



---

# Introduction to Embedded Systems

## Unit 1.1: Embedded System Applications, Design, Organization, and Architecture

Alexander Yemane

BCIT

INCS 3610

Fall 2025

---

# Embedded Systems...

## What Are Those?

# Consumer Devices



**PHILIPS HUE – SMART BULBS**

**SAMSUNG - SMART THINGS**

**SMART DOOR LOCKS**

**AMAZON ALEXA/ECHO**

**APPLE IPHONE AND IWATCH**

**BELKIN WEMO – BABY MONITORS**



# Commercial Devices



**TESLA MODEL X / MODEL S**

**CARDIAC PACEMAKERS**



**INSULIN PUMPS**



**AIR QUALITY / TEMP CONTROL**



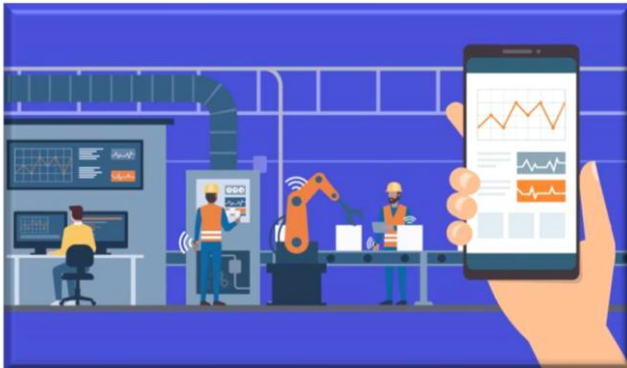
# Industrial Devices

HMI

**SMART AGRICULTURE**

**STATISTICAL CONTROL**

PLC



# Infrastructure Devices

**TRAFFIC LIGHTS**

**ELECTRICAL GRID**

**CITY LIGHTING**



# Military Devices

**GUNS AND WEAPONS**

**SURVEILLANCE TECH**

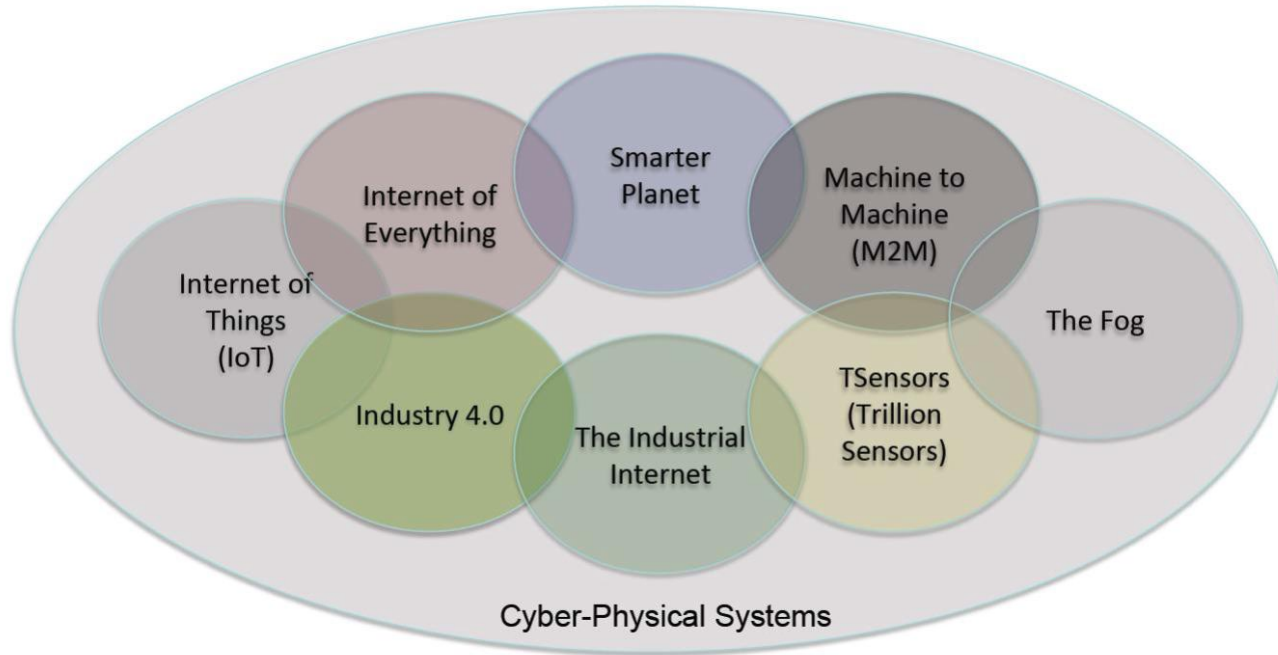
**BEACONS**

**BIOMETRICS**





# Many names – similar meanings

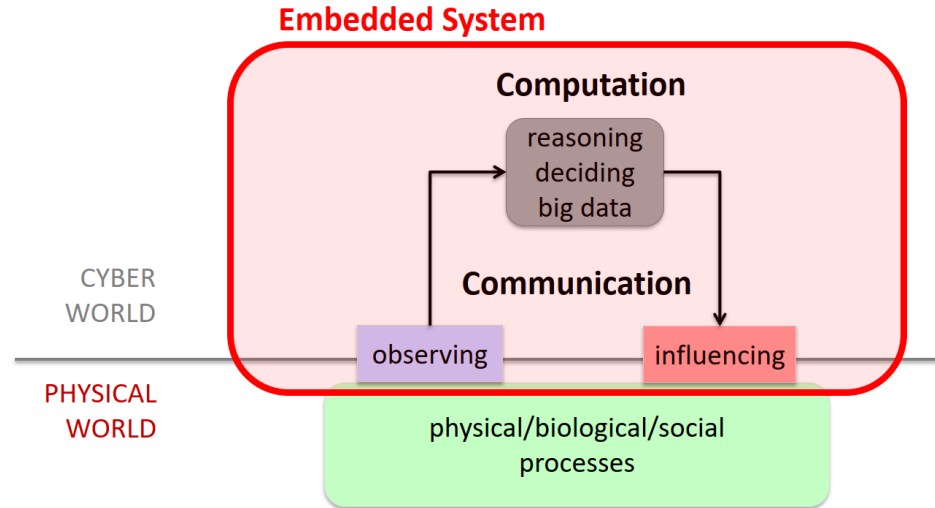




An embedded system is nearly any computing system other than a desktop, laptop, or mainframe computer.

# Intersection of cyber and physical processes

- An embedded system combines three main processes:
  - Computation
  - Communication
  - Physical dynamics



# Common characteristics of embedded systems

- Predictability & Dependability
- Efficiency & Specialization
- Reactivity & Timing

# Predictability & Dependability

- Embedded systems often execute a single program, repeatedly
- “It is essential to *predict* how a CPS is going to behave under any circumstances [...] *before* it is deployed.” R. Majumdar & B. Brandenburg (2014). Foundations of Cyber-Physical Systems

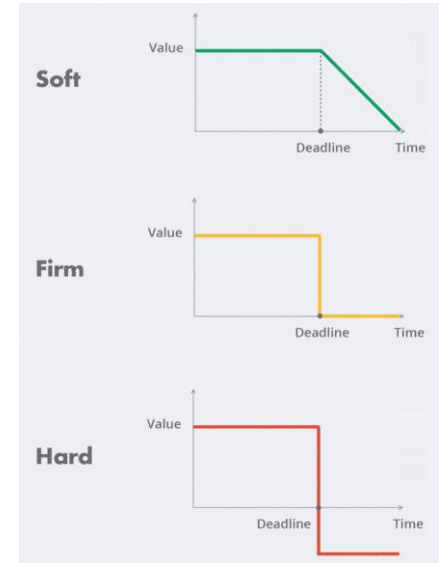
# Efficiency & Specialization

- Embedded systems must be efficient:
  - *Energy* efficient
  - *Code-size* and *data memory* efficient
  - *Run-time* efficient
  - *Weight* efficient
  - *Cost* efficient
- Embedded systems are often specialized towards a certain application or application domain

# Reactivity & Timing

- Embedded systems are often reactive:
  - Reactive systems must *react to stimuli* from the system environment
- Embedded systems often must meet real-time constraints:
  - For *hard* real-time systems, right answers arriving too late are wrong.

“A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe” [Kopetz, 1997].
  - Other time-constraints are called *soft* or *firm*.



# Typical design criteria

- Unit cost
- Non-recurring engineering (NRE) cost
- Size
- Performance
- Power consumption
- Flexibility
- Time-to-prototype
- Time-to-market
- Maintainability
- Correctness
- Safety
- Security



## Segue: This week's headline on *The Registrar*

**The Registrar®**

# Frostbyte10 bugs put thousands of refrigerators at major grocery chains at risk

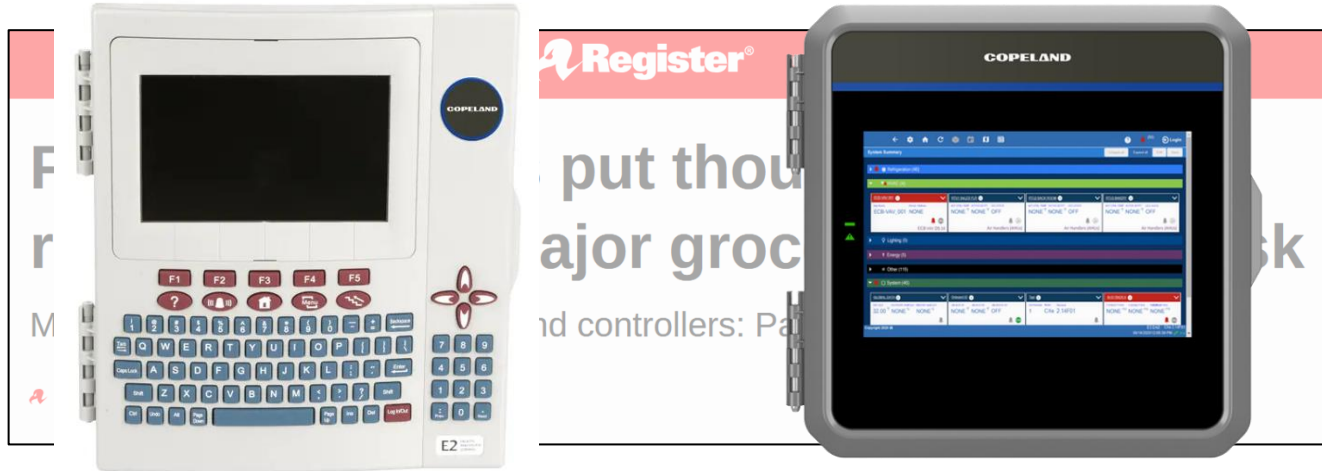
Major flaws uncovered in Copeland controllers: Patch now

 [Jessica Lyons](#)

Tue 2 Sep 2025 // 09:00 UTC

Copeland controllers are widespread in North American grocery chains

# Segue: This week's headline on *The Registrar*



Copeland E2  
Facility Management System

Copeland E3  
Supervisory Control

# Understanding the severity and criticality of Frostbyte10 – early preview into embedded system security

- Attackers could manipulate temperatures, disrupt lighting/HVAC, spoil food/medicine, and cause operational downtime
- Some vulnerabilities:
  - Predictable admin account and root password
  - Possible remote access via SSH
  - Unsigned firmware
- Fortunately, no known exploitation in the wild at disclosure time, but the large scale raises stakes

<sup>1</sup> [https://www.theregister.com/2025/09/02/frostbyte10\\_copeland\\_controller\\_bugs/](https://www.theregister.com/2025/09/02/frostbyte10_copeland_controller_bugs/)

<sup>2</sup> <https://www.armis.com/research/frostbyte10/>

# Embedded systems vs General-purpose computing

## Embedded systems

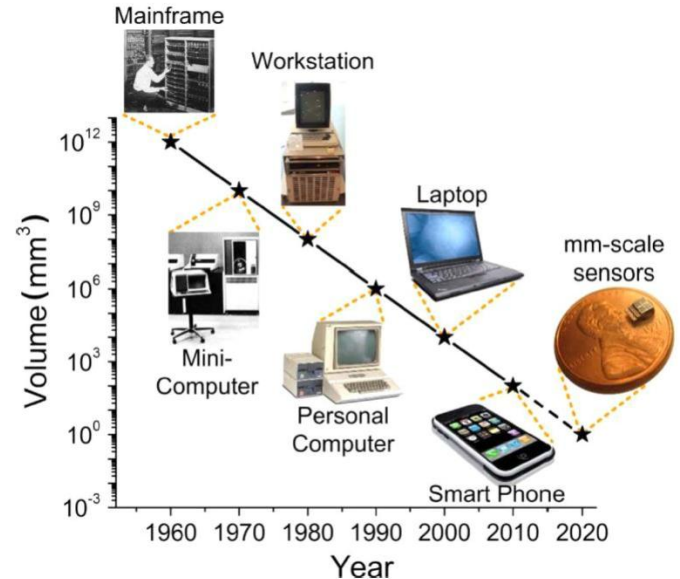
- Few applications that are known at design-time.
- Not programmable by end user.
- Fixed run-time requirements (additional computing power often not useful).

## General-purpose computing

- Broad class of applications.
- Programmable by end user.
- Faster is better.

# Trends

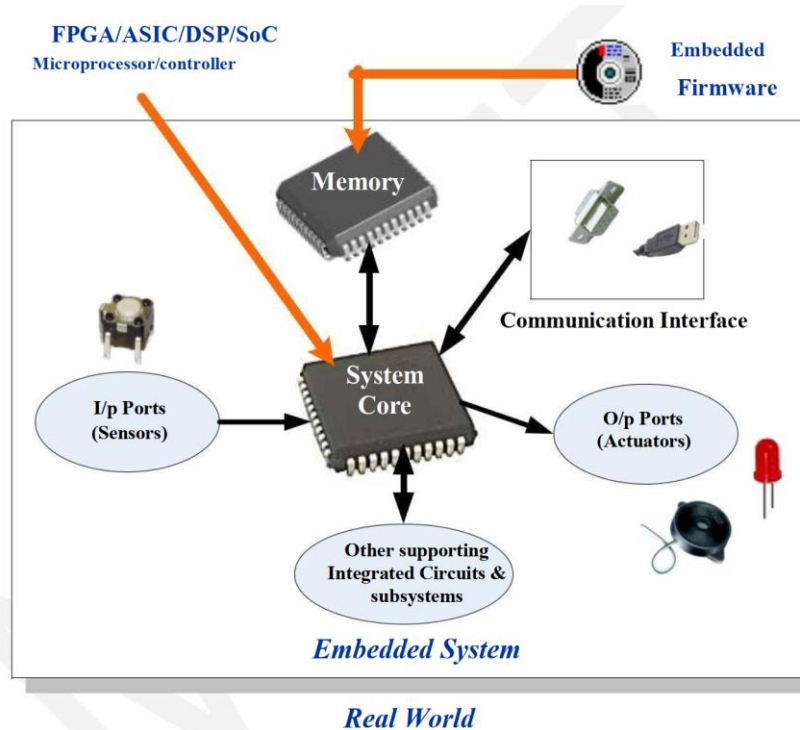
- *Embedded systems are communicating with each other*, with servers or with the cloud. Communication is increasingly wireless.
- *Higher degree of integration* on a single chip or integrated components
- *Low power and energy constraints* (portable or unattended devices) are increasingly important, as well as temperature constraints (overheating).
- There is increasing interest in *energy harvesting* to achieve long term autonomous operation.



---

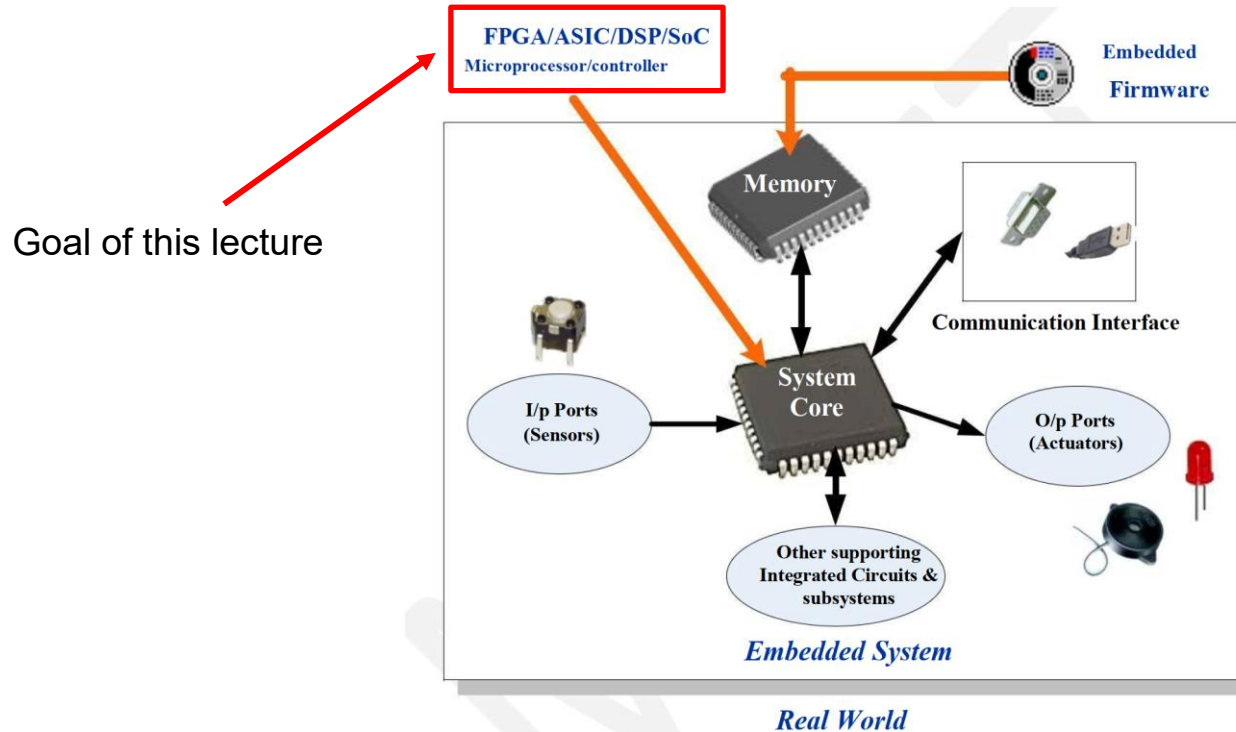
# The Core of an Embedded System

# Elements of a typical embedded system





# Elements of a typical embedded system



An embedded processor is the core or “brain” of an embedded system that performs most of the computational tasks.

# Types of embedded processors

- **Application-specific integrated circuits (ASIC)**
  - Energy efficient
  - Low flexibility, designed for a specific task-application high cost to market
- **Digital signal processor (DSP)**
  - Specialized microprocessor (i.e. to audio streams, video stream, etc.)
  - Cost-energy effective
- **Microcontroller (MCU)**
  - Low cost and fully integrated solutions including control
  - Toward general purpose but still special purpose



MCU drone

# Types of embedded processors

- **Microprocessors (MPU)**
  - General purpose (i.e. Intel processors)
  - High performance
- **System on a chip (SoC)**
  - Can include all the above architecture in a single integrated chip
- **Field programmable gate array (FPGA)**
  - Based on rewritable memory technology, can be quickly reprogrammed for various applications
  - High logic density, performance, and range of features



FPGA drone



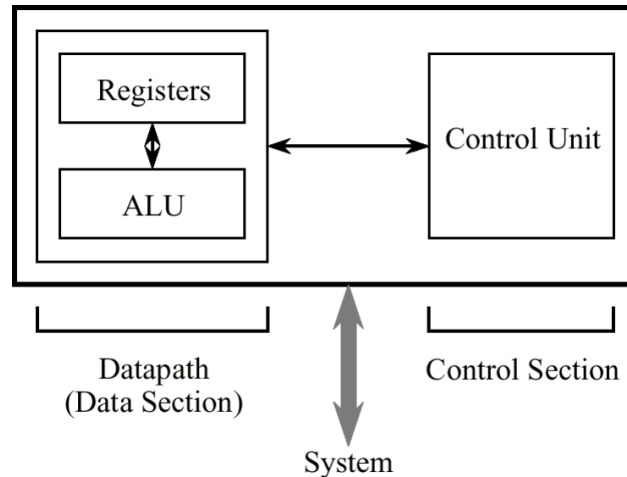
GPU drone

---

# The Central Processing Unit & Processor Architecture

# Central processing unit

- A central processing unit (CPU) consists of a *data section* containing registers and an arithmetic logic unit (ALU), and a *control section*, which interprets instructions and effects register transfers. The data section is also known as the *datapath*.



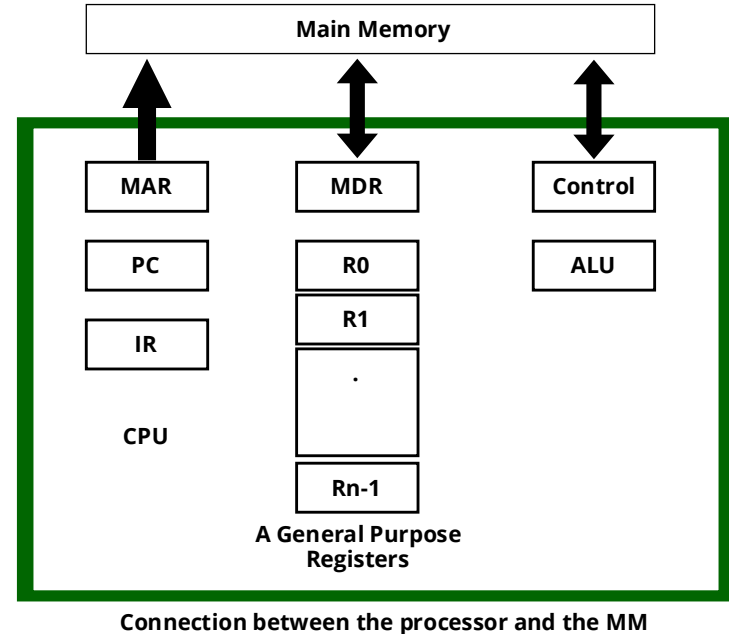
# Registers

- Registers are a fundamental building block within a computer system
- Very fast computer memory
- Used to store data/instruction in-execution



# Types of CPU registers

- **General purpose registers**
  - $R_0 \dots R_{n-1}$
- **Special purpose registers**
  - Memory Address Register (MAR)
  - Memory Data Register (MDR)
  - Accumulator
  - Instruction Register (IR)
  - Program Counter (PC)
  - Flag Register

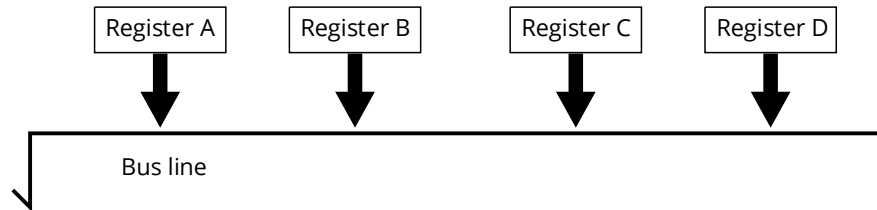


# Buses

- Address bus
  - Carries the address of a specified location to read or write to.
  - The address width (e.g., 16-bit, 32-bit) determines the maximum memory capacity the system can access.
    - E.g., a 16-bit address bus can access  $2^{16} = 65,536$  memory locations
- Data bus
  - Carries the actual information between the CPU and memory or I/O devices.
- Control bus
  - Carries control signals supplied by the CPU to synchronize the movement of information on the address and data bus.

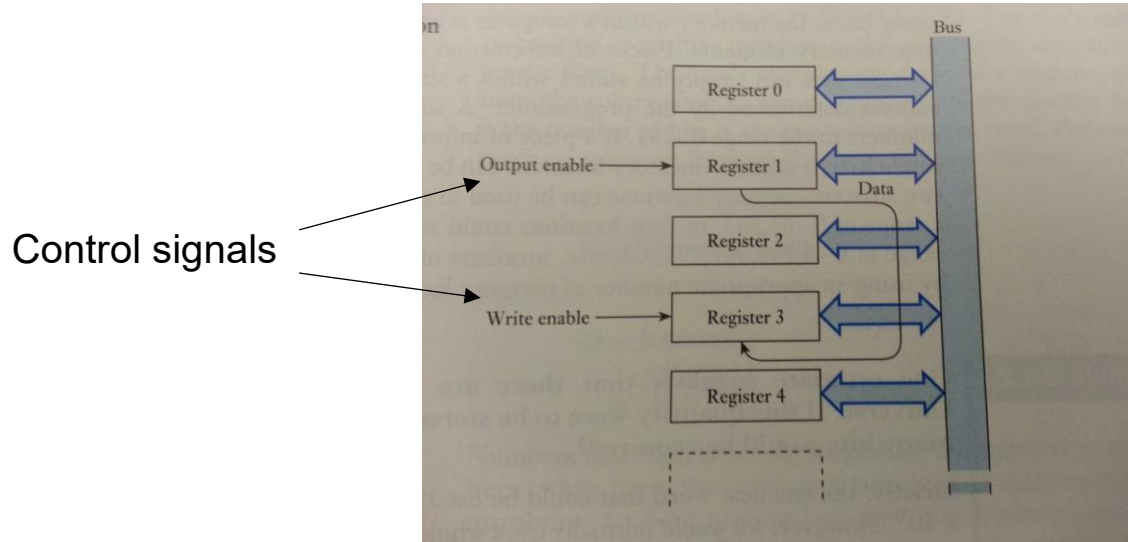
# Bus transfer

- It is impractical to have data and control lines to directly allow each register to be loaded with the contents of every possible other register
- This is why computer systems use a shared communication pathway, aka a *bus*, allowing efficient data transfer
- A bus is a path (of a group of wires) over which information is transferred, from any of several sources to any of several destinations.
  - It has control circuits to select which register is the source, and which is the destination.



# Data busing

- Reading: The process of taking information from register and placing it on the data bus
- Writing: The process of storing information in a register



# Interrupt system

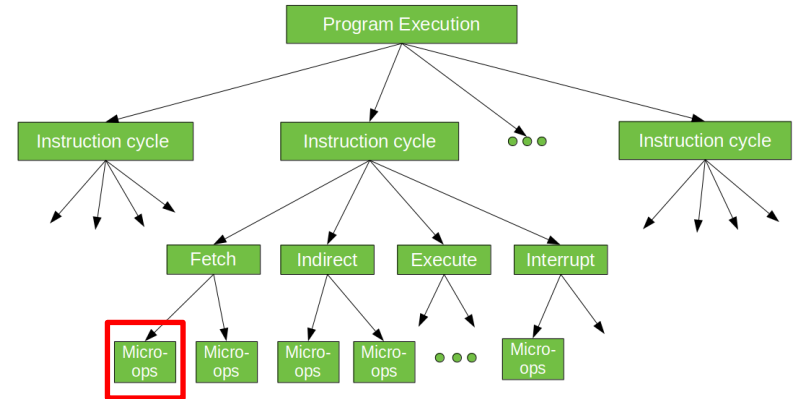
- An interrupt is a mechanism by which internal/external modules may interrupt the normal execution of a program.
- 'Wakes up' CPU when it is required for an urgent task or I/O operation.
- Avoids wasting time polling slow devices (particularly useful for microcontrollers) and save power.
- More details explained in future lecture.

Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?

Polling example

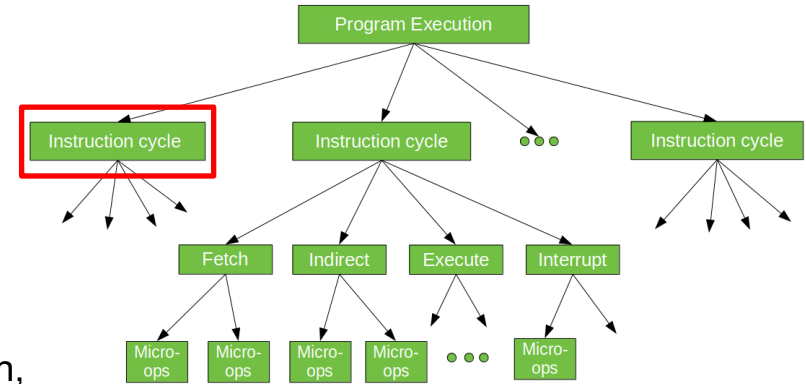
# Micro-operations

- Micro-operations are elementary/atomic operations performed (during one clock pulse) on the data stored in one or more registers
- Break down complex machine instructions into smaller, executable steps like:
  - Data transfer between registers or between external buses of the CPU, e.g. load, store
  - Arithmetic operations, e.g. add, subtract
  - Logic operations, e.g. AND, OR
  - Shift operations, e.g. shift R, rotate R



# Instruction cycle

- Every program is a collection of instructions, where each instruction is executed with the help of an *instruction cycle*.
- An instruction cycle is divided into 4 phases:
  1. *Fetch* the instruction from memory
  2. *Decode* the instruction into commands
  3. *Execute* the commands
  4. *Store* the processed data in memory
- After going through each phase for the first instruction, the CPU again starts from phase 1 for the second instruction and repeats the cycle.



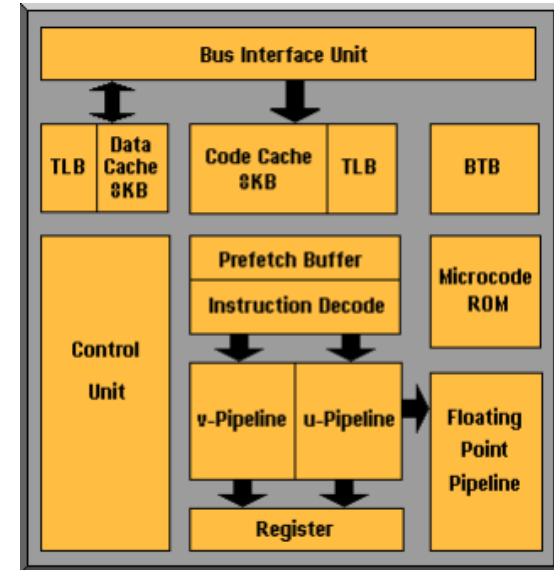


# Internal hardware organization of the CPU

- Set of registers and their functions
- Set of allowable microoperations performed on data stored in registers.
- Control signals that initiate the sequence of microoperations that we want to perform

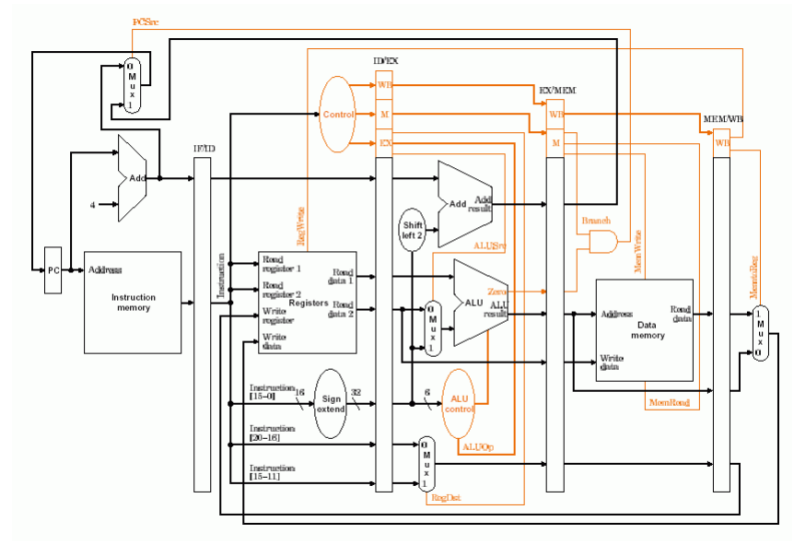
# CPU characteristics and examples

- **Instruction set architecture (ISA)**
  - Complex instruction set computer (CISC),  
e.g. Intel x86 family, Motorola 680x0 family
  - Reduced instruction set computer (RISC),  
e.g. PowerPC, ARM family, ATMEL AVR family
- **Word size**
  - 8-bit, e.g. Intel 8051, Motorola 6800, ATMEL AVR
  - 16-bit, e.g. Intel 8088, Motorola 68000, TI MSP430
  - 32-bit, e.g. x86 family, Motorola 680x0 family, PowerPC, ARM32
  - 64-bit, e.g. x86-64 family, PowerPC, ARM64



# CPU characteristics and examples

- **Instruction set architecture (ISA)**
  - Complex instruction set computer (CISC), e.g. Intel x86 family, Motorola 680x0 family
  - Reduced instruction set computer (RISC), e.g. PowerPC, ARM family, ATMEL AVR family
- **Word size**
  - 8-bit, e.g. Intel 8051, Motorola 6800, ATMEL AVR
  - 16-bit, e.g. Intel 8088, Motorola 68000, TI MSP430
  - 32-bit, e.g. x86 family, Motorola 680x0 family, PowerPC, ARM32
  - 64-bit, e.g. x86-64 family, PowerPC, ARM64



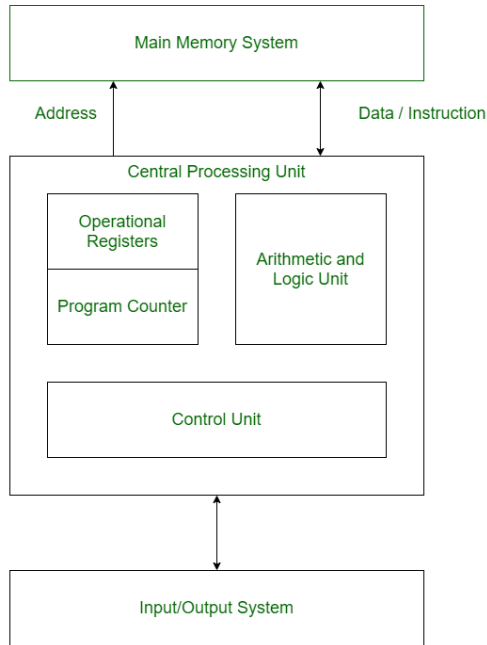
# Word size

- Word size is described in terms of bits
  - Corresponds to the data size that can be manipulated at a time by the processor
  - Typically reflected in the size of the processor datapath and register bank

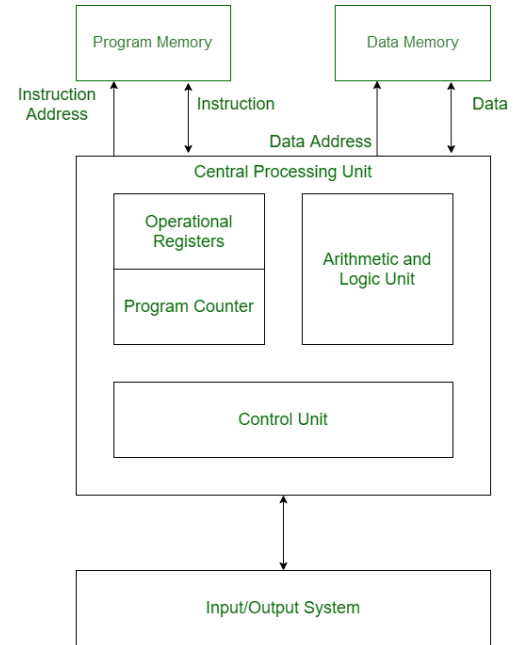
# Processor architecture

- Two types of information are found in program code:
  - *Instruction codes* for execution
  - *Data* that is used by the instruction codes
- Two common processor architecture models define how data is exchanged between the CPU and memory:
  - *von Neumann architecture*
  - *Harvard architecture*

# Processor architecture



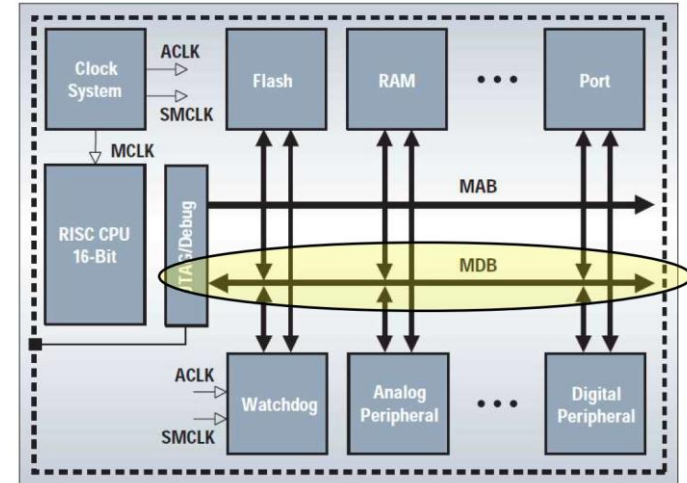
von Neumann architecture



Harvard architecture

# von Neumann architecture

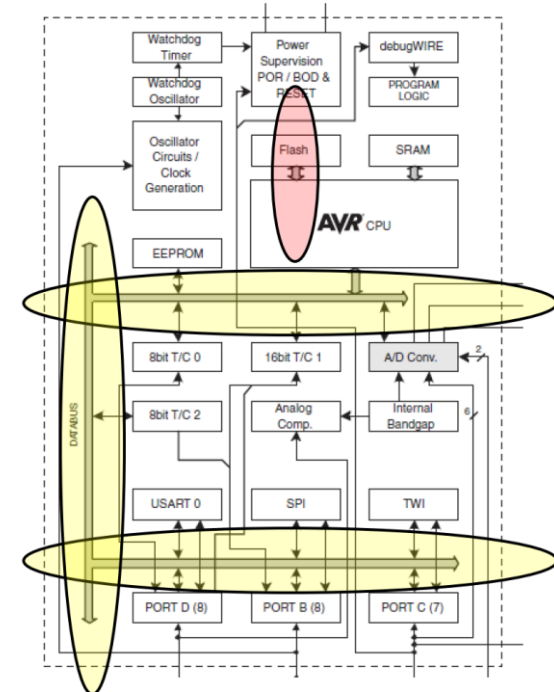
- Single memory interface bus for both instruction fetching and data access
- More efficient use of memory
- More flexible programming style
- Data may overwrite code (e.g. due to program bug)
- Bottleneck in code and data transfer



TI MSP430

# Harvard architecture

- Separate instruction and data buses
- Allows code and data access at the same time
- Allows different sizes of data and instructions to be used
- Does not incur any code corruption by data
- But more sophisticated hardware glue logic is required to support multiple interface buses



ATMEGA AVR



# Other approaches to processor architecture: RISC and CISC

## Reduced instruction set computer (RISC)

- Simple operations
- Simple *addressing modes*
- Longer compiled program but faster to execute
- Uses *pipelining*
- Used in most embedded systems

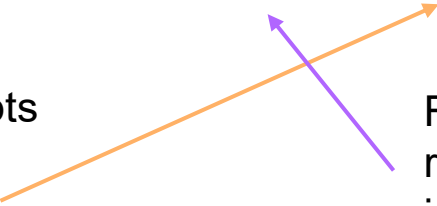
## Complex instruction set computer (CISC)

- More complex instructions (closer to high-level language support)
- x86 standard (Intel, AMD, etc.), but even in the mainframe territory CISC is dominant via the IBM/390 chip

Note: RISC and CISC architectures are becoming more and more alike.

# CISC vs RISC: The performance equation

- The following equation is commonly used for expressing a computer's performance ability:

$$\frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$$


The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction.

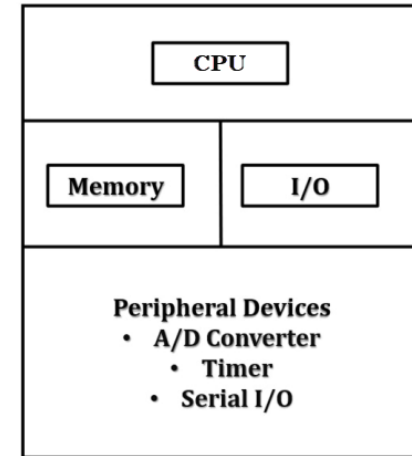
RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program.

---

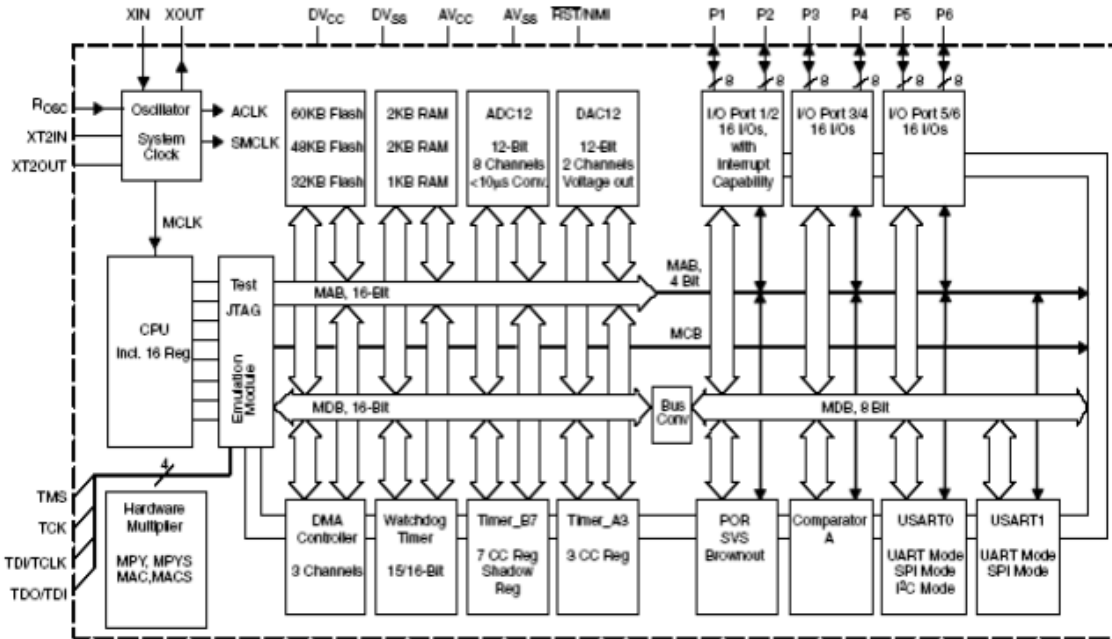
# Microcontrollers & Microprocessors

# Microcontroller

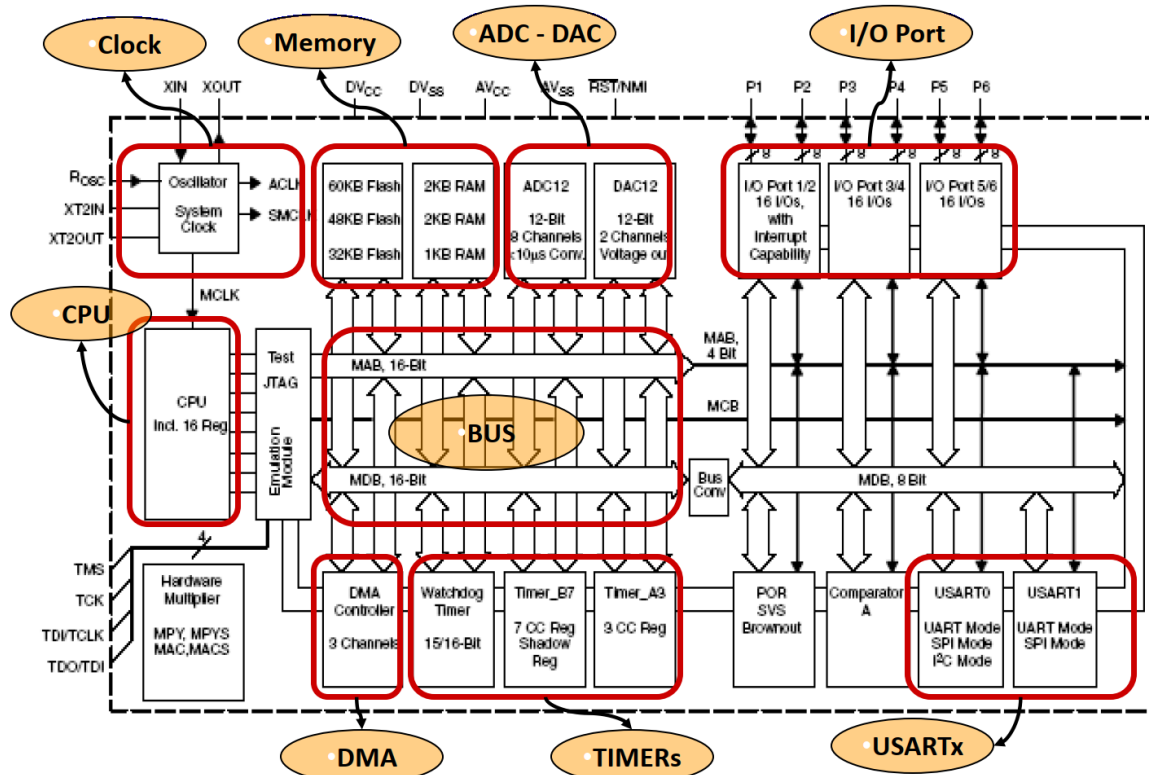
- A microcontroller (MCU) is a CPU with many support devices *built into the same chip*
  - Self-contained (CPU, memory, I/O)
  - Application- or task-specific (not a general-purpose computer)
  - Appropriately scaled for the job
  - Small power consumption
  - Low cost (\$0.50–\$5.00)



# Example of MCU architecture



# Example of MCU architecture



# Many MCUs

- ARM core processors (from many vendors)
- Atmel AVR (8-bit), AVR32 (32-bit), and AT91SAM (32-bit)
- Cypress Semiconductor PSoC (Programmable System-on-Chip)
- Freescale ColdFire (32-bit) and S08 (8-bit)
- Freescale 68HC11 (8-bit)
- Intel 8051
- Infineon: 8, 16, 32 Bit microcontrollers<sup>[9]</sup>
- MIPS
- Microchip Technology PIC, (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24), (32-bit PIC32)
- NXP Semiconductors LPC1000, LPC2000, LPC3000, LPC4000 (32-bit), LPC900, LPC700 (8-bit)
- Parallax Propeller
- PowerPC ISE
- Rabbit 2000 (8-bit)
- Renesas RX, V850, Hitachi H8, Hitachi SuperH (32-bit), M16C (16-bit), RL78, R8C, 78K0/78K0R (8-bit)
- Silicon Laboratories Pipelined 8051 Microcontrollers
- STMicroelectronics ST8 (8-bit), ST10 (16-bit) and STM32 (32-bit)
- Texas Instruments TI MSP430 (16-bit)
- Toshiba TLCS-870 (8-bit/16-bit).

# A bit about Advanced RISC Machine

- Advanced RISC Machine (ARM) is a British semiconductor (and software) design company that designs and licenses ARM processor cores to semiconductor manufacturers
  - They just sell the ARM core
  - Other manufacturers license the core from them and then design microcontrollers around that core by adding in peripherals and memory to suit their design goals
- Originally conceived to be a processor for the Acorn desktop system, now widespread in embedded systems
- Based on RISC architecture



TI MSP432



# A bit about Advanced RISC Machine

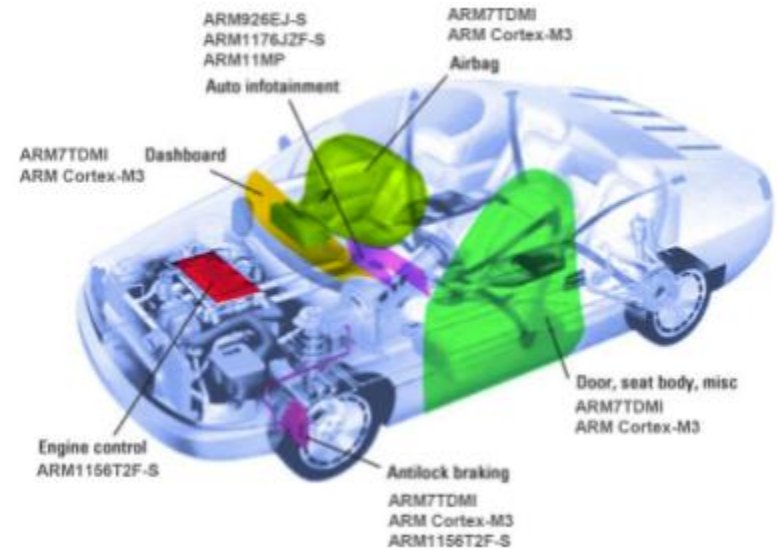
- Advanced RISC Machine (ARM) is a British semiconductor (and software) design company that designs and licenses ARM processor cores to semiconductor manufacturers
  - They just sell the ARM core
  - Other manufacturers license the core from them and then design microcontrollers around that core by adding in peripherals and memory to suit their design goals
- Originally conceived to be a processor for the Acorn desktop system, now widespread in embedded systems
- Based on RISC architecture



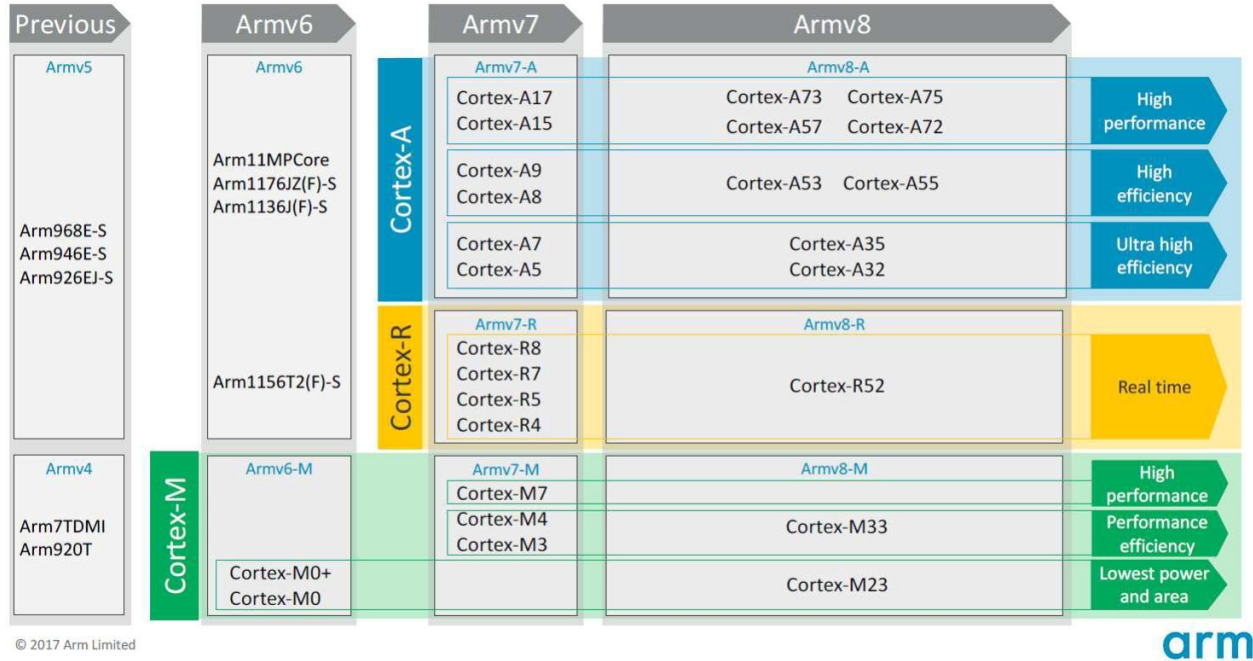
STM32U585  
(ARM Cortex-M33 core)

TI MSP432

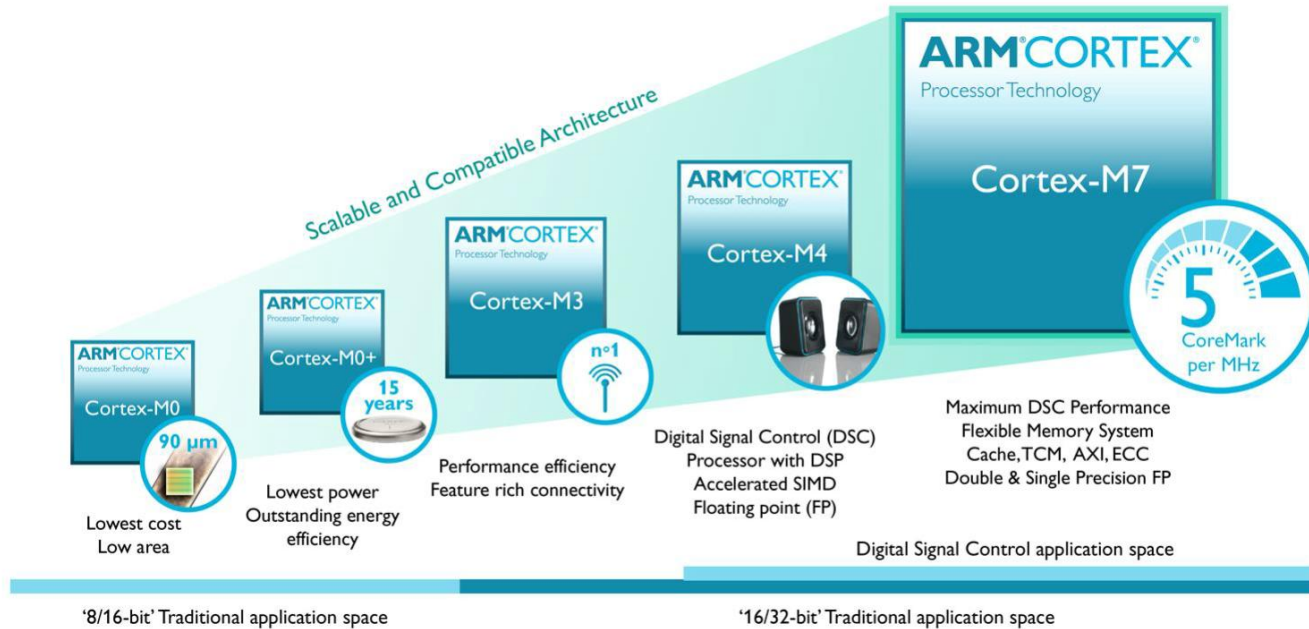
## Applications of ARM-based MCUs



# Different cores for different applications

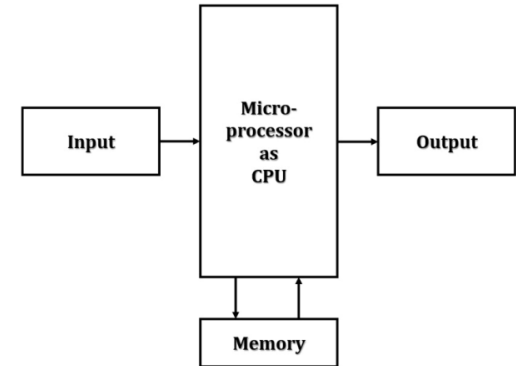


# ARM Cortex-M MCUs



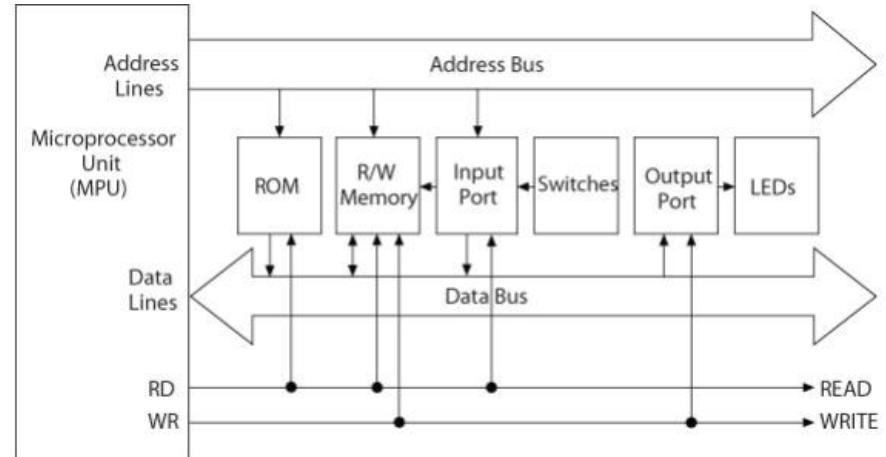
# Microprocessor

- A microprocessor (MPU) is a CPU *built into a single chip*
  - Relies on external components for memory and peripherals (I/O devices)
  - Contains millions of transistors connected by wires



# MPU with bus organization

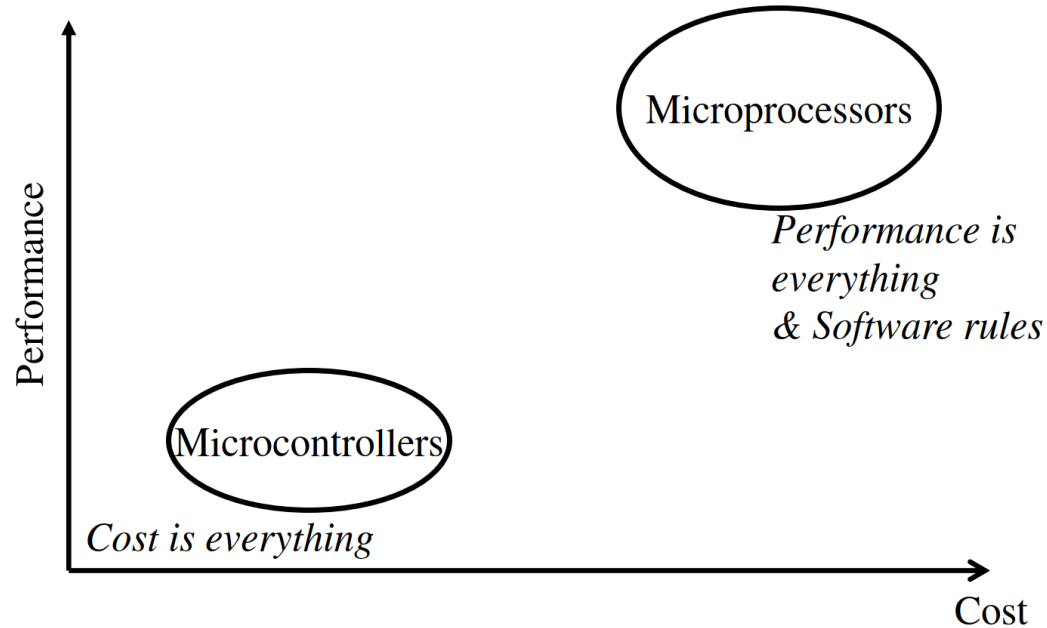
- Note the direction of the buses
  - Exception: control (RD/WR) buses can be uni-directional or bi-directional
- Note the width of the address and data buses



# Intel microprocessors historical perspective

Processor	Year of Introduction	Number of Transistors	Initial Clock Speed	Address Bus	Data Bus	Addressable Memory
4004	1971	2,300	108 kHz	10-bit	4-bit	640 bytes
8008	1972	3,500	200 kHz	14-bit	8-bit	16 K
8080	1974	6,000	2 MHz	16-bit	8-bit	64 K
8085	1976	6,500	5 MHz	16-bit	8-bit	64 K
8086	1978	29,000	5 MHz	20-bit	16-bit	1 M
...						
Pentium Pro	1995	5.5 M	150 MHz	36-bit	32/64-bit	64 G
Pentium II	1997	8.8 M	233 MHz	36-bit	64-bit	64 G
Pentium III	1999	9.5 M	650 MHz	36-bit	64-bit	64 G
Pentium 4	2000	42 M	1.4 GHz	36-bit	64-bit	64 G

# Microcontrollers vs Microprocessors





# Microcontrollers vs Microprocessors

## Microcontroller

- Includes RAM, ROM, serial and parallel interface, timer, interrupt schedule circuitry (in addition to CPU) in a single chip
- Used in small, minimum component designs performing control-oriented activities
- Have instruction sets catering to the control of inputs and outputs.
  - Instructions operate on a single bit
- Often use a Harvard architecture for better efficiency and real-time speed

## Microprocessor

- Single chip CPU
- Most commonly used as the CPU in (micro)computer systems
- Have instruction sets that are processing intensive, catering to large volumes of data
  - Instructions operate on nibbles, bytes, etc
- Often use a von Neumann architecture for better scalability and performance