure the MCU has enough ADC (Analog-to-Digital Converter) channels and DAC (Digital-to-Analog Converter) support if needed.
- **Power Consumption**
  - **Active/Idle Power**: Essential for battery-powered applications. Some MCUs offer low-power modes to conserve energy.
  - **Power Modes**: Look for ultra-low-power or sleep modes if low energy consumption is critical, as in IoT or wearable devices.
- **Communication Interfaces**
  - **UART, SPI, I2C**: Basic communication protocols. Multiple peripherals often require various protocols.
  - **CAN, USB, Ethernet, Wi-Fi, Bluetooth**: Specialized protocols for specific applications, such as automotive (CAN), networking (Ethernet), or wireless communication (Bluetooth/Wi-Fi).
- **Operating Voltage**
  - Ensure compatibility with other components in the circuit, as some MCUs operate at 5V, while others operate at 3.3V or lower.

Criteria of Choosing a Microcontroller (Continued)

- **Package Type and Size**
  - **Physical Size and Form Factor**: Depending on the project, consider the size (e.g., DIP, QFN, or BGA). Small form factors may be essential for wearables or portable devices.
  - **Ease of Soldering**: Some packages are easier to solder manually, while others might require specialized tools or manufacturing processes.
- **Cost**
  - Choose an MCU that fits within the project's budget. Consider production volumes, as costs per unit may vary greatly.
- **Development Tools and Ecosystem**
  - **IDE Support**: Look for compatible Integrated Development Environments (IDEs) and libraries.
  - **Documentation and Community**: Choose an MCU with good documentation, resources, and community support, which can simplify development and debugging.
- **Temperature and Environmental Specifications**
  - For applications in harsh environments, choose MCUs rated for extreme temperatures or high humidity conditions.
- **Security Features**
  - For applications where data security is critical, consider MCUs with encryption modules, secure boot, and tamper detection features.
- **Production Availability and Longevity**
  - Verify that the MCU will be available for the expected production lifetime of the product, as some MCUs may become obsolete quickly.

QUESTIONS?

THANK YOU