Module 1:

Introduction to Embedded Systems

BCSE305L

Dr. Nitish Katal

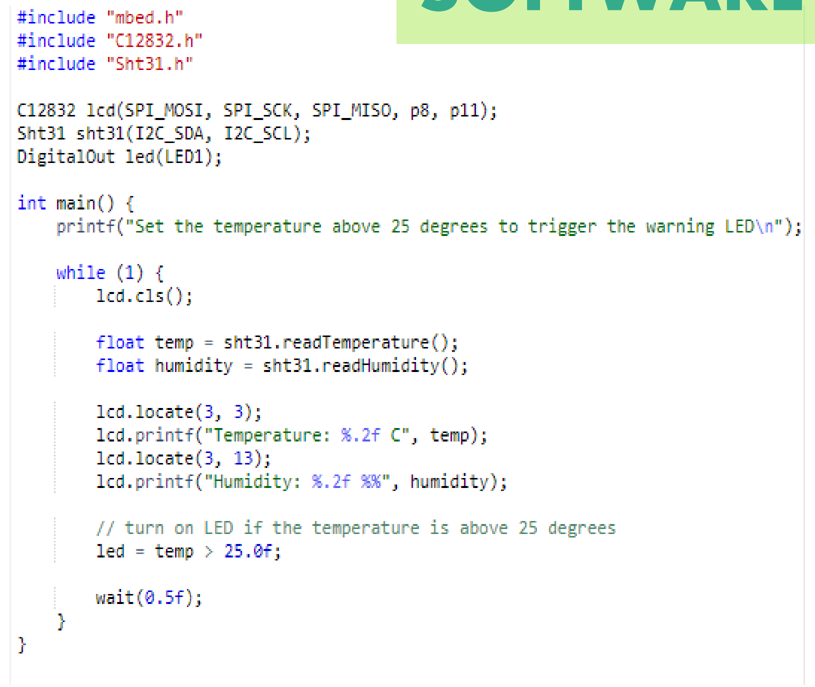
What is “Embedded System”

- **Embedded:**
  - ***Hidden*** inside, or
  - To ***fix*** something ***firmly*** and ***deeply***
- **System:**
  - ***Multiple components***/subsystems ***interfaced together*** for a ***special purpose***.
- **Embedded + System:**
  - ***Special-purpose computing system*** designed to ***perform certain dedicated functions***.
  - **Functionalities** : are done by *dedicated HW and SW* with *limited resources*.

What is "Embedded System"

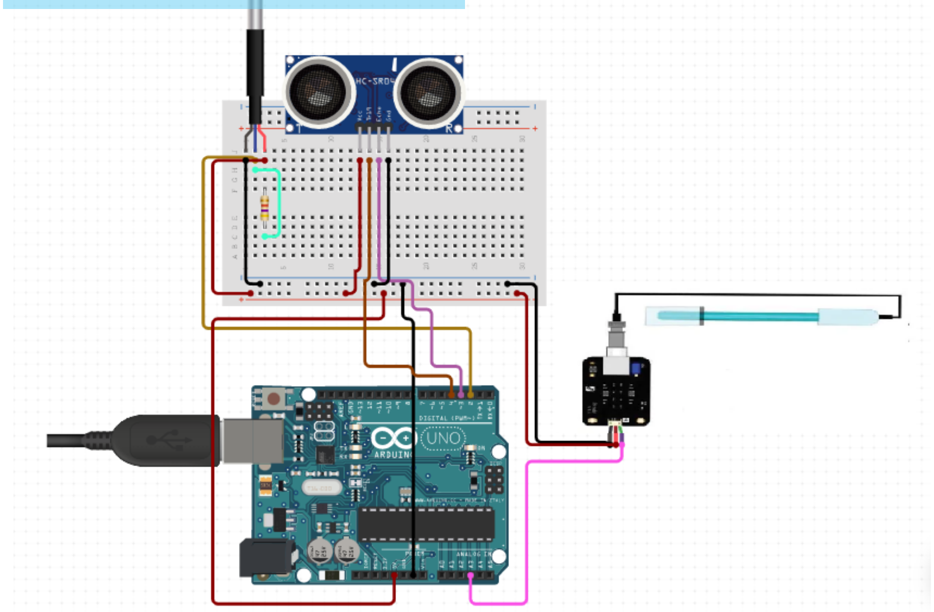
- **Embedded + System:** ***Special-purpose computing system*** designed to ***perform certain dedicated functions***.

SOFTWARE

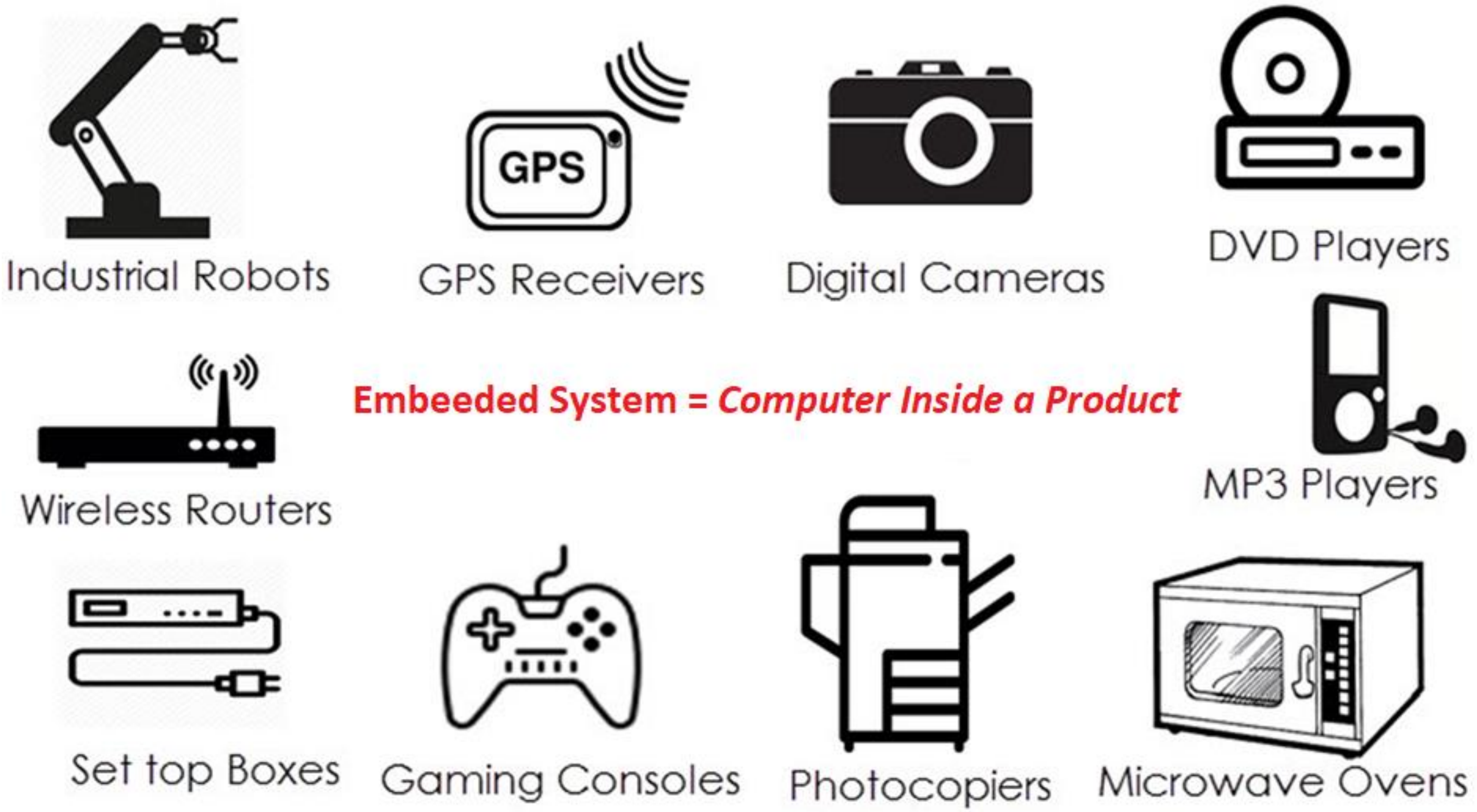


```
#include "mbed.h"
#include "C12832.h"
#include "Sht31.h"

C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Sht31 sht31(I2C_SDA, I2C_SCL);
DigitalOut led(LED1);

int main() {
    printf("Set the temperature above 25 degrees to trigger the warning LED\n");

    while (1) {
        lcd.cls();

        float temp = sht31.readTemperature();
        float humidity = sht31.readHumidity();

        lcd.locate(3, 3);
        lcd.printf("Temperature: %.2f C", temp);
        lcd.locate(3, 13);
        lcd.printf("Humidity: %.2f %%", humidity);

        // turn on LED if the temperature is above 25 degrees
        led = temp > 25.0f;

        wait(0.5f);
    }
}
```

HARDWARE



Examples of Embedded System

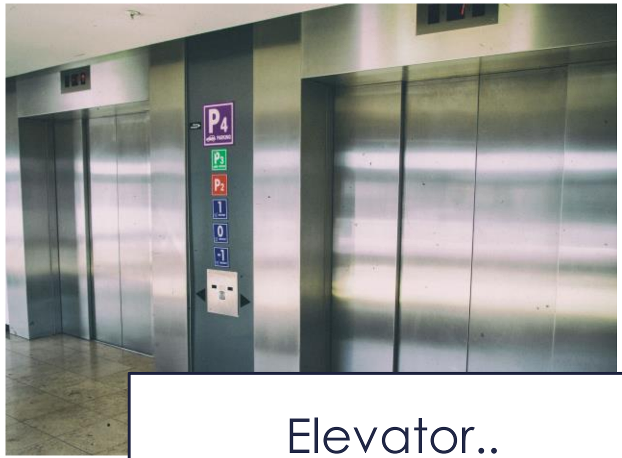


Embedded system in daily life....

Wake up...

Breakfast...

Washing...

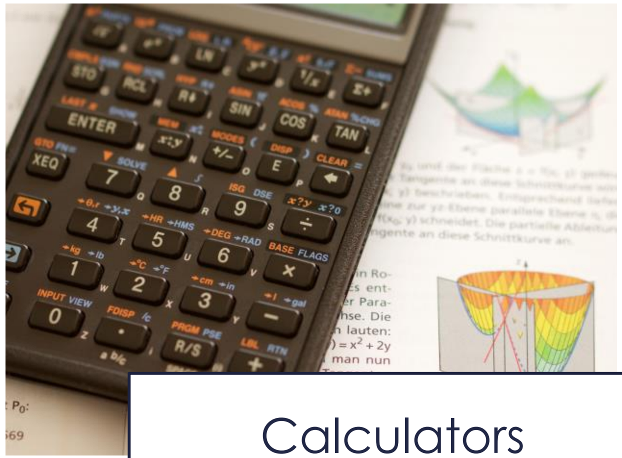


Elevator..

Car...

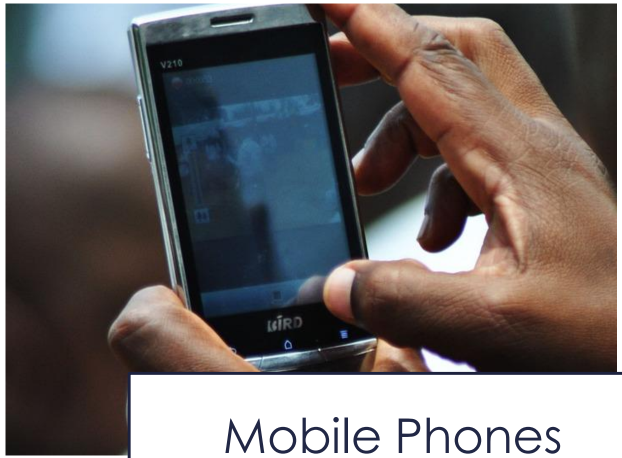
Traffic lights...

Embedded systems in Classroom….


Calculators

Projectors

Audio System


Mobile Phones

Scopes

Anything Else?

Embedded systems in Homes….

Smart TV

Setup Box

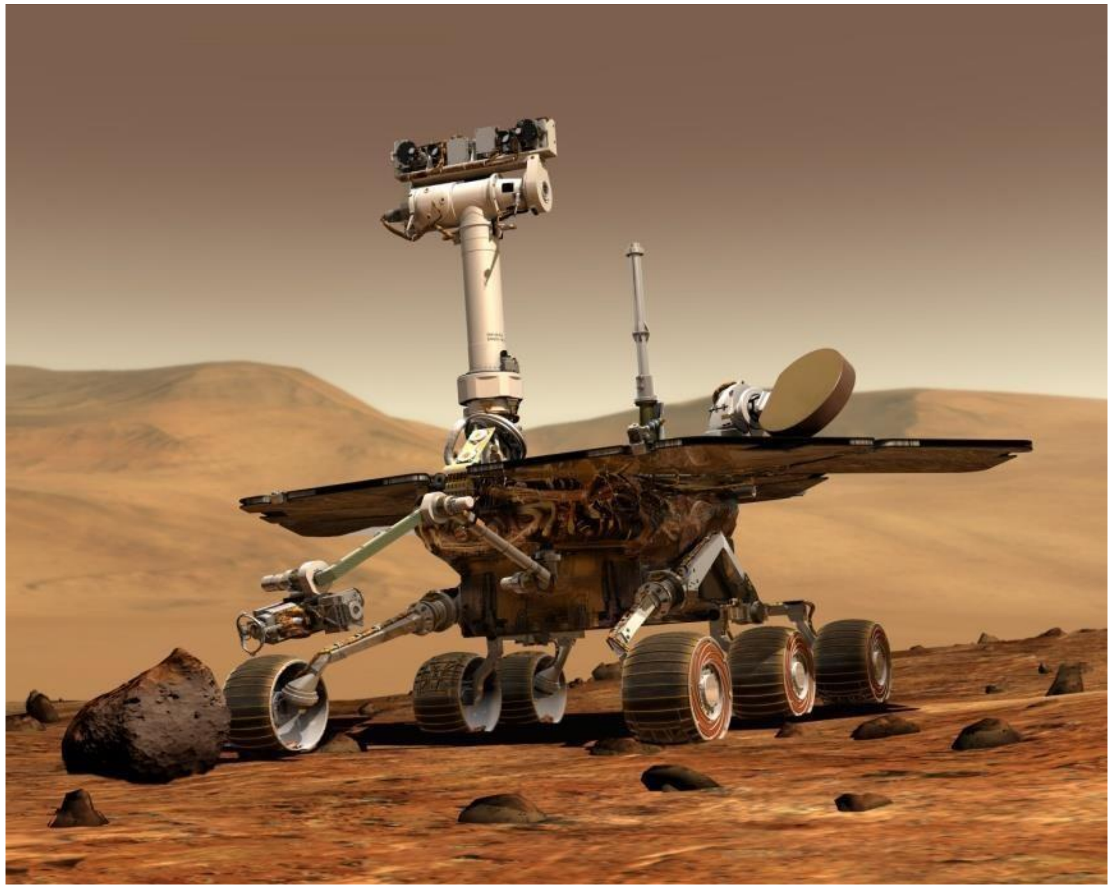
CCTVs

Cameras

Printers

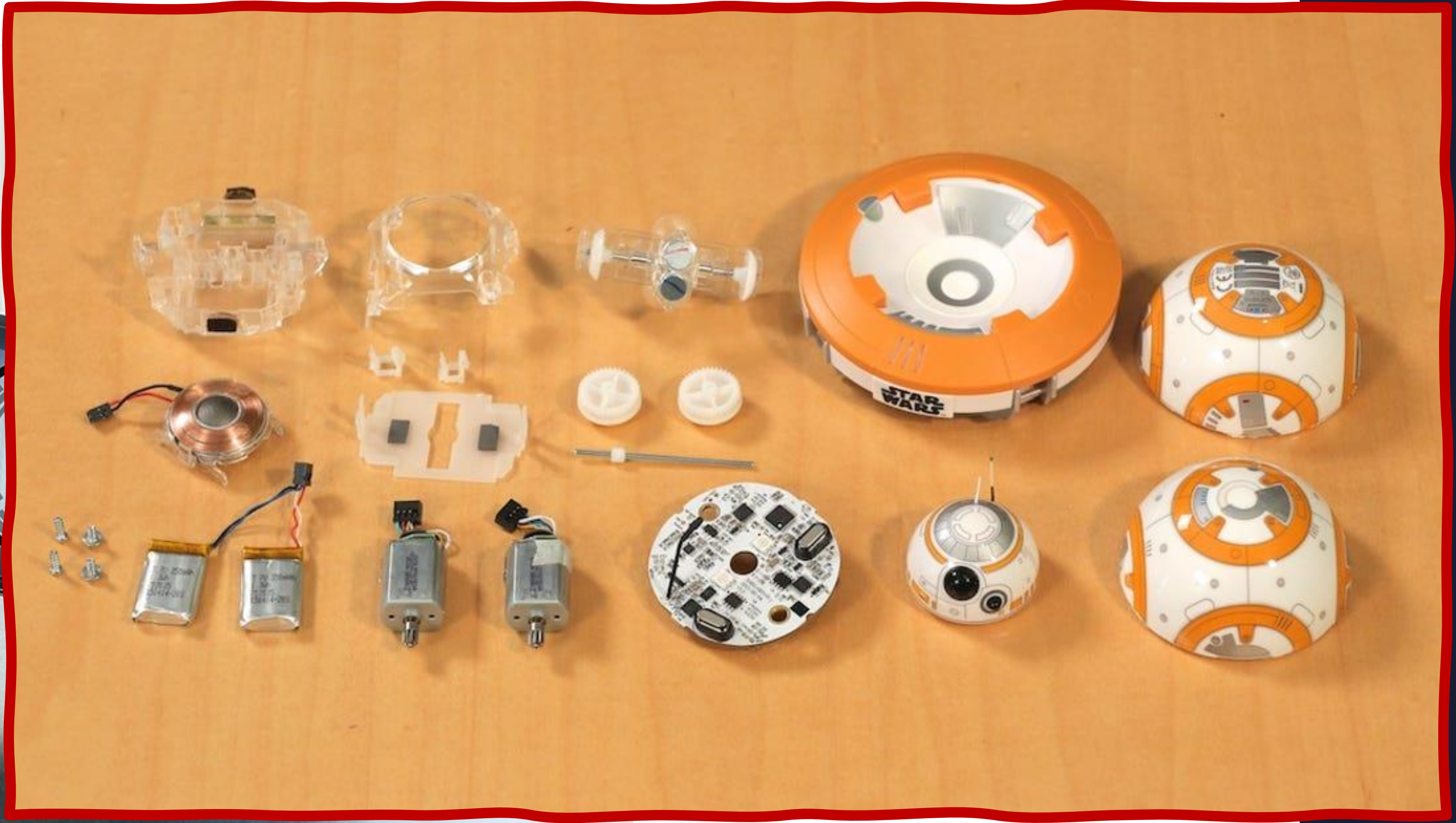
Examples

- **NASA's Twin Mars Rovers.**
  - ***Microprocessor***: Radiation Hardened 20MHz PowerPC From **IBM**
  - ***Commercial Real-time OS.***
  - Software and OS was developed during multi-year flight to Mars and downloaded using a radio link.

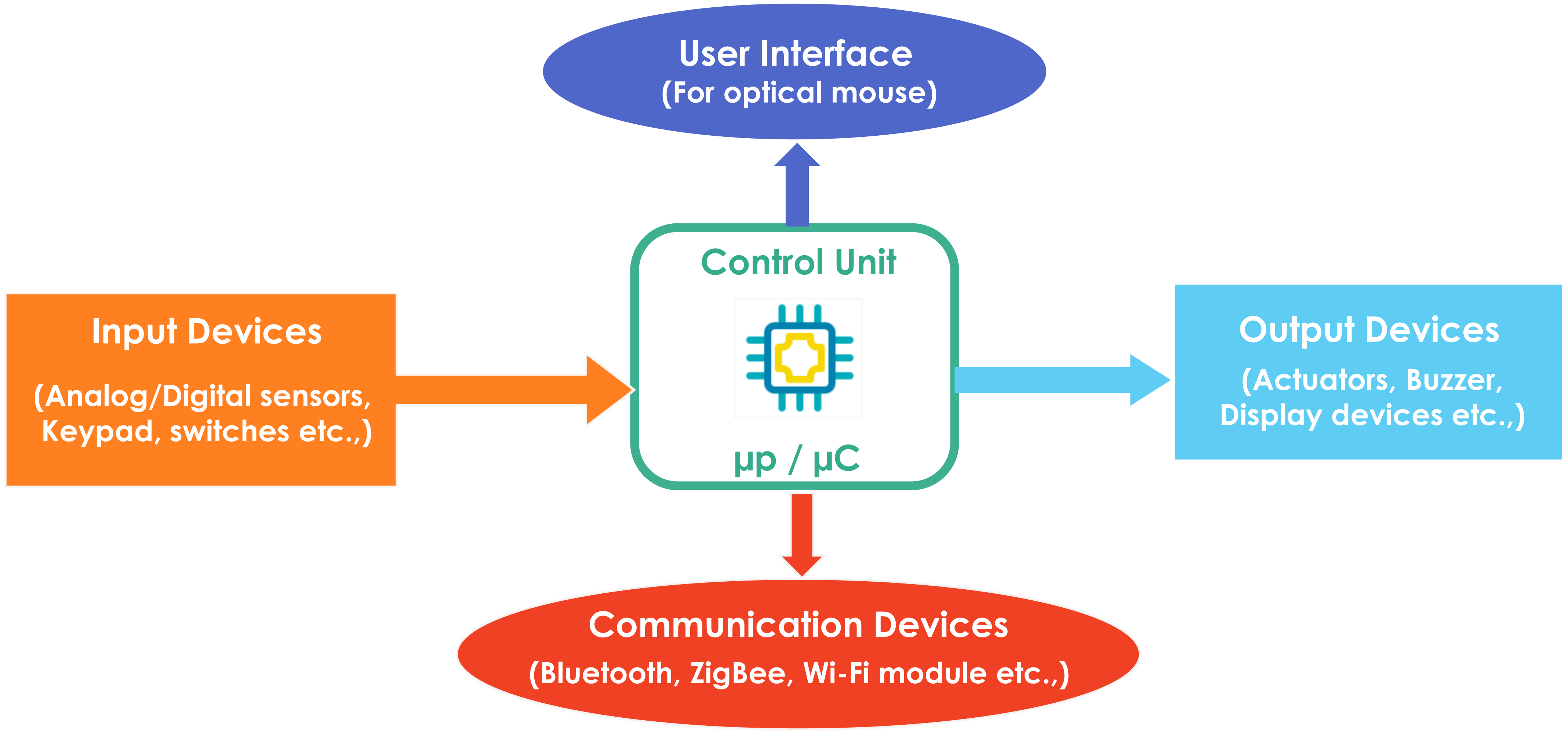


Examples

- Sphero



Block Diagram


User Interface
(For optical mouse)
Control Unit
µp / µC
Input Devices
(Analog/Digital sensors, Keypad, switches etc.,)
Output Devices
(Actuators, Buzzer, Display devices etc.,)
Communication Devices
(Bluetooth, ZigBee, Wi-Fi module etc.,)

Purpose of Embedded Systems

- *Each embedded system is designed to serve the purpose of any one or combinations of following task:*
  - Data collection / storage / representation
  - Data communication
  - Data (signal) processing
  - Monitoring
  - Controlling
  - Application specific user interface

Applications of Embedded Systems

- ❑ *Home Appliances*
- ❑ *Consumer Electronics*
- ❑ *Home Automation / Security*
- ❑ *Computer components*
- ❑ *Automotive*
- ❑ *Health care*
- ❑ *Military*
- ❑ *Industrial*
- ❑ *Business*
- ❑ *Retail*
- ❑ *Communications*
- ❑ *Avionics*
- ❑ *Banking*

Characteristics of Embedded Systems

- ❑ *Dedicated to* ***specific tasks****.*
- ❑ *Supported by a* ***wide array of processors and processor architectures.***
- ❑ *Usually* ***cost sensitive****.*
- ❑ ***Real-time constraints.***
- ❑ ***Require specialized tools*** *and methods*
- ❑ ***Software failure*** *is much more severe.*
- ❑ ***Power*** *&* ***memory constraints****.*
- ❑ *Operate under* ***extreme environmental conditions****.*
- ❑ ***Fewer system resources.***
- ❑ *Real Time Operating System*

Why Embedded Systems *are Special?*

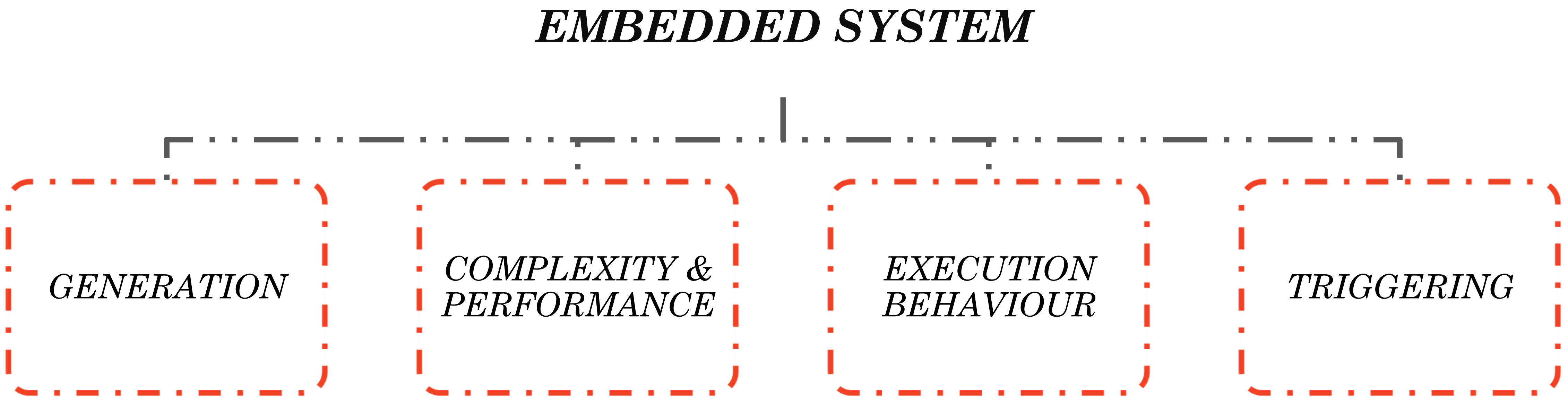
- ❑ *Complex algorithms* – *(Automotive Engine ECU)*
- ❑ *User interface* – *(GPS)*
- ❑ *Real time* – *(ABS)*
- ❑ *Multi-rate* – *(Multimedia Application)*
- ❑ *Manufacturing cost* – *(Mobile Phone)*
- ❑ *Power and energy* – *(Data Logger)*

Difference

S. NO.	**GENERAL PURPOSE COMPUTING SYSTEMS**	**EMBEDDED SYSTEM**
1	A ***combination of generic hardware and general-purpose operating system*** for **executing variety of applications**	Combination of **special purpose hardware and embedded OS** for executing a specific set of application
2	Contains General Purpose Operating System (GPOS)	***May or may not contain operating system*** for function.
3	***Applications*** can be ***modified*** by ***user***	***Firmware*** of the embedded system ***is pre-programmed*** and it is ***non-modifiable*** by the ***end user***
4	***Performance*** is a ***key deciding factor*** in the selection of the system. Always "***Faster is better***".	***Application specific requirements*** (like performance, power, memory, cost) are key deciding factor
5	***Need not*** to be ***deterministic*** in execution behavior	***Execution behavior*** is ***deterministic*** for certain type of system like "***hard real time systems***" (Missile guidance or airbag system)
6	Personal computer, Laptops	DVD player, Digital camera

Classification of **Embedded Systems**

Classification



Classification Based on *Generation*

First Generation:	Build around 4-bit & 8-bit processor like 8085, Z80
	Simple hardware & firmware developed in assembly code
	Ex: Digital telephone keypads, calculator etc.,
Second Generation:	Build around 8-bit and 16-bit processor
	Complex and powerful instruction set
	Ex: Data acquisition system, SCADA system etc.,

Classification Based on *Generation*

Third Generation:	Build around 16-bit and 32-bit processor
	Application specific processor DSP & ASIC were introduced
	Complex and powerful instruction set with pipelining
	Dedicated embedded real-time OS entered into market
	Ex: robotics, industrial process control, networking etc
Fourth Generation:	Advent of SOC, reconfigurable & multicore processors
	High performance, tight integration and miniaturization µP
	High performance RTOS for functioning
	Ex: smart phone, PDA etc.,

Classification Based on *Complexity & System Performance*

Small Scale:

Simple in application needs

Performance requirements are **not time critical**

Build around **low cost & low performance 8 or 16-bit µp /µC**

Ex: Electronic toys, digital multimeters etc.

Medium Scale:

Slightly complex in hardware & software requirements

Build on **medium performance & low cost 16/32-bit µp /µC**

Usually contains **embedded OS**

Ex: DVD players, network routers etc.,

Large Scale:

Highly complex hardware and software requirement

Employed in mission **critical application demanding high performance**

Build using **high performance 32 or 64-bit µp /µC or SOC or PLDs or multi-core processors**

May contain complex **high performance RTOS**

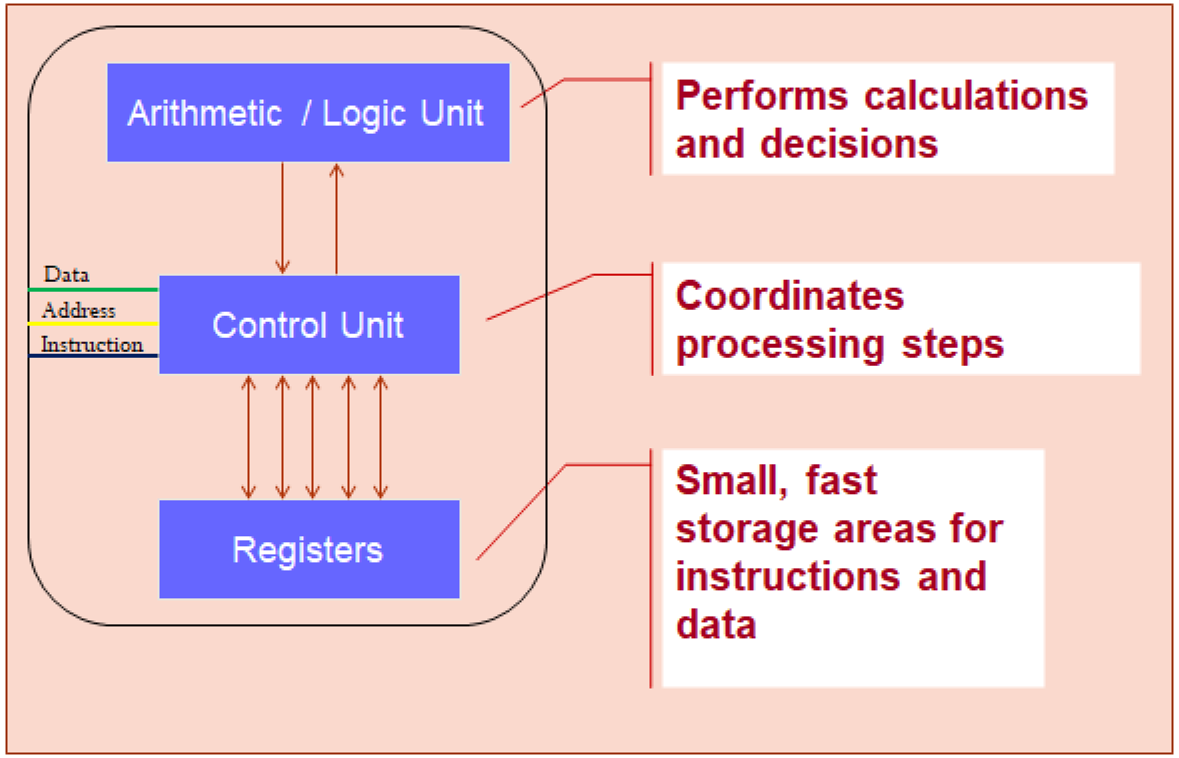
Classification Based on *Execution*

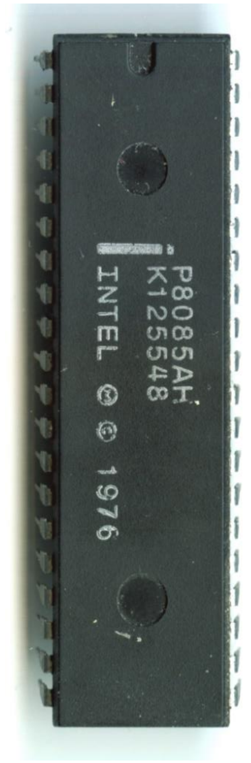
- ❑ **Deterministic:**
  - Real-time systems
  - Non Real-time systems
- ❑ **Triggering mechanism:**
  - Event triggered
  - Time triggered

Microprocessors & Microcontrollers

Microprocessors

- It is a central processing unit (CPU) on a single integrated circuit (IC) chip containing millions of very small components including transistors, resistors, and diodes that work together.


Arithmetic / Logic Unit
Performs calculations and decisions
Data
Address
Instruction
Control Unit
Coordinates processing steps
Registers
Small, fast storage areas for instructions and data



Examples: 8085, 8086, Intel *i* series, AMD, ARM, Qualcomm Snapdragon

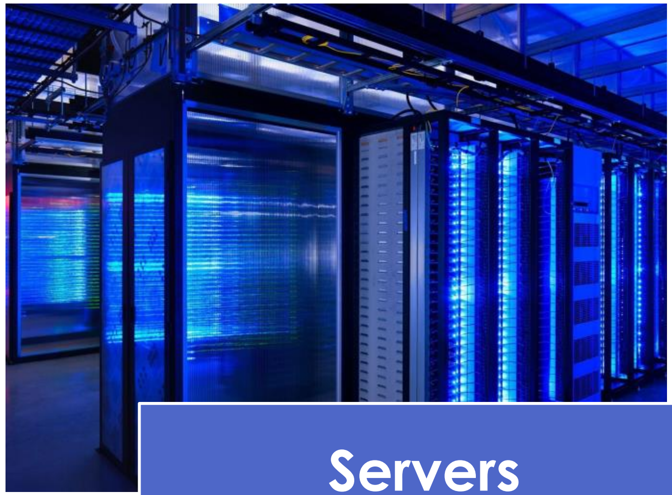
Microprocessors

- **1st Microprocessor:**
  - **1969:** The assignment - ***Nippon Calculating Machine Corporation*** approached **Intel** to design **12 custom chips** for its new Busicom 141-PF* **printing calculator**.
  - ***The Intel solution*** - Intel designed a ***set of four chips*** known as the ***MCS-4***. It *included CPU(4004), Shift registers(4003) RAM(4002), ROM(4001)*
  - ***1971: Era of integrated electronics*** - Intel purchased the rights from Nippon Calculating Machine Corporation and launched the ***Intel 4004 processor*** and its chipset with an advertisement in the November 15, 1971

Intel 4004

μP : Application

Laptop



Servers

Networking devices

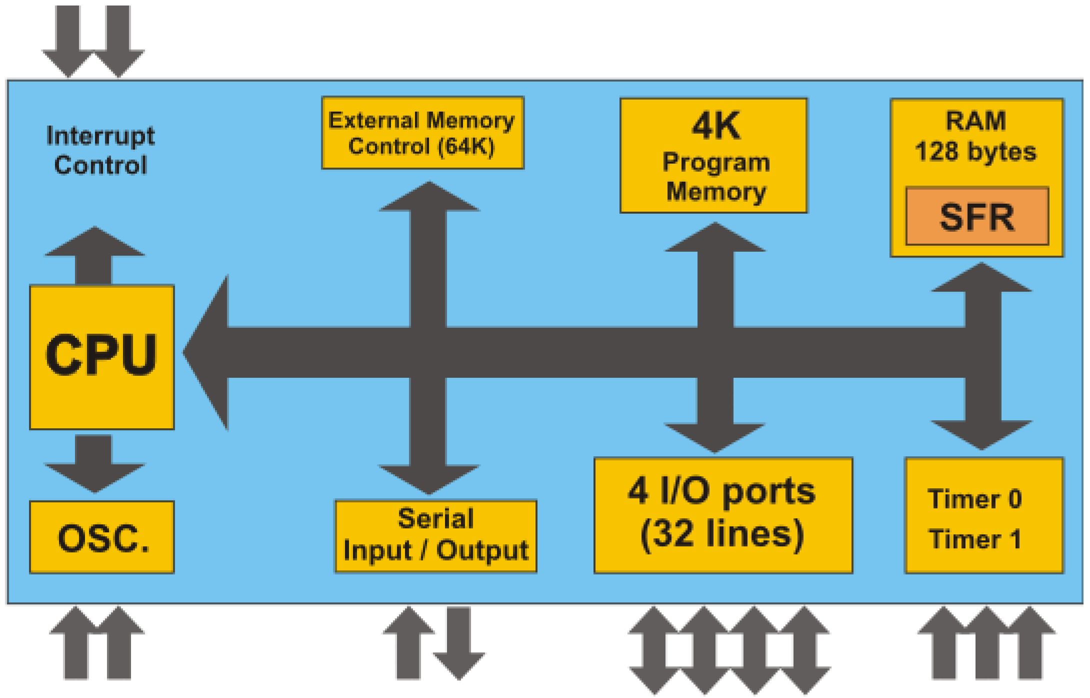
Mobile phones

Games machine

Military applications

Microcontroller

- ❑ μC is a compact IC designed to govern a specific operation in an embedded system.
- ❑ A typical μC includes a processor, memory and input/output (I/O) peripherals on a single chip.

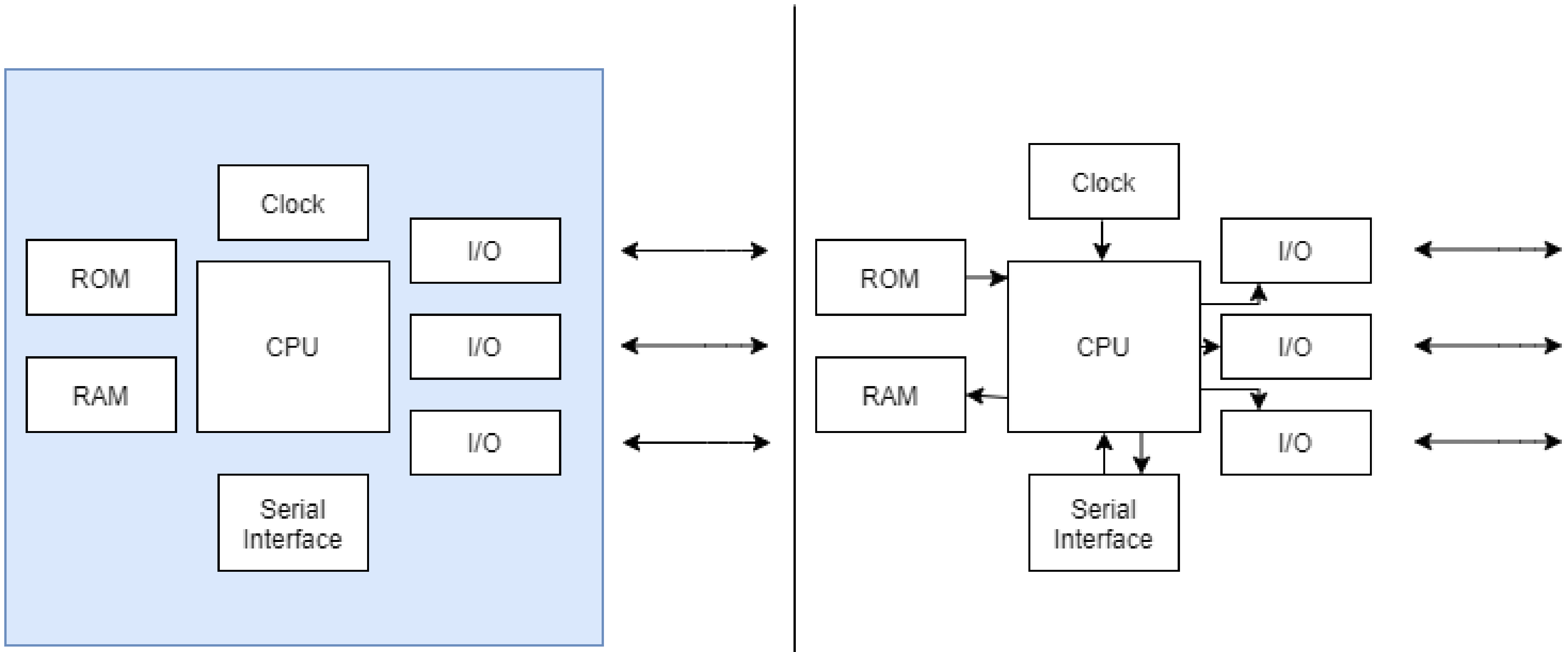


Examples: Intel 8051, PIC 16F877, AVR32, ARM LPC2148

Microcontroller

- ❑ **Commonly used μC**

μC v/s μP



Microcontroller - CPU and other pheripheral in a single chip

Micropocessor - CPU and several supporting chip

μC v/s μP

Feature	**Microcontroller (μC)**	**Microprocessor (μP)**
Purpose	Designed for specific **embedded system applications**	Designed for **general-purpose computing applications**
Architecture	**Single-chip computer system** with ***on-board memory, peripherals, and I/O interfaces***	**CPU** with ***minimal on-board memory, peripherals***, and ***I/O interfaces***
Integration level	Highly integrated	Less integrated
System architecture	**Single-chip system**	**CPU + support chips**
Processing power	Lower power	Higher power
Instruction set	Fixed instruction set	More flexible
On-board memory	On-chip memory	No on-board memory
(I/O)	More I/O ports	Fewer I/O ports

μC v/s μP

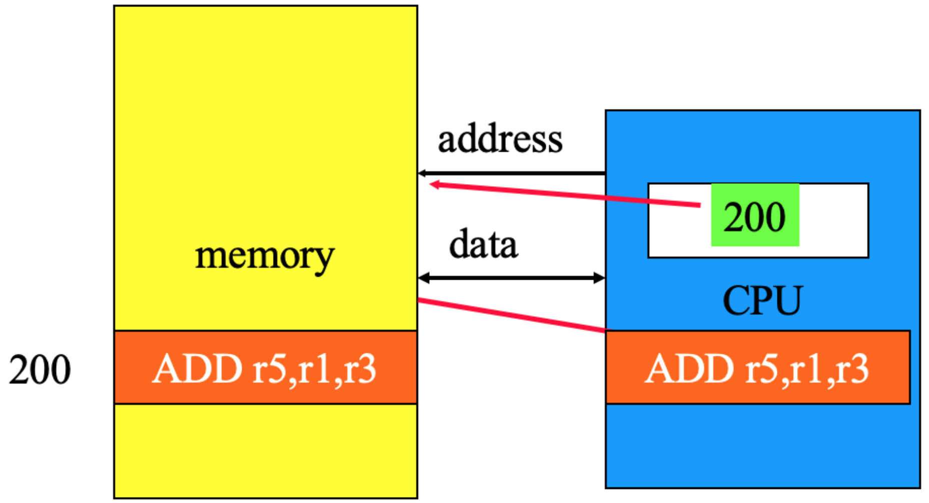
Feature	Microcontroller (μC)	Microprocessor (μP)
Peripheral devices	On-board peripherals	External peripherals
Cost	Lower cost	Higher cost
Power consumption	Lower power	Higher power
Applications	Embedded systems	General-purpose
Development	Integrated development environment (IDE) provided by manufacturers, with specialized programming languages and tools	Standard development tools and languages such as C, C++, and assembly
Clock speed	Lower clock speed, typically less than 100 MHz	Higher clock speed, typically greater than 1 GHz

Architectural Diff. in μC & μP

Van Neumann Architecture

- **Memory holds data, instructions.**
- **Central processing unit** (CPU) **fetches instructions from memory**.
- **Separate CPU** and **memory** distinguishes programmable computer.
- **CPU registers help out**: program counter (PC), instruction register (IR), general-purpose registers, etc.

Widely used in μP



Architectural Diff. in μC & μP

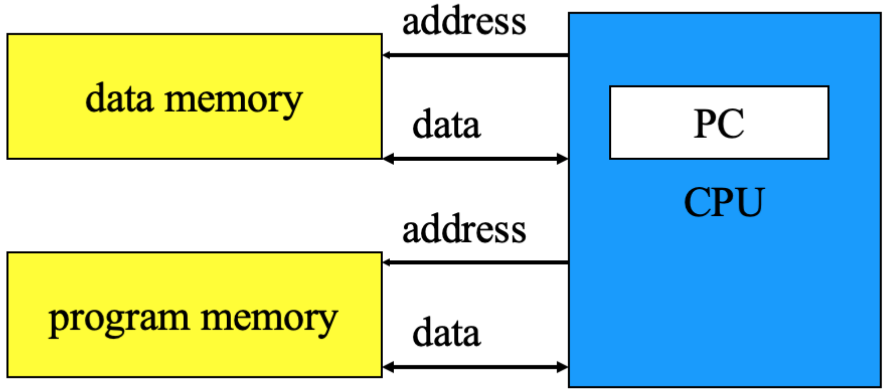
Harvard Architecture

Widely used in μC

- Architecture contains **separate storage and separate buses** (signal path) **for instruction and data**.

- It was basically developed to overcome the bottleneck of Von Neumann's Architecture.

- The **main advantage** of having ***separate buses*** for ***instruction*** and ***data*** is that the **CPU can access instructions** and **read/write data at the same time**.



Architectural Diff. in μC & μP

Van Neumann *vs.* Harvard Architecture

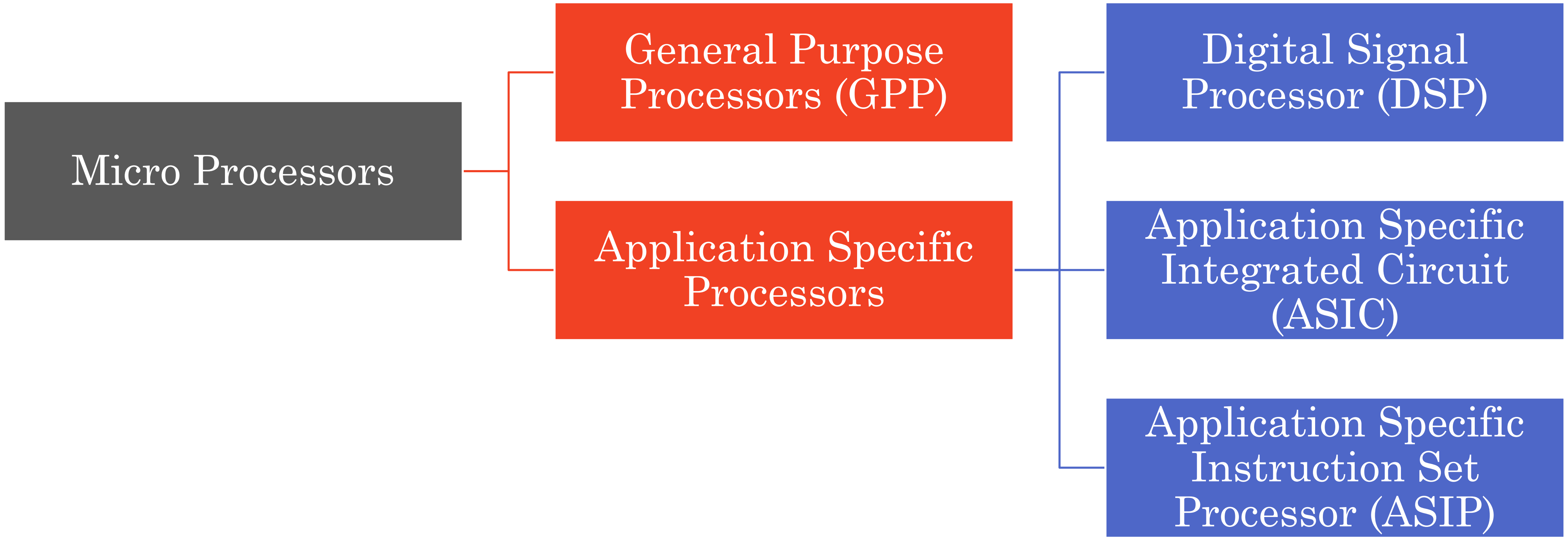
- ❑ Harvard can’t use self-modifying code.
- ❑ Harvard allows two simultaneous memory fetches.
- ❑ Most DSPs use Harvard architecture for streaming data:
  - ❑ Greater memory bandwidth;
  - ❑ More predictable bandwidth.

Why μC in Embedded Sys.

- Low cost
- Small size
- Easy to program
- Highly reliable
- Less design time
- Low power

Classification of **Microprocessors**

Classification of μP


Micro Processors
General Purpose Processors (GPP)
Application Specific Processors
Digital Signal Processor (DSP)
Application Specific Integrated Circuit (ASIC)
Application Specific Instruction Set Processor (ASIP)

Classification of μP

General Purpose Processors (GPP)

- ❑ Programmable device used in a variety of Applications
- ❑ GPP are designed to perform multiple tasks
- ❑ The system designer only needs to program the processor's memory to carry out the required functionality
- ❑ Biggest advantage of such system is the flexibility
- ❑ Lack high performance that a certain task required

- ❑ Advantages
  - ❑ Easy to design and use
  - ❑ Design time & cost is low
  - ❑ Reprogramability
- ❑ Disadvantages:
  - ❑ Performance is not very good
  - ❑ Size is high
  - ❑ Consume much power
- ❑ Intel " core *i*" is the most well-known example

Classification of µP

Application Specific Processors (ASP)

- ASPs emerged as a solution for **high performance** and **cost effective processors**.
- This processor is designed to **execute exactly one program**
- Contains only the **components needed to execute** a single program
- An embedded designer creates a single-purpose processor by designing a **custom digital circuit**

- Advantages
  - Performance is very good
  - Small size (exact to fit one solution)
  - They consume little power
- Disadvantages:
  - Not very easy to design
  - Design time is thus high
  - Design cost is thus higher
  - Reprogramming is difficult
  - Limited flexibility: not easy to make changes, accommodate features

Classification of μP

Application Specific Processors (ASP)

- Classification of ASPs

Digital Signal Processor (DSP):	Programmable microprocessor for extensive real time mathematical computations.
Application Specific Instruction Set Processors (ASIP):	Programmable microprocessor where hardware and instruction set are designed together for one special application.
Application Specific Integrated Circuit (ASIC):	Algorithm completely implemented in hardware.

Classification of µP (ASPs)
DIGITAL SIGNAL PROCESSORS (DSP)

- ❑ DSPs are specialized microprocessors optimized for the *need of* ***performing digital signal processing***.
- ❑ DSP gained their importance with the increased demand on data intensive applications such as video and internet browsing on mobile devices.
- ❑ DSPs satisfy the need for ***powerful processor*** while maintaining low cost and low power consumption

- ❑ **DSP Architecture Features:**
  - ❑ **Memory architecture** designed for ***streaming data***, using **DMA** extensively.
  - ❑ Deliberate **exclusion** of a **memory management unit** (MMU)
  - ❑ **Bit-reversed addressing**, a special addressing mode useful for calculating FFTs.
  - ❑ **Special arithmetic operations**, such as fast multiply–accumulates (MACs)
  - ❑ **Separate program and data memories** (Harvard architecture)

Classification of µP (ASPs)
Application-specific Instruction Set Processors (ASIP)

- Typically, a programmable architecture that is designed in a specific way to perform certain tasks more efficiently
- As the name suggests, the Instruction set seems to be the core characteristic of any ASIP based platform; but this is entirely not true.
- Considering a whole platform, other very important attributes like interfaces and micro-architecture do contribute a lot to the overall system performance.
- The term "Application" in ASIP is *not necessarily related to software applications*,
  - It actually describe the class of tasks the ASIP platform was designed to efficiently accomplish
  - This extra efficiency is not exclusively associated with faster performance.
- Other factors like reduced production costs, simplified manufacturing process and less power consumption can all be considered efficiency qualities for ASIP

Classification of µP (ASPs)
Application-specific Integrated Circuits (ASIC)

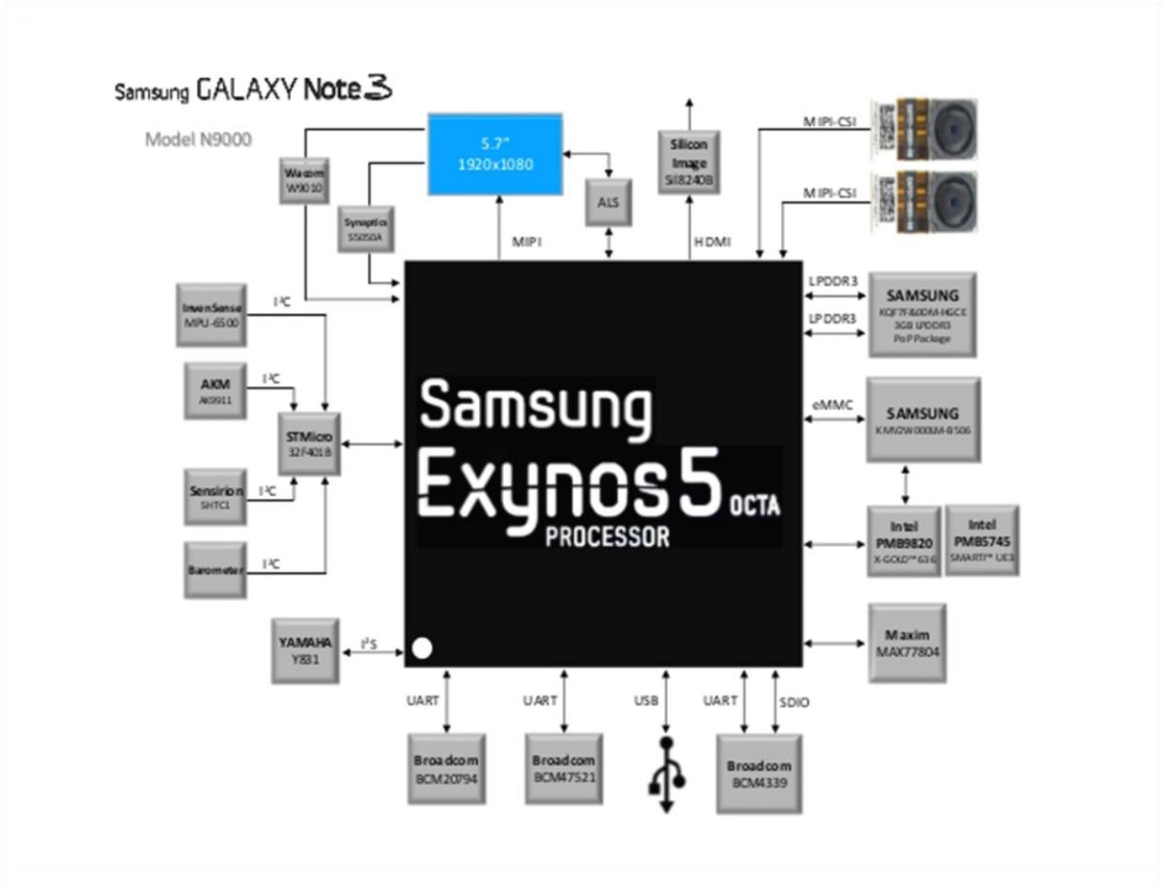
- ❑ An *IC* designed and used by a *single company* in a *specific system*.
- ❑ *Example*: An IC designed for a specific line of cellular phones of a company, whereby no other company can use it
- ❑ They are incredibly expensive, time-consuming, and resource-intensive to develop
- ❑ But extremely high performance coupled with low power consumption.

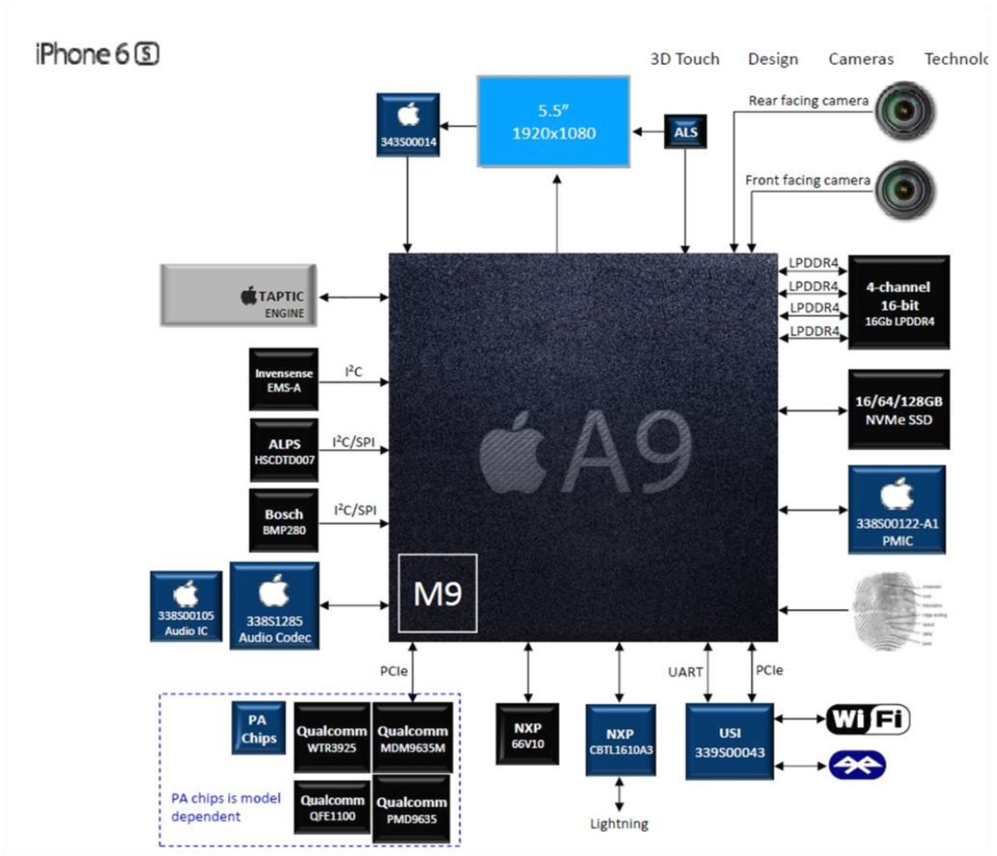
- ❑ **Full-custom:** entirely tailor-fitted to a particular application from the very start
- ❑ **Semi-custom:** designed to allow a certain degree of modification during the manufacturing process.
- ❑ **Structured:** built from a group of 'platform slices', with a 'platform slice' being defined as a pre-manufactured device, system, or logic for that platform.
- ❑ **Gate-array:** ASIC are transistors which are predefined on the silicon wafer.

Applications of ASIC:
- ❑ An IC that encodes and decodes digital data using a proprietary encoding/decoding algorithm.
- ❑ A medical IC designed to monitor a specific human biometric parameter.
- ❑ An IC that's custom-made for a particular automated test equipment.

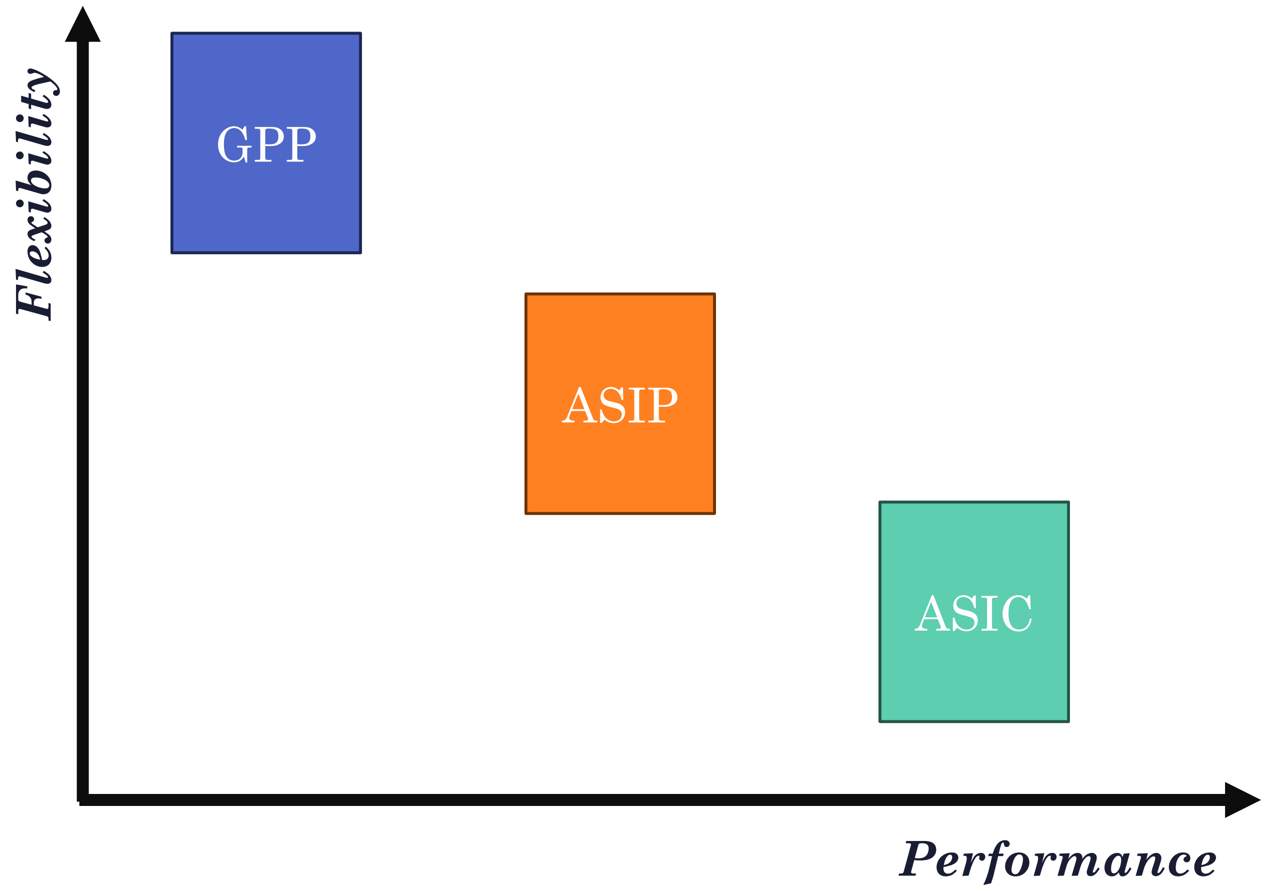
Classification of µP (ASPs)
APPLICATION-SPECIFIC INTEGRATED CIRCUITS (ASIC)

- ❑ **Example: ASIC**





Classification of μP

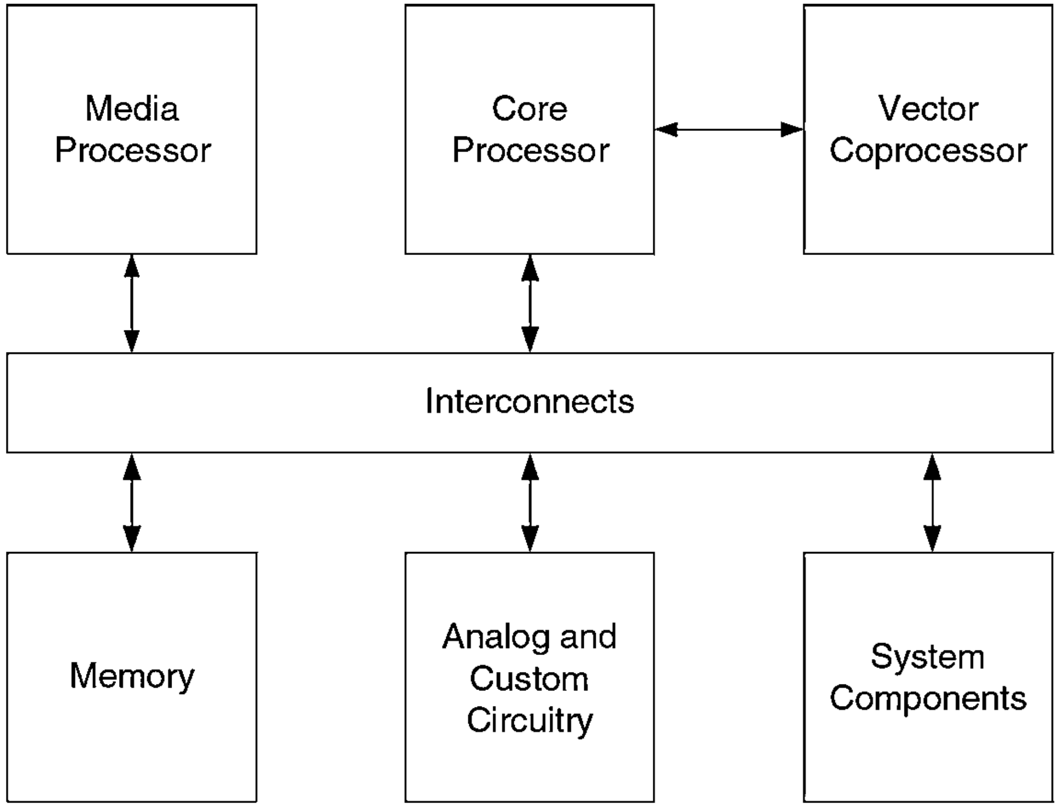


Classification of µP

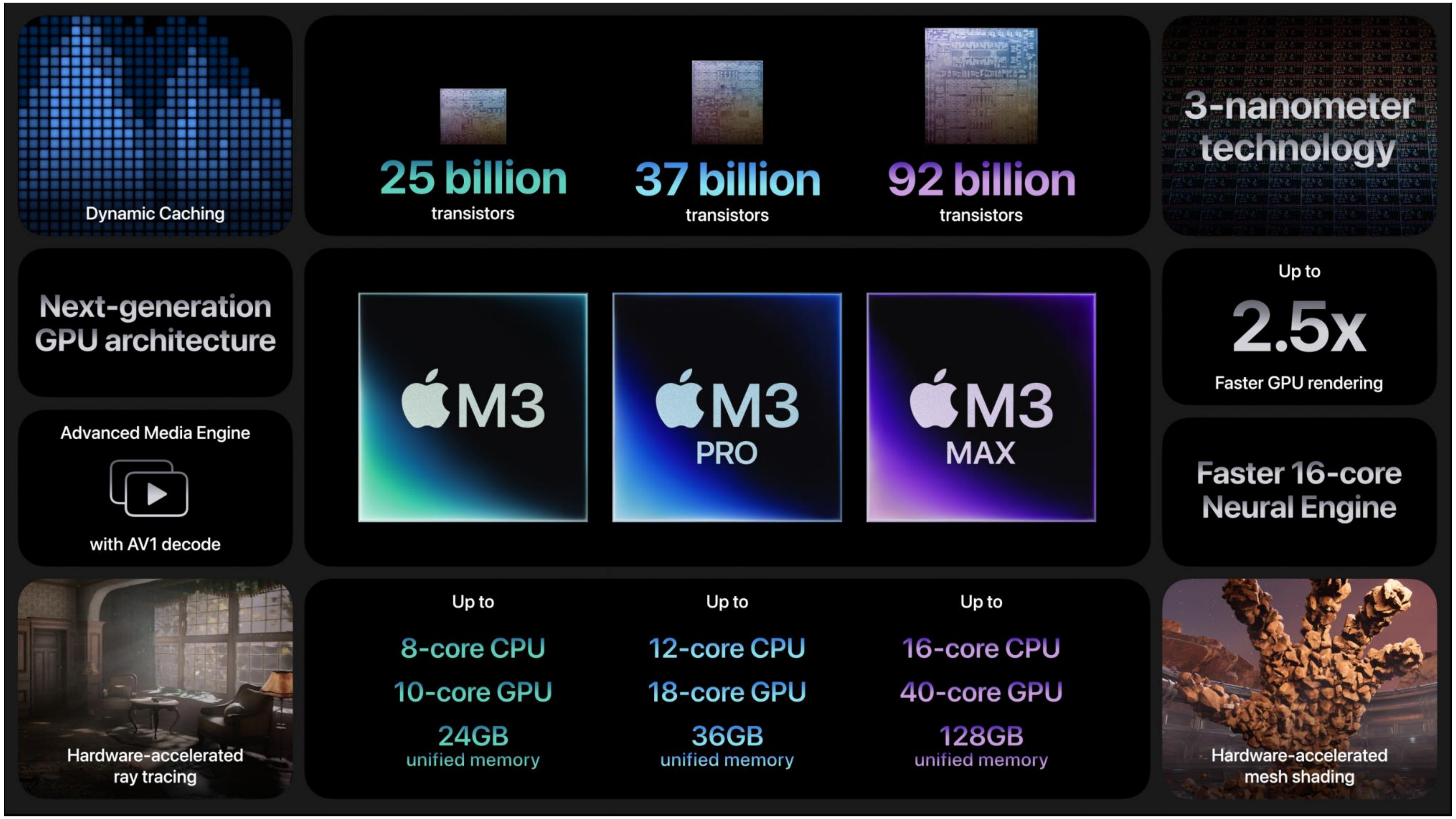
	GPP	ASIP	ASIC
Performance	Low	High	Very High
Flexibility	Excellent	Good	Poor
HW Design	None	Large	Very Large
SW Design	Small	Large	None
Power	Large	Medium	Small
Reuse	Excellent	Good	Poor
Market	Very Large	Relatively Large	Small
Cost	Mainly on SW	SOC	Volume sensitive

Classification of μP
System on Chip (SOC)

- ❑ An IC that takes a single platform and integrates an entire electronic system onto it.
- ❑ SoC contains:
  - ❑ One or more processor cores — *μP* and/or *μC* and/or DSPs
  - ❑ Along with on-chip memory, hardware accelerator functions, peripheral functions etc.,
- ❑ SoC can perform a variety of functions including signal processing, wireless communication, artificial intelligence etc.



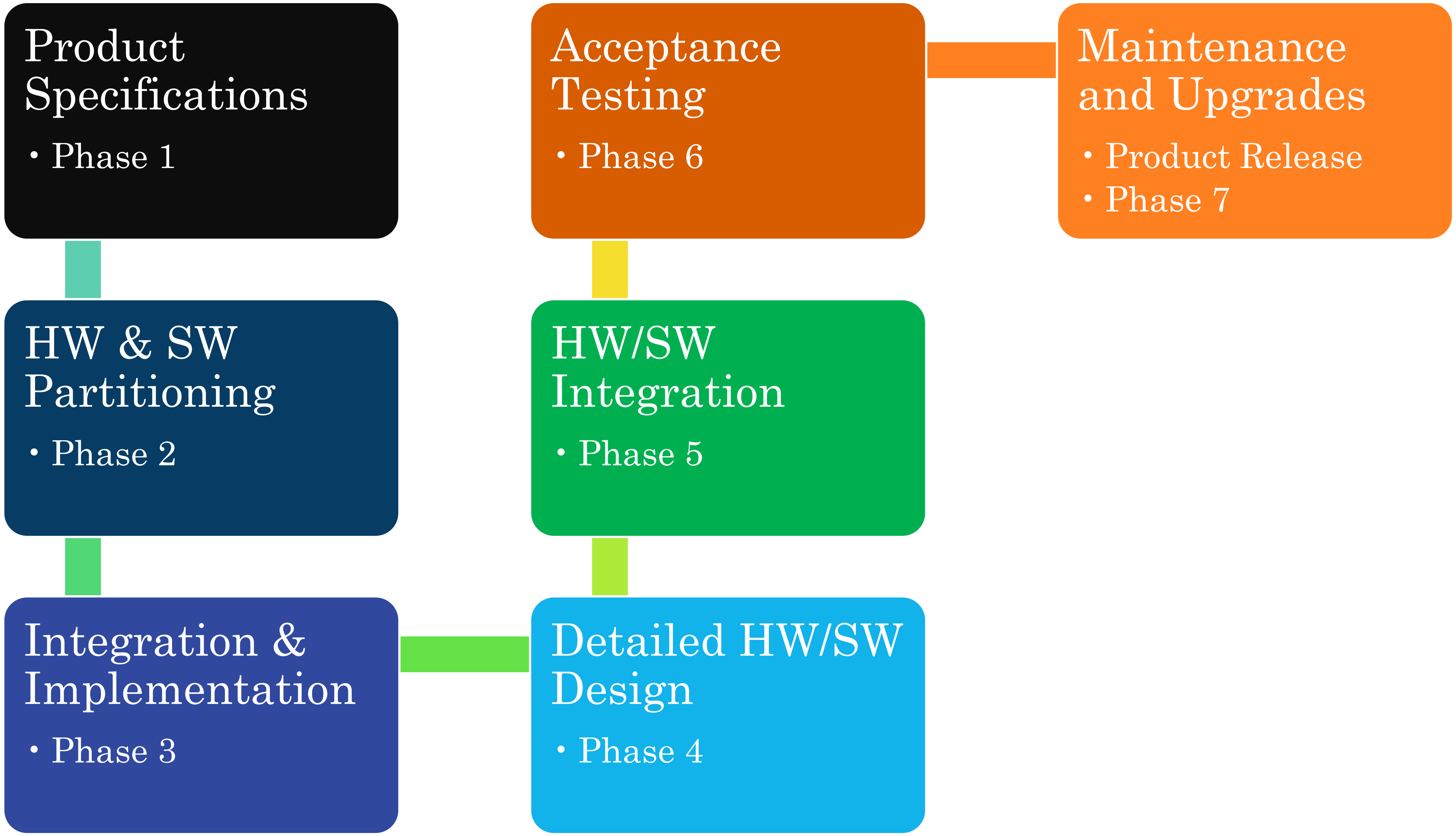
Classification of μP
System on Chip (SOC)



Embedded Systems

Design Lifecycle

ES : Design Lifecycle



ES : Design Lifecycle

Phase 1: Product Specifications

- **Objective**: Capture a ***formal description*** of the ***complete system***
  - Transforming the customer's wishes into the final product
  - Costumer Research Tour
  - Sales and Marketing Department take the lead.
  - **R&D are not allowed**(in general)
  - Help to arrive a common Vision of the Product

Phase 2: HW & SW Partitioning

- **Objective**: To decide which ***Portion in H/W and which in S/W***.
- ***Example***, FPU (H/W) faster, Floating Point (S/W) slower.
- Partitioning decision is a complex optimization problem
- Choice of CPU impacts the partitioning decision.

ES : Design Lifecycle

Phase 3: Iteration & Implementation

- **Objective**: Perform ***initial design*** ***before*** the ***detailed implementation***
- Final stage before H/W and S/W team stop communication
- Hardware designers might be using simulation tools
- Software designers running code on single-board computers that use the target micro processor
- Design can be extended in H/W or S/W according to convenience.

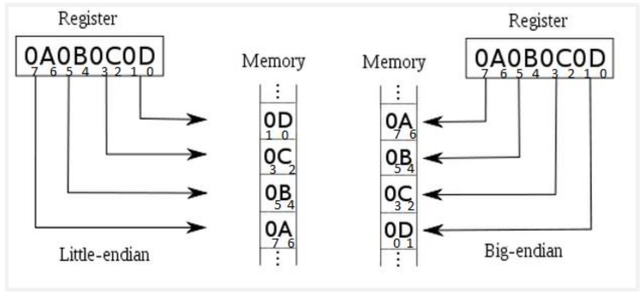
Phase 4: Detailed HW & SW Design

- **Objective**: Getting the hardware and software done for the next phase
- ***S/W embedded developer*** must understand about *execution environment, development tools, and run-time package*.
- ***H/W embedded systems developer*** must understand about,
  - *Memory usage.*
  - *System startup*
  - *Interrupts and exceptions*

ES : Design Lifecycle

Phase 5: HW & SW Iteration

- ❑ Objective: To ***integrate both H/W and S/W to carryout testing***
- ❑ This stage require ***special tools*** and methods to ***manage the complexity***.
- ❑ ***Helps*** perform ***debugging*** and ***identify errors***, Also discover whether S/W team understood the H/W description.
- ❑ *E.g* Big Endian/Little Endian problem



Phase 6: Product Testing & Release

- ❑ Objective: Verify designed system meet its requirement or not also ensure software doesn't crash at a critical moment
- ❑ Testing & reliability requirements- more stringent in embedded
- ❑ Testing must determine whether the system is performing close to its optimal capabilities
- ❑ Who does the testing? Not the designer

ES : Design Lifecycle

Phase 7: **Maintaining & Upgrading Existing Products**

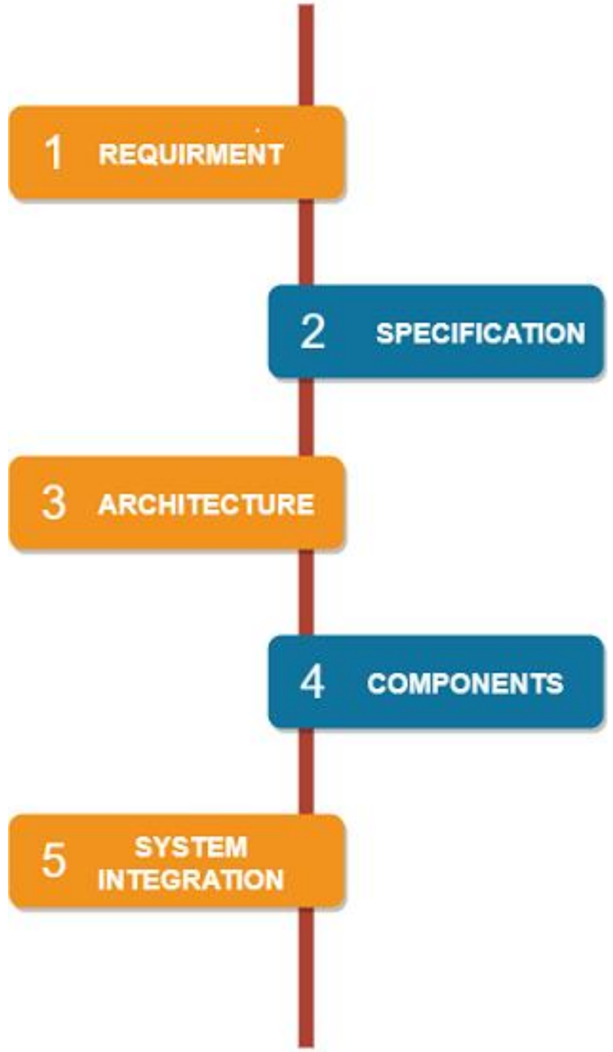
- ❑ **Objective**: To ***maintain and add additional features*** on the existing product design
- ❑ Carried out by non-members of the original design team for a particular product
- ❑ Experience, skills, the existing documentation, and the old product to understand the original design well enough to maintain and improve it.

Embedded Systems

Design Process

ES : Design Process

Top–down - From description of the system to concrete details.



Bottom–up
Start with components to build a system.

Ref. "Computers as components: principles of embedded computing system design" by Wayne Wolf

ES : Design Process

- **Major goals** of the design to be considered are :
  - Manufacturing cost
  - Performance
  - Power consumption
- **Tasks which need to be performed at each step,**
  - Analyze the design at each step - meet specification
  - Refine the design - add details
  - Verify the design - system goals

ES : Design Process

- **Step 1: Requirements**
  - ***Informal descriptions*** gathered from the ***customer*** are known as ***requirements***
  - Requirements are refined into a ***specification*** to ***begin*** the ***designing*** of the ***system architecture***
  - **Classified to 2 categories**
    - ***Functional***: Needed **output as a function** of input
    - ***Non-functional***: **performance, cost, physical size, weight, and power consumption**

ES : Design Process

- ❑ **Step 1: Requirements :** *Example*
- ❑ ***GPS Navigation System***

Product Name	**GPS Navigation System**
Purpose	Consumer grade moving map for driving use
Inputs	Power button; two control button
Outputs	Back-lit LCD Display 400x600
Functions	Uses 5-receiver GPS system; three user-selectable resolutions; always displays current latitude and longitude
Performance	Updates screen within 0.25 seconds upon movement
Power	100mW
Cost	30 USD
Size & Weight	2” x 6” and 250 grams

ES : Design Process

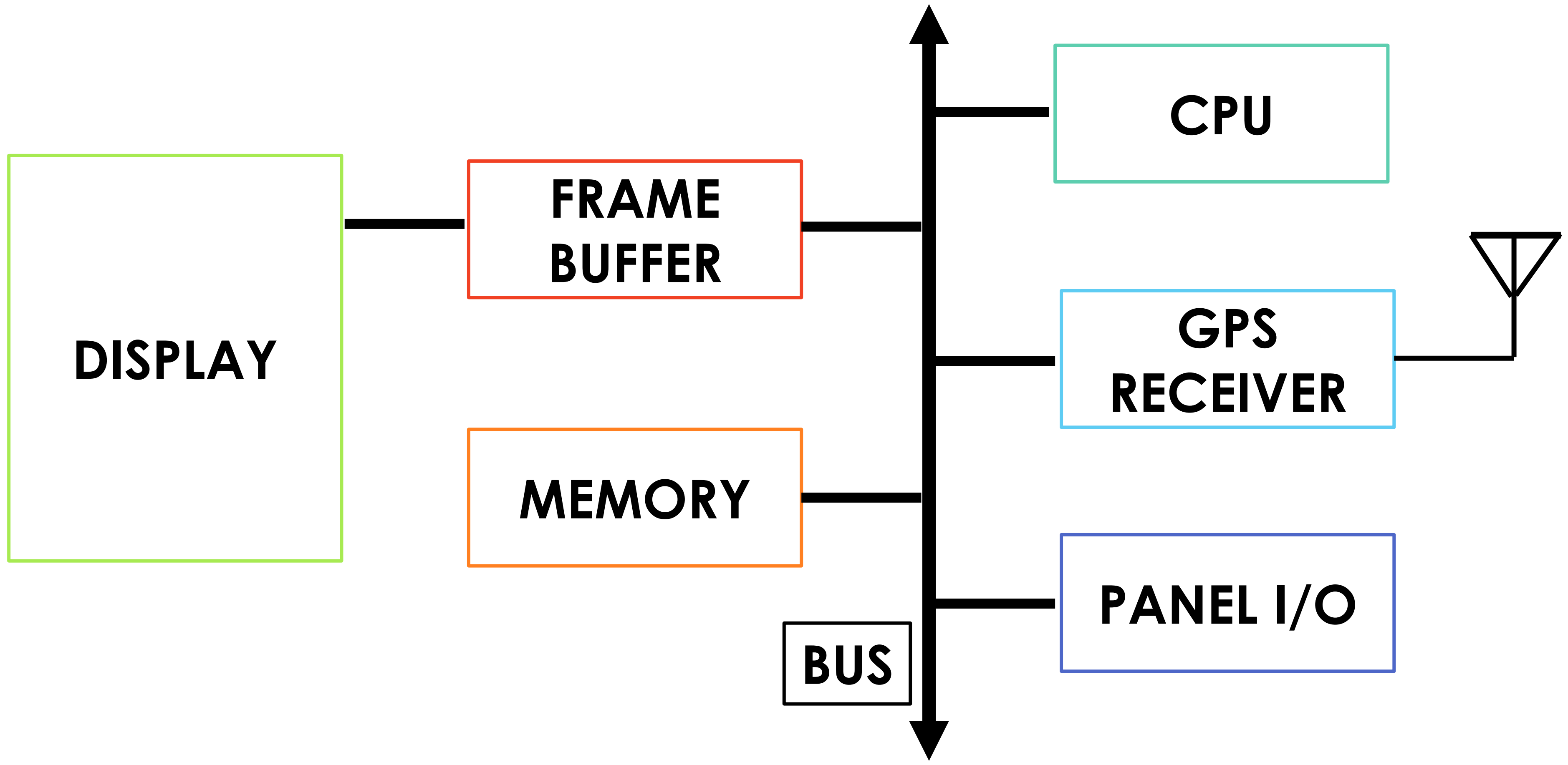
- **Step 2: Specifications**
  - Requirements gathered is ***refined*** into a ***specification***.
  - Specification serves as the *contract between the customers* and the *architects*.
  - Are essential to *create working systems* with a *minimum of designer effort*.
  - *Specific*, *understandable* and *accurately* reflect the *customer's requirements*.
  - ***Specification would include details for several components:***
    - Data received from the GPS satellite
    - Map data
    - User interface
    - Operations that must be performed to satisfy customer requests
    - Background actions

ES : Design Process

- **Step 3: Architecture**
  - ***Specification*** describes only the *functions of the system*
  - **Implementation of the system** is described by the **Architecture**
    - **Architecture** is a plan for the **overall structure of the system**.
    - It will be used later to design the components
    - **System block diagram** may be refined into **two block diagrams** –
      - **Hardware & Software**

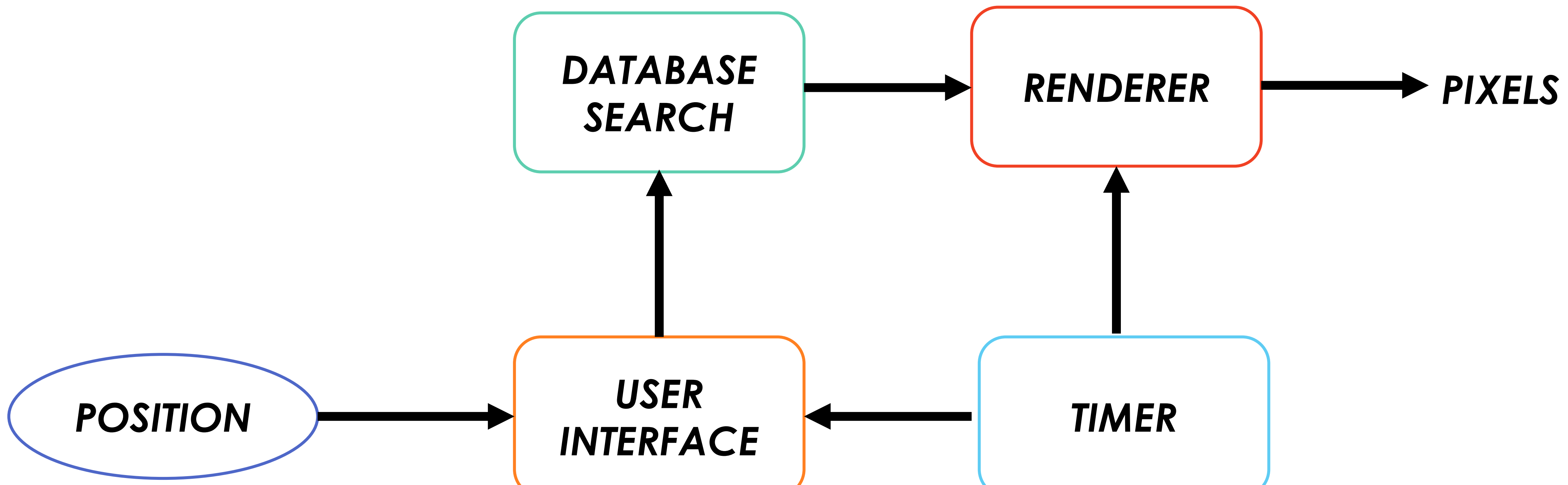
ES : Design Process

- **Step 3: Architecture :** ***System Design : Hardware***



ES : Design Process

- **Step 3: Architecture :** ***System Design : Software***



ES : Design Process

- **Step 4: Hardware & Software Components**
  - **Architectural** description **specify** what **components we need**
    - *The component design effort builds those components in conformance to the architecture and specification*
  - ***Includes both hardware*** and ***software modules***
  - Some of the components will be ready-made (example :CPU, memory chips)

ES : Design Process

- **Step 4: Hardware & Software Components**
  - In example ***GPS Receiver***
  - In the moving map, *GPS receiver* is a ***predesigned*** standard hardware component.
  - ***Topographic software*** is a ***standard software module*** which uses standard routines to access the database.
  - **Printed circuit board** are the components which **needs to be designed**.
  - When **creating** these **embedded software modules**, ***ensure the system runs properly in real time*** and that it ***does not take up more memory space than allowed***.

ES : Design Process

- **Step 5: System Integration**
  - After the components are built, they are integrated
  - ***Bugs*** are typically ***found*** during the ***system integration***
  - By debugging a few modules at a time, simple bugs can be discovered
  - By fixing the simple bugs early, more complex bugs can be discovered
  - ***Appropriate debugging facilities*** during design which ***can help to ease system integration problems***.

ES Design Process

Example

ES : Design Process Example: *Automatic Chocolate Vending Machine*

- **Requirements**
  - ***Purpose:***
    - To sell chocolate through an ACVM from which children can automatically purchase the chocolates.
    - The payment is by inserting the coins into a coin-slot.
  - ***Inputs:***
    - Coins of different denominations through a coin slot.
    - User commands.
  - ***Outputs:***
    - Chocolate and signal to the system that subtracts the cost from the value of amount collected.
    - Display of the menus for GUIs, time and date, advertisements, welcome and thank messages.

ES : Design Process Example:
Automatic Chocolate Vending Machine

- **Requirements**
  - A child sends commands to the system using a GUI.
    - It must consists of the LCD display and keypad units.
  - The child inserts the coins for cost of chocolate and the machine delivers the chocolate.
  - If the coins are not inserted as per the cost of chocolate in reasonable times then all coins are refunded.
  - If the coin amount more, the excess amount is refunded along with chocolate.
  - The coins for the chocolates purchased collect inside the machine in a collector channel, so that owner can get the money, again through appropriate commands using the GUI.
  - USB wireless modem enables communication through Internet to the ACVM owner.

ES : Design Process Example:
Automatic Chocolate Vending Machine

- ❑ **Design Metrics**
  - ❑ ***Power Dissipation:*** As required by mechanical units, display units and computer system
  - ❑ ***Performance:*** One chocolate in two minutes and 256 chocolates before next filling of chocolates into the machine.[Assumed]
  - ❑ ***Process Deadlines:*** Machine waits for maximum 30 s for the coins and machine should deliver the chocolate within 60 s.
  - ❑ ***User Interfaces:*** Graphic at LCD or touch screen display on LCD and commands by children or machine owner through fingers on keypad or touch screen
  - ❑ ***Engineering Cost:*** US$ 50000 (assumed)
  - ❑ ***Manufacturing Cost:*** US$ 1000 (assumed)

ES : Design Process Example:
Automatic Chocolate Vending Machine

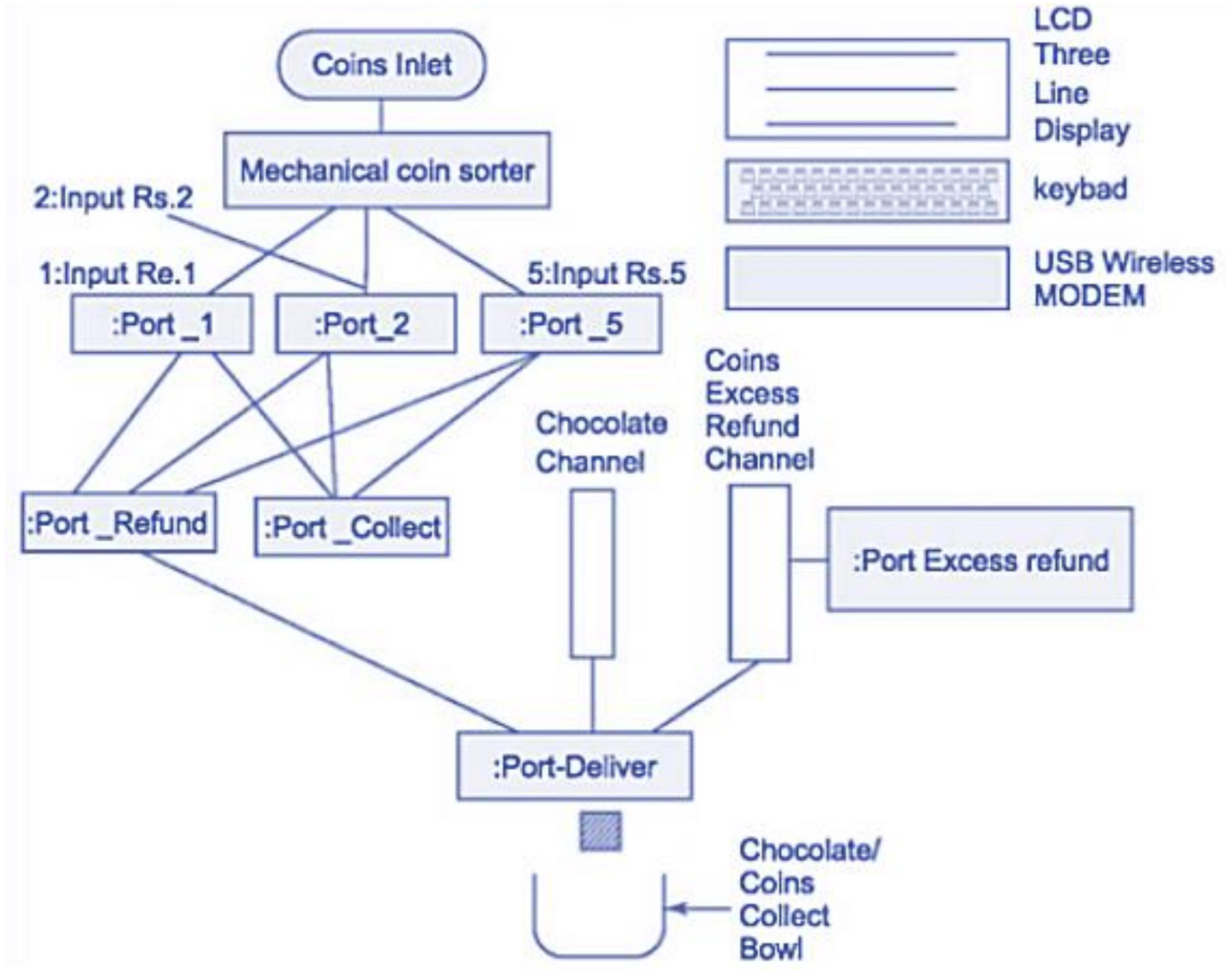
- ❑ **Specifications**
  - ❑ **Alphanumeric keypad on the top of the machine.**
    - ❑ A child interaction with it when buying a chocolate. Owner commands and interaction with the machine.
  - ❑ **Three line LCD display unit on the top of the machine.**
    - ❑ Displays menus, entered text, pictograms, and welcome, thank and other messages, and time and date.
    - ❑ Child as well as the ACVM owner GUIs with the machine using keypad and display

ES : Design Process Example:
Automatic Chocolate Vending Machine

- **Specifications**
  - **Coin insertion slot:**
    - So that the child can insert the coins to buy a chocolate.
  - **Delivery slot**
    - To collect the chocolate, and coins if refunded.
  - **Internet connection port**
    - So that owner can interact with ACVM from remote location.

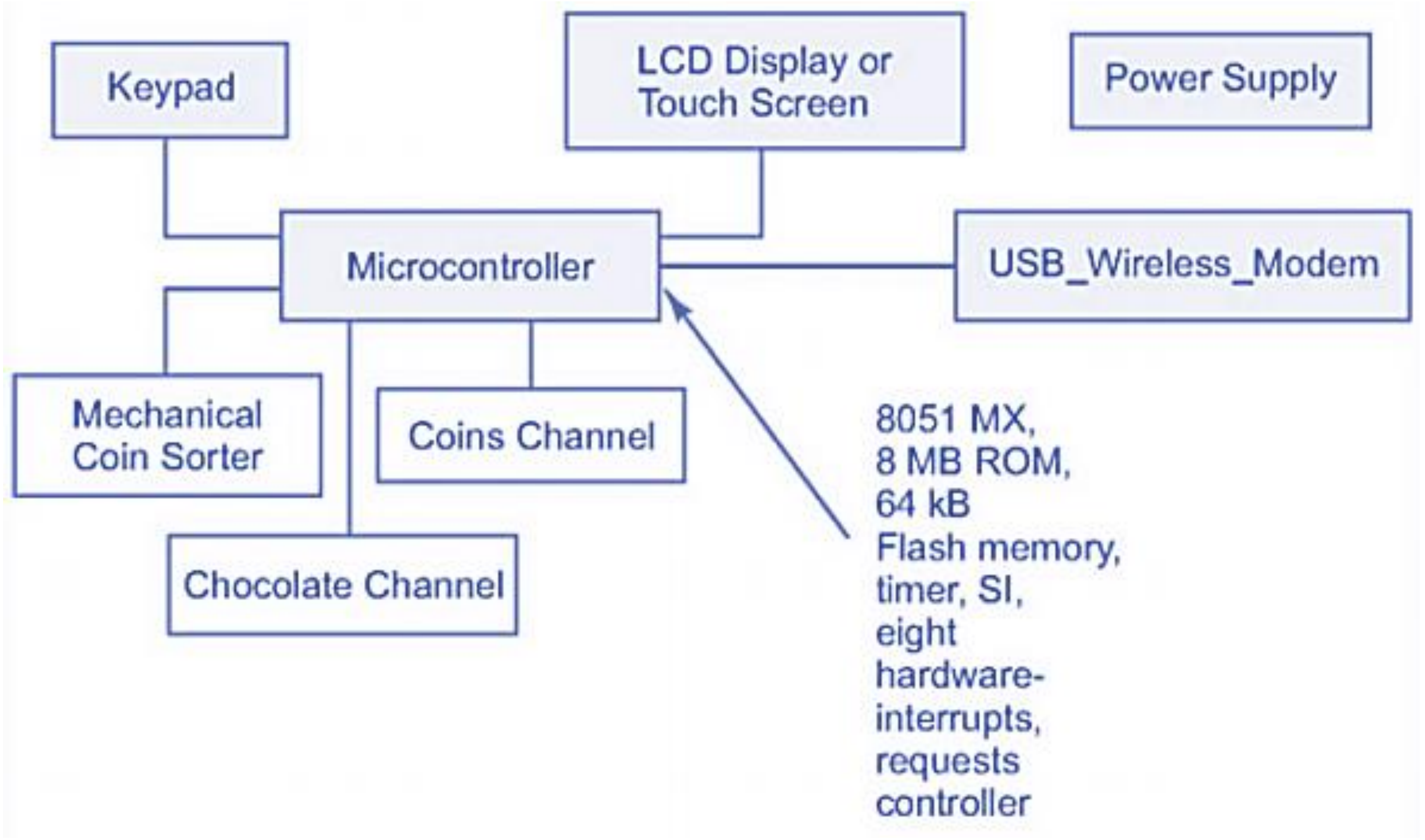
ES : Design Process Example:
Automatic Chocolate Vending Machine

- **Specifications : Overall System Architecture**



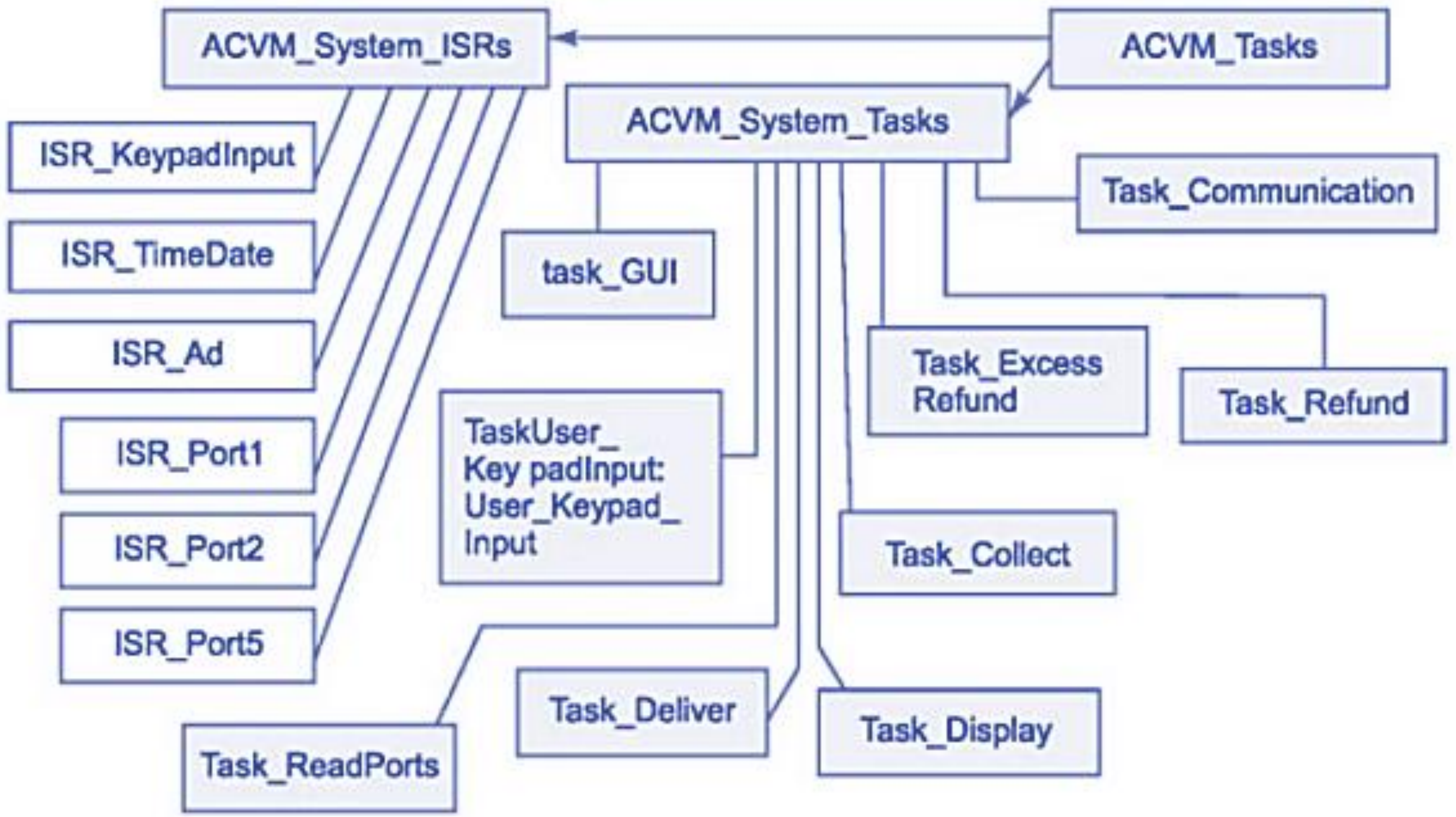
ES : Design Process Example:
Automatic Chocolate Vending Machine

- **Specifications : Hardware Design**


Keypad
LCD Display or Touch Screen
Power Supply
Microcontroller
USB_Wireless_Modem
Mechanical Coin Sorter
Coins Channel
Chocolate Channel
8051 MX, 8 MB ROM, 64 kB Flash memory, timer, SI, eight hardware-interrupts, requests controller

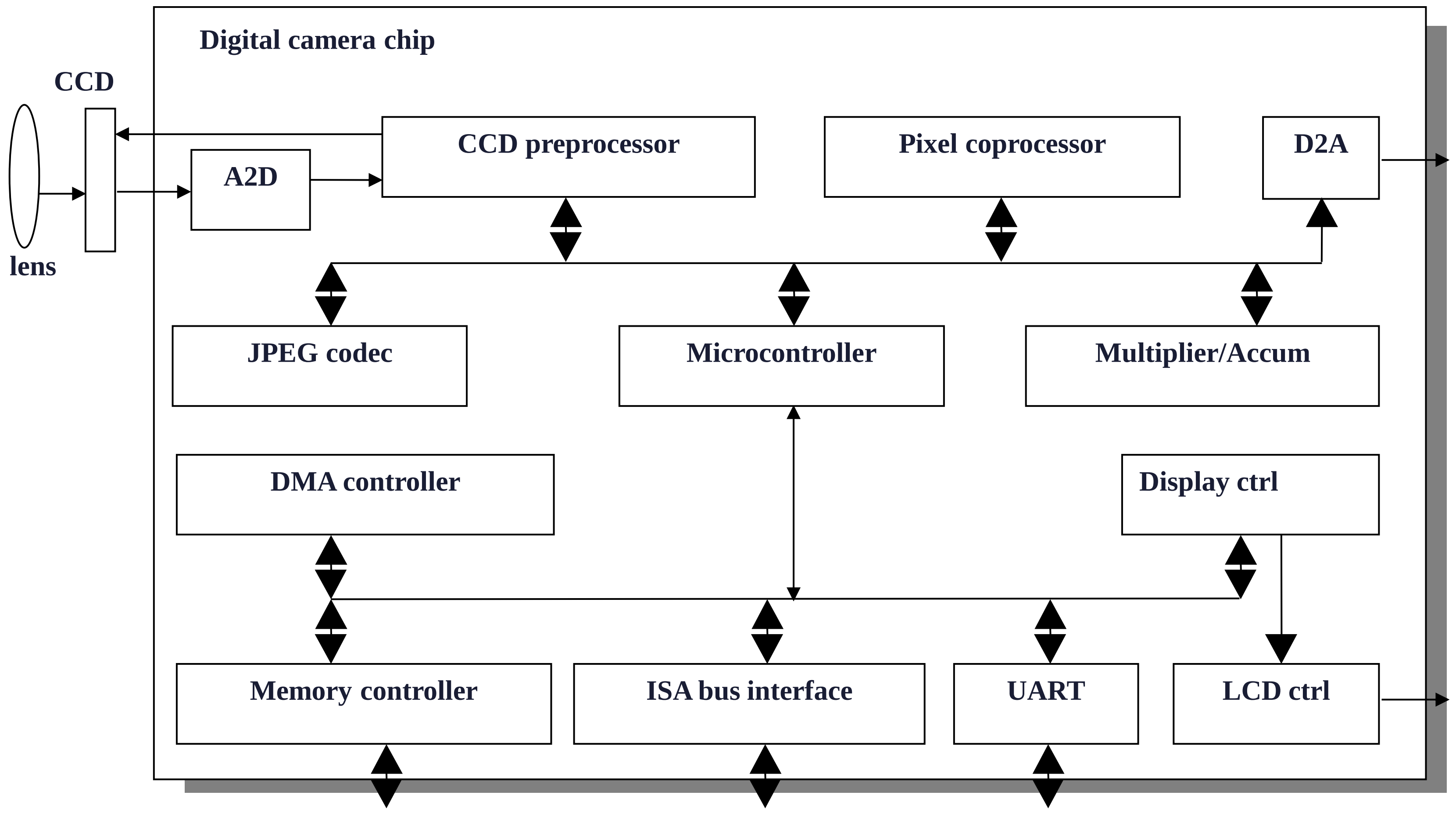
ES : Design Process Example:
Automatic Chocolate Vending Machine

- ❑ **Specifications : Software Design**

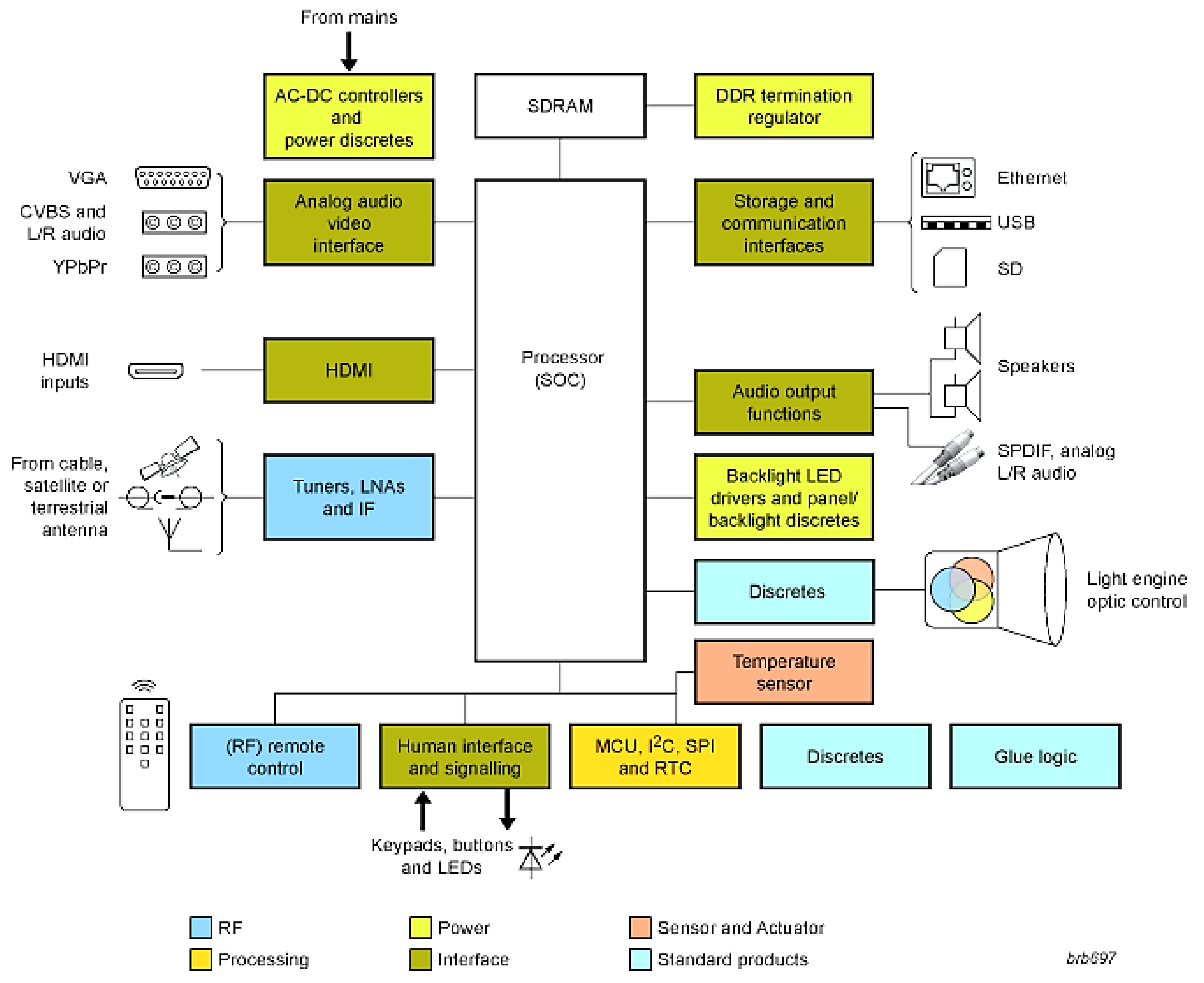


ES : Design Process Example: *Digital Camera*

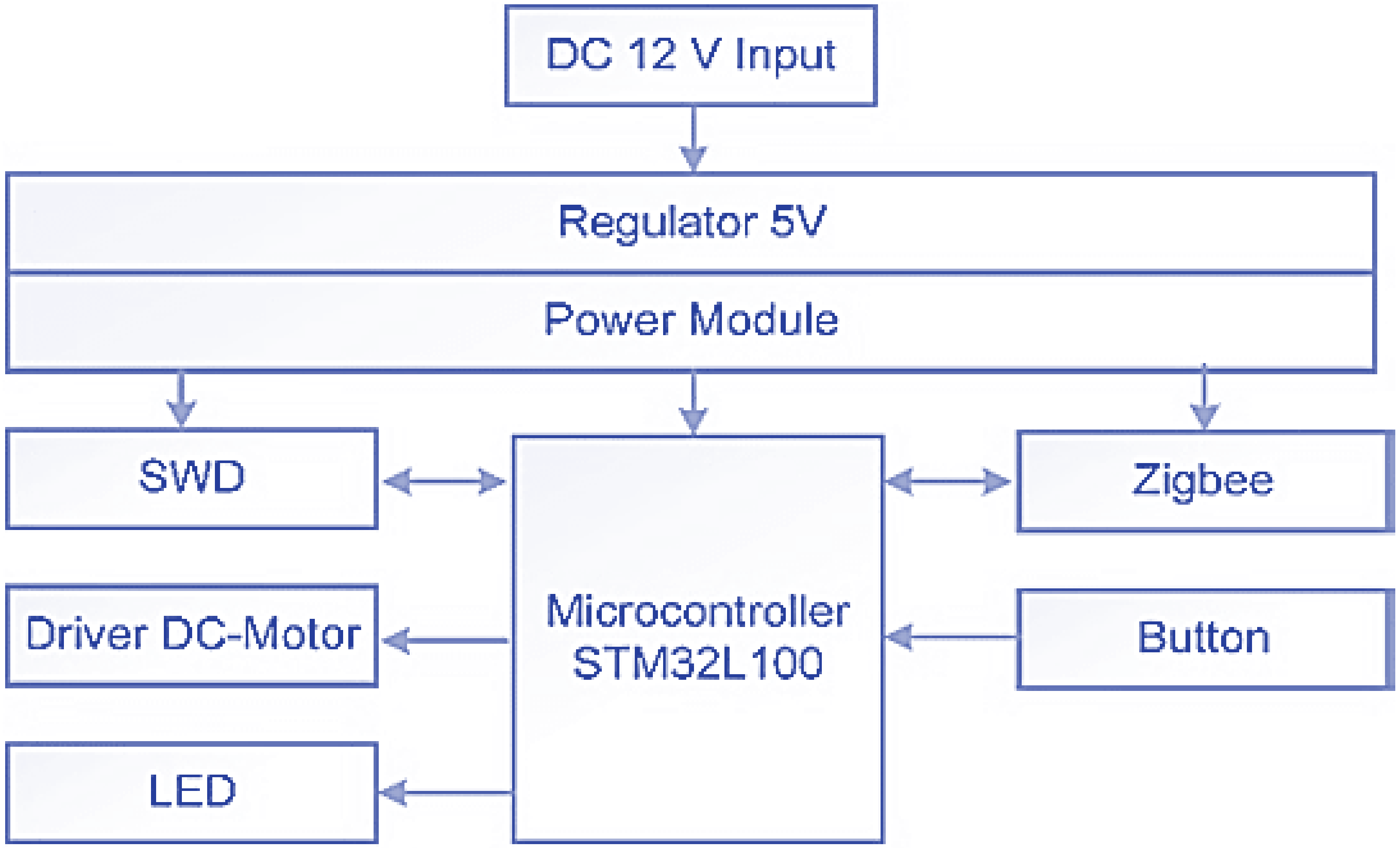
- *Single-functioned* -- always a digital camera
- *Tightly-constrained* -- Low cost, low power, small, fast
- *Reactive and real-time* -- only to a small extent



ES : Design Process Example:
Projector



ES : Design Process Example:
Digital Fan



Introduction to
Intel MCS 51 μC

Intel MCS 51 µC

- **Intel MCS-51** (commonly termed **8051**) is a
  - ***Harvard architecture***,
  - ***Complex instruction set computing (CISC)*** Architecture,
  - ***Single chip*** microcontroller (*µC*) series
  - Developed by ***Intel*** in ***1980*** for use in embedded systems

- Intel's original MCS-51 family was developed using ***N-type metal-oxide-semiconductor (NMOS)*** technology

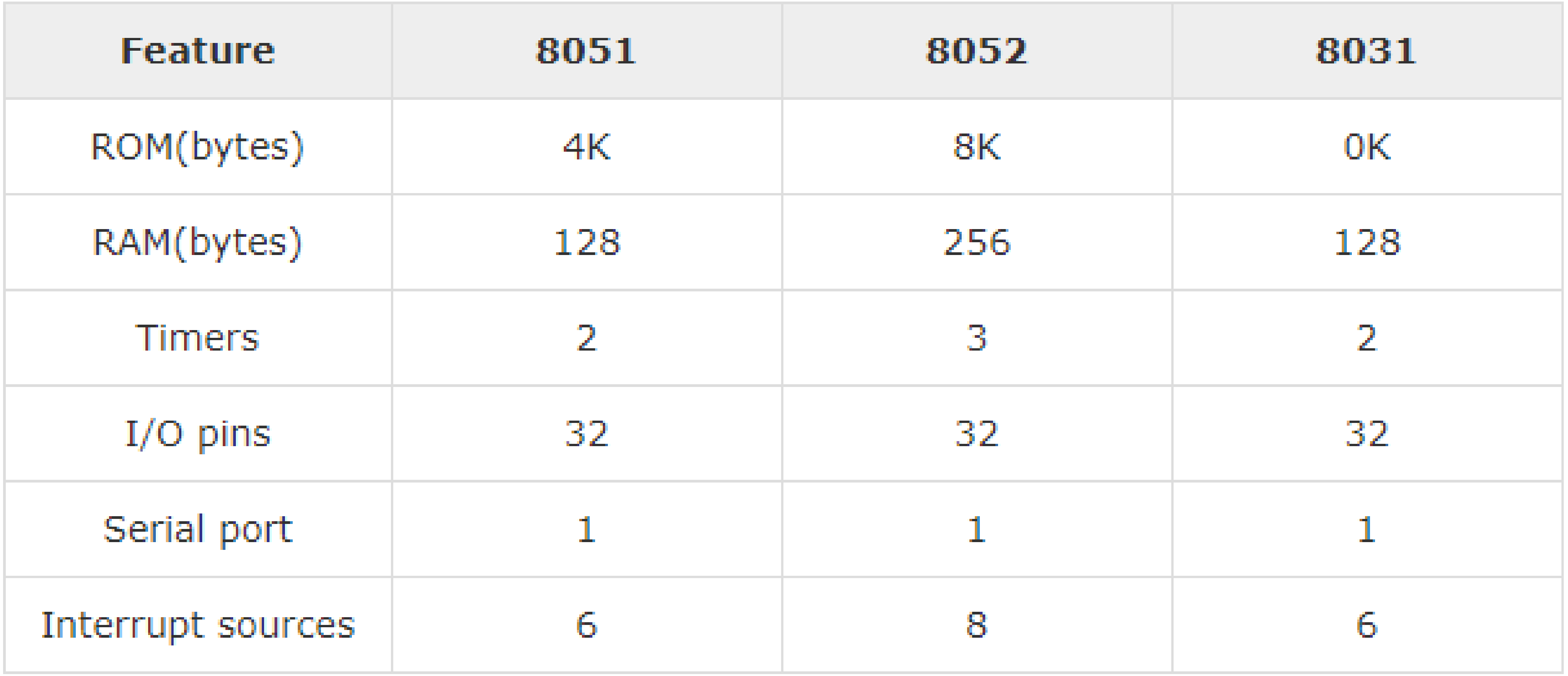
- But later versions, identified by a ***letter C*** in their name (e.g., ***80C51***) used ***complementary metal–oxide–semiconductor (CMOS)*** technology and **consume less power**

Intel MCS 51 µC

- **Features:**
  - **8-bit CPU**
  - **4K bytes** on-chip program memory (**ROM**)
  - **128 bytes** on-chip data memory (**RAM**)
  - **32 I/O pins** arranged as **four 8-bit ports (P0 - P3)**
  - **32 general purpose registers** each of **8-bit**
  - **Special Function Registers** (SFRs) of **128 bytes**
  - **16-bit Program Counter**
  - **8-bit** Processor Status Word (**PSW**) & **Stack Pointer**
  - **Two 16-bit timer/counters** : T0 and T1
  - **Two external** and **three internal vectored interrupts**
  - **One full duplex serial I/O** (UART)

Intel MCS 51 µC

- **8051 Family:**



Feature	8051	8052	8031
ROM(bytes)	4K	8K	0K
RAM(bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

Intel MCS 51 µC

- **8051 Family:**
  - 8051 is the most popular member of the 8051 family,
    - *You will not see “8051" in the part number.*
  - Because the **8051 is available** in **different memory types**, such as
    - ***UV-EPROM***,
    - ***Flash***, *and*
    - ***NV-RAM***,
      - All of which have different part numbers.

Intel MCS 51 µC

- **8051 Family:**
  - **UV-EPROM** version of the **8051** is the **8751**.
  - **Flash ROM version** is **marketed** by **many companies** including ***Atmel Corp***. and ***Dallas Semiconductor***.
    - **Atmel Flash 8051** is called **AT89C51**,
    - **Dallas Semiconductor** calls theirs **DS89C4xO** (DS89C420/430/440).
  - **NV-RAM** version of the **8051** made by **Dallas Semiconductor** is called **DS5000**.
  - **OTP** (one-time programmable) version of the **8051** made by various manufacturers.

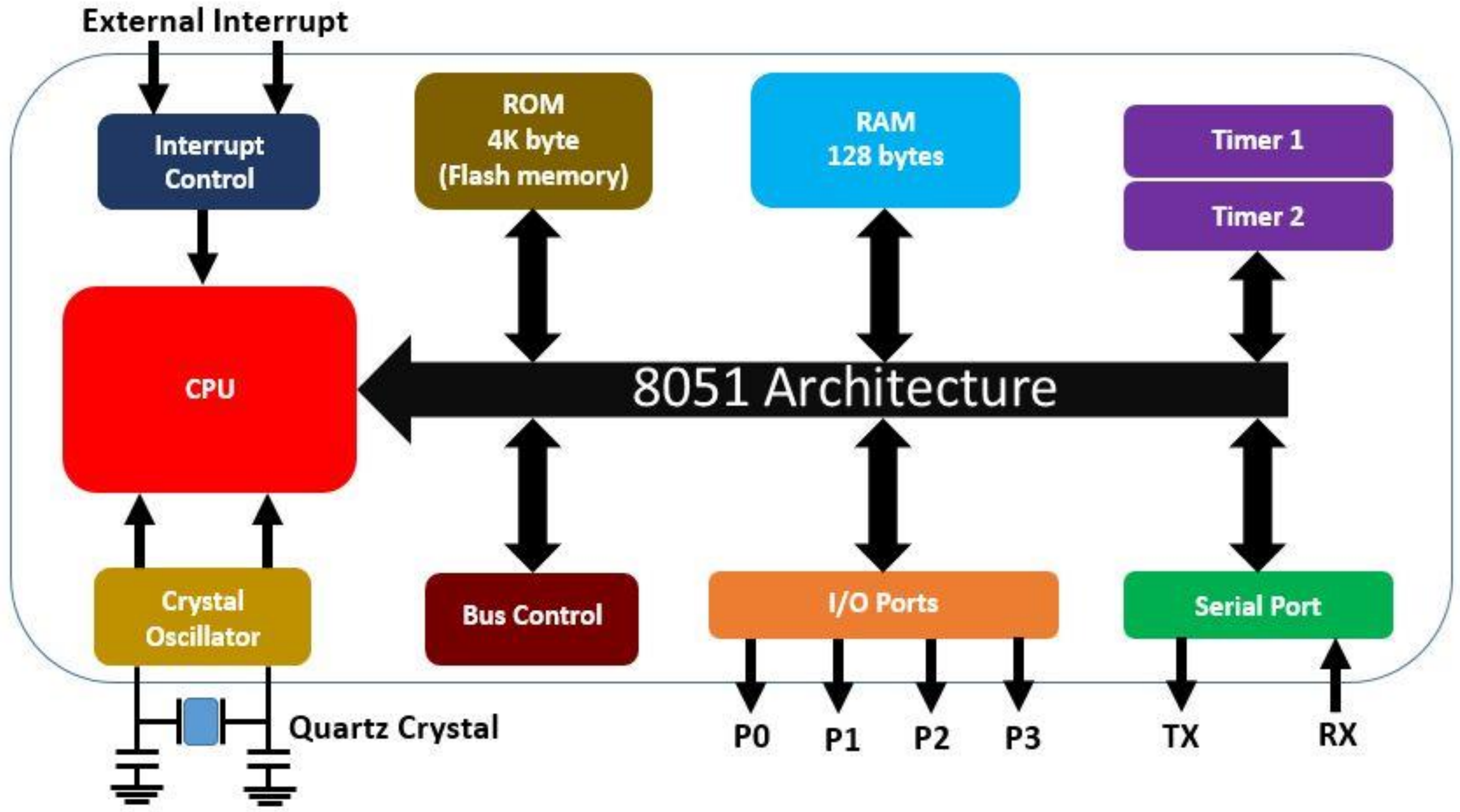
Intel MCS 51 µC

- **8051 Manufactures:** Atmel, Analog Devices, St Microelectronics, Dallas, Maxim, Silicon Labs, Texas Instruments, Microchip, Zilog
- **Atmel 8051 Series**

Part Number	ROM	RAM	I/O pins	Timer	Interrupt	Vcc	Packaging
AT89C51	4K	128	32	2	6	5V	40
AT89C52	8K	256	32	3	8	5V	40
AT89C1051	1K	64	15	1	3	3V	20
AT89C2051	2K	128	32	3	8	3V	20
AT89LV51	4K	128	32	2	6	3V	40
AT89LV52	8K	128	32	3	8	3V	40

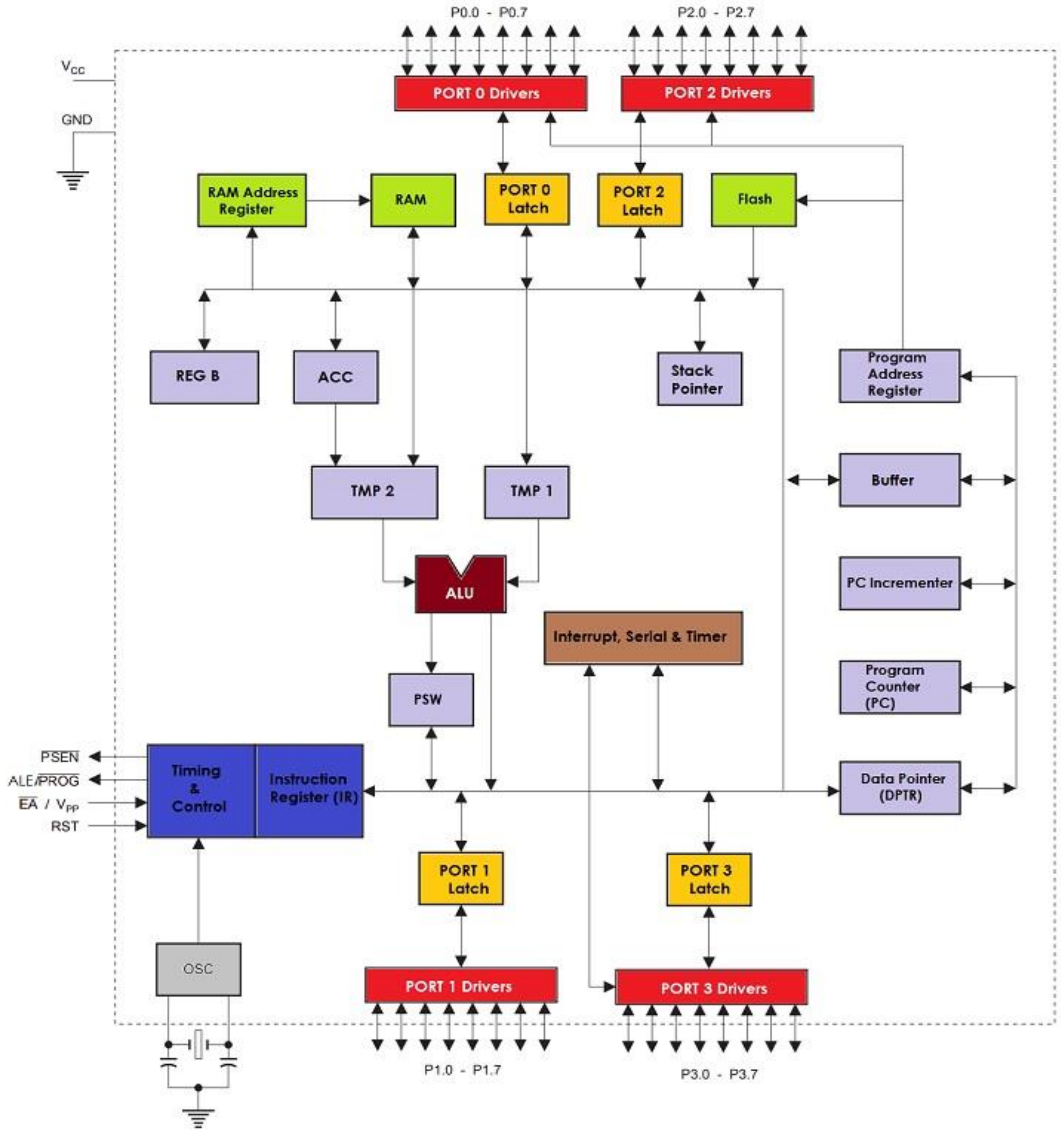
8051 μC Architecture

8051 Architecture (*Simplified*)



8051 Arch.

(*Detailed*)



8051 Architecture

- **Central Processing Unit**
  - CPU is the brain of any processing device of the µC.
  - It **monitors** and **controls** all **operations** that are performed on the μC units.
  - ***User*** has ***no control*** over the work of the CPU ***directly***.
  - It ***reads program*** written in ***ROM*** memory and ***executes them*** and ***do the expected task*** of that application**.**

- **Interrupt Control**
  - **5 interrupt** sources in 8051 Microcontroller and interrupt control section control them.
    - ***Two external*** interrupts (INT0 & INT1),
    - ***Two timer*** (TF0 & TF1) interrupts and
    - ***One serial port*** (RI / TI) interrupt.
  - ***8051 can be configured*** in such a way that it ***temporarily terminates*** or ***pause the main program*** at ***the occurrence of interrupt***.
    - When ***subroutine*** is ***completed*** then the ***execution*** of ***main program starts*** as usual.

8051 Architecture

- **RAM & ROM**
  - Has **4K** of **Code Memory** or Program memory that is it has **4KB ROM** and
  - **RAM** of **128 bytes**.
  - **ROM**: The memory which is used to store the program of Microcontroller, is known as code memory or **Program memory** .
  - **RAM:** Also requires a memory to store data or operands temporarily. This memory is known as **Data Memory**.

- **Bus Control**
  - Responsible for **controlling the operation of address** and **data bus**.
  - **Bus**: is a ***collection of wires*** which ***work*** as a ***communication channel*** or ***medium*** for ***transfer*** of ***Data***.

  - **2 Type of Buses:**
  - **Address Bus**: Microcontroller 8051 has a ***16 bit*** address bus. It used to address memory locations.
  - **Data Bus:** Microcontroller 8051 has ***8 bits*** data bus. It is used to carry data.

8051 Architecture

- **Crystal Oscillator**
  - Since Microcontroller is a digital circuit device, therefore it ***requires clock for its operation***.
  - For this purpose, Microcontroller 8051 has oscillator circuitry section which works as a ***clock source*** for ***CPU***.
  - As the **output pulses** of oscillator are **stable** therefore it enables **synchronized work** of all parts of 8051 Microcontroller.

- **I/O Ports**
  - To connect any ***external devices or peripherals*** we require I/O interfacing ports in the microcontroller.
  - All 8051 microcontrollers have ***4 I/O ports*** each comprising **8 bits** which can be configured as input (1) or an output (0), depends on its logic state.
  - Accordingly, in total **of 32 input/output pin**s enabling the microcontroller to be connected to peripheral devices are available for use.

8051 Architecture

- Timers & Counters
  - Two ***16-bit timers*** and ***counters***: Timer 0 and Timer 1.
  - They can be ***used either as timers*** to generate a ***time delay*** or as ***counters to count*** events happening outside the microcontroller.
  - Since the ***8051*** has an ***8-bit*** architecture, each ***16-bi***t is ***accessed*** as ***two separate registers*** of low byte and high byte.

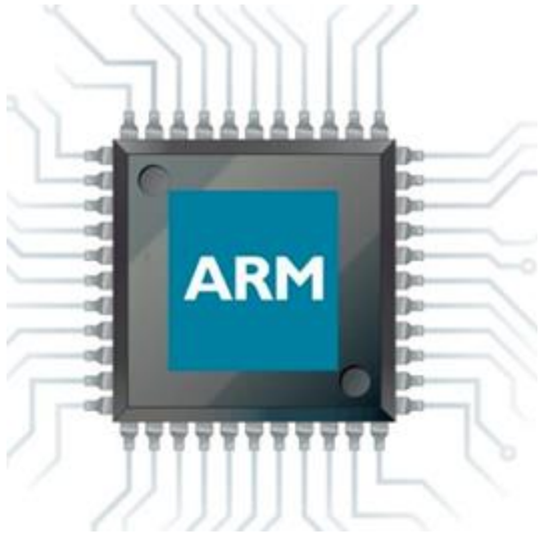
- Serial Ports
  - Contains ***one Serial port*** or ***UART*** (Universal Asynchronous Receiver Transmitter)
  - The serial port is ***full-duplex*** so, it can transmit and receive simultaneously
  - **Two port pins** are used to provide the serial interface
    - ***P3.0*** is the ***receive*** pin (RXD)
    - ***P3.1*** is the ***transmit*** pin (TXD)
  - Can be programmed to ***operate in one of four different modes*** and at a ***range of frequencies***.

ARM

ARM

- ARM was founded as **Advanced RISC Machines** in **1990**
- **ARM** is the world's **leading provider** of **RISC** (Reduced Instruction Set Computer) based **microprocessor**
  - *ARM does not manufacture silicon by itself, instead it license ARM core design to many semiconductor partners around the world.*
- **Used** especially in ***portable devices*** due to ***low power consumption*** and ***reasonable performance*** (MIPS/watt)

- *ARM also develop technologies to assist designing of ARM architecture* which includes, *software tools, boards, debug hardware, application software, bus architectures, peripherals* etc.,



Advanced RISC Machines

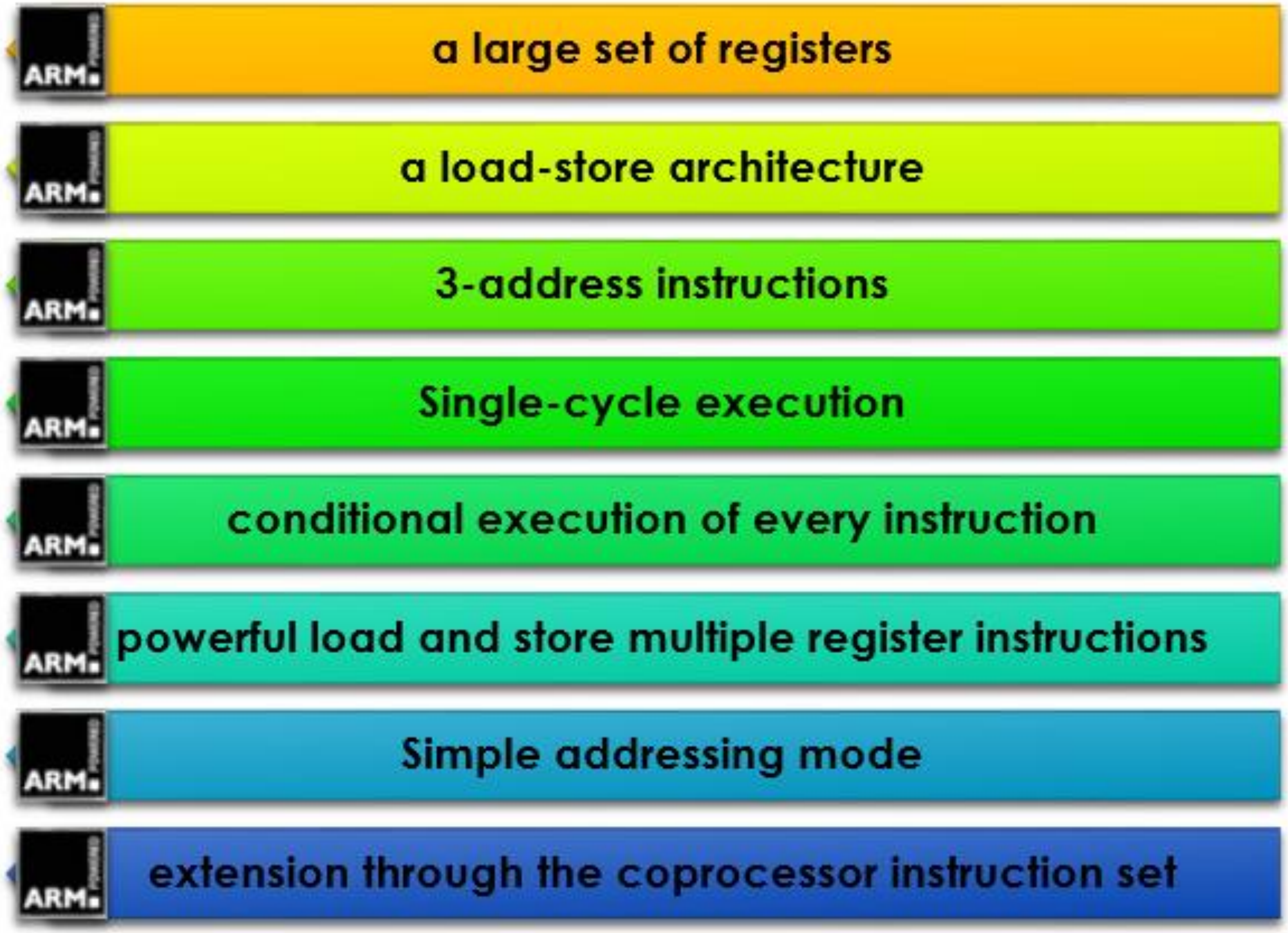
ARM Processor : Features

- ❑ **32/64-bit RISC Processor Core**
- ❑ **Load** and **Store** type **architecture**
- ❑ **Fast interrupt** response
- ❑ **Uniform** and **fixed length instruction** set (32-bit ARM & 16-bit Thumb)
- ❑ **Small core size** and **power-efficiency**
- ❑ **Multiple stage Pipelined** operation
- ❑ **Wide range** of **clock frequency** ranging from **1MHz** to few **GHz**
- ❑ **High performance** and **High code density**
- ❑ **8 / 16 / 32/ 64 -bit data types**
- ❑ **Various modes of operation** (USR, SYS, FIQ,IRQ, SVC, ABT, UND)
- ❑ **Good speed** / **power consumption ratio**

ARM Processor : Applications

- Home Appliances
- Consumer Electronics
- Home Automation / Security
- Computer components
- Automotive
- Health care
- Military
- Industrial
- Business
- Retail
- Communications
- Avionics
- Banking

ARM Architecture : Features



a large set of registers

a load-store architecture

3-address instructions

Single-cycle execution

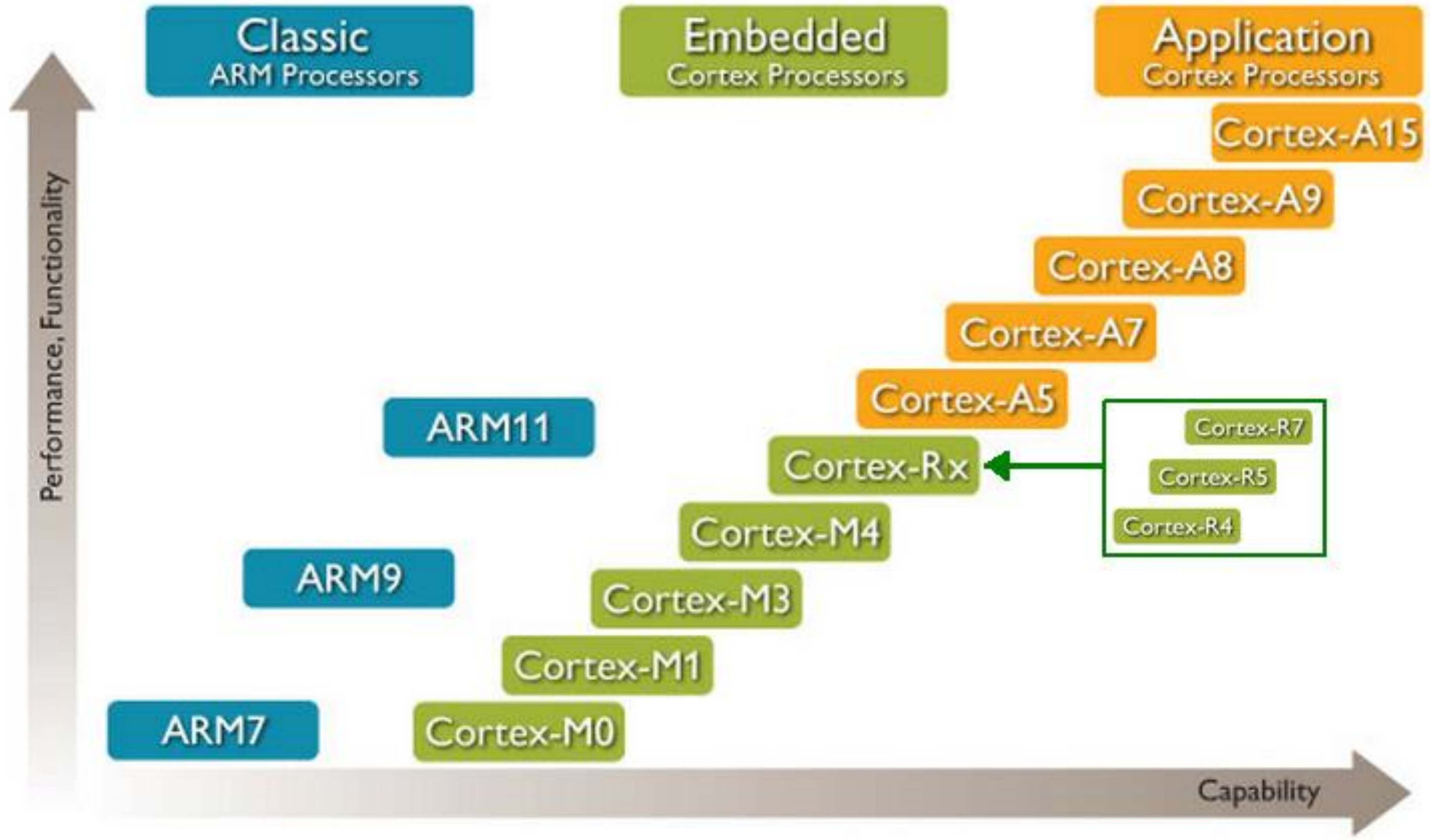
conditional execution of every instruction

powerful load and store multiple register instructions

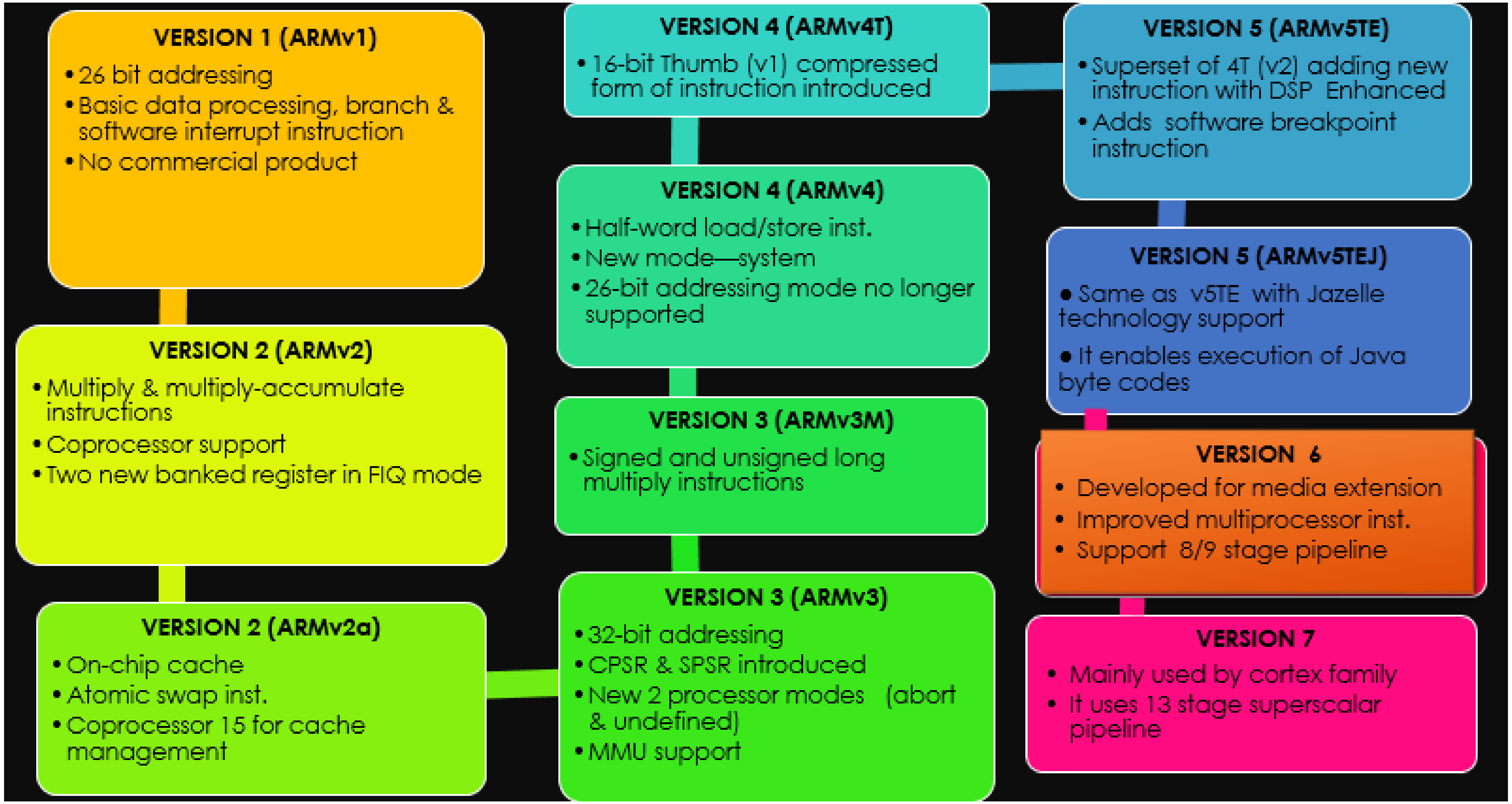
Simple addressing mode

extension through the coprocessor instruction set

ARM Processor : Families



ARM Processor : Versions



VERSION 1 (ARMv1)
- 26 bit addressing
- Basic data processing, branch & software interrupt instruction
- No commercial product

VERSION 2 (ARMv2)
- Multiply & multiply-accumulate instructions
- Coprocessor support
- Two new banked register in FIQ mode

VERSION 2 (ARMv2a)
- On-chip cache
- Atomic swap inst.
- Coprocessor 15 for cache management

VERSION 3 (ARMv3)
- 32-bit addressing
- CPSR & SPSR introduced
- New 2 processor modes (abort & undefined)
- MMU support

VERSION 3 (ARMv3M)
- Signed and unsigned long multiply instructions

VERSION 4 (ARMv4)
- Half-word load/store inst.
- New mode—system
- 26-bit addressing mode no longer supported

VERSION 4 (ARMv4T)
- 16-bit Thumb (v1) compressed form of instruction introduced

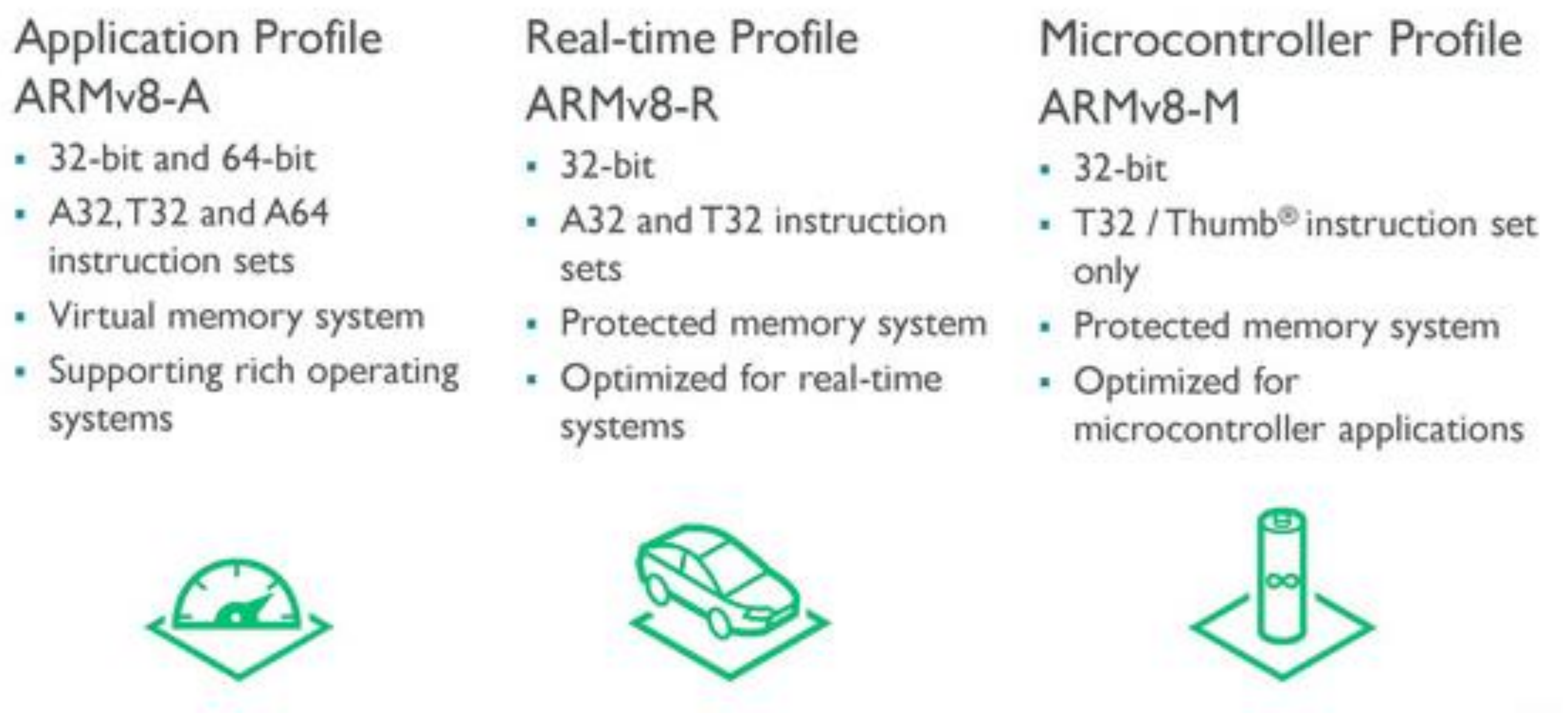
VERSION 5 (ARMv5TE)
- Superset of 4T (v2) adding new instruction with DSP Enhanced
- Adds software breakpoint instruction

VERSION 5 (ARMv5TEJ)
- Same as v5TE with Jazelle technology support
- It enables execution of Java byte codes

VERSION 6
- Developed for media extension
- Improved multiprocessor inst.
- Support 8/9 stage pipeline

VERSION 7
- Mainly used by cortex family
- It uses 13 stage superscalar pipeline

ARM Processor : Current Versions



Application Profile
ARMv8-A

- 32-bit and 64-bit
- A32, T32 and A64 instruction sets
- Virtual memory system
- Supporting rich operating systems

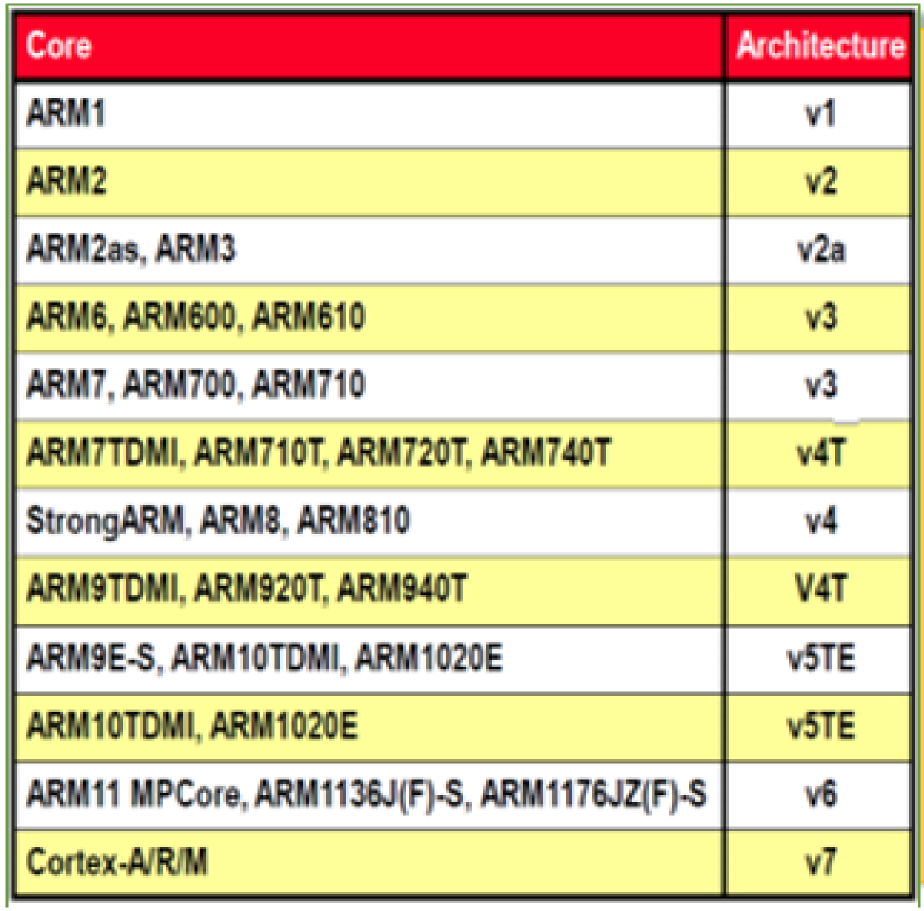
Real-time Profile
ARMv8-R

- 32-bit
- A32 and T32 instruction sets
- Protected memory system
- Optimized for real-time systems

Microcontroller Profile
ARMv8-M

- 32-bit
- T32 / Thumb® instruction set only
- Protected memory system
- Optimized for microcontroller applications

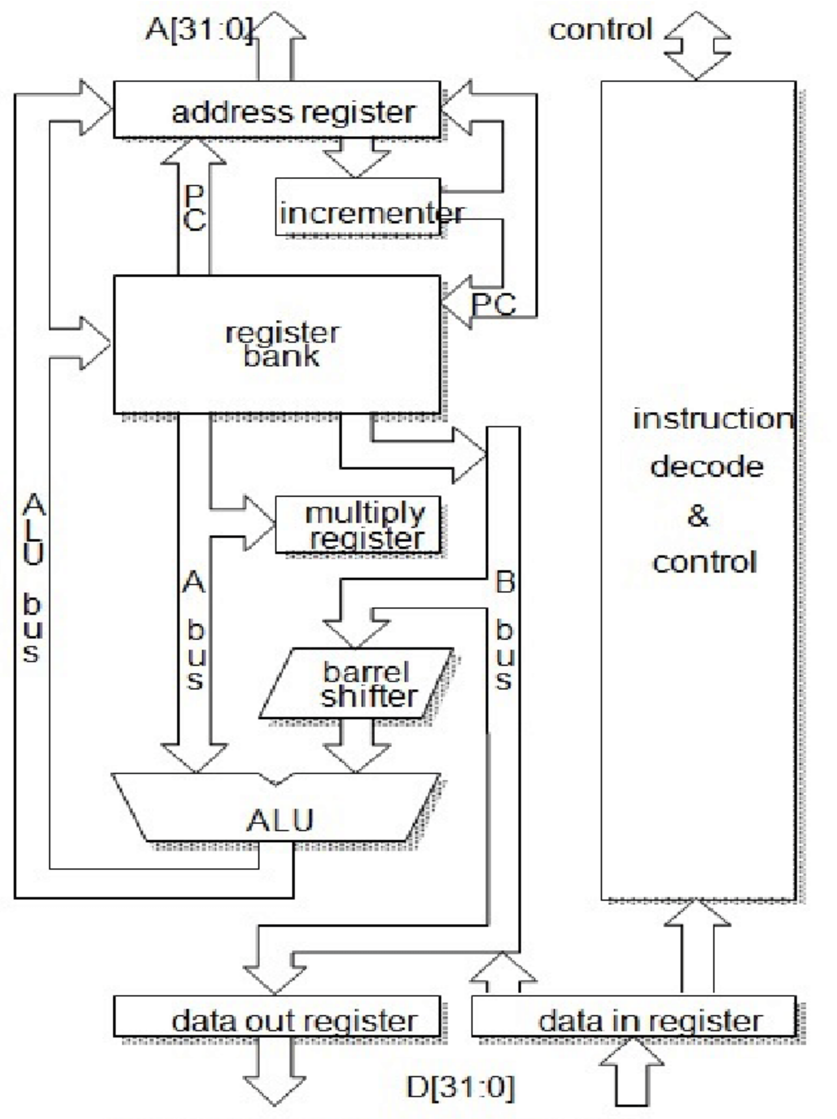
ARM Processor : Evolution



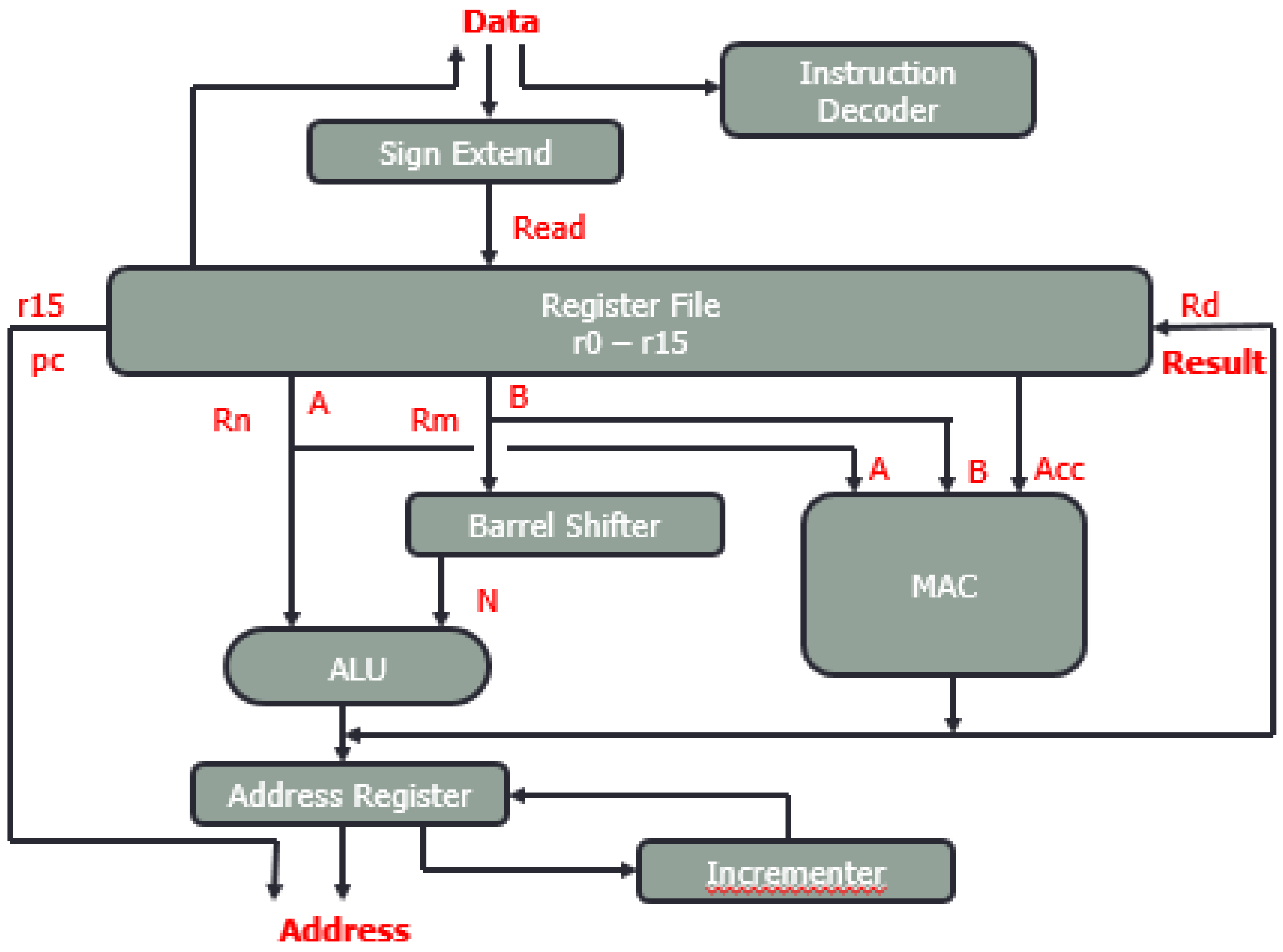
Core	Architecture
ARM1	v1
ARM2	v2
ARM2as, ARM3	v2a
ARM6, ARM600, ARM610	v3
ARM7, ARM700, ARM710	v3
ARM7TDMI, ARM710T, ARM720T, ARM740T	v4T
StrongARM, ARM8, ARM810	v4
ARM9TDMI, ARM920T, ARM940T	V4T
ARM9E-S, ARM10TDMI, ARM1020E	v5TE
ARM10TDMI, ARM1020E	v5TE
ARM11 MPCore, ARM1136J(F)-S, ARM1176JZ(F)-S	v6
Cortex-A/R/M	v7

- ARM{x}{y}{z}{T}{D}{M}{I}{E}{J}{F}{-S}
- x : Family
- y : MMU/Protection unit
- z : Cache
- T : Thumb 16-bit decoder
- D : JTAG Debug
- M : Fast multiplier
- I : Embedded ICE macrocell
- E : DSP Enhanced
- J : Jazelle Technology
- F : Vector floating point unit
- S : Synthesizable version

ARM7 Basic Architecture



ARM Core Dataflow Model



ARM Core Dataflow Model

- **Instruction Decoder:**
  - It **translates instructions** before they are **executed**.
  - The ARM processor, ***like all RISC processors***, uses a ***load-store architecture***.
  - **Load instructions copy data from memory to registers in the core**, and conversely the store

- **Register File**
  - A storage bank made up of **32-bit registers**.
  - Most instructions treat the registers as holding **signed or unsigned 32-bit values**

ARM Core Dataflow Model

- **Sign extend hardware**
  - Converts signed ***8-bit and 16-bit numbers*** to **32-bit values**

- **ALU & MAC:**
  - **Takes** the **register values** ***Rn*** and ***Rm*** from the ***A*** and ***B buses*** and ***computes*** a ***result***.
  - ***Data processing instructions write*** the ***result*** in ***Rd*** **directly** to the **register file**.
  - ***Load*** and ***store instructions*** **use** the **ALU** to **generate** an **address** to be ***held*** in the ***address register*** and ***broadcast*** on the ***Address bus***.

ARM Core Dataflow Model

- **Source Registers:**
  - **ARM instructions** have **two source registers**, **Rn & Rm**, and
  - A **single result** or destination **register**, Rd.
  - ***Source operands*** are ***read*** from the ***register file*** using the ***internal buses A*** and ***B***, respectively.

- **Address Register & Incrementor**
  - For ***load and store instructions*** the ***incrementor updates*** the ***address register before*** the ***core reads*** or ***writes*** the ***next register*** value ***from*** or to ***the next sequential memory location***.
  - The **processor continues executing instructions** until an **exception** or **interrupt** changes the normal execution flow

ARM Core Dataflow Model

- **Barrel Shifter:**
  - ***Register Rm*** alternatively can be ***preprocessed*** in the ***barrel shifter*** before it ***enters*** the ***ALU***.
  - Provides **five types of shift**
    - Logic Shift left (LSL)
    - Logic Shift Right (LSR)
    - Arithmetic Shift Right (ASR)
    - Rotate Right(ROR)
    - Rotate Right Extended (RRX)

- **After passing through the functional units,** the **result** in **Rd** is **written back to the register file** using the ***Result bus***.

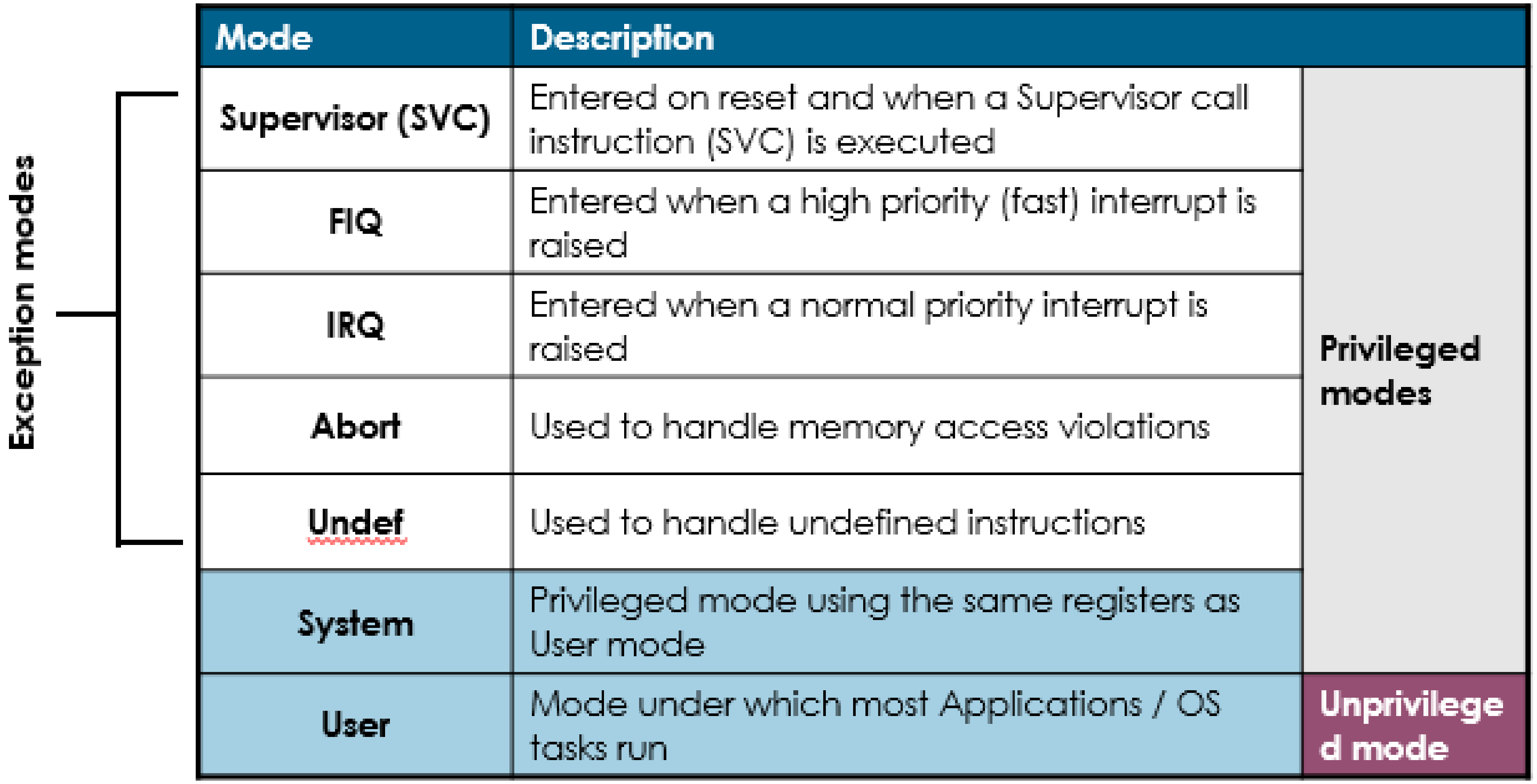
ARM Processor : Modes

- The processor mode determines which ***registers are active*** and the ***access rights*** to the **Current Program Status Register *cpsr register*** itself.

- Each processor has mode:
  - ***PRIVILEGED*** mode allows **full read-write** access to the **CPSR**
  - ***NONPRIVILEGED*** mode only **allows read access** to the **control field** in the **CPDR** but still allows ***read-write access to the condition flags***.

- **7 processor modes:**
  - **Six privileged modes** (*abort, fast interrupt request, interrupt request, supervisor, system, and undefined*) and
  - **One nonprivileged mode** (*user*).

ARM Processor : Modes


Exception modes

Mode	Description	
Supervisor (SVC)	Entered on reset and when a Supervisor call instruction (SVC) is executed	Privileged modes
FIQ	Entered when a high priority (fast) interrupt is raised	
IRQ	Entered when a normal priority interrupt is raised	
Abort	Used to handle memory access violations	
Undef	Used to handle undefined instructions	
System	Privileged mode using the same registers as User mode	
User	Mode under which most Applications / OS tasks run	Unprivileged mode

ARM Processor : Register Set

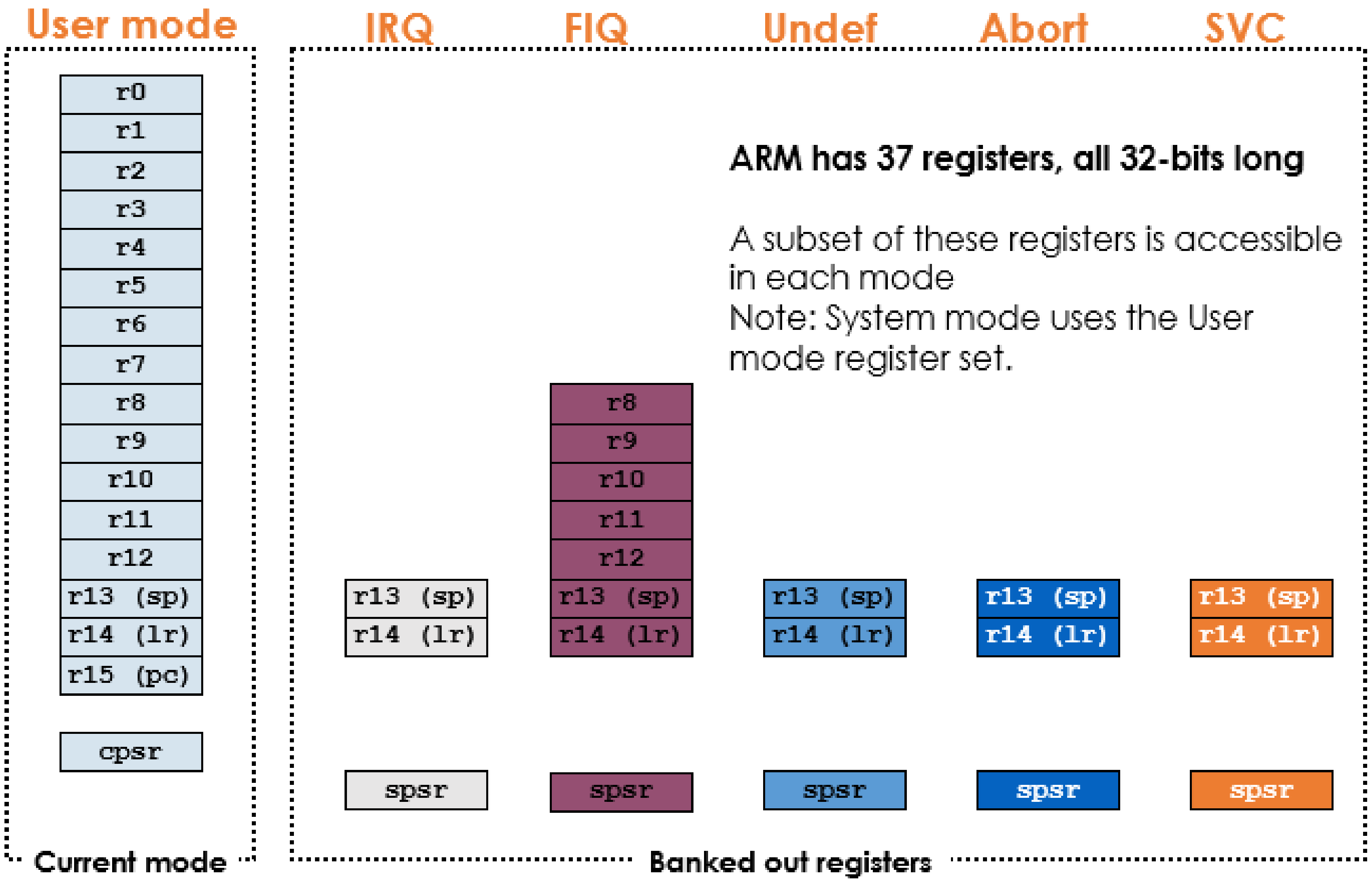
UNBANKED REGISTER

This means that each of them refer to the same 32-bit physical register in all processor modes. They are completely general purpose registers, and can be used whenever an instruction allows GPR to be specified

BANKED REGISTER

They are available only when the processor is in a particular mode. Where particular register is intended, without depending on the current processing mode a more specific name is used.

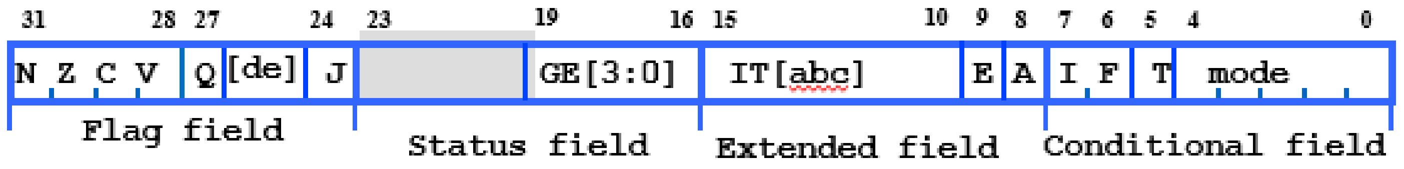
ARM Processor : Register Set



ARM has 37 registers, all 32-bits long

A subset of these registers is accessible in each mode
Note: System mode uses the User mode register set.

ARM Processor : CPSR Register



- Condition code flags
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- Sticky Overflow flag - Q flag
  - Indicates if saturation has occurred
- SIMD Condition code bits – GE[3:0]
  - Used by some SIMD instructions
- IF THEN status bits – IT[abcde]
  - Controls conditional execution of Thumb instructions

- T bit
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state
- J bit
  - J = 1: Processor in Jazelle state
- Mode bits
  - Specify the processor mode
- Interrupt Disable bits
  - I = 1: Disables IRQ
  - F = 1: Disables FIQ
- E bit
  - E = 0: Data load/store is little endian
  - E = 1: Data load/store is bigendian
- A bit
  - A = 1: Disable imprecise data aborts

Mode bits	
M[4:0]	Mode
0b10000	User
0b11111	System
0b10001	FIQ
0b10010	IRQ
0b10011	Supervisor
0b10111	Abort
0b11011	Undefined

PIC

PIC – Features

- This ***powerful*** (***200 nanosecond instruction execution***) yet easy-to-program (***RISC instructions***).
- **CMOS FLASH**-based **8-bit microcontroller** packs Microchip's powerful PIC® architecture into an **40-pin dual in-line (DIP)**
- **PIC16F877A features**
  - **256 bytes** of **EEPROM data memory**,
  - Self programming,
  - An **ICD** - *In-Circuit Debugging*
  - 2 Comparators,
  - **8 channels** of **10-bit Analog-to-Digital** (A/D) **converter**,
  - 2 input capture/output compare/PWM functions,
  - The **synchronous serial port** can be configured as **either 3-wire Serial Peripheral Interface** (SPI™) or the **2-wire Inter-Integrated Circuit** (I²C™) bus and A **Universal Asynchronous Receiver Transmitter** (USART).

PIC – Memory

- **8 K** of **flash program ROM** (14-bits wide).

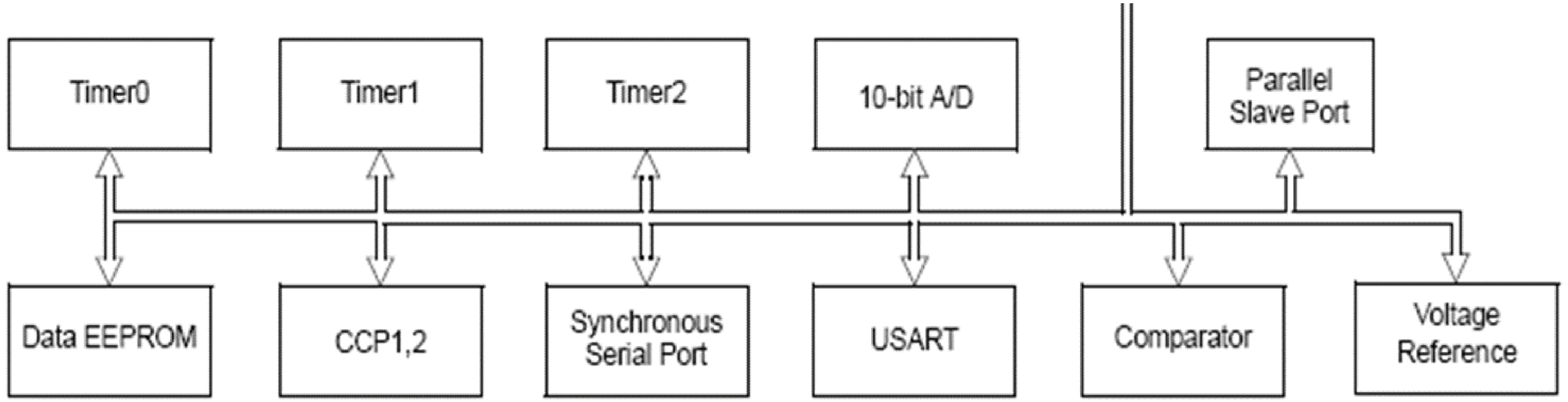
- Each of the **35 RISC instructions** fits in just **one 14-bit wide location**, **permitting fast (200 ns) instruction execution time**, if microcontroller is clocked at its maximum allowable clock rate of 20 MHz.

- **368 bytes** of **volatile data RAM** “File Register” locations (8-bits wide) for **temporary storage of data**.

- **256 bytes** of **nonvolatile EEROM** for **storage of calibration data, user-customized option settings**, etc.

PIC – Memory

- ❑ **PIC16F877A HARDWARE BLOCK DIAGRAM**



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

PIC

Special Functions of PIC16F877A I/O Pins

- **Three hardware timers:** **Timer 0** (RA4/TOCK1 pin), **Timer 1** (RC0/T1OSO/TICKI and RC1/T1OSI pins), and **Timer 2** (no external pins are associated with this timer – this timer can be set to ***periodically interrupt*** via the PR2 period register).

- **10-bit, 8-channel analog-to-digital converter** (analog inputs on Ports A and E)

- **8-bit parallel slave port (PSP**) for making the PIC look like an **addressable 8-bit peripheral register** on a **host computer's bus.**
  - **Port D** for the 8-bit data bus connection and
  - **Port E** for the RD\, WR\, and CS\ bus control lines).

PIC

Special Functions of PIC16F877A I/O Pins

- ❑ **Input Capture – Output Compare** – **Pulse Width Modulation** pins (RC2/CCP1 and RC1/T1OSI/CCP2).

- ❑ **Synchronous serial data port** (SDI, SDO, and SCK pins) used for **implementing master and slave serial peripheral** interface SPI and I2C serial interfaces.

- ❑ **Asynchronous serial data port**, or Universal Asynchronous Receiver-Transmitter, (RC6/TX and RC7/RX pins).

- ❑ **Two analog comparators** with **programmable analog reference voltage** PORT A pins are used for the analog comparator inputs and also the digital comparator outputs.

PIC

	8051	PIC	AVR	ARM
Bus width	8-bit for standard core	8/16/32-bit	8/32-bit	32-bit mostly also available in 64-bit
Communication Protocols	UART, USART,SPI,I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S	UART, USART, SPI, I2C, (special purpose AVR support CAN, USB, Ethernet)	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI (serial audio interface), IrDA
Speed	12 Clock/instruction cycle	4 Clock/instruction cycle	1 clock/instruction cycle	1 clock/ instruction cycle
Memory	ROM, SRAM, FLA SH	SRAM, FLASH	Flash, SRAM, EEPROM	Flash, SDRAM, EEPROM
ISA	CISC	Some feature of RISC	RISC	RISC
Memory Architecture	Von Neumann architecture	Harvard architecture	Modified	Modified Harvard architecture
Power Consumption	Average	Low	Low	Low

PIC

	8051	PIC	AVR	ARM
Families	8051 variants	PIC16,PIC17, PIC18, PIC24, PIC32	Tiny, Atmega, Xmega, special purpose AVR	ARMv4,5,6,7 and series
Community	Vast	Very Good	Very Good	Vast
Manufacturer	NXP, Atmel, Silicon Labs, Dallas, Cyprus, Infineon, etc.	Microchip Average	Atmel	Apple, Nvidia, Qualcomm, Samsung Electronics, and TI etc.
Cost (as compared to features provide)	Very Low	Average	Average	Low
Other Feature	Known for its Standard	Cheap	Cheap, effective	High speed operation
Popular Microcontrollers	AT89C51, P89v51, etc.	PIC18fXX8, PIC16f88X, PIC32MXX	Atmega8, 16, 32, Arduino Community	LPC2148, ARM Cortex-M0 to ARM Cortex-M7, etc.