Introduction to Embedded Systems

by:
Eng. Mohamed Tarek.

Instructor

- BSc. degree in **Communications and Electronics** Engineering from **Cairo** university.
- Software Engineer at **Intel Mobile Communications** company from 8/2013 to 4/2014.
- Embedded Software Engineer at **Mentor Graphics** company from 4/2014 till 7/2017.
- Senior Embedded Software Engineer at **Mentor Graphics** company from 7/2014 till now.
- Former Part Time Embedded Systems Instructor at **ITI Suez canal (Ismailia)** branch.
- Former Part Time Embedded Systems Instructor at **AMIT-Learning** and **SGEC** courses centers.

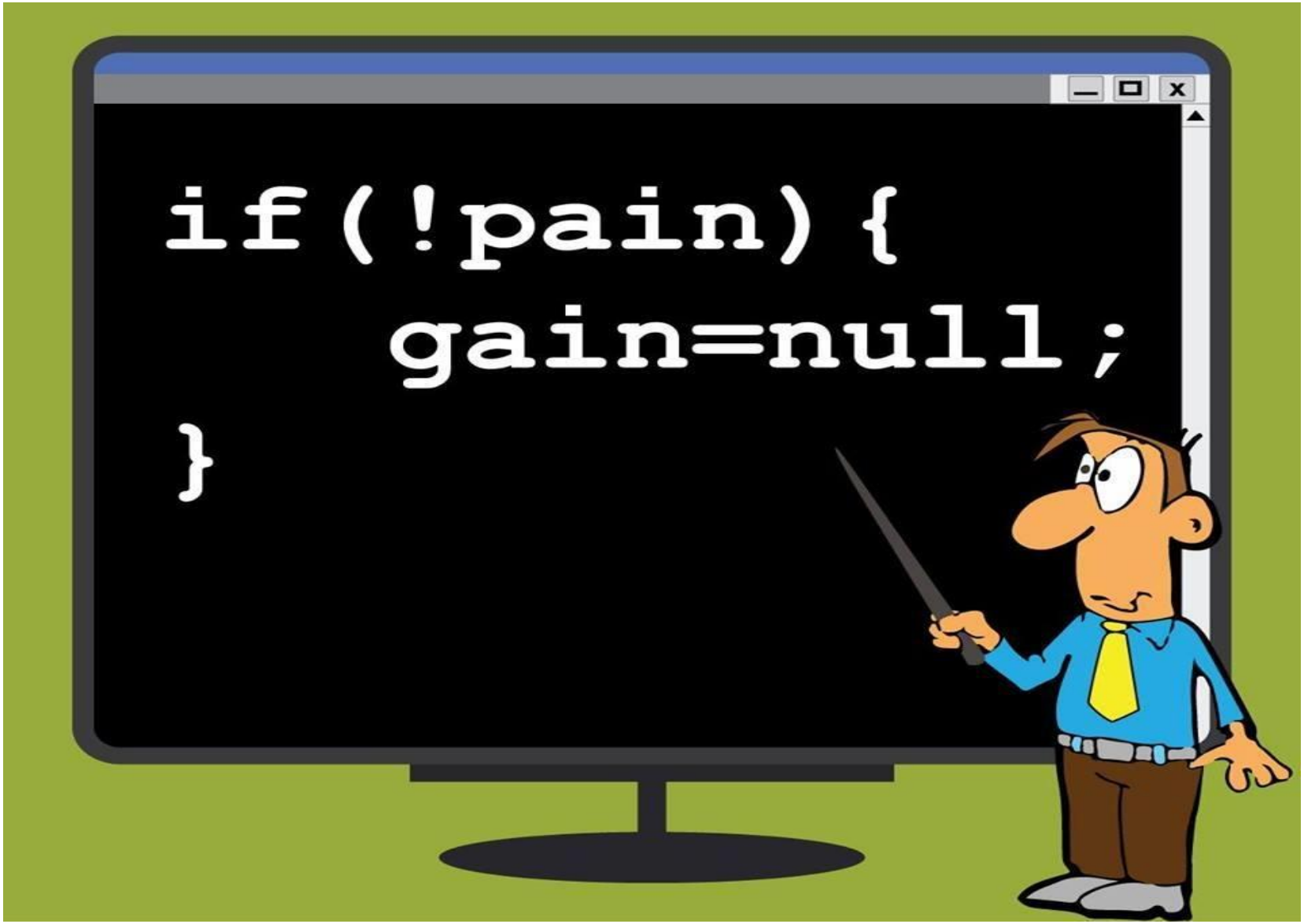
EVERY

EXPERT

WAS ONCE A

BEGINNER

Quotes



Quotes

- “Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven’t found it yet, keep looking. Don’t settle. As with all matters of the heart, you’ll know when you find it”

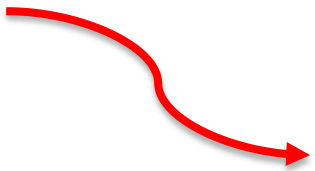
Steve Jobs, Apple co-founder

Quotes

- “There are people who **make** things happen, there are people who **watch** things happen, and there are people who **wonder** what happened.”

  Jim Lovell, Apollo 13 mission commander

- **Watch 75%**
  Wonder 20%
  Make 5%



I hope I am here

Notes

- Ask any time during the session by raising your hand.
- Turn your cell silent.
- Feel free to refer to anything; internet for example.

Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

Agenda

- **Embedded Systems Definition.**
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

Embedded Systems Definition

- All systems that contain one or more **processing units** usually it is micro-controller to do specific functionalities and give responses upon receiving inputs, This processor is not for general purposes like general purpose Processor in PC’s and notebooks.

- Computing systems with tightly coupled hardware and software integration, that are designed to perform a **dedicated and repeated function**.

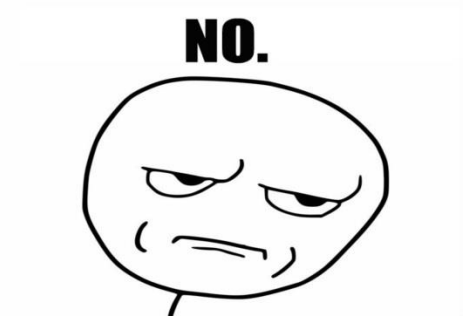
Embedded Systems Definition

- The word embedded reflects the fact that these systems are usually an integral part of a larger system, known as the embedding system. Multiple embedded systems can co-exist in an embedding system.

- Embedded Systems are Application-specific systems which contain hardware and software tailored for a particular task and are generally part of a larger system.

- Historically, the term embedded systems was used to describe any non-PC-related system, but in time, the definition has grown to include any system that is dedicated to a particular functionality.

- **Can Personal Computer be considered as an Embedded System as it integrates hardware and software to perform functions? Why?**



  - PC cannot be considered as an embedded system because :
    1. It uses a General-Purpose Processor.
    2. The system is built independently from the software runs on it.

Agenda

- Embedded Systems Definition.
- **Embedded Systems Characteristics.**
- Embedded Systems Applications.
- Embedded Systems Design.
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

- **Reliability**
  - Personal computer programs such as word processors and games do not need to achieve the same standard of reliability that a microcontroller application must. Errors in programs such as word processors may result in errors in a document or loss of data. An error in an Embedded system application such as a TV remote control or compact disc player will result in a product that does not work and consequently does not sell.
  - An error in Embedded system application such as an antilock braking system or autopilot could be fatal.

Efficiency

- Considered in real time applications.
- A real time application is one in which must be able to act at a speed corresponding with the occurrence of an actual process. must compute certain results in certain real-time without delay.
- Some Embedded systems may have real-time performance constraints that must be met, for reasons such as safety and usability. others may have low or no timing performance requirements, allowing the system hardware to be simplified to reduce costs.

- **Tightly-constrained**

  Embedded Systems normally come with constraints in hardware resources:

  - Processing(speed)
  - Memory(Data)
  - Storage(Code Size)
  - Most of the time it targets real time objectives, this means,
    - It needs to be fast and efficient(Response Time).
    - It needs to be predictable (execution time known ahead, and almost constant)
  - Power limited(battery operated devices).
  - Cost
  - Size

Embedded Systems Characteristics

- The developer has to deal with all of these constraints
  - Development should take into consideration code efficiency and code foot print.
  - Debugging tools are “closer to the metal”
  - Special attention to power consumption in some cases.

- **Single-functionality**

  Dedicated to perform a specific function and include processors dedicated to specific function.

- **Complex functionality**

  Often have to run sophisticated algorithms or multiple algorithms Cell phone, laser printer, automotive, etc.

- **Safety-critical**

  Must not endanger human life and the environment

- **Maintainability**

  The ability to modify the system after its initial release and enhance its performance like execution time, code and memory size.

- **Interactive System**

  An Embedded system should be as interactive as possible so that it is easily understandable, Operated and Handled by a user and provide ease of task.

- **Strong association between the HW and SW.**

Comparison

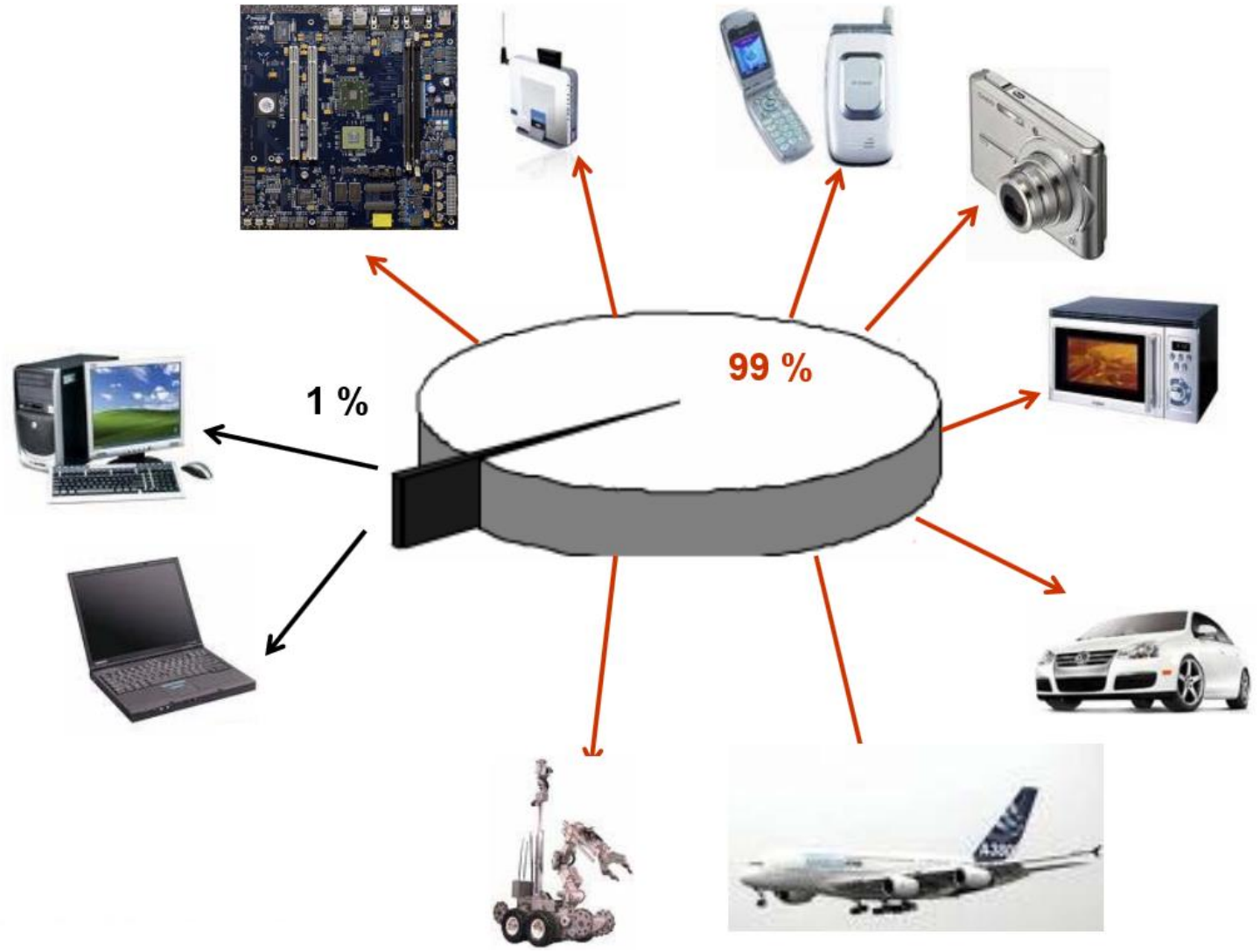
Embedded Systems

- Perform **ONLY** Few applications that are known at design-time (i.e. special purpose).
- Not programmable by end user.
- Fixed run-time requirements (additional computing power not useful).
- Criteria
  - Cost
  - Power consumption
  - Deterministic behavior
  - Meeting time bounds
  - Size

General Purpose Systems

- Computing a lot of applications (i.e. general purpose).
- Programmable by end user
- Faster is better
- Criteria
  - Cost
  - Average speed

Comparison



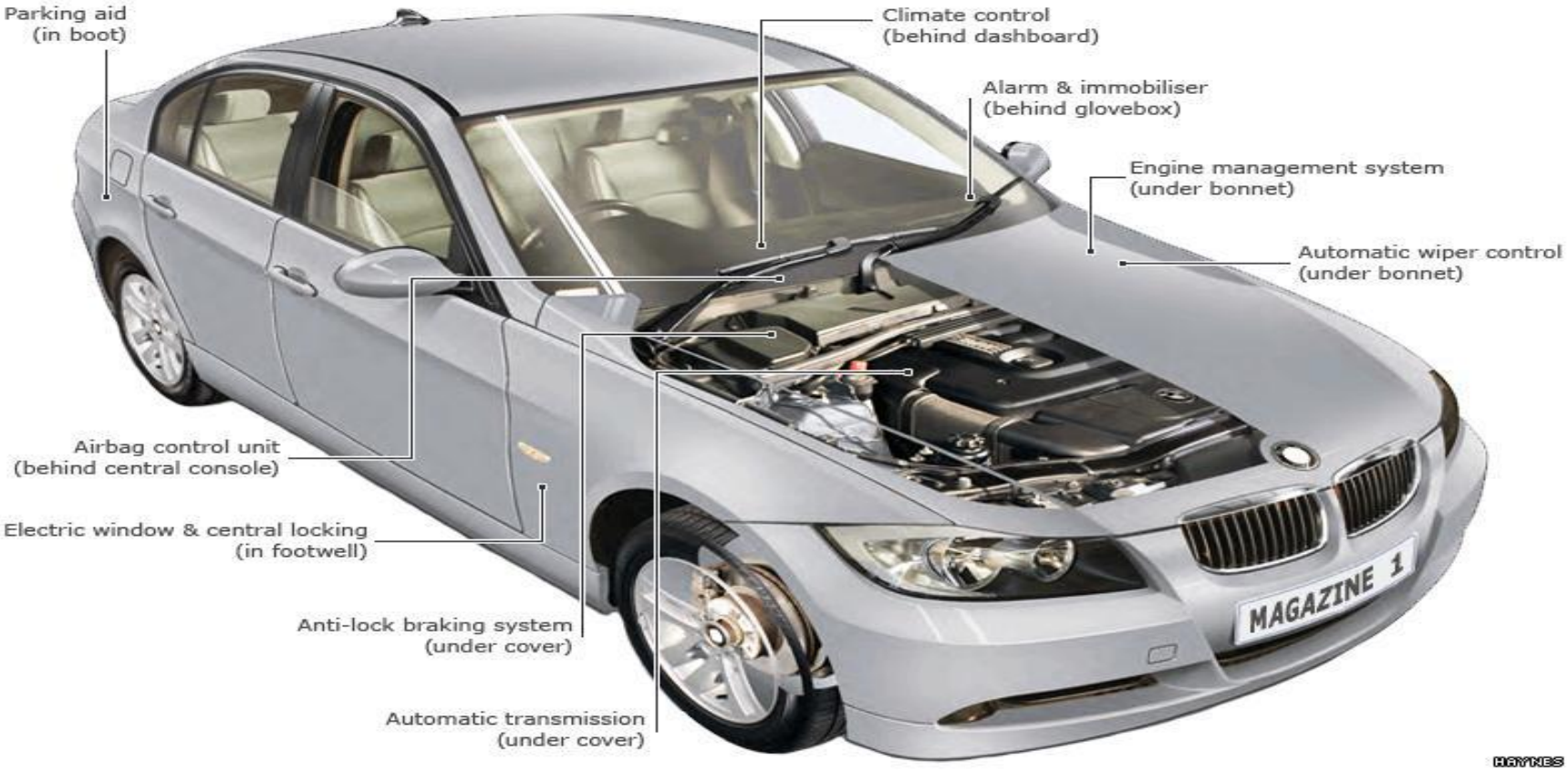
Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- **Embedded Systems Applications.**
- Embedded Systems Design.
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

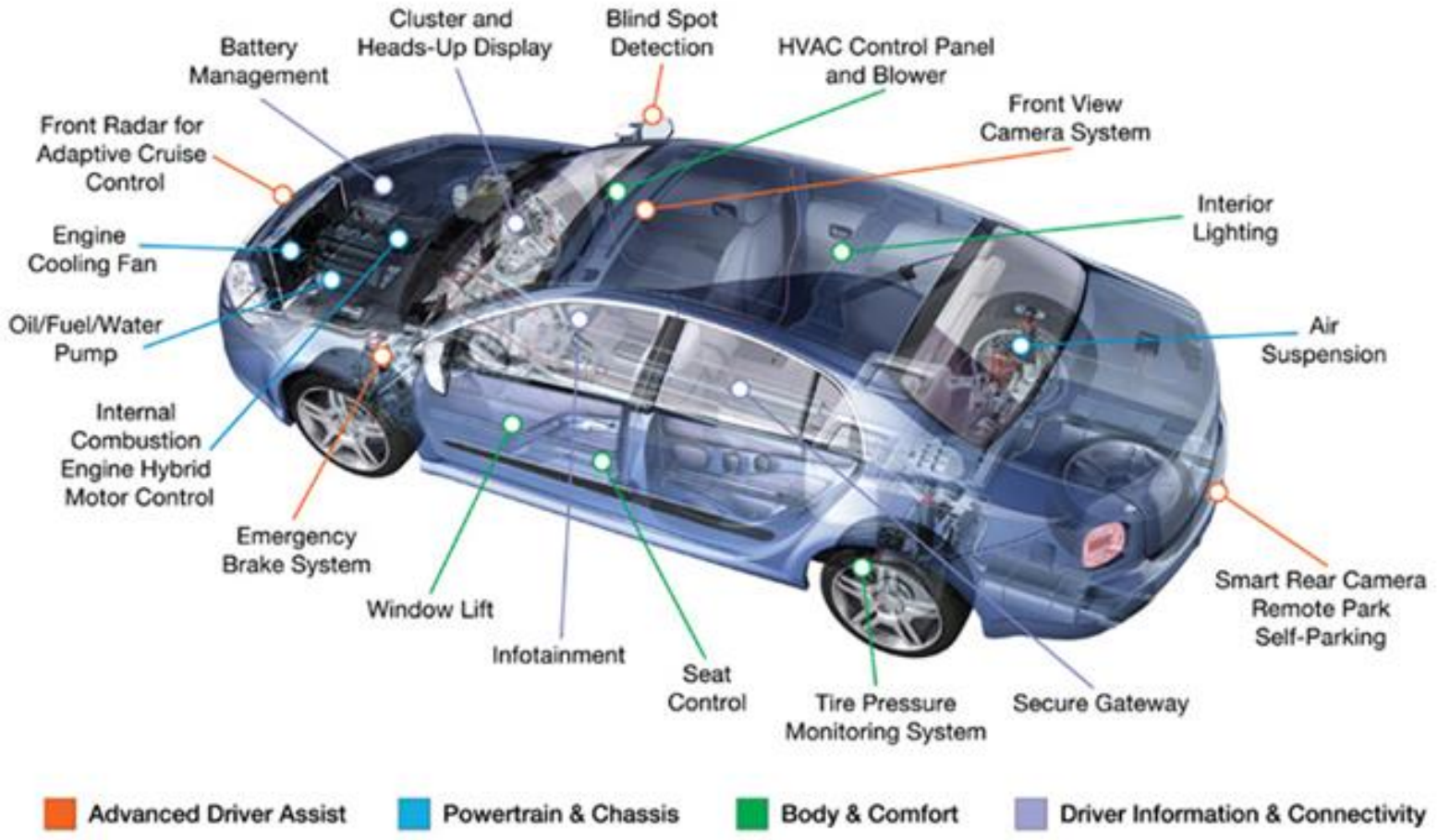
- **Exercise**

  Choose two Embedded Systems and list the main functionalities of each one.

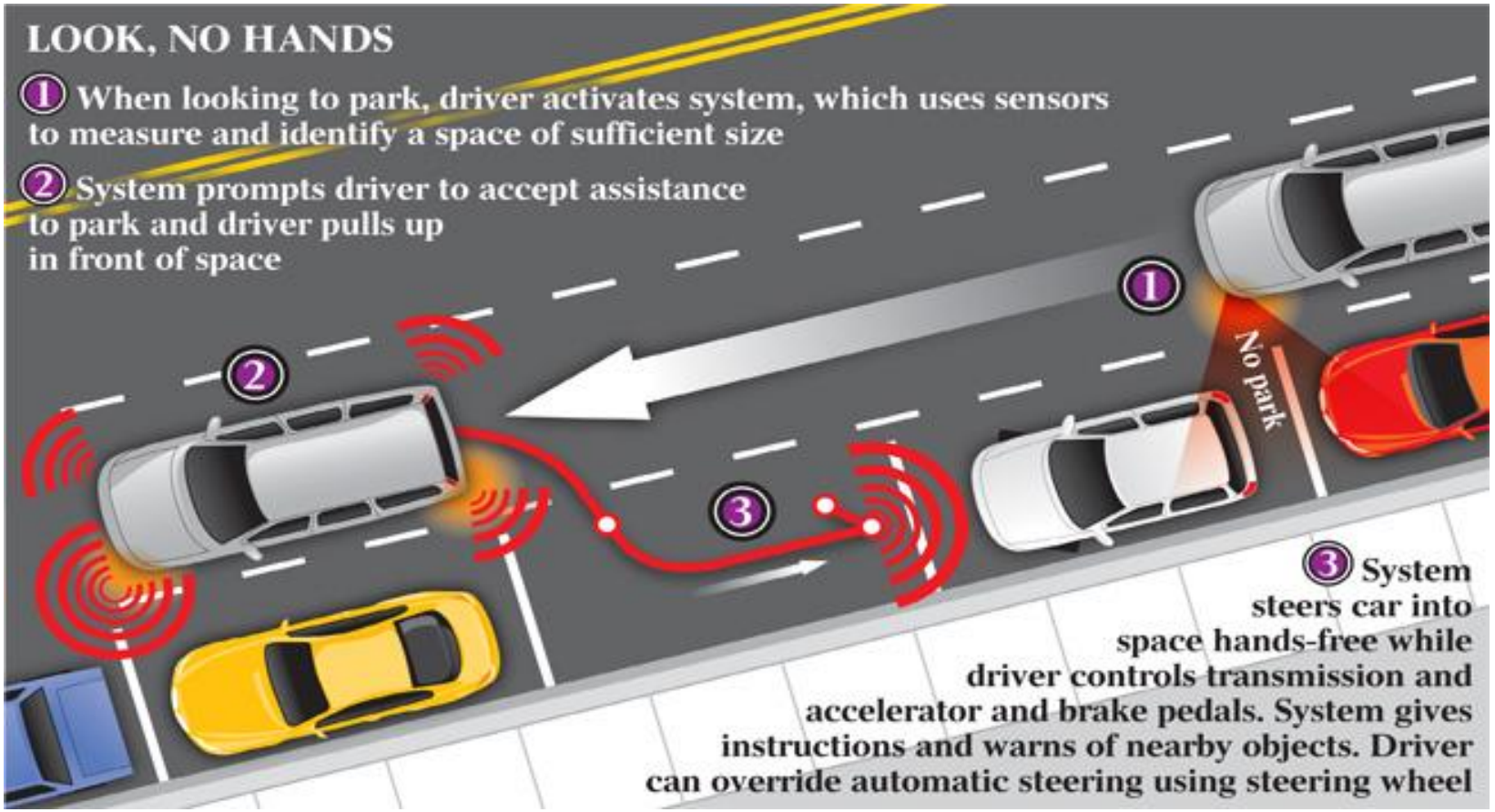
- **Automotive**



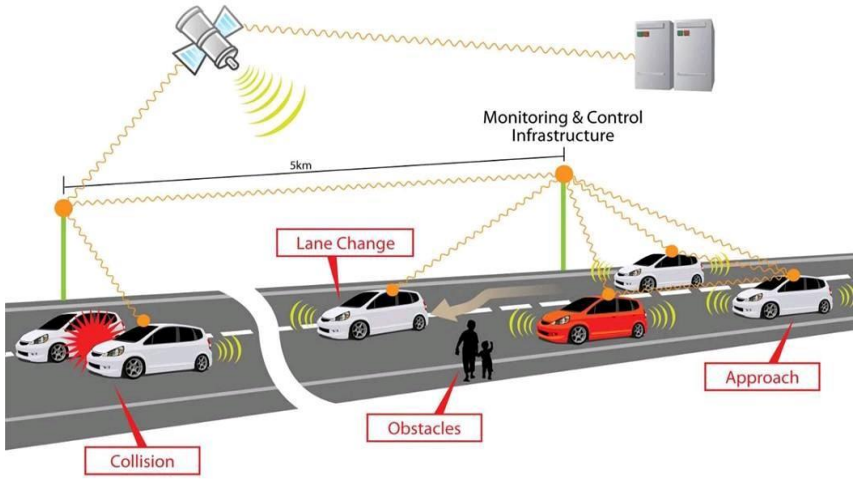
- **Automotive**

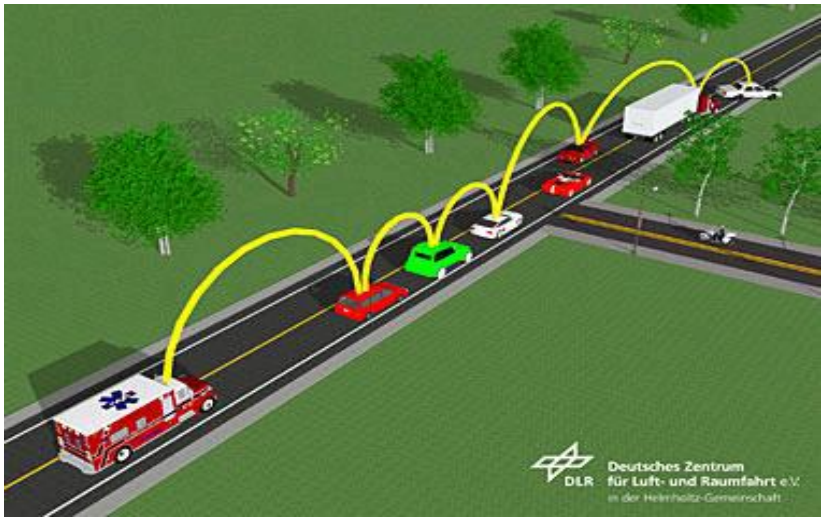


- **Automotive**
  - Electronics represents 40% of total cost of a car.
  - 90% of new car features require software.



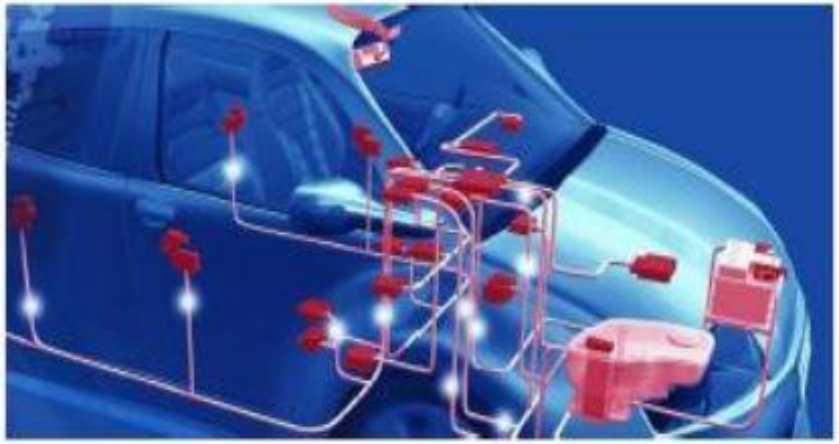
- **Automotive**

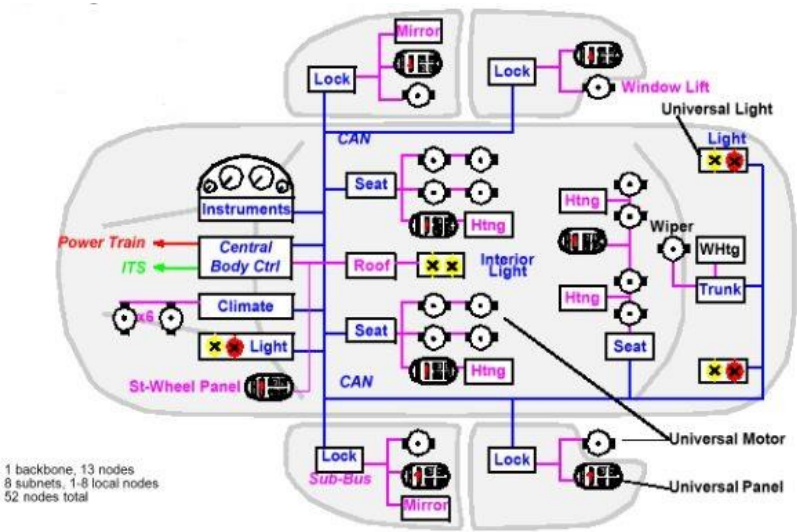




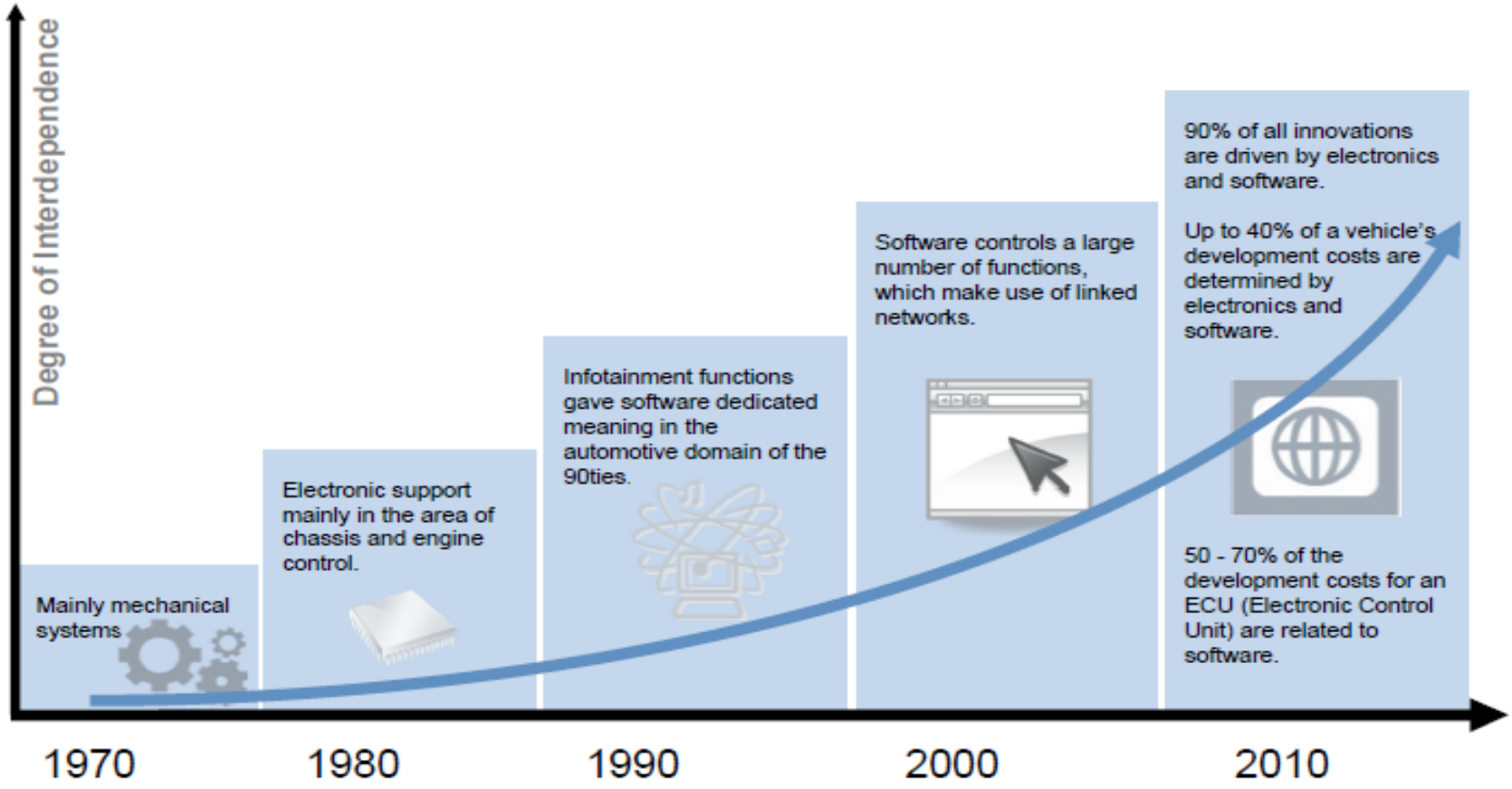
- ECUs are used in different functions of the car :
  - Air Bag.
  - Anti lock braking system(ABS)
  - Engine Control
  - Lighting System.
  - Transmission Control
  - Fuel Efficiency Control
  - Electric Power Steering
  - Speed Control
  - Suspension System Control
  - Battery Management
  - Seat Control
  - Door Control
  - Electric Windows Control
  - Lighting Control
  - Airbag Control
  - Telemetric Control
  - Mirror Control
  - Security System Control
  - Windshield Blades Control
  - Entertainment
  - Human-machine interface (HMI)
  - And a lot more …..

- Embedded Automotive Challenges
  - Reducing number of ECUs by integration many of functionalities in a single ECU.
  - Sharing use of computation power.
  - Optimizing the wiring.
  - Reducing weight.
  - Move from combustion engines to hybrid and full electric engines.





- # Increasing Complexity in Vehicle Development



Phones and Tablets

Contains mainly three types of processors:

1. Communication Processor(s)
   - Wifi
   - GSM/3G/LTE
   - Bluetooth/NFC
2. Audio/Graphics Processor(s)
   - Audio Processing
   - Graphics and Video Processing
3. Application Processor
   - Android
   - Windows Phone
   - iOS

- **Telecommunication systems**

- **Networking Devices**

- **Robotics and Toys**

Embedded Systems Applications

- Product: NASA's Mars Sojourner Rover 1996,low cost spacecraft . Using 8-bit Intel Microprocessor 80C85.

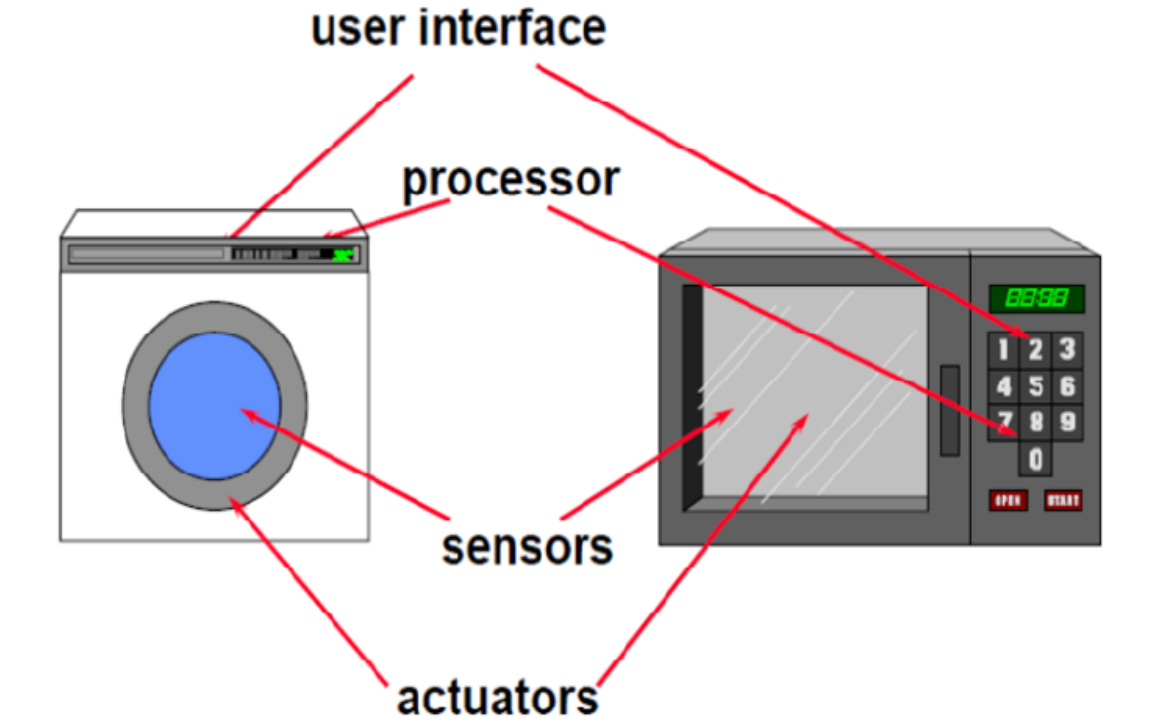
- **Industry**

- # Image Processing

- **Image Processing**

- **Electronic Devices**

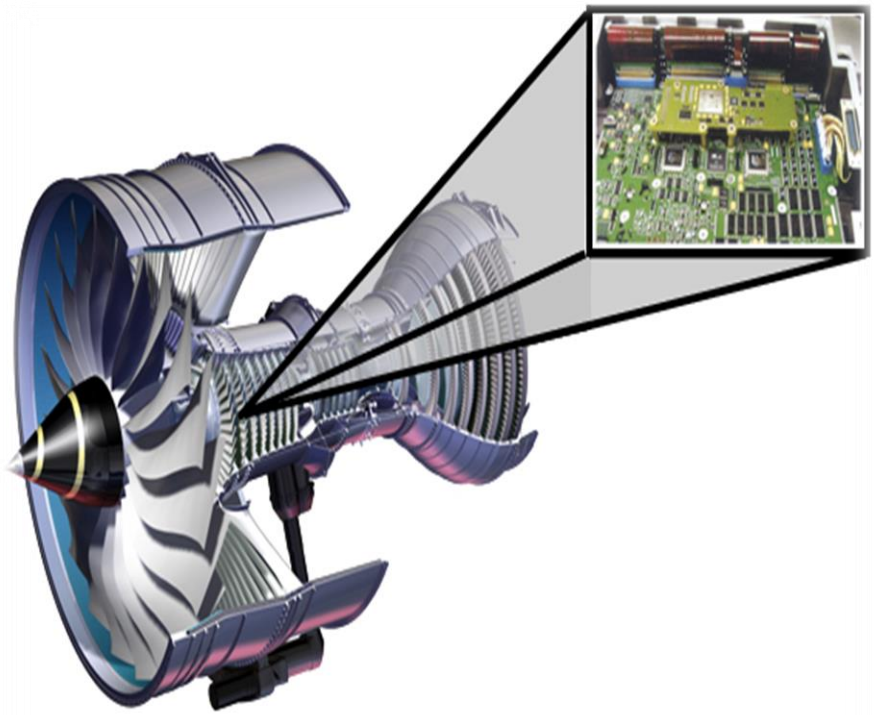
- **Appliances**



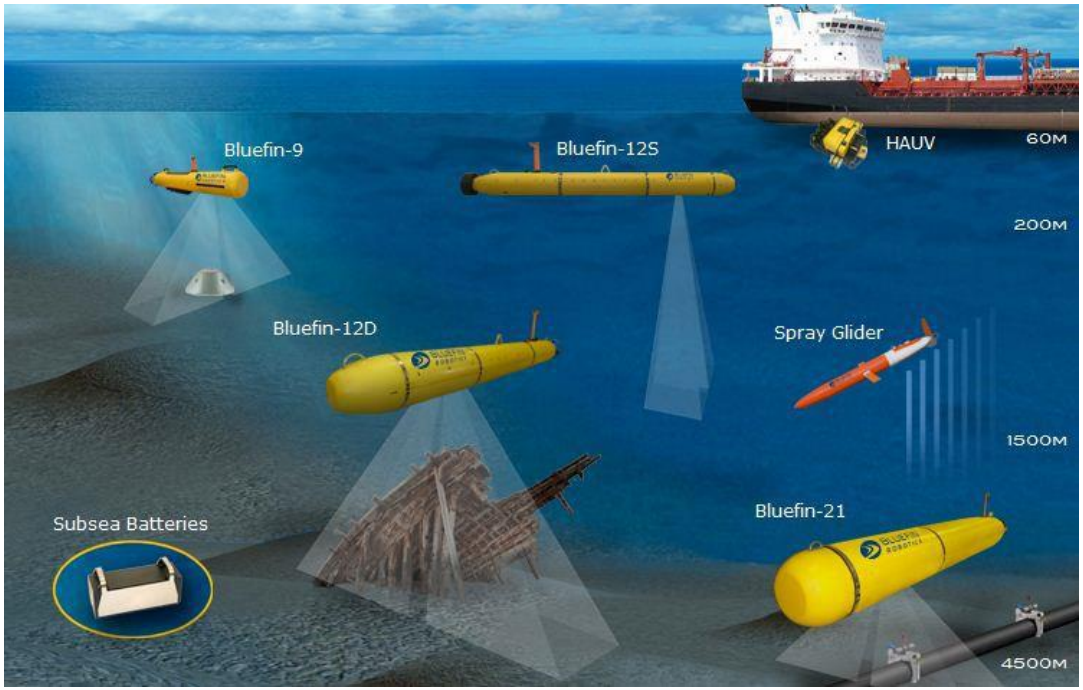
- # Home Automation

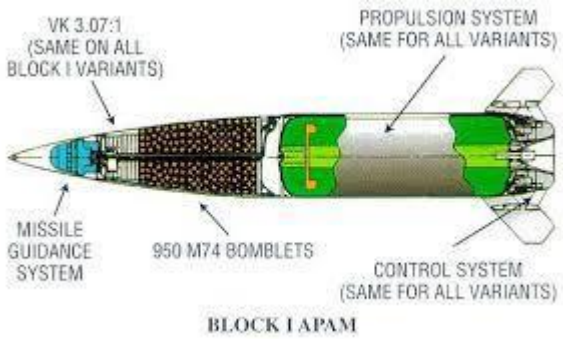
- **Banking**

- **Aircraft Control**



- **Military**





- **Wearable**
  - Google Glasses.
  - Smart watches.
  - Temperature adjustable clothes.

Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- **Embedded Systems Design.**
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

Trade off between SW & HW

For a certain application:

- Which functional blocks should be performed in Hardware??
- Which functional blocks should be performed in software??

- **Software characteristics**
  - Advantages
    - Highly configurable
    - Shorter development cycle
    - Easier in versions updates
    - Cheaper
  - Disadvantages
    - Constrained with processor speed which may satisfy real time application and may not.

- **Hardware characteristics**
  - Advantages
    - Better performance in high speed real time application
  - Disadvantages
    - Longer development cycle.
    - Customized for specific application, not updatable (unchangeable).

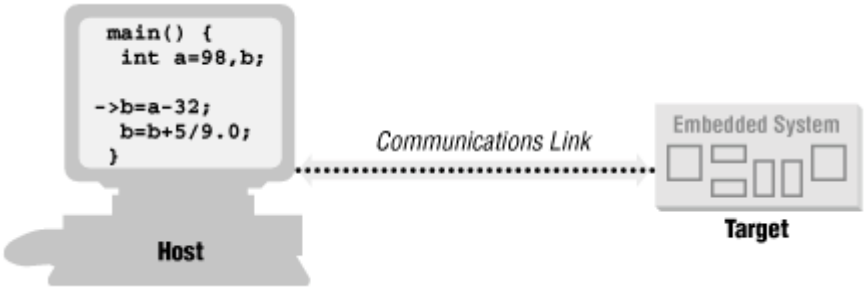
HW/SW Partitioning

In complicated systems functional blocks could be:

- Level I: external discrete hardware component on board
- Level II: hardware integrated with CPU on chip system on chip (SOC)
- Level III: done by software running on CPU

HW/SW Partitioning

- Design and build the target hardware
- Develop the software independently
- Integrate them and hope it works



Embedded Systems Design Steps

1. Modeling via any modeling language (System C or even Matlab)
2. HW/SW partitioning (Determine which blocks must be H/W & which may be S/W)
3. Global Design of the S/W part
4. Unit Design of the S/W part
5. Coding.
6. Unit Testing.
7. Integration Testing.
8. Verification and Validation.
9. Maintaining.

Embedded Systems Design Challenges

- Find an implementation that can perform the computation such that the requirements are satisfied.
- Embedded systems perform computations (software) that are subject to physical constraints (hardware)
- Execution on a physical platform: processor speed, power, storage, reliability.
- Computer science separates computation from physical constraints

- **Design goals**
  - Performance(overall speed execution time and throughput).
  - Functionality and user interface.
  - Manufacturing cost.
  - Power consumption.
  - Safety.
  - Other requirements. (physical size, etc.)

Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- **Embedded HW**
  - **Processing Engines.**
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

1.General Purpose Processor

- CPU = ALU + Registers + Control unit
- 32 or 64-bit data path.
- Modern Processors have:
  - Advanced cache logic.
  - Built-in math co-processor capable of performing fast floating-point operations(FPU).
  - A built-in memory management unit (MMU) to provide memory protection and virtual memory for multitasking-capable.
  - Interfaces to support a variety of external peripheral devices because Microprocessor alone is useless.

1.General Purpose Processor(Cont.)

- Complex in design because these processors provide a full scale of features and a wide spectrum of functionalities.
- These processors result in large power consumption, heat production, and large size.
- Intel X86, Intel Pentium , AMD, PowerPC and SPARC.

2.Embedded General Purpose Processor

- 16/32-bit data path.
- Designed for a wide range of applications (consumer and communication).
- Limited functionality depending on the application.
- Not Flashable the program memory is external and have CASH memory and MMU.
- Usually integrated into larger dedicated systems in a SOC (System on Chip), also called core-based ASIC.
- Have main peripherals Like Timers, Ethernet, USB and GPIO
- Examples: ARM, MIPS and Renesas RH850.
- ARM has seized the lion’s share of the market.

3.Digital Signal Processor

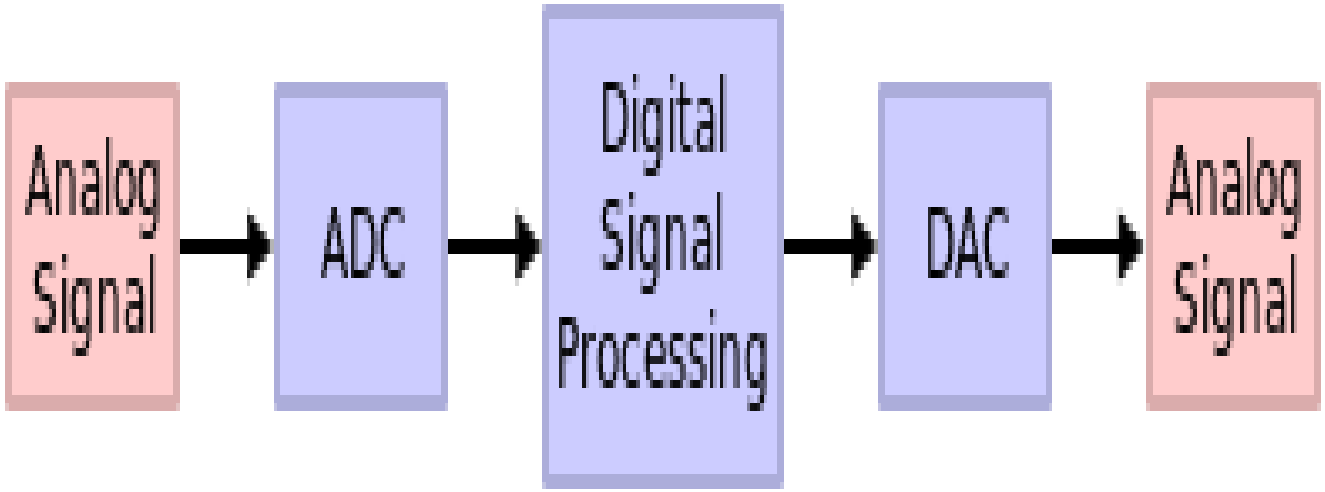
- Like microprocessor but focus on very efficient execution of arithmetic operations.

- Digital signal processing algorithms typically require a large number of mathematical operations to be performed quickly and repeatedly on a series of data samples. Signals (perhaps from audio or video sensors) are constantly converted from analog to digital, manipulated digitally by DSP processor, and then converted back to analog form.

3.Digital Signal Processor

- Used widely in digital signal processing in communication systems such as Cell phones, Image and video processing.
- Even though DSPs are incredibly fast and powerful embedded processors,
- Examples: TI (Texas Instruments), Motorola.
- TI has been the dominant player in the DSP market for several years.

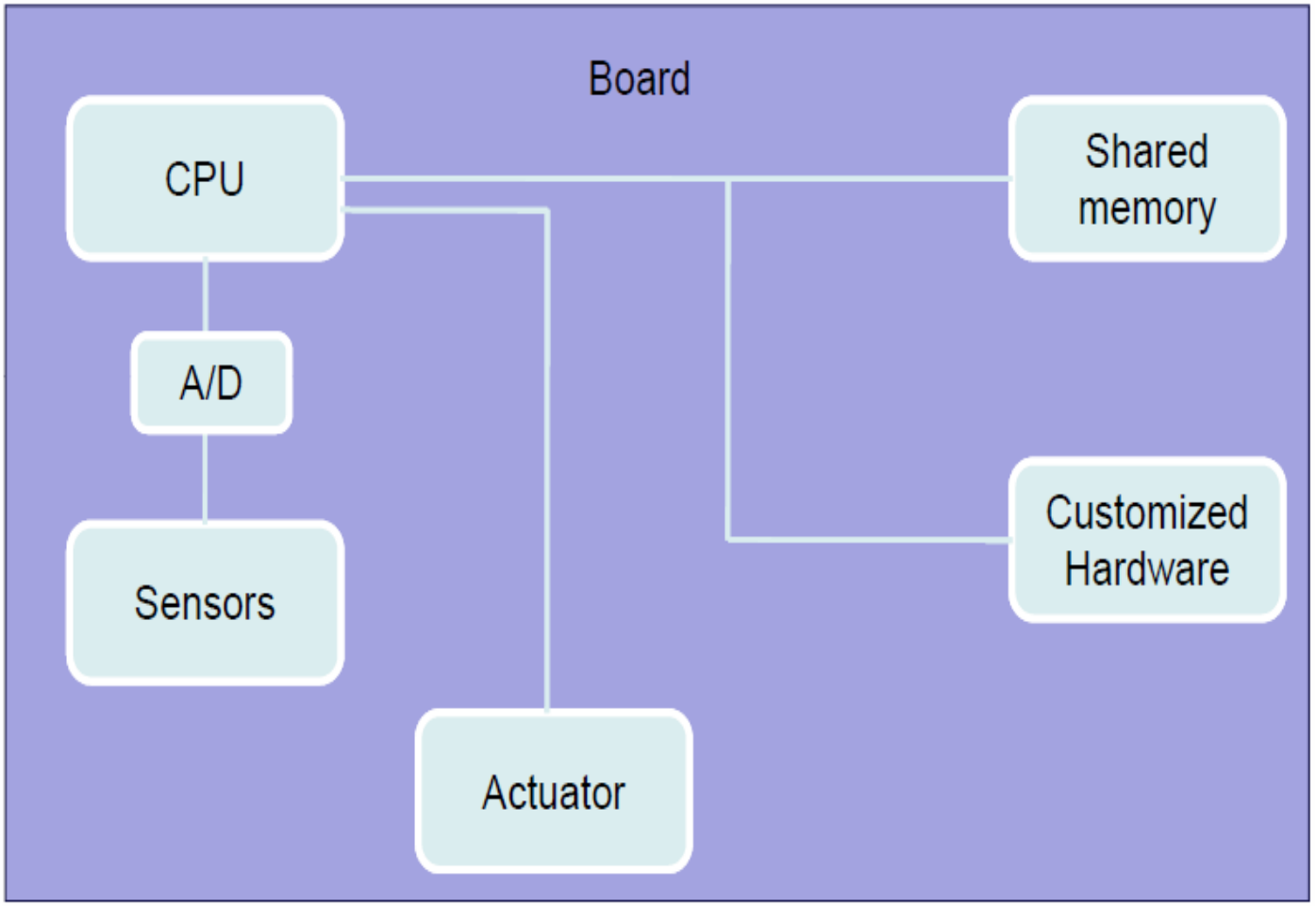
3.Digital Signal Processor



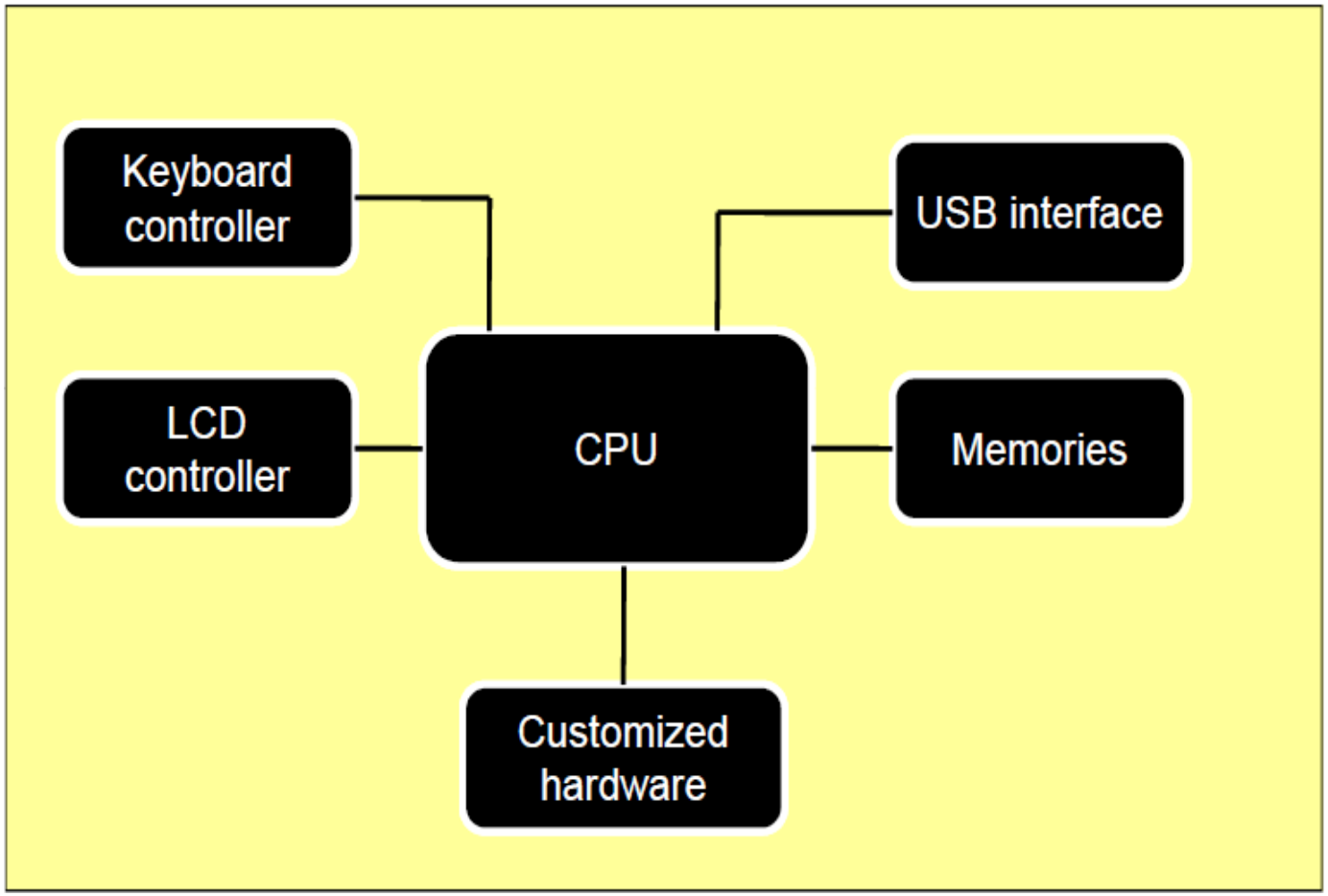
4.Microcontrollers

- Microcontroller = CPU + Memory + Peripherals.
- Microcontrollers are often referred to as single chip devices or single chip computers in a small size that its resources are more limited than those of a desktop personal computer.
- Included memories (RAM, ROM), I/O Ports, buses and peripherals depending on the application designed for (ADC, Timers, USART, I2C , SPI and ..).
- The workhorse of industrial electronics.
- Designed for standalone operation.
- Include a processing unit of 8-bit, 16-bit or 32-bit.

System on Board

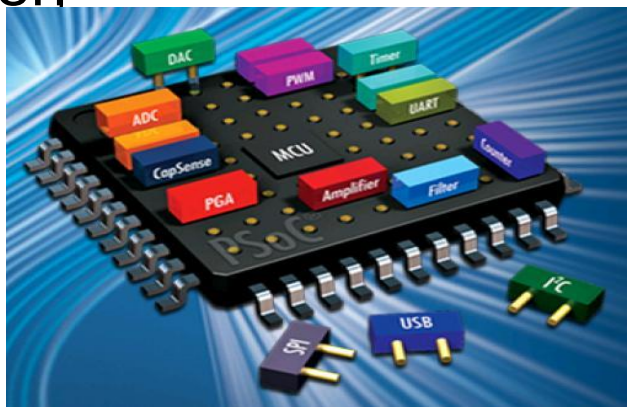


System on chip(SOC)



System on chip(SOC)

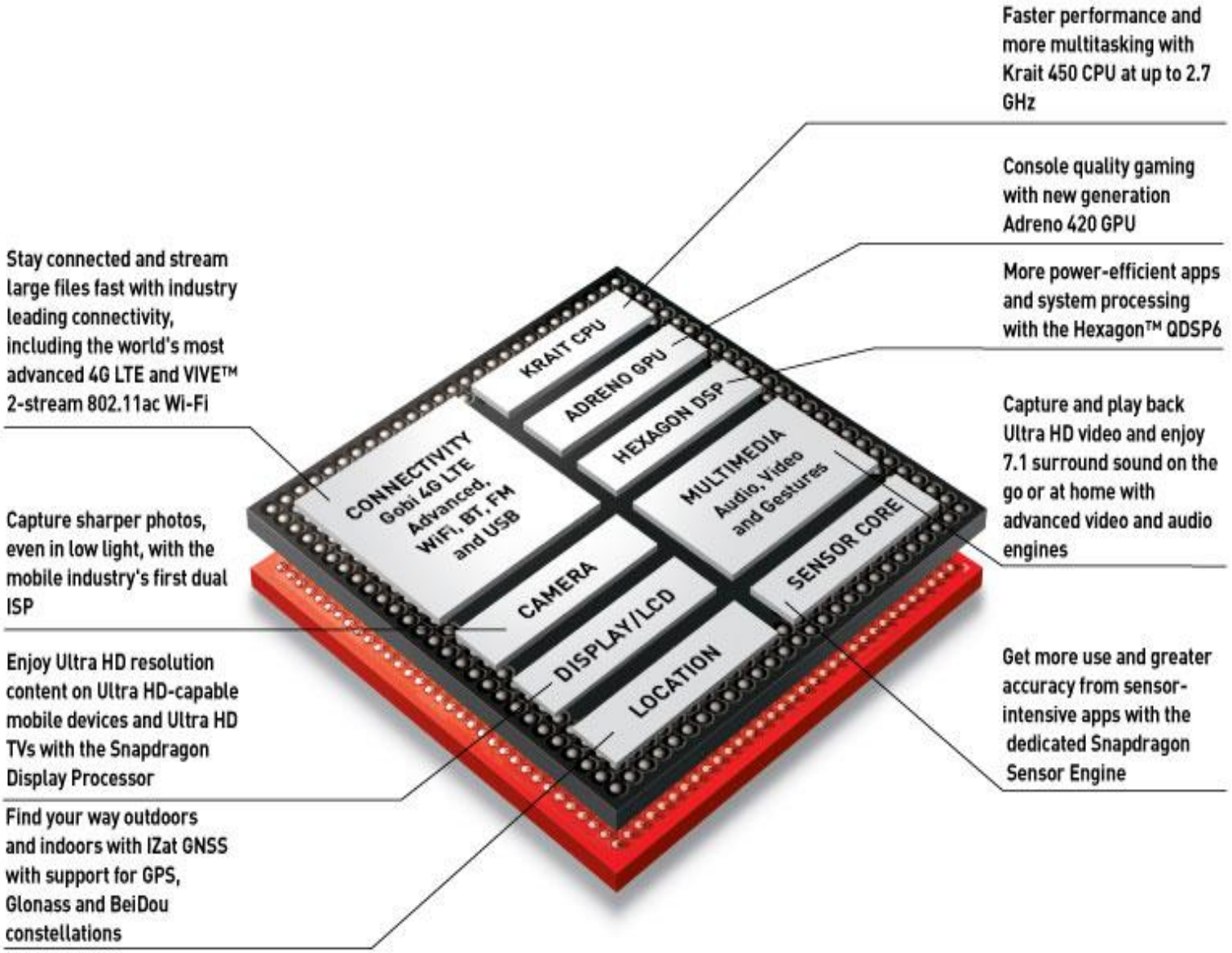
- A SOC has multiple functional units on one piece of silicon, Integrating all components of a computer or other electronic systems into a single integrated circuit (chip).
- Due to advancements in silicon technologies, it is possible to add more functionalities to the processor system on chip like Co-processors, Memories, Internal I/O devices and Interrupt circuit.
- For example, a system-on-a-chip for a sound-detecting device might include an audio receiver, an analog-to-digital converter (ADC), a microprocessor, memory, and the input/output logic control for a user all on a single micro-chip.
- Typically application is in the area of embedded systems.



- **Network of embedded micro-controllers on board**

Many micro-controllers on one / many board(s) communicate together through specific bus protocol like LIN bus and CAN bus in automotive applications.

- **Multi-Core system on chip like mobile handset which has one chip contains:**
  - DSP Processor for graphics and image processing.
  - Embedded Processors like: ARM.
  - Custom hardware for Wi-Fi, Bluetooth and LTE(4G Network).
  - Custom peripherals for board interface:
    - GPS.
    - Touchscreen interface.
    - Memory card interface.

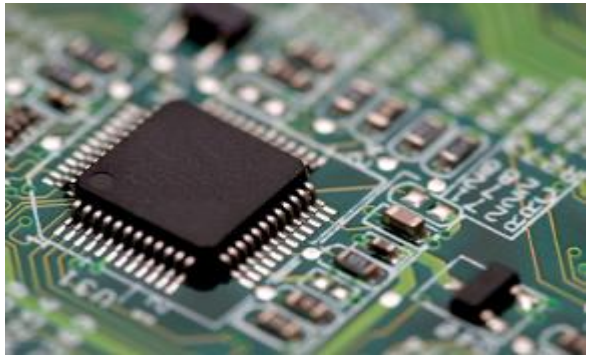


Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- **Embedded HW**
  - Processing Engines.
  - **Micro-processor vs. Micro-controller.**
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

Micro-controller

- Basically a microcontroller can be described as a **computer on a chip**. a single chip containing a CPU, non-volatile memory (ROM), volatile memory(RAM), a timer and an I/O control unit.
- A microcontroller apart from the above mentioned components usually also include serial communication capabilities, interrupt controls and analog I/O capabilities.
- Used for a few **dedicated functions** determined by the system designer.

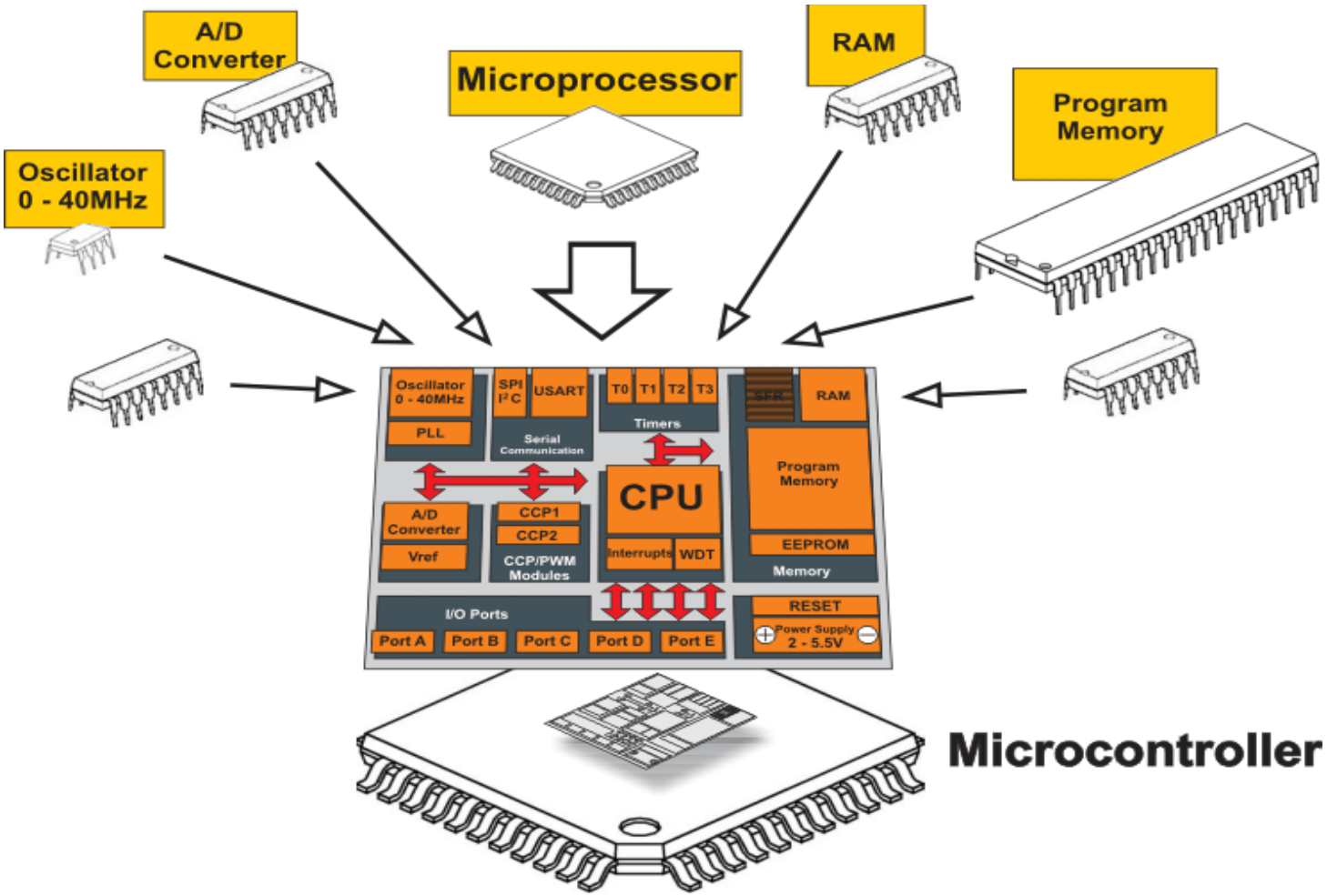


Micro-controller vs. Micro-processor

- Microcontrollers were developed to have almost all needed peripherals in one chip.

- This allowed to decrease the size of embedded system and increased communication speeds between different system components.

Many IC's into one IC



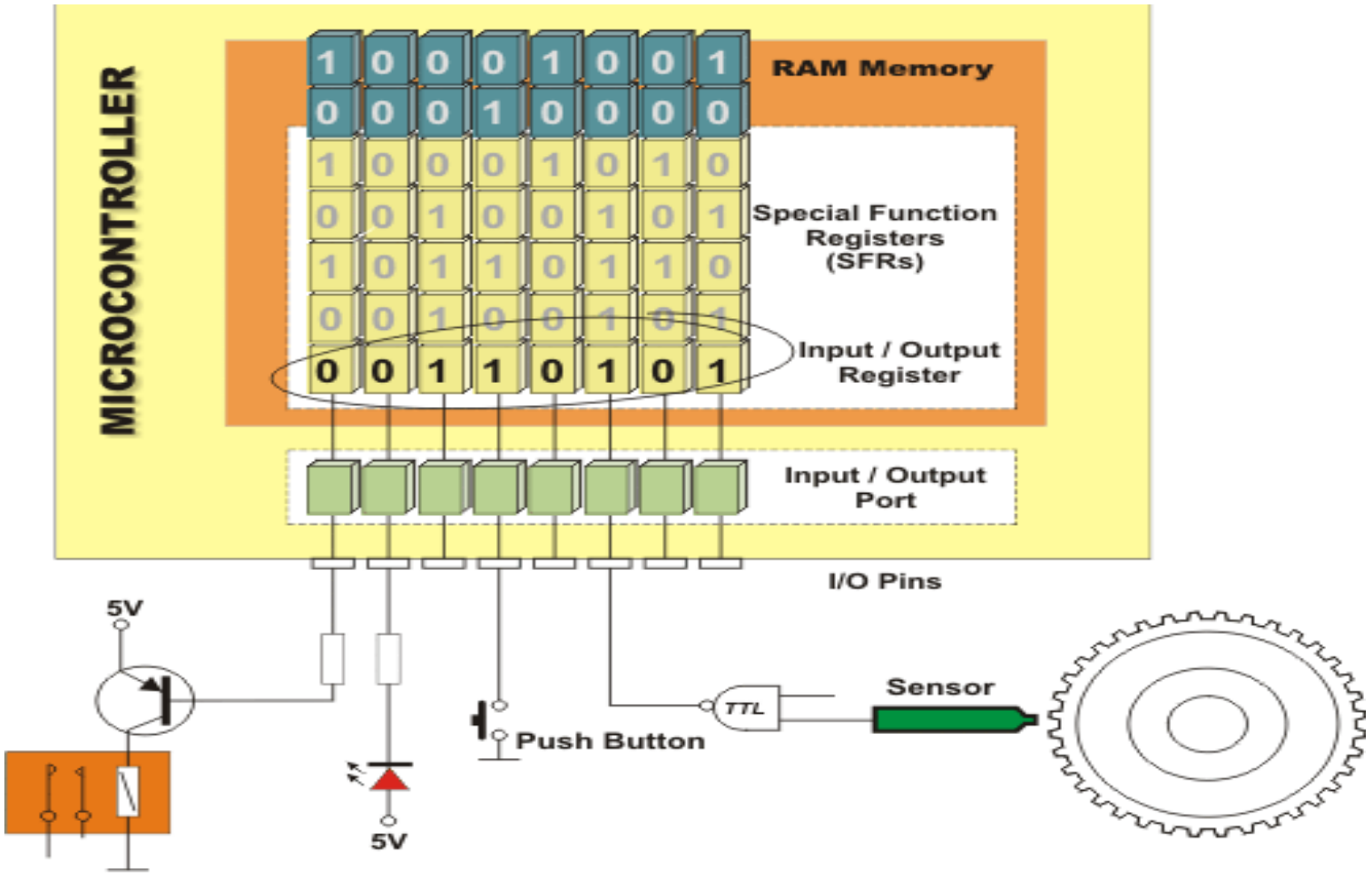
Micro-controller vs. Micro-processor

- Microcontrollers don’t work alone in the circuit it must interfaces with other on chip devices like Sensors, Switches, Leds, LCD, Keypad and DC Motor.

- Microcontroller can accept inputs from some components and provide outputs to other components within any given system.

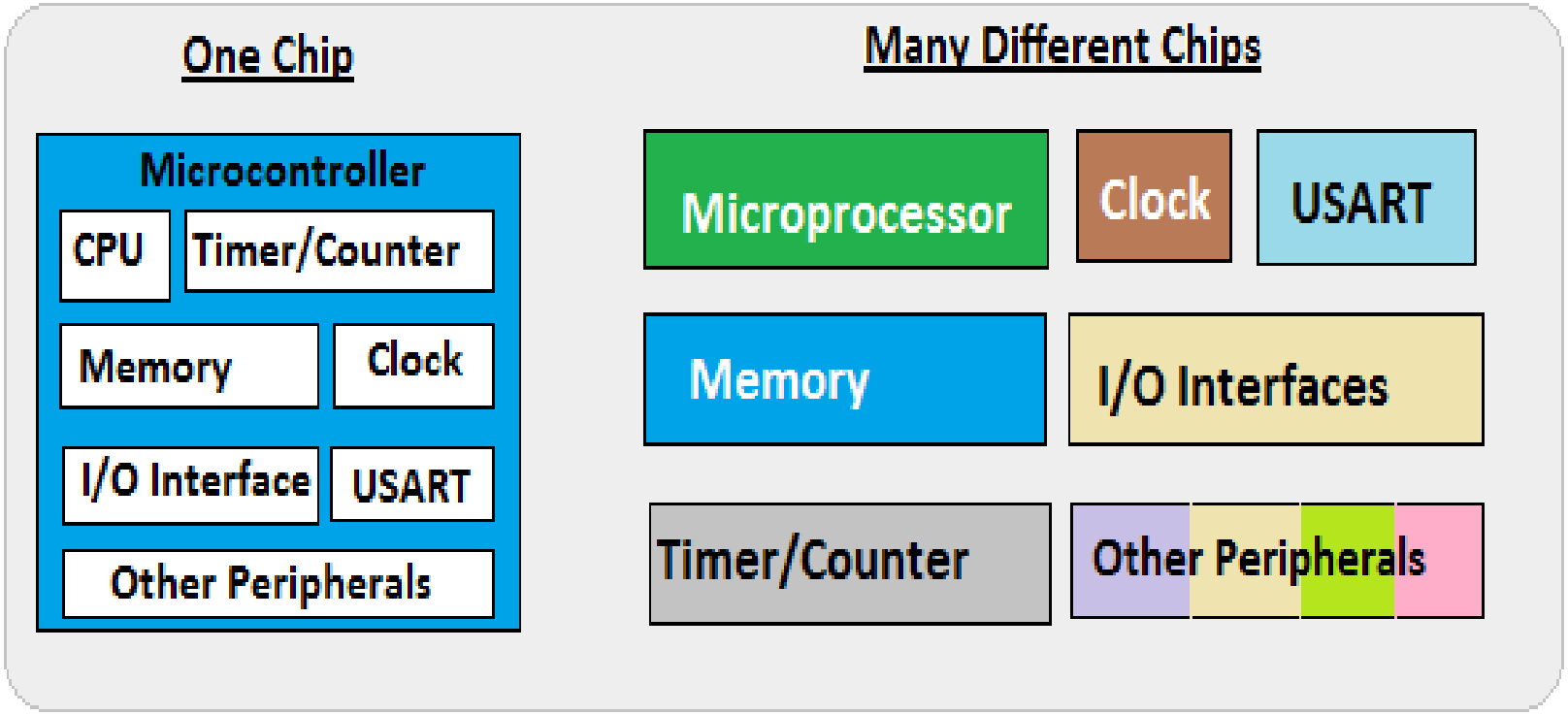
- Differences in requirements, make the manufacturers produce different microcontrollers with different memory sizes, number of I/O lines and number of integrated peripheral devices. Other wise they are all similar to use.

Micro-controller vs. Micro-processor



- **Micro-processor**
  - Just a **CPU** has to add externally memory, clock, input/output interfaces, timer and all other needed peripheral. This is the reason a microprocessor has so many pins.
  - The difference between a microcontroller and a microprocessor is that the microprocessor is a **general purpose** computer while a microcontroller is a computer **dedicated** to one or just a few tasks.

Micro-controller vs. Micro-processor

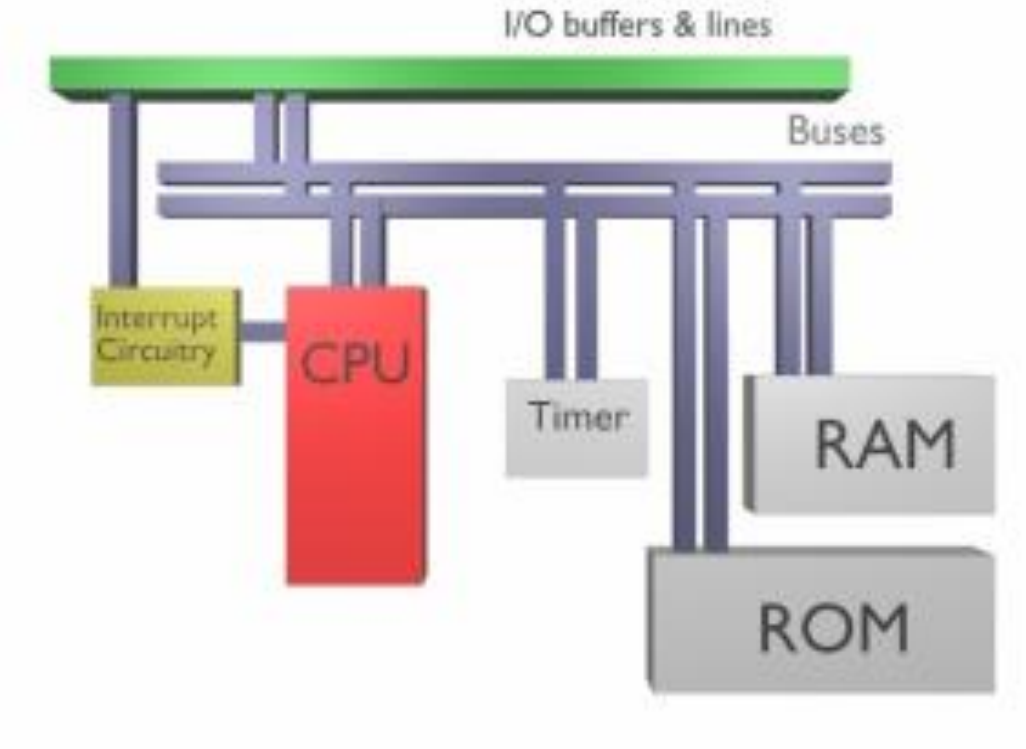


Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- **Embedded HW**
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - **Micro-controller main components.**
  - Micro-controller other components.
- Embedded Systems Constrains.
- Embedded Systems Market.

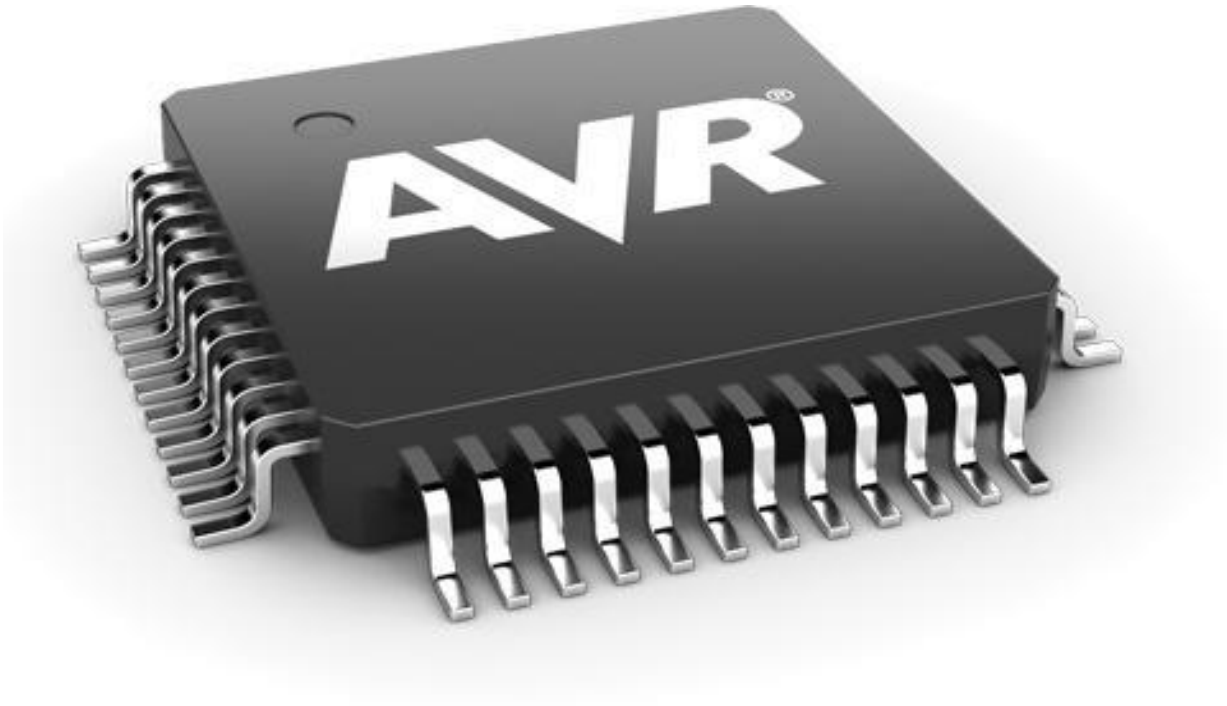
Micro-controller main components

1. Central processing unit (CPU).
2. Memory units.
3. Input and Output ports (GPIO or DIO).
4. Timers.
5. Watch dog timer
6. Buses.
7. Interrupt circuit.



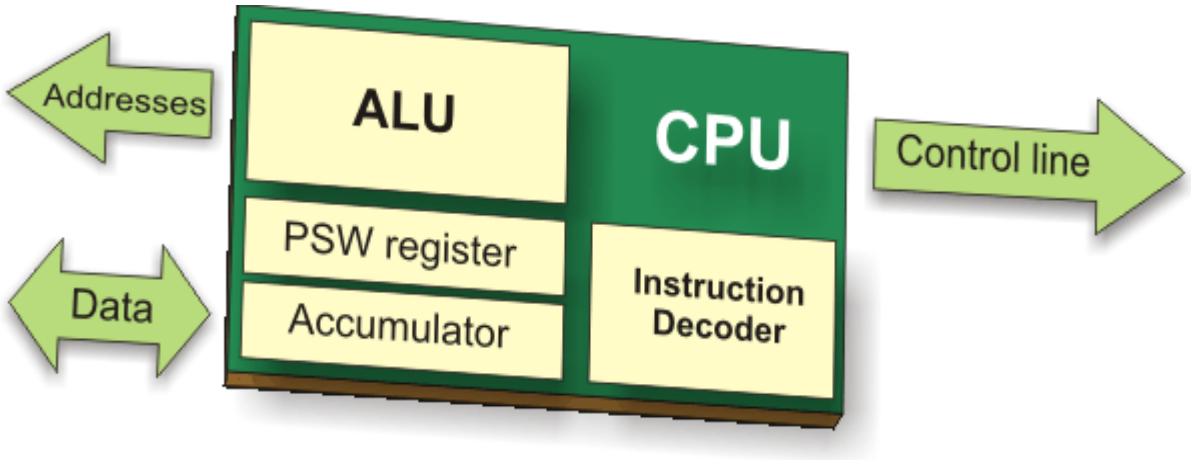
Micro-controller main components

- In the following slides we will try to understand the different components in any microcontroller then map these information to AVR microcontrollers



1.Cental Processing Unit(CPU)

- The **central processing unit** (CPU) does all the computing: it fetches, decodes and executes program instructions and directs the flow of data to and from memory. The CPU performs the calculations required by program instructions and places the results of these calculations, if required into memory space.
- The unit which monitors and controls all processes inside the microcontroller. It consists of several smaller units.



- **Example Assembly Instruction for AVR CPU**

Instruction	OP-Code
LDI Rd, k (0≤k≤255)	1110 – kkkk – dddd – kkkk
LDI R10, 255	1110 – 1111 – 1010 – 1111

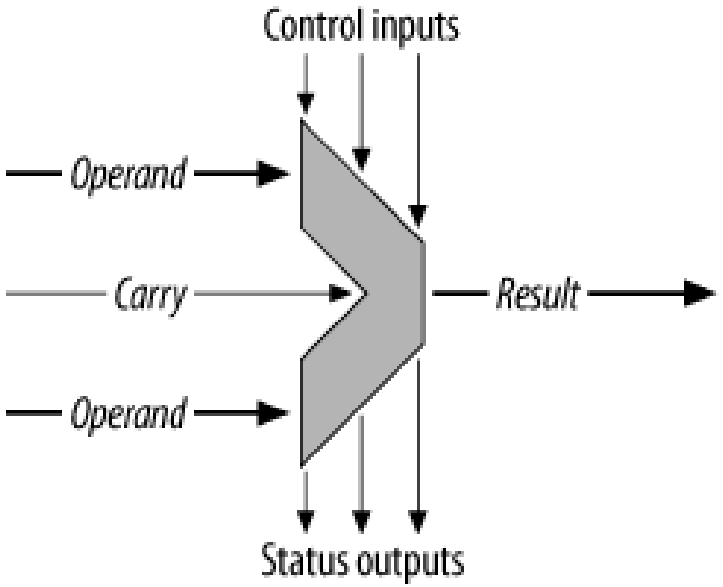
1.Cental Processing Unit(CPU)

- The most important units inside CPU are:
  - Arithmetic Logic Unit(ALU).
  - Instruction Decoder(Control Unit).
  - Accumulator Register.
  - Register File.
  - SFR Registers.

Arithmetic Logic Unit

Part of the CPU is responsible for performing calculations and execution for Arithmetic, Logic and Shift instructions.

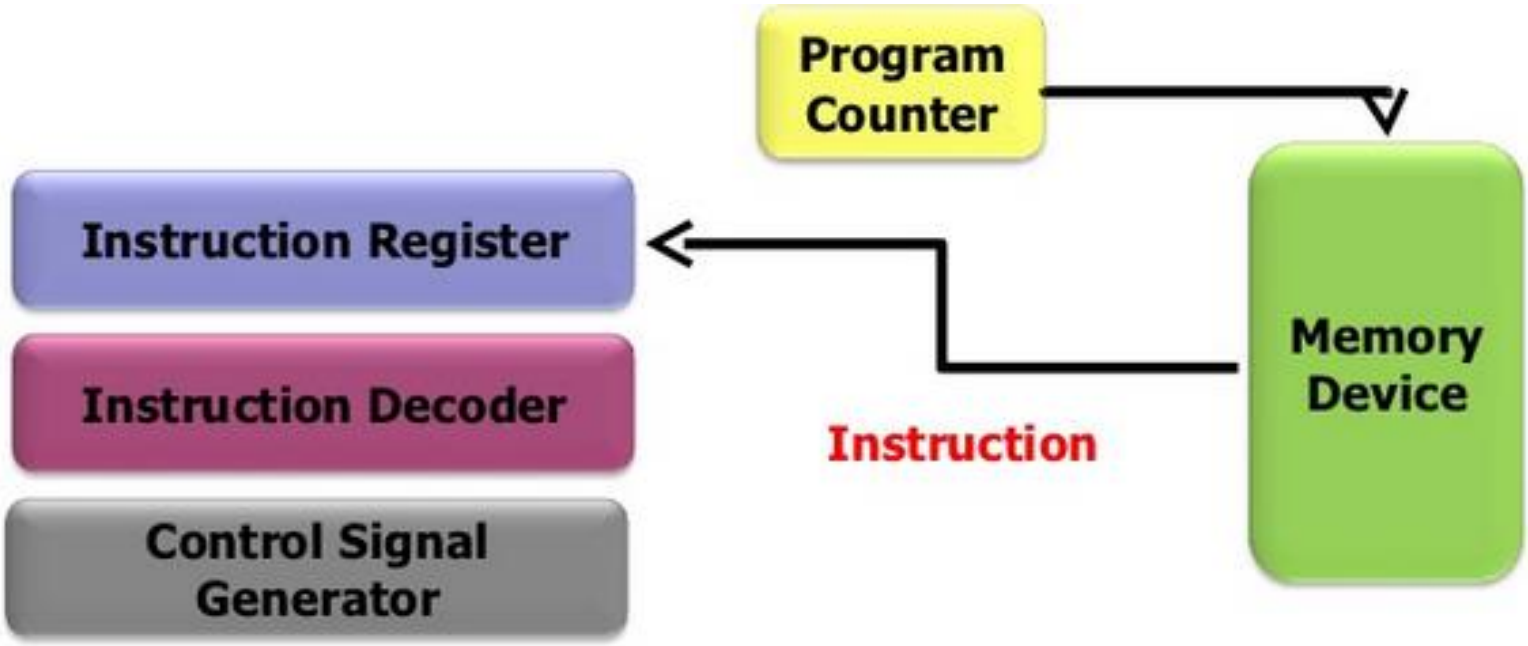
- **Arithmetic instructions include:**
  - Addition
  - Subtraction
  - shifting operations
- **Logic instructions include:**
  - AND
  - OR
  - XOR
  - NOT operations



Instruction Decoder(Control Unit)

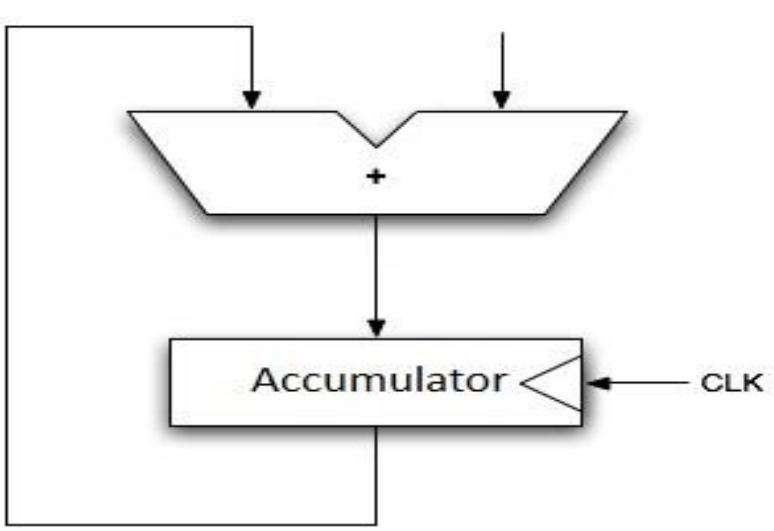
- Converts instructions stored or fetched in program memory into codes which the ALU can understand and generate the control signals of this instruction.
- The control unit is the circuitry that controls the flow of data through the processor, and coordinates the activities of the other units within it.
- In other word it is **in charge of the entire Instruction (Machine) cycle**
- The **“instruction set”** differs from microcontroller family to another expresses the abilities of this circuit.

- **Instruction Decoder(Control Unit)**



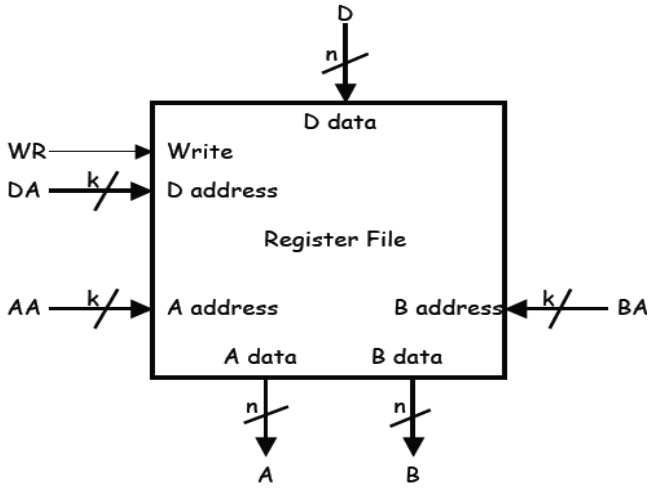
- **Accumulator Register**

Is an SFR closely related to the operation of the ALU. It is a kind of working desk used for storing all data upon which some operations should be performed (addition , shift, Subtraction, etc.).



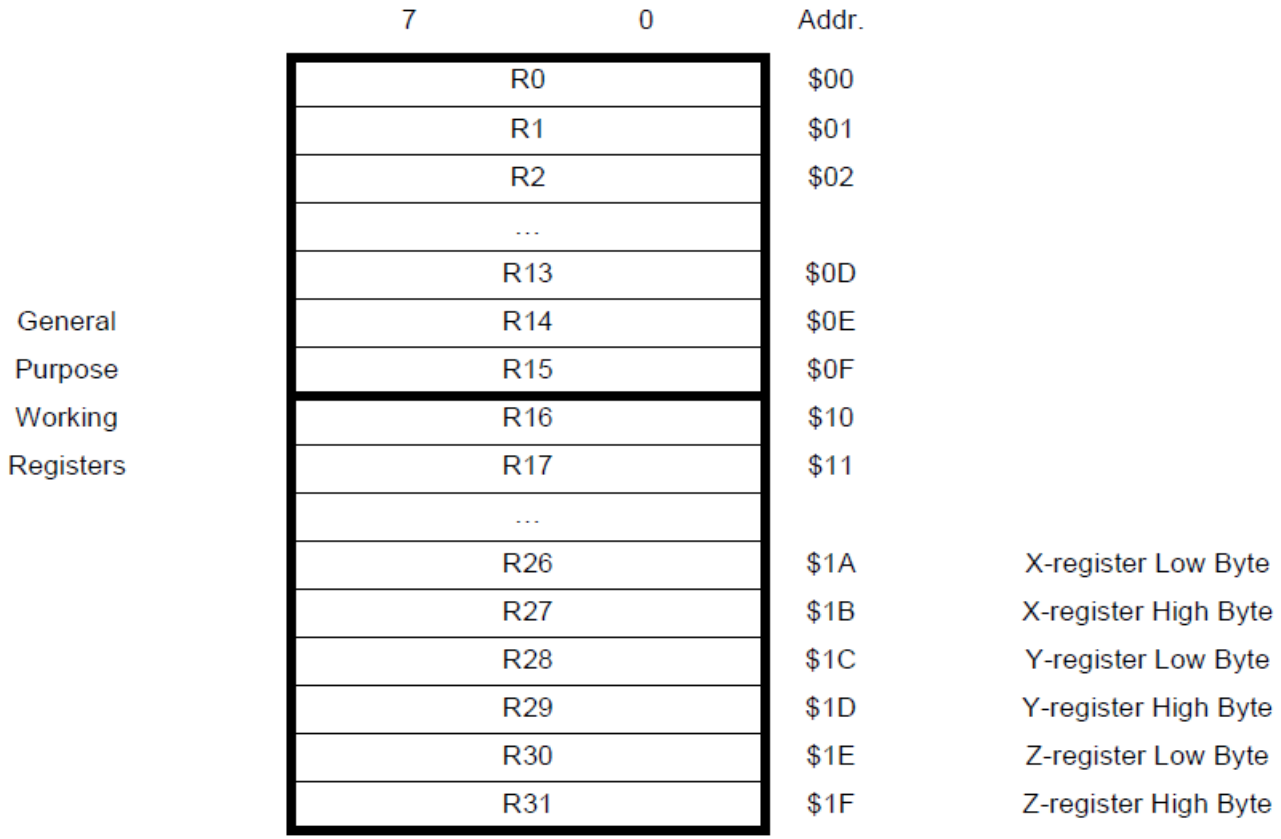
Register File

- A number of registers grouped together.
- n data bits and k address bits.
- We can read two registers at once by supplying AA, BA and WR = 0.
- We can write at a register using the DA and D inputs and setting WR = 1



Register File in AVR CPU

- AVR CPU contains 32 registers R0 → R31.



- ## SFR Registers

The numbers and names of registers vary drastically among microcontrollers. However there are certain registers which are common to most of microcontrollers, although the names may vary.

1. Instruction Register(IR)

- Register which contains the instruction after getting it from program memory (Fetch stage).
- The ATmega16 contains 16 Kbytes On-chip In-System Reprogrammable Flash EEPROM memory for program storage. Since Most AVR instructions are 16 bits wide, the Flash is organized as 8K × 16. so The ATmega16 Instruction Register is 16 bits wide.

2. Program Counter(PC)

- Perhaps the most important CPU register is the program counter(PC).
- The PC holds the address of the next instruction in program memory space, which the CPU will process. As each instruction is fetched and processed by the ALU, the CPU increments the PC and there by steps Through the program stored in the program memory space.
- The ATmega16 contains 16 Kbytes On-chip In-System Reprogrammable Flash EEPROM memory for program storage. Since most AVR instructions are 16 wide, the Flash is organized as 8K × 16. so The ATmega16 Program Counter (PC) is 13 bits.

3. Stack Pointer(SP)

- The **stack pointer** is basically a register that holds either "the memory address of the last location on that stack where data was stored" or "the memory address of the next available location on the stack to store data.“ last case is more common in CPU architectures.
- The Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer and a Stack POP command increases the Stack Pointer(and may be vice versa for some processors).

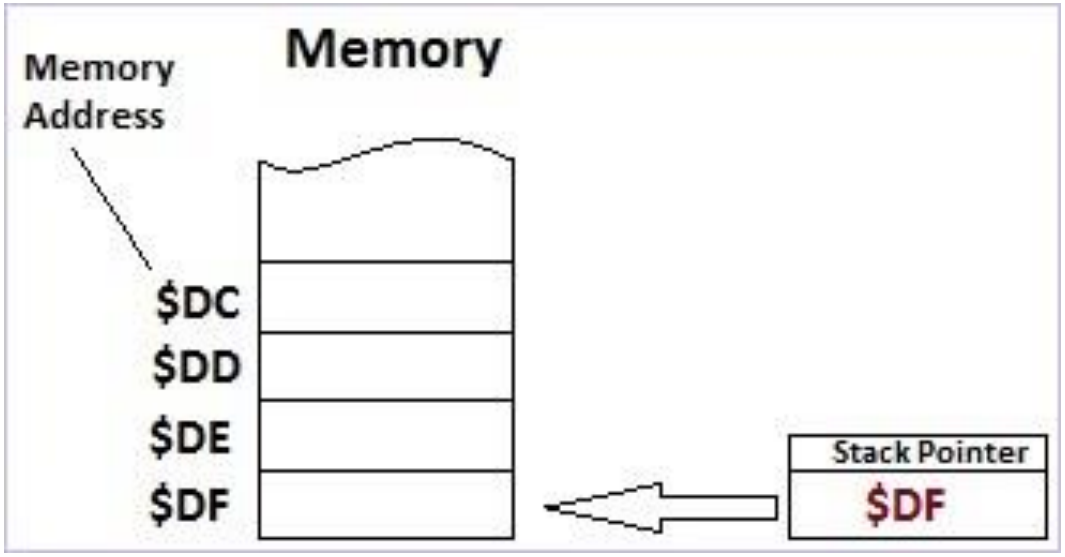
Stack

- The Stack is mainly used for storing temporary data, for storing **local variables** and for storing **return addresses** after **interrupts** and **subroutine** calls.
- Store data in stack using **PUSH** and **POP** operators.
- A stack is a **LIFO** (last in, first out) mechanism the last thing store on the stack is the first thing to be retrieved from the stack.

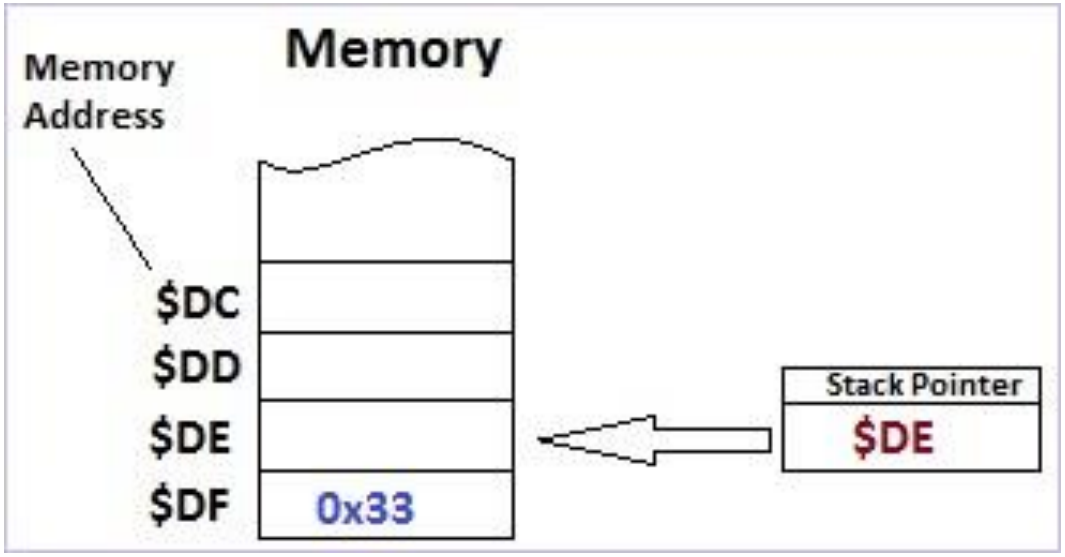
- **Stack Pointer Register in AVR CPU**
  - The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent on SRAM Data memory size. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.
  - SRAM in ATmega16 is 1KB it divided to 1K x 8 so the Stack Pointer is 10 bits wide.

15	14	13	12	11	10	9	8	
SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
7	6	5	4	3	2	1	0	

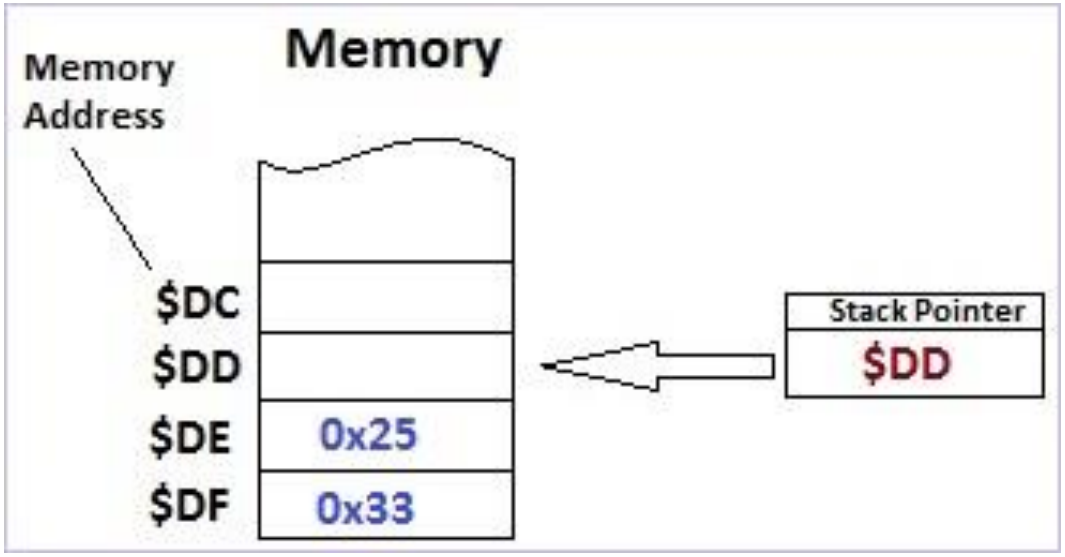
- Stack Pointer have to be initialized to point to the last memory location in SRAM (i.e. first empty location in the stack).



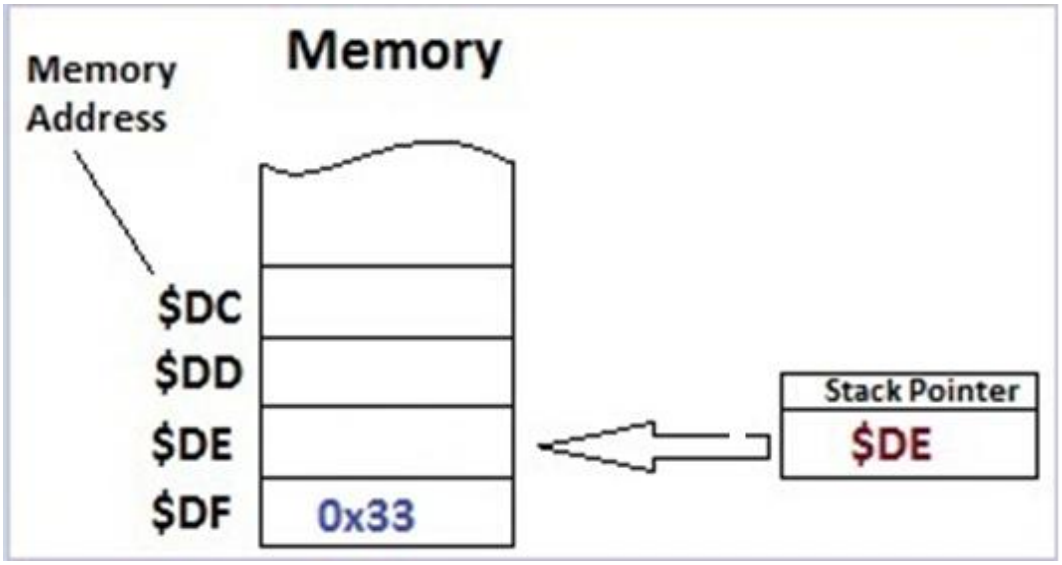
- Stack after first PUSH



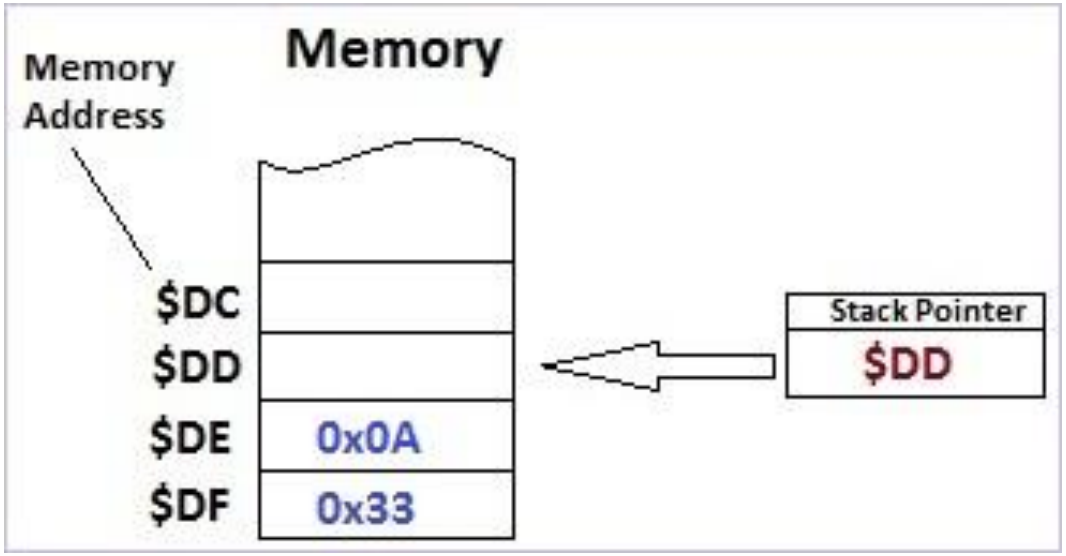
- Stack after Second PUSH



- Stack after First POP.

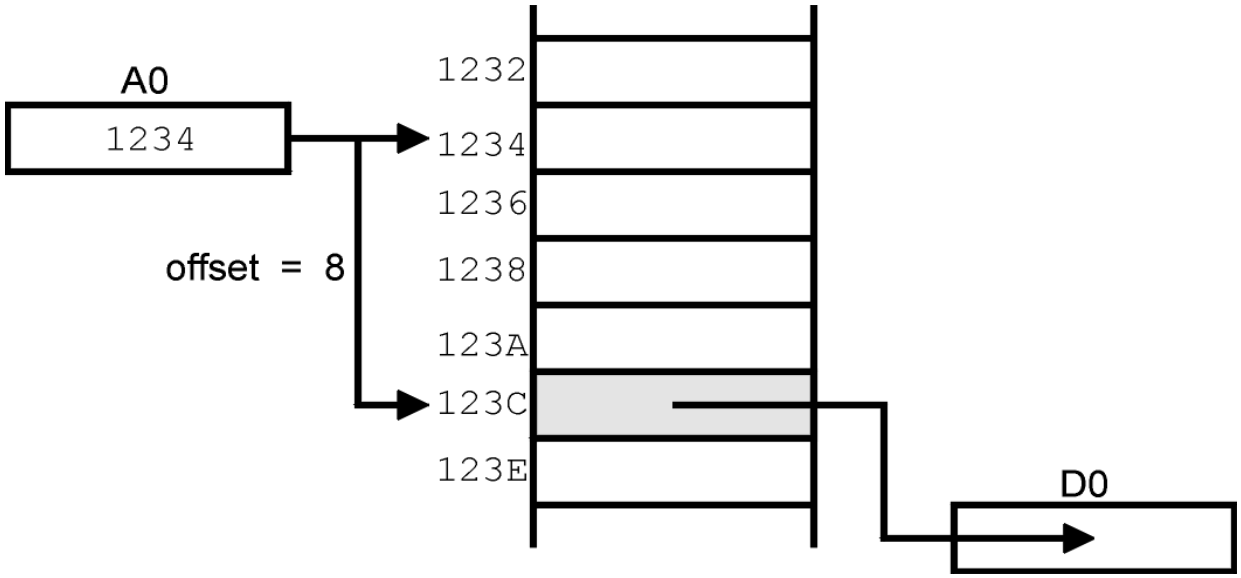


- Stack after Third PUSH.



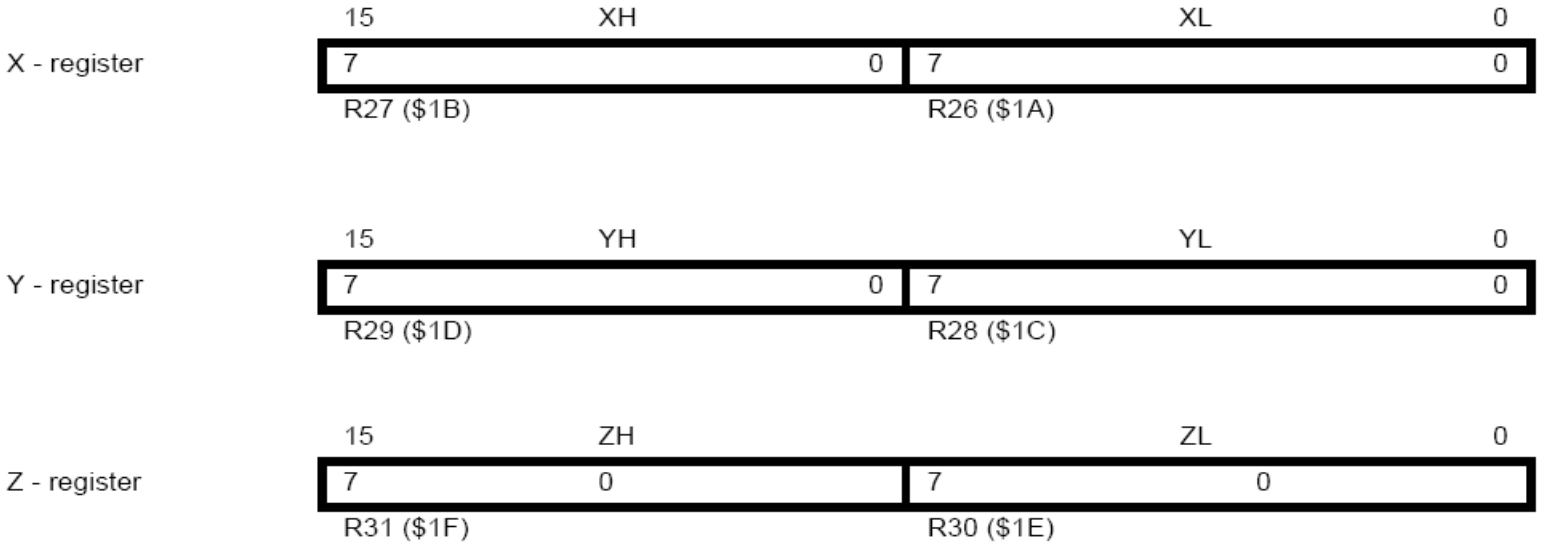
4. Index Register

- Used to specify an address when certain addressing modes are used, And Known as pointer register like X,Y and Z registers in AVR CPU.
- Also called **Base** Register or **Pointer** Register.
- Used in indirect addressing modes, as the accessed address = index register + offset mentioned in code.



X,Y and Z Index Registers in AVR CPU

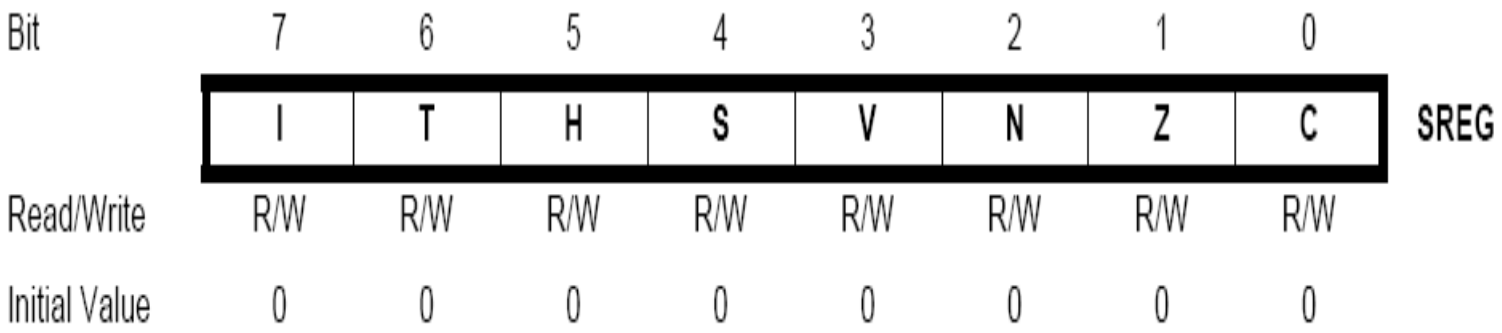
- The R26..R31 registers have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space.



5. Processor Status Word/Register(PSW)

- The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations.
- Status Register is updated after ALU operations.

Status Register in AVR CPU



- Flags as:
  - **Bit 0 –C: Carry flag**
    indicates a carry in an arithmetic or logic operation.
  - **Bit 1 –Z : Zero flag**
    indicates a zero result in an arithmetic or logic operation.
  - **Bit 2 –N: Negative flag**
    indicates a negative result in an arithmetic or logic operation.
  - **Bit 3 –V : Overflow flag**
    indicates an overflow occurs in an arithmetic operation.

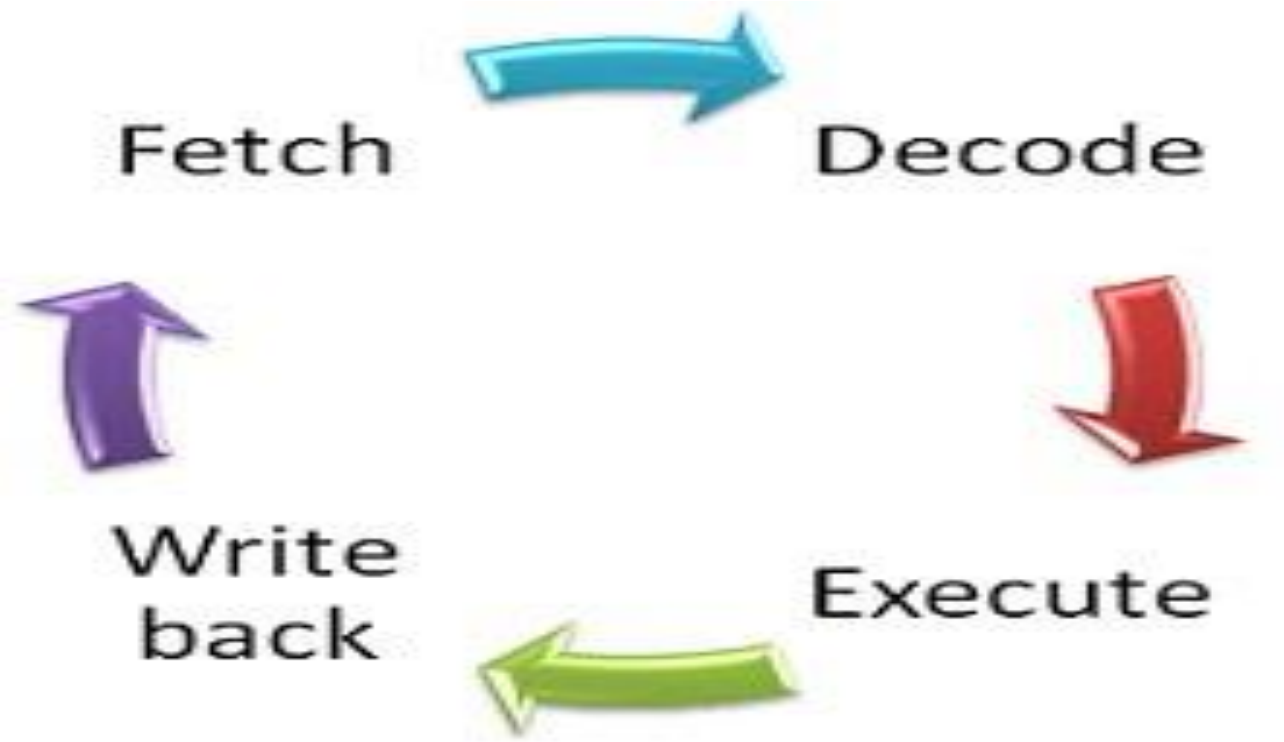
- **Bit 4 –S: Sign Flag**

  indicates the result sing of arithmetic operation.
- **Bit 5 –H: Half-carry flag**

  indicates a Half Carry in the least significant four bits in arithmetic operations.
- **Bit 6 –T: Bit Copy Storage**

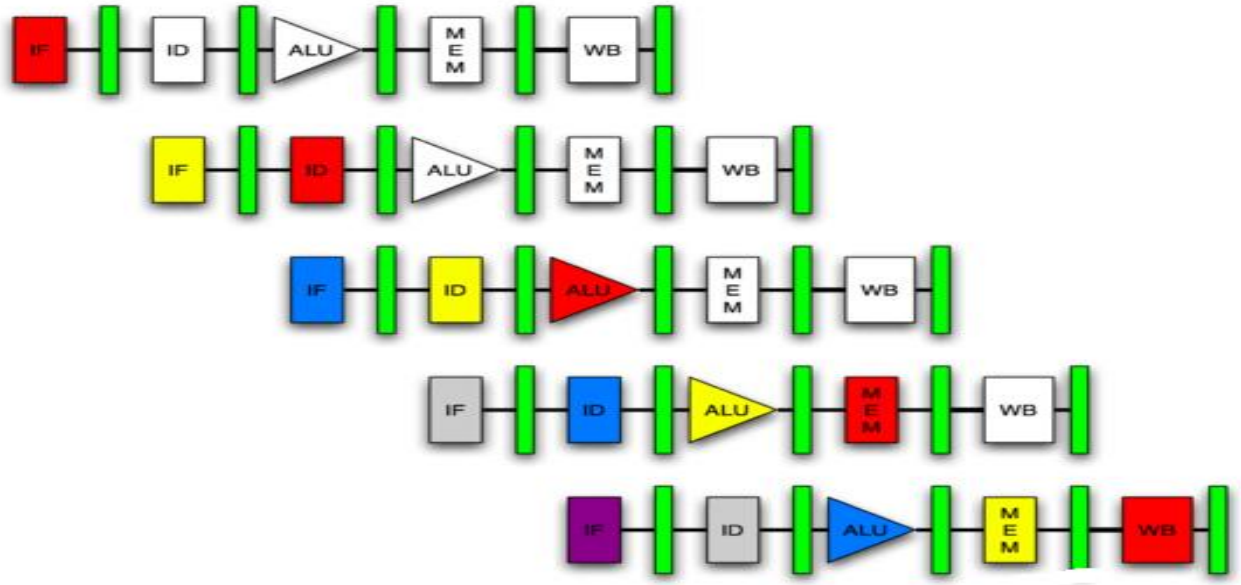
  The Bit Copy instructions **BLD** (Bit Load) and **BST** (Bit store) use the T-bit as source or destination for the operated bit. A bit from a register in the Register file can be copied into T by the **BST** instruction, and a bit in T can be copied into a bit in a register in the Register file by the **BLD** instruction.
- **Bit 7 –I: Global Interrupt Enable: Global Interrupt mask bit**

  must be set for the interrupts to be enabled.

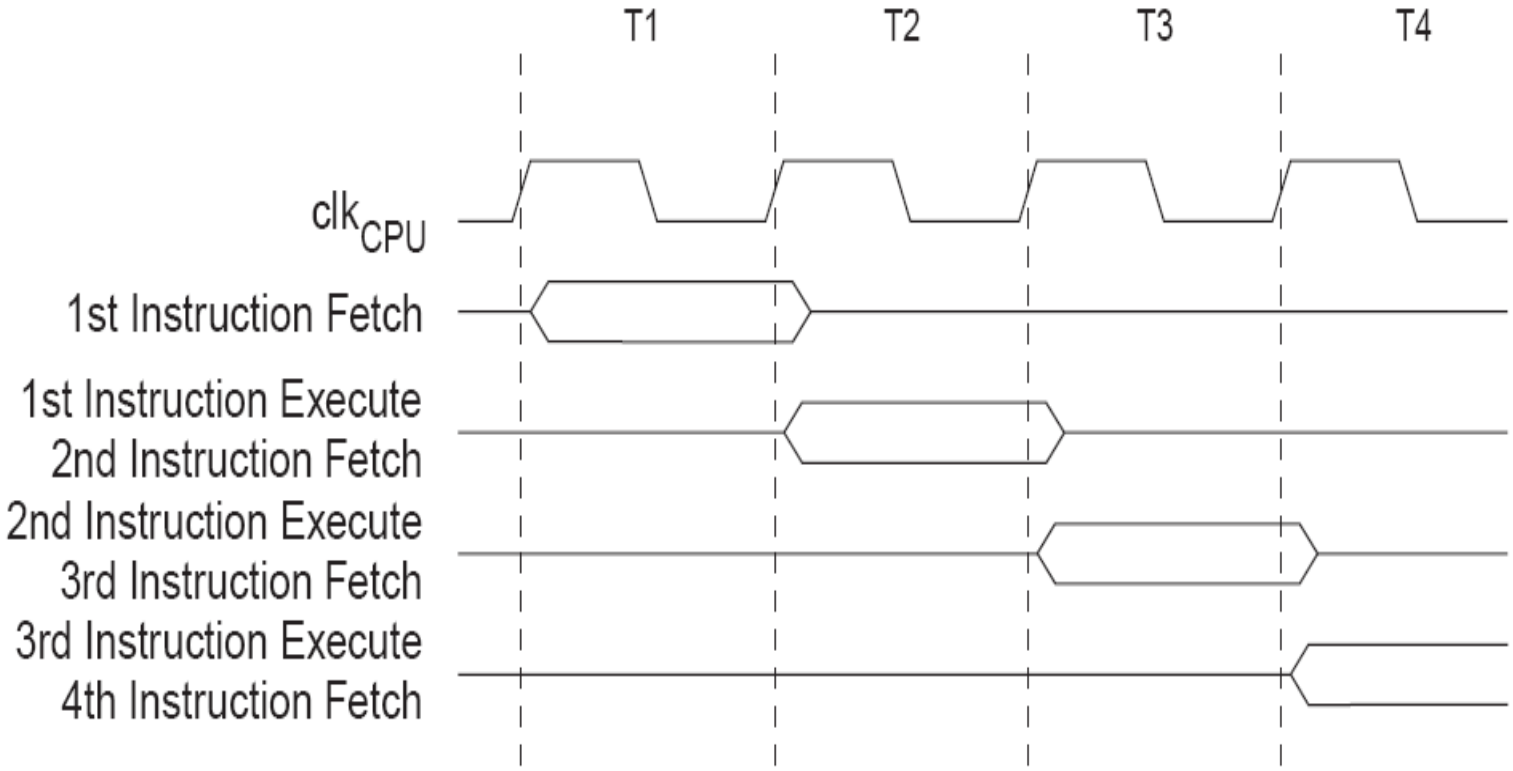


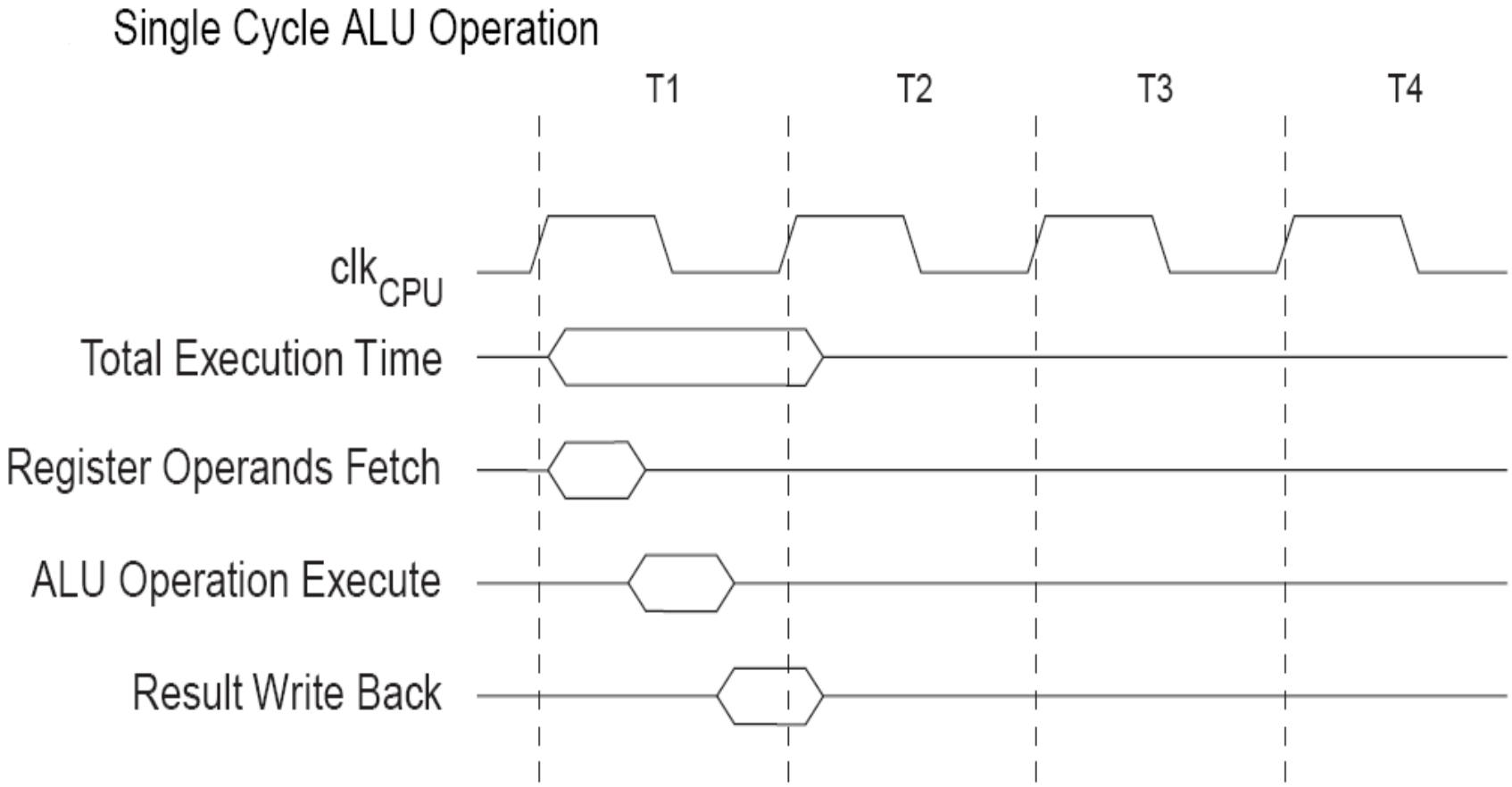
Instruction life cycle

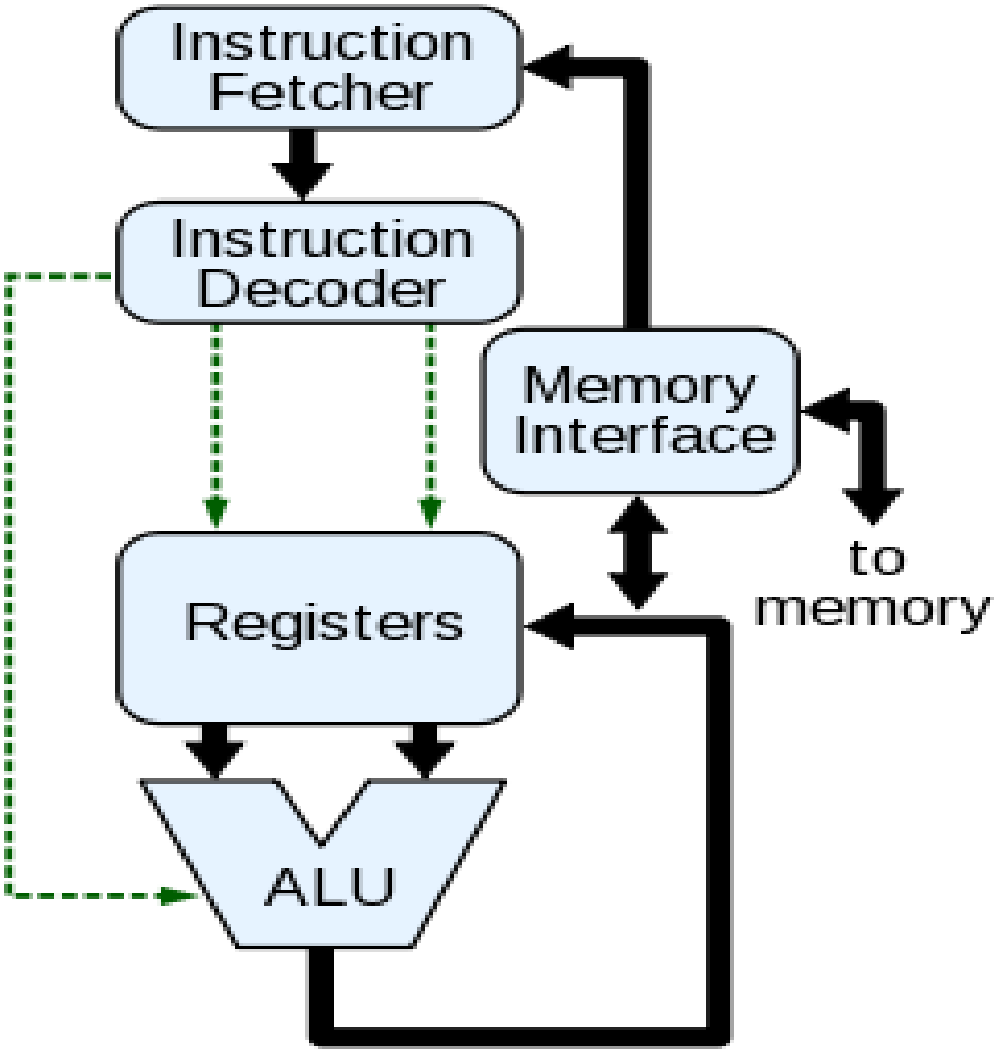
- **Instruction Pipelining:**
  - Continuous and parallel streaming of instruction to the CPU.
  - A method of achieving higher execution speed at same clock speed.



- **The Parallel Instruction Fetches and Instruction Executions in AVR CPU**







Simple CPU Architecture

CPU Architecture(Memory accessing):

- There are two basic types of architecture:
  - Von-Neumann Architecture.
  - Harvard Architecture.
- Microcontrollers most often use a **Harvard** Architecture or a modified Harvard-based architecture.

Von-Neumann Architecture

- Von-Neumann architecture has a single and common memory space where both program instructions and data are stored.
- There is a single data bus which fetches both instructions and data.

Harvard Architecture

- Harvard architecture computers have separate memory areas for program instructions and data.
- One bus connects CPU to RAM memory(Data Memory). The other bus connects CPU to ROM memory(Program Memory).

Von-Neumann Architecture

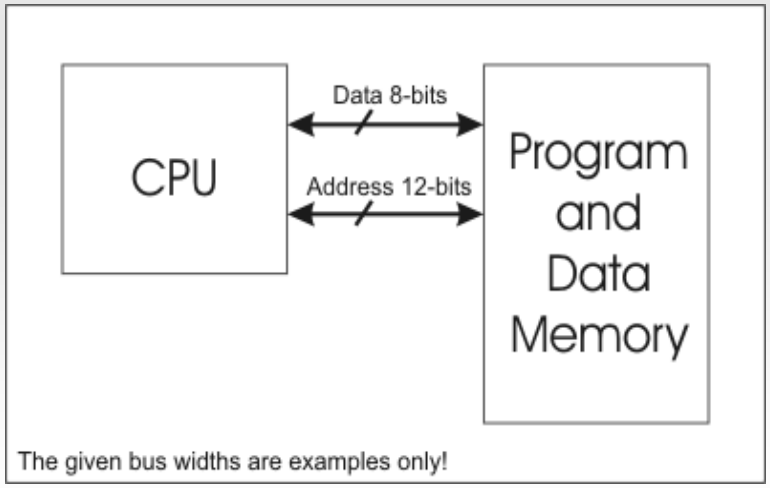
- Each time the CPU fetches a program instruction it may have to perform one or more read/write operations to data memory space, it must wait until these subsequent operations are completed before it can fetch and decode the next program instructions.

Harvard Architecture

- The CPU can read an instruction and perform a data memory access at the same time.
- This speeds up execution time but increases the cost of more hardware complexity.

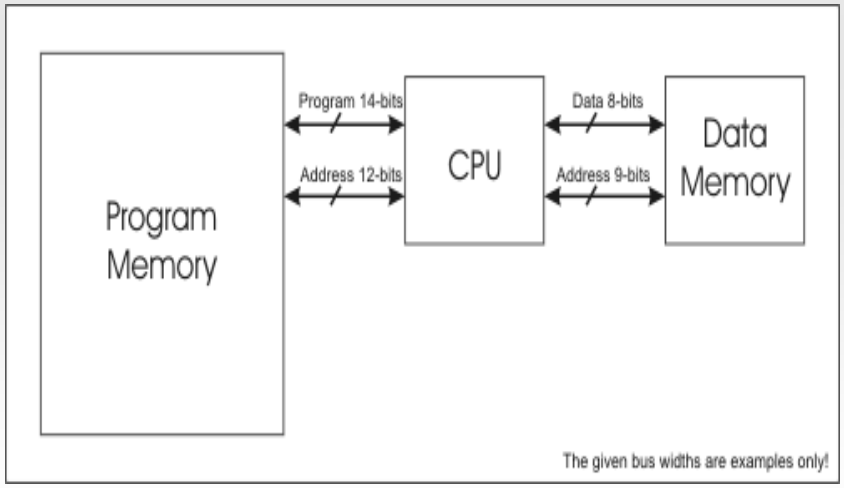
Von-Neumann Architecture

- The advantages of this architecture lies in it's simplicity and economy (only 1 memory + few number of wires).



Harvard Architecture

- The advantages of this architecture lies in its speed. But with more HW, cost and complexity.



CPU Architecture(Number of instructions):

- There are two basic types of architecture:
  - Reduced Instruction Set Computing(RISC).
  - Complex Instruction Set Computing(CISC).
- Microcontrollers most often use a **RISC** Architecture.

RISC Architecture

- CPU design that recognizes only a limited number of instructions, Simple in design.
- Simple instructions.
- Instructions are executed quickly.
- Only the load and store instructions operate directly onto memory.

CISC Architecture

- CPU design that recognizes a large number of instructions, Complex in design.
- Complex and Large no. of instructions.
- Large execution time for each instruction.
- Instructions can operate directly on memory.

RISC Architecture

- RISC aims to optimize execution of instructions by limiting the capabilities of a single instruction and having consistent instructions' execution time (Instruction Pipelining)
- thus gaining speed from execution point of view but code size will be large for a dedicated function. But it will increase the number of memory access.

CISC Architecture

- CISC is an old concepts that dates back when memory access was slow.
- CISC aimed to integrate several functionalities in one instruction, in order to limit the program size. and thus limit memory access in order to gain some speed.

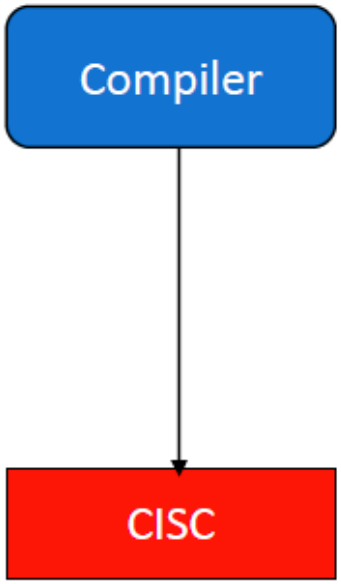
RISC Architecture

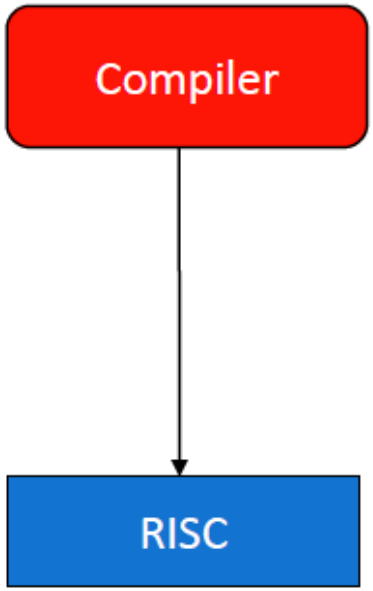
- So It's harder to write powerful optimized compilers, since large number of instructions needed.
- Combine a large number of general registers for arithmetic operations to avoid storing variables on a stack in memory.
- Example: ARM, AVR, PIC,POWER-PC and SPARC.

CISC Architecture

- It's easier to write powerful optimized compilers, since fewer instructions exist.
- Small number of general purpose registers.
- Example: Intel X86, AMD and Motorola 68000.

1.Cental Processing Unit(CPU)





More complex

Less complex

- **Consider this example:**

This **CISC** assembly code

```
clear 0x1000  ;clear memory location 0x1000
load r1,#5       ;load register 1 with the value 5
```

Becomes the following **RISC** assembly code

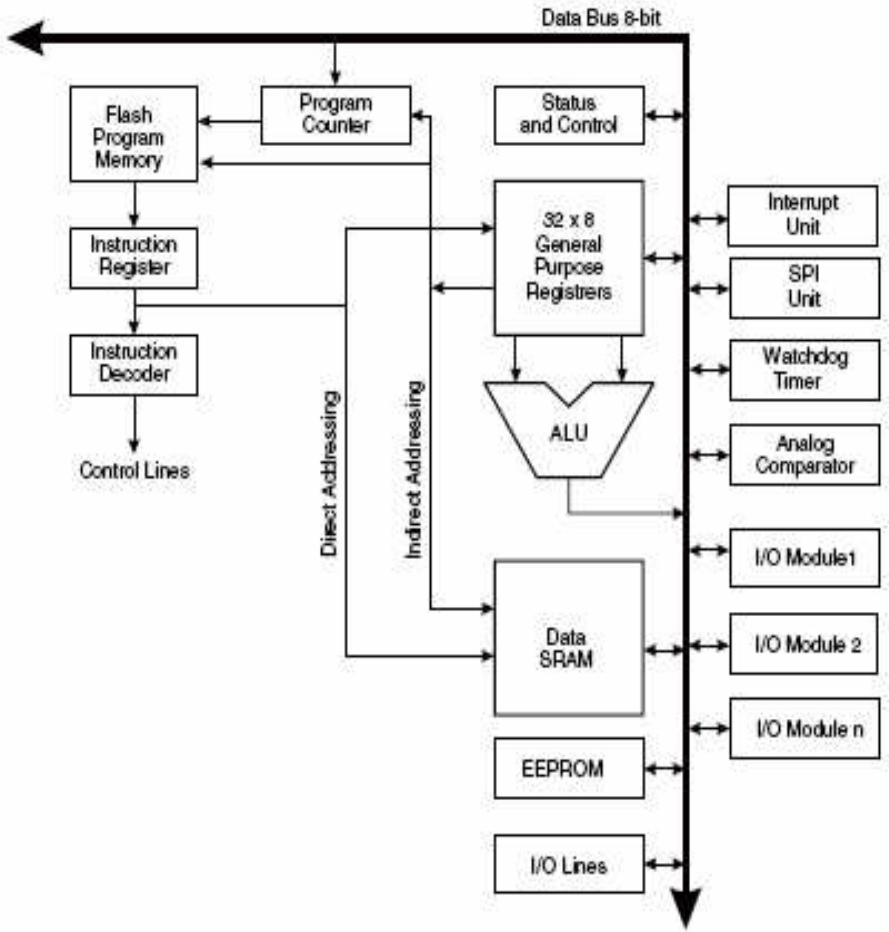
```
xor   r1,r1      ; clear register 1
store r1,0x1000  ; clear memory location 0x1000
load r1,#5       ; load register 1 with the value 5
```

1.Cental Processing Unit(CPU)

- Consider multiplying two numbers, then adding the same two numbers. A CISC machine will get the two values from memory, perform the multiplication, then store the result. It would then have to re-retrieve the same two values from memory and perform the addition and store the result. A RISC machine would pull the two values from memory, perform the multiplication, then store the result, but an optimizer would see that the values are already local to the processor and would skip re-retrieving these values from memory thus saving a number of internal clock cycles, and the more expensive memory access cycles.

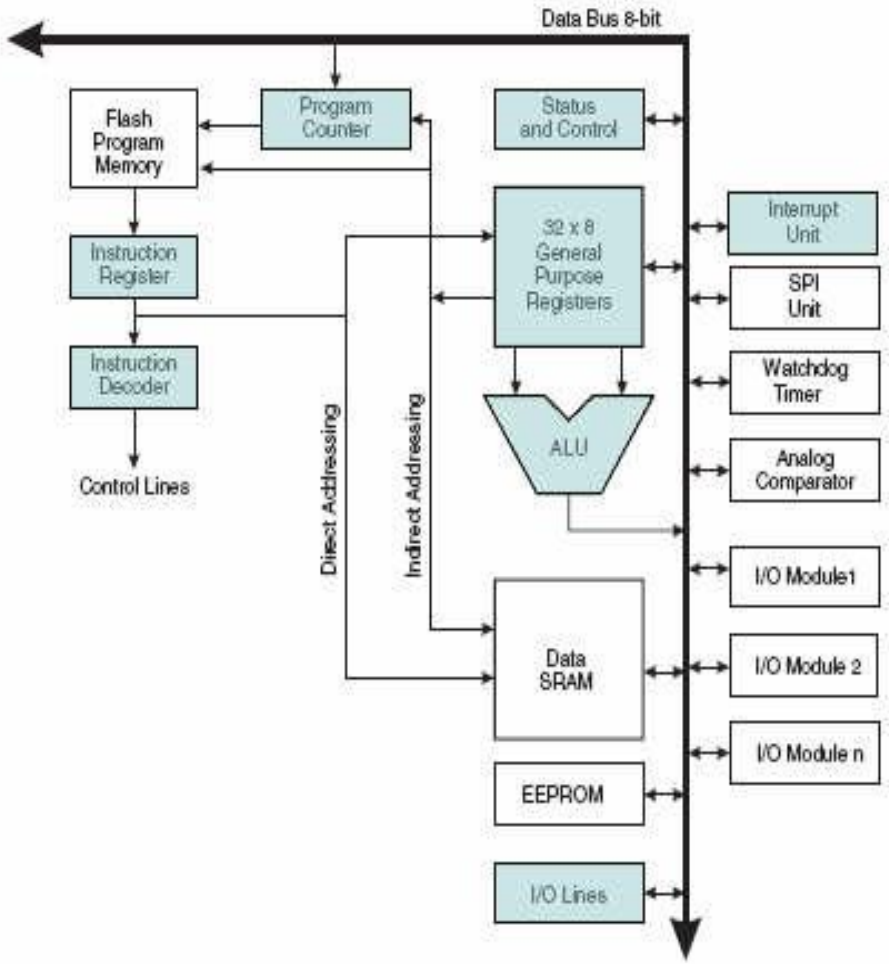
Block diagram of the AVR Core architecture.

- Note the:
  - Different memories
  - Special Purpose Registers
  - General Purpose Registers
  - Special Purpose “units”
  - ALU



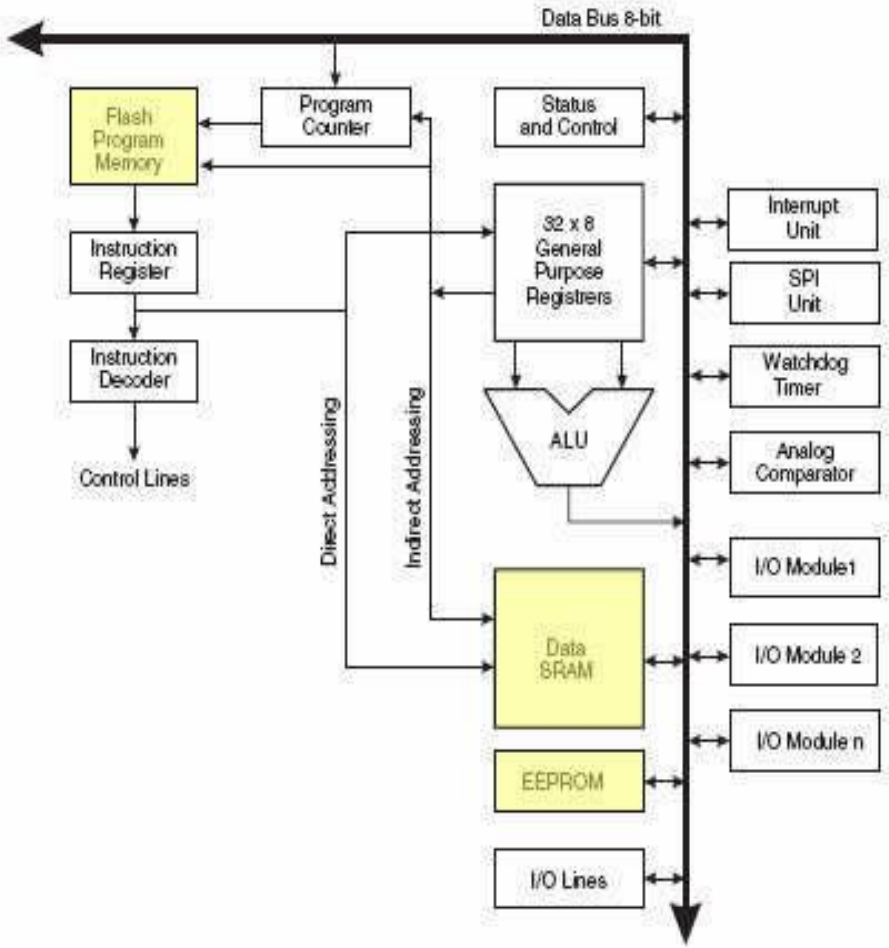
1.Cental Processing Unit(CPU)

- The blocks shown in blue are typical of a microprocessor. The other blocks, e.g. memories and special purpose units are what defines the difference between a microprocessor and a microcontroller.
- The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals and handle interrupts.



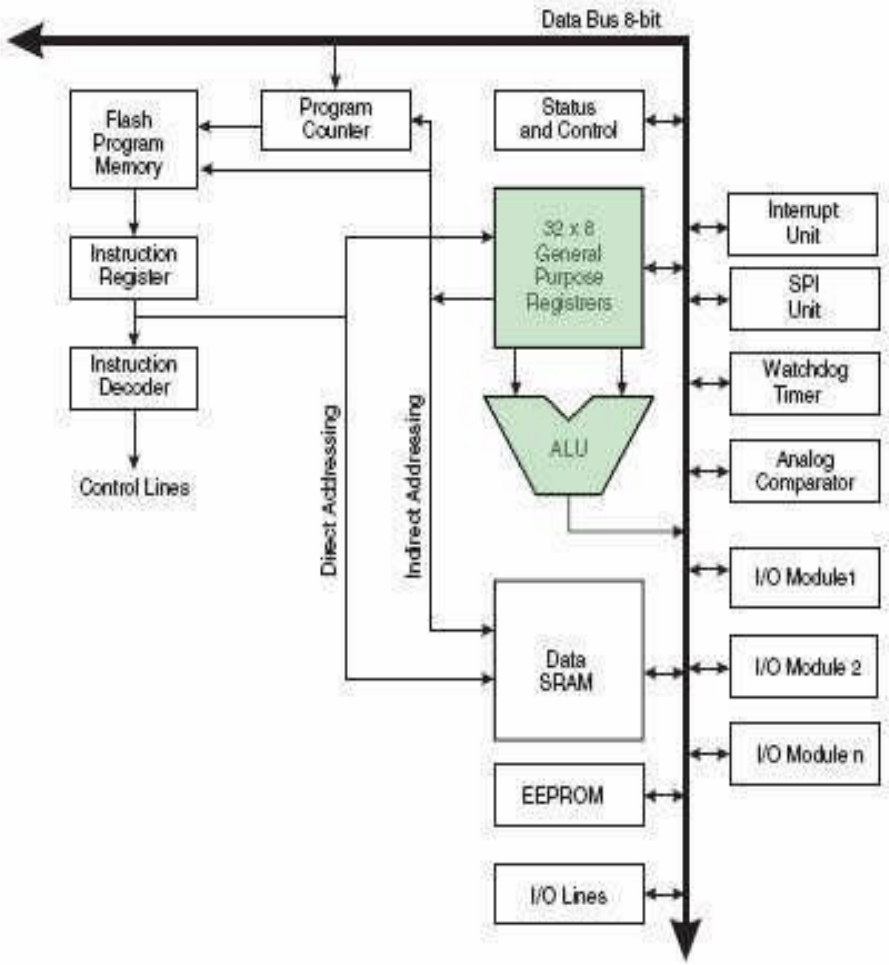
1.Cental Processing Unit(CPU)

- In order to maximize performance and parallelism, the AVR uses a **Harvard** architecture with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.



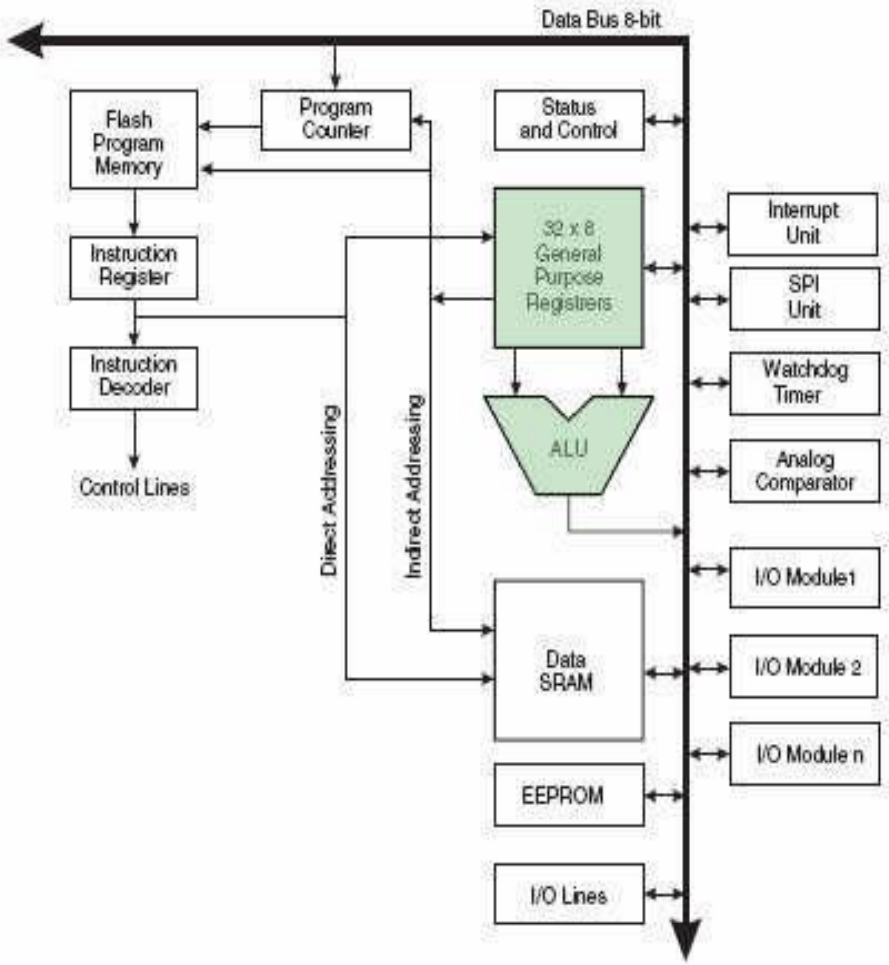
1.Cental Processing Unit(CPU)

- The fast-access Register file contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation In a typical ALU operation, two operands are output from the Register file, the operation is executed, and the result is stored back in the Register file in one clock cycle.



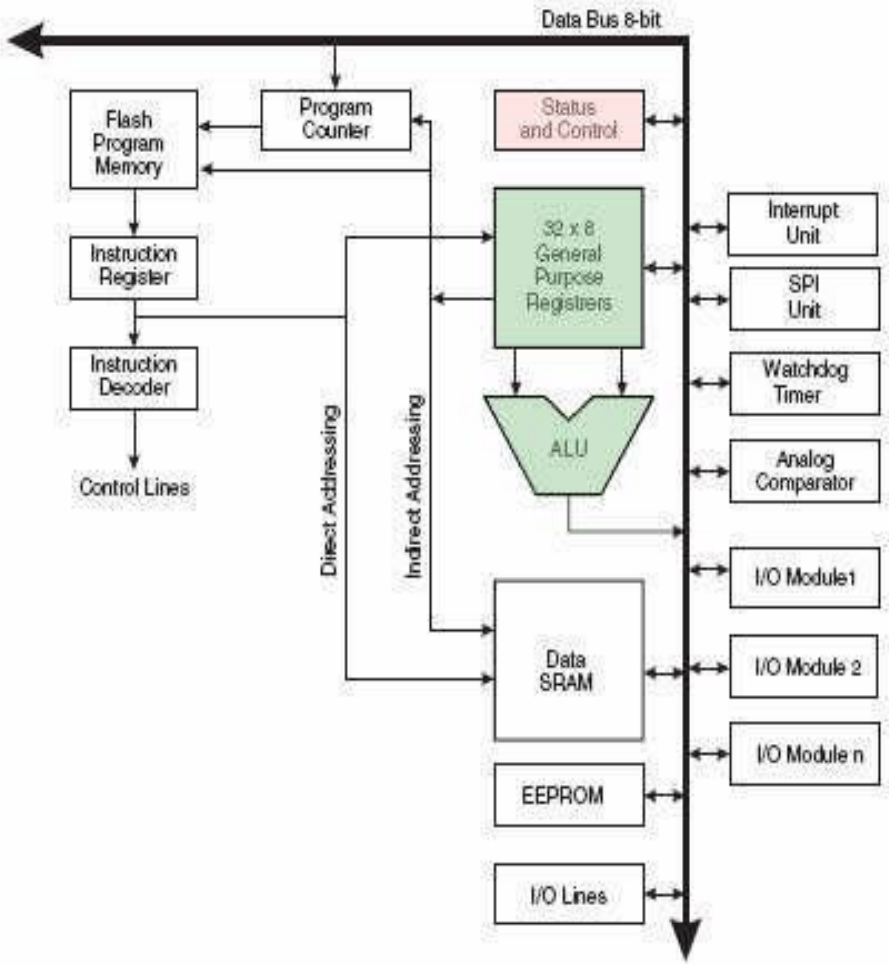
1.Cental Processing Unit(CPU)

- Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing –enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-register, Y-register and Z-register.



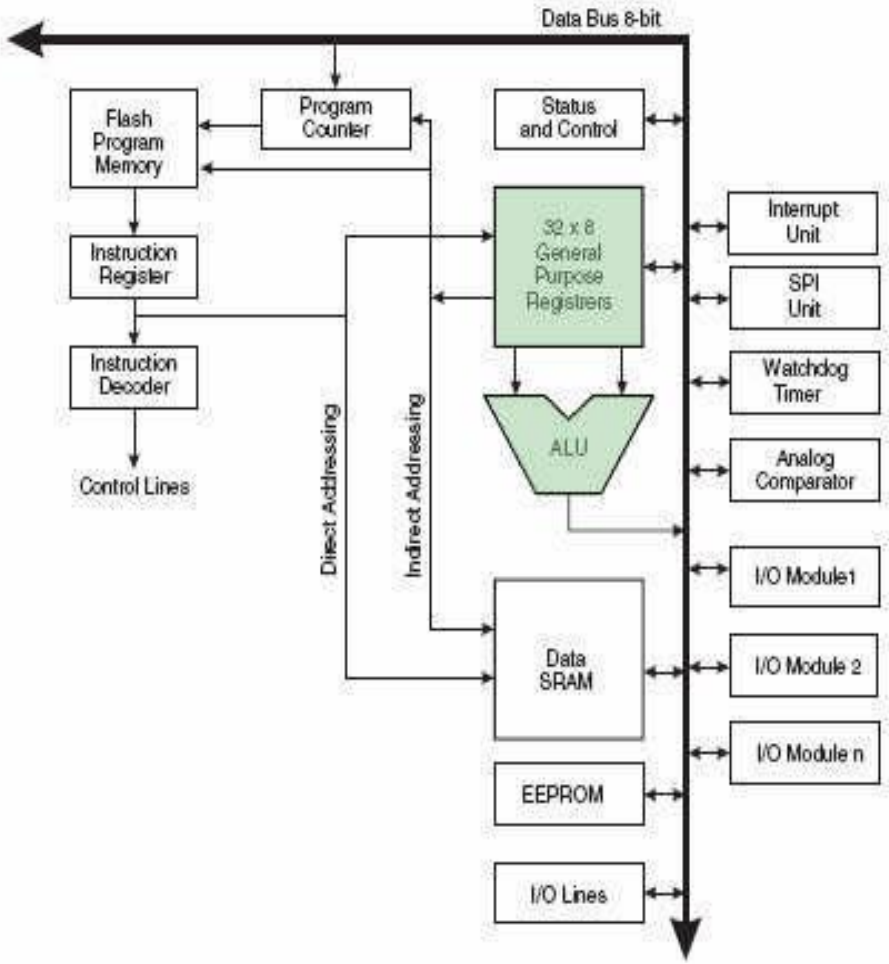
1.Cental Processing Unit(CPU)

- The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.



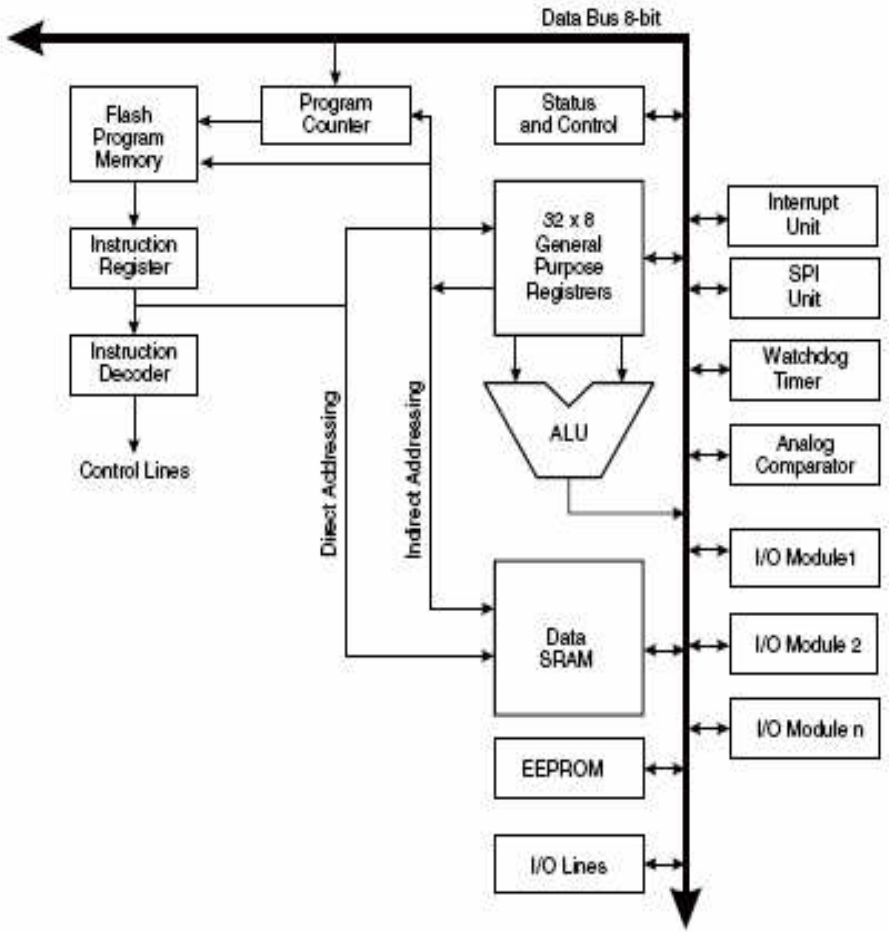
1.Cental Processing Unit(CPU)

- The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format.



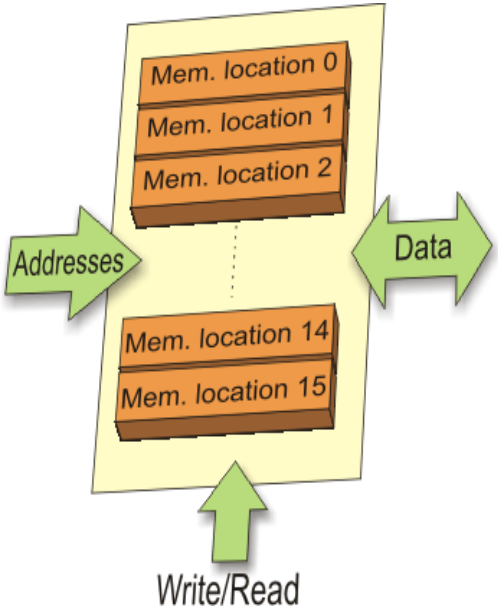
1.Cental Processing Unit(CPU)

- Program Execution: Hex Code is what is stored in the Flash Program memory. When the program runs, the hex code is accessed by the Program Counter. It loads the next instruction into a special Instruction Register. The operands of the instruction are fed into the ALU , while the instruction itself is decoded by Instruction Decoder and then executed by the ALU.



2.Memory Units

- Memory is a part of the microcontroller used for data and program storage.
- Each memory address corresponds to one memory location. The contents of any location can be accessed and read by its addressing. Memory can either be written to or read from.
- There are different types of memories inside the microcontroller:
  - RAM memory (Random Access Memory) used as Data memory.
  - ROM memory (Read Only Memory) used as program memory.



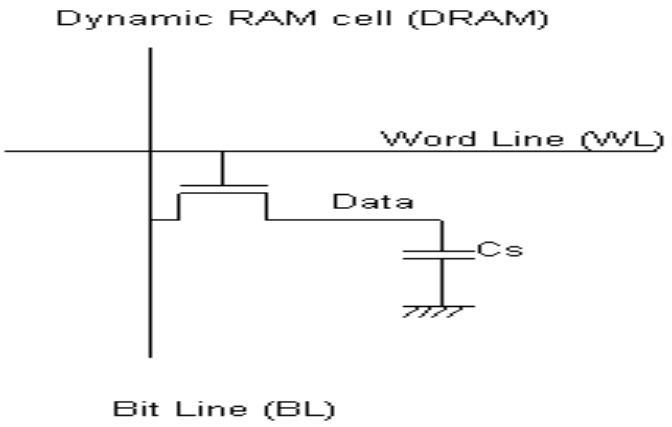
RAM(Random Access Memory)

- Also called Read/Write Memory, The term random access refers to the ability to access any memory cell directly.
- RAM is volatile: if you remove the power supply its contents are lost.
- Used to store data as long as Microcontroller is powered and the program is running.
- Any variable used in a program is allocated into RAM.
- Modifiable through program instructions.

RAM Types

1. Dynamic RAM(DRAM)

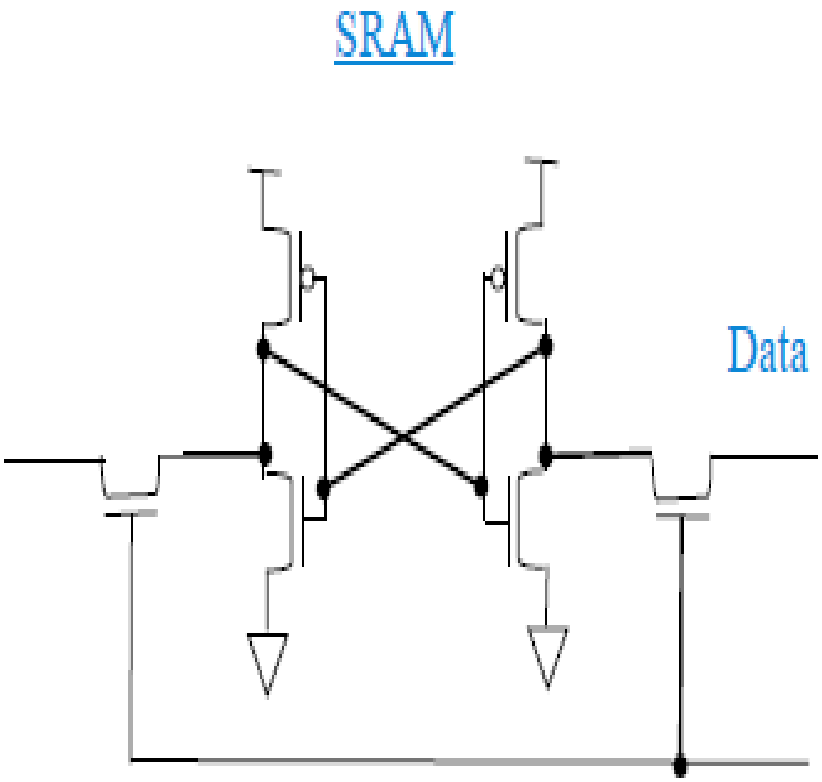
- DRAM is a RAM device that has memory cells with a paired transistor and capacitor.
- The capacitor arrays will hold their charge only for a short period before it begins to diminish.
- it is required periodic refreshing every few milliseconds to retain its content.
- If a processor access conflicts with the need to refresh the DRAM, the refresh cycle must take precedence.

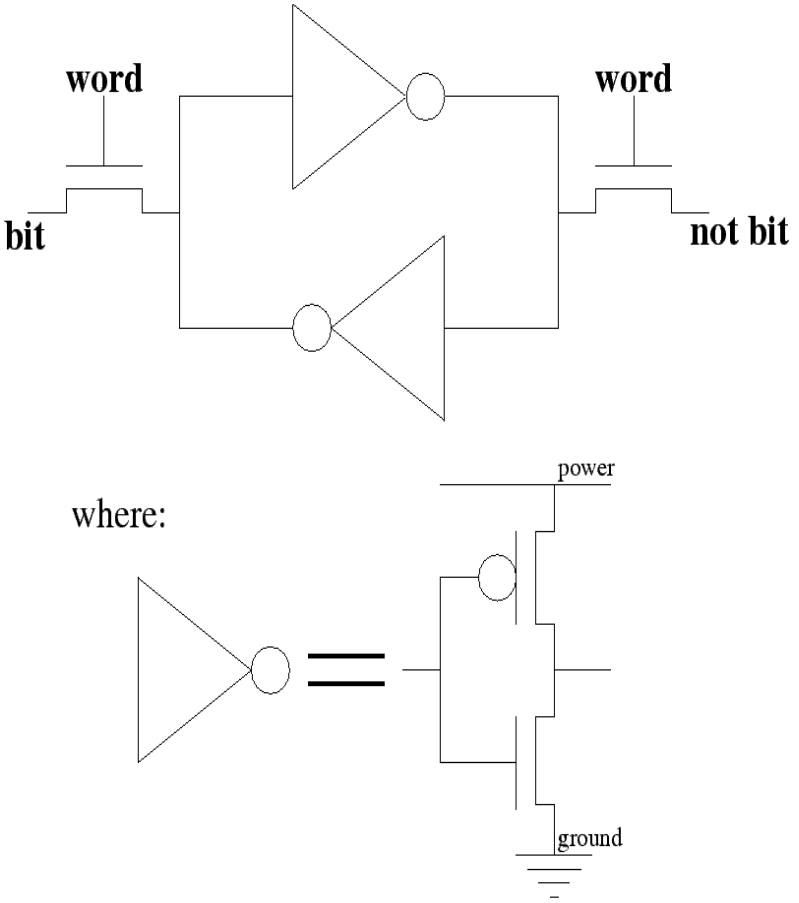


2. Static RAM(SRAM)

- SRAM uses multiple transistors, typically four to six, for each memory cell.
- SRAMs use pairs of logic gates to hold each bit of data.
- SRAM does not require periodic refreshing and it is faster than DRAM.
- Their drawbacks are that their density is considerably less than DRAM, while being much more expensive.

2.Memory Units





2.Memory Units

- Even though both SRAMs and DRAMs are volatile memories, they have some important differences:
  - Since the DRAM requires a single capacitor and a transistor for each memory cell, it is much simpler in the structure than the SRAM, which uses six transistors for each memory cell.
  - On the other hand, due to the use of capacitors, DRAM requires to be refreshed periodically as opposed to the SRAM.
  - DRAMs are less expensive and slower than SRAMs.
  - Therefore they are used for the large main memory of personal computers, workstations, etc., while SRAM are used for the smaller and faster cache memory.

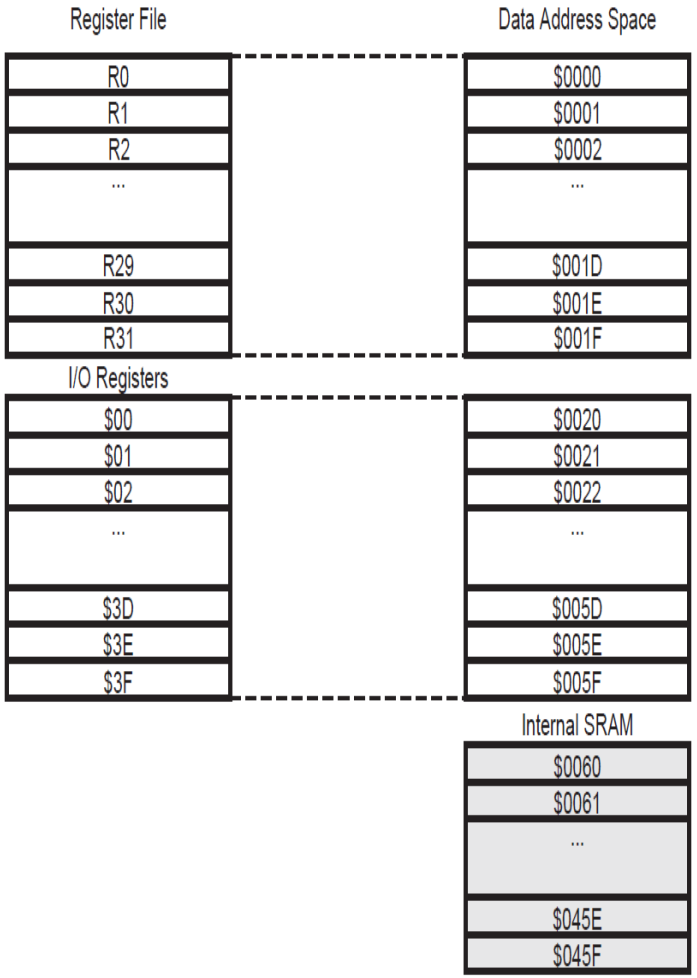
3. Non-Volatile RAM (NVRAM)

- NVRAM is a special type of SRAM that has backup battery power so it can retain its content after the main system power is shut off.
- Another variation of NVARM combines SRAM and EEPROM so that its content is written into the EEPROM when power is shut off and is read back from the EEPROM when power is restored.

- **RAM Divided virtually to**
  - General purpose registers for CPU acts as accumulators (AVR Architecture).
  - Peripherals control special registers like I/O PORTS, timer, UART, etc. registers.
  - All global and static variables are stored in RAM.
  - Stack: store local variables and used to save the present context state when interrupt occurs.

SRAM Memory Mapping in ATmega16 µC

- The 1120 Data bytes Memory locations address the Register File, the I/O Memory, and the internal data SRAM.
- The first 32 locations address the Register File.
- The next 64 locations address the I/O functions, ADC, Timers, USART,SPI and other peripherals
- The next 1024 locations address the internal data SRAM.



2.Memory Units

- Memory Mapped I/O is to use the same address bus to address Peripherals (i.e. I/O Devices), CPU registers and Memory.
- In Memory Mapped I/O when address is accessed by the CPU, it may refer to a portion of physical RAM, but it can also refer to peripherals control special registers.
- To accommodate the I/O devices, areas of the addresses used by the I/O Devices must be reserved for and must not be available for normal data memory.
- It takes memory from main memory which reduces the memory available for applications.

- **ROM(Read Only Memory)**
  - Permanent memory(Non-Volatile memory).
  - Used as Program Memory in Micro-Controller.
  - ROM generally slower than RAM.
  - The size of program that can be written depends on the size of this memory. Today's microcontrollers commonly use 16-bit addressing, which means that they are able to address up to 64 Kb of memory, i.e. 65535 locations.
  - Written upon programming the microcontroller.
  - Can't be written/modified at run time.

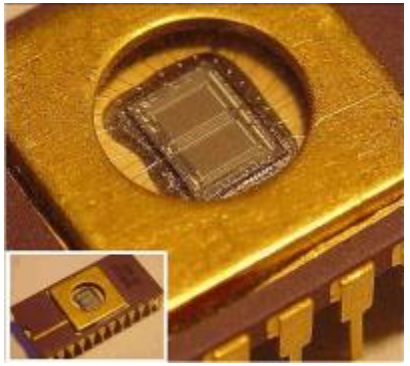
- **ROM Types**
  1. **Masked ROM**
     - Programmed by the manufacturer.
     - The term ‘masked’ comes from the manufacturing process.
     - In case of a large-scale production, the price is very low.

2. OTP(one time programmable)

- Also called programmable ROM(PROM).
- enables programmer to download a program into it one time only.
- Used when the firmware is stable and the product is shipping in bulk to customers.
- If an error is detected after downloading, the only thing you can do is to download the correct program to another chip.

3. UV EPROM(UV Erasable Programmable ROM)

- It enables data to be erased under strong ultraviolet light.
- After a few minutes it is possible to download a new program.
- the package of this microcontroller has recognizable “window” on the upper side. It enables surface of the silicon chip to be lit by an UV lamp, which has as a result that complete program cleared and a new program download enabled.



4. EEPROM (Electrically Erasable Programmable ROM)

- Can be erased by exposing it to an electrical charge.
- The contents of this memory may be changed during run time (similar to RAM), but remains permanently saved even if the power supply is off (similar to ROM). Accordingly.
- EEPROM is often used to read and store values, created during operation, which must be permanently saved.
- Acts as peripheral of microcontroller.
- Accessed through special registers in Micro-Controller.

2.Memory Units

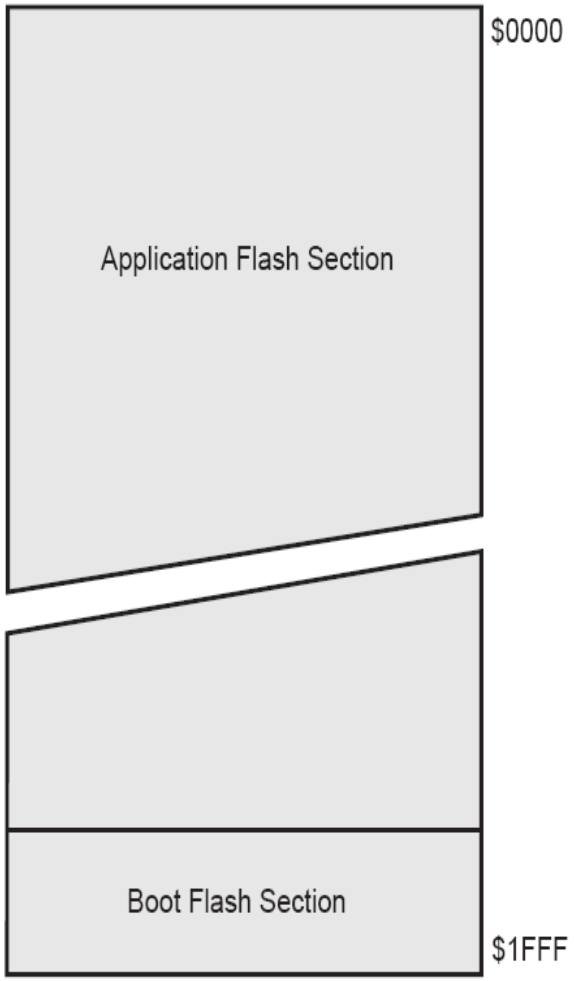
- Take more time in read/write access than RAM.
- Finite number of Write/Erase Cycles 100,000 in **ATmega16** µC.
- **ATmega16** µC contains 512 Bytes EEPROM.
- Example, if you design an electronic lock or an alarm, it would be great to enable the user to create and enter the password, but it's useless if lost every time the power supply goes off. The ideal solution is a microcontroller with an embedded EEPROM.

5. Flash EEPROM

- Invented in the 80s in the laboratories of **Intel.**
- Represented as the successor to the EEPROM.
- Flash is normally organized as sectors (256B - 16KB)
- Large blocks of memory erased at once, rather than one word at a time like EEPROM, So FLASH is much faster than EEPROM.
- The large block sizes used in flash memory give it a significant speed advantage over old-style EEPROM when writing large amounts of data like downloading a µC application code.

Flash EEPROM Memory in Atmega16 µC

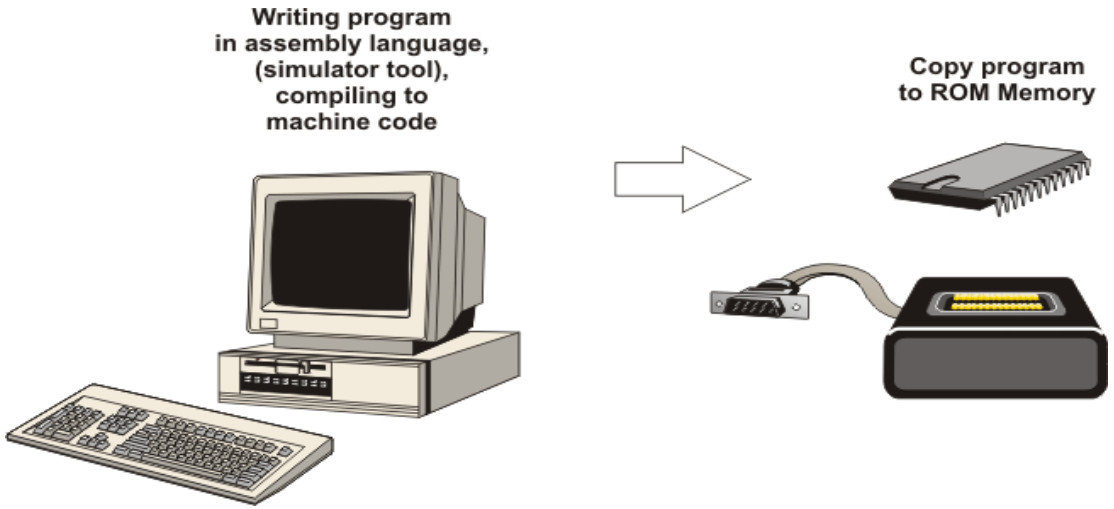
- The ATmega16 contains 16K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8K x 16.
- the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.
- The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations.



2.Memory Units

- Finite number of Write/Erase Cycles 10,000 Flash in **ATmega16** µC.
- Since the contents of this memory can be written and cleared practically unlimited number of times most microcontrollers program memory are manufactured using flash technology.

- **ROM Contents**
  - Program code.
  - Interrupt Vector Table.
  - Constant data Handled through ***const*** keyword in C.
  - Bootloader.



2.Memory Units

- Most of micro-controllers contain **SRAM** memory as a **Data** memory.
- Most of micro-controllers contain **Flash EEPROM** memory as a **program** memory and **EEPROM** memory as a **peripheral**. EEPROM is used to read and store values created during operation, which must be permanently saved.

2.Memory Units

- There are normally 3 types of memory present in a microcontrollers. These are SRAM, FLASH, and EEPROM memories.
- MCU related data that will not change shall be stored in **FLASH**, like code and constant variables.
- Data that must be read from and written to repetitively in a program shall be stored in **SRAM**.
- Application data that must be retained even when power is removed from the system shall be stored in **EEPROM**.

2.Memory Units

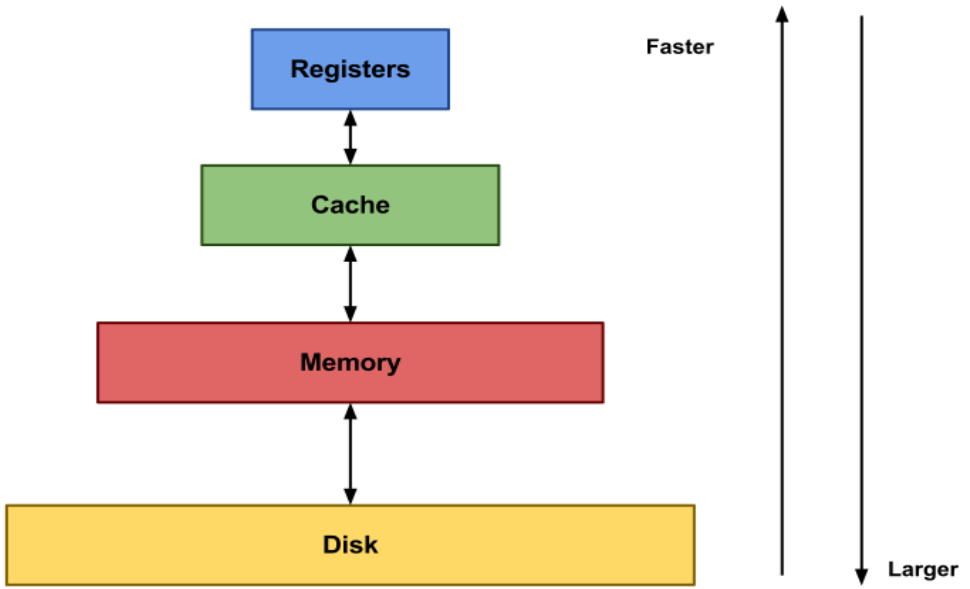
Type	Volatile?	Writeable?	Erase Size	Max Erase Cycles	Cost (per Byte)	Speed
SRAM	Yes	Yes	Byte	Unlimited	Expensive	Fast
DRAM	Yes	Yes	Byte	Unlimited	Moderate	Moderate
NVRAM	No	Yes	Byte	Unlimited	Expensive (SRAM + battery)	Fast
Masked ROM	No	No	n/a	n/a	Inexpensive	Fast
PROM	No	Once, with a device programmer	n/a	n/a	Moderate	Fast
EPROM	No	Yes, with a device programmer	Entire Chip	Limited (consult datasheet)	Moderate	Fast
EEPROM	No	Yes	Byte	Limited (consult datasheet)	Expensive	Fast to read, slow to erase/write
Flash	No	Yes	Sector	Limited (consult datasheet)	Moderate	Fast to read, slow to erase/write

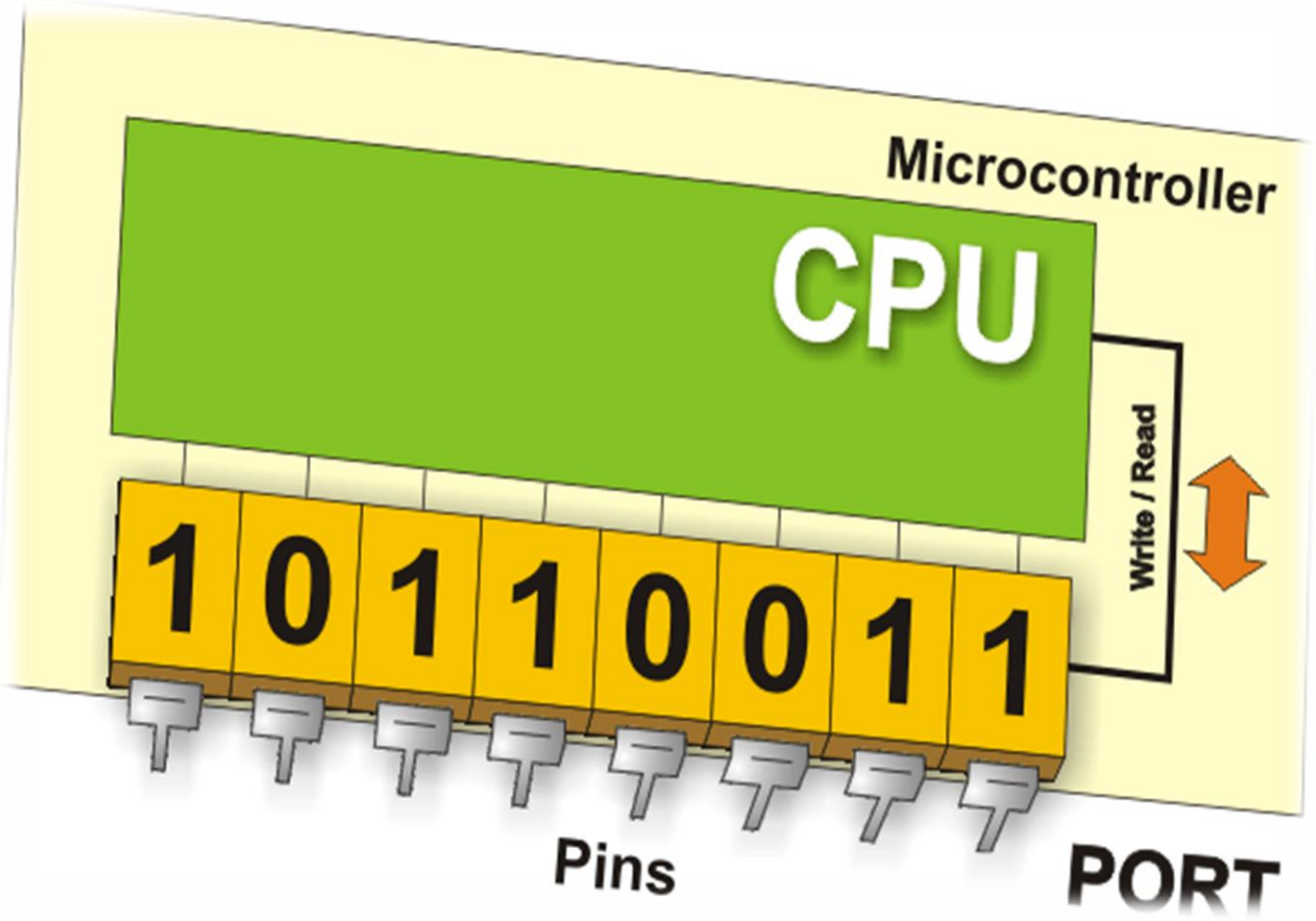
- **Main memory**

Large, inexpensive(DRAM), slow memory stores entire program and data.

- **Cache**

Small, expensive(SRAM), fast memory stores copy of likely accessed parts of larger memory, Can be multiple levels of cache.

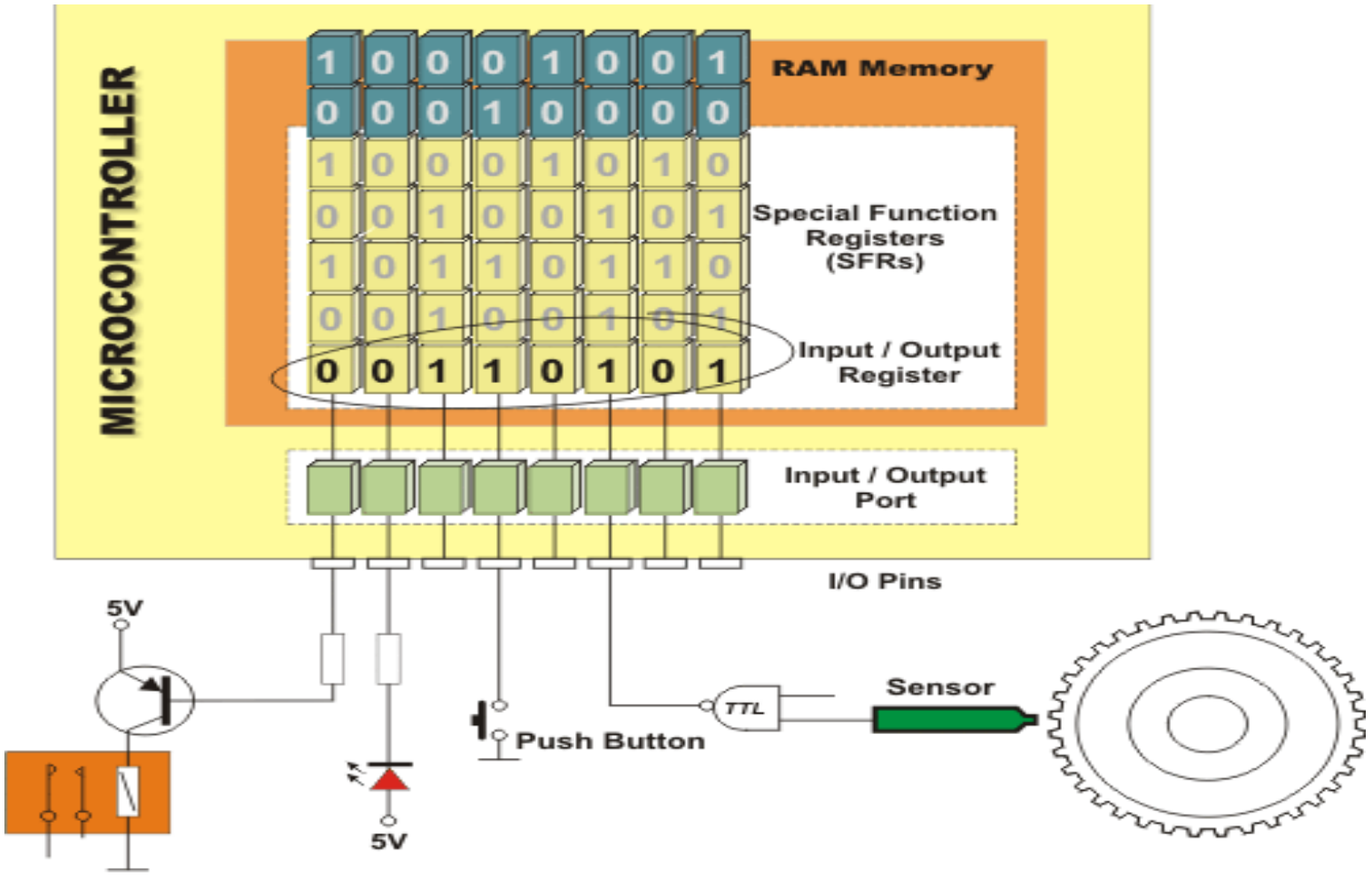


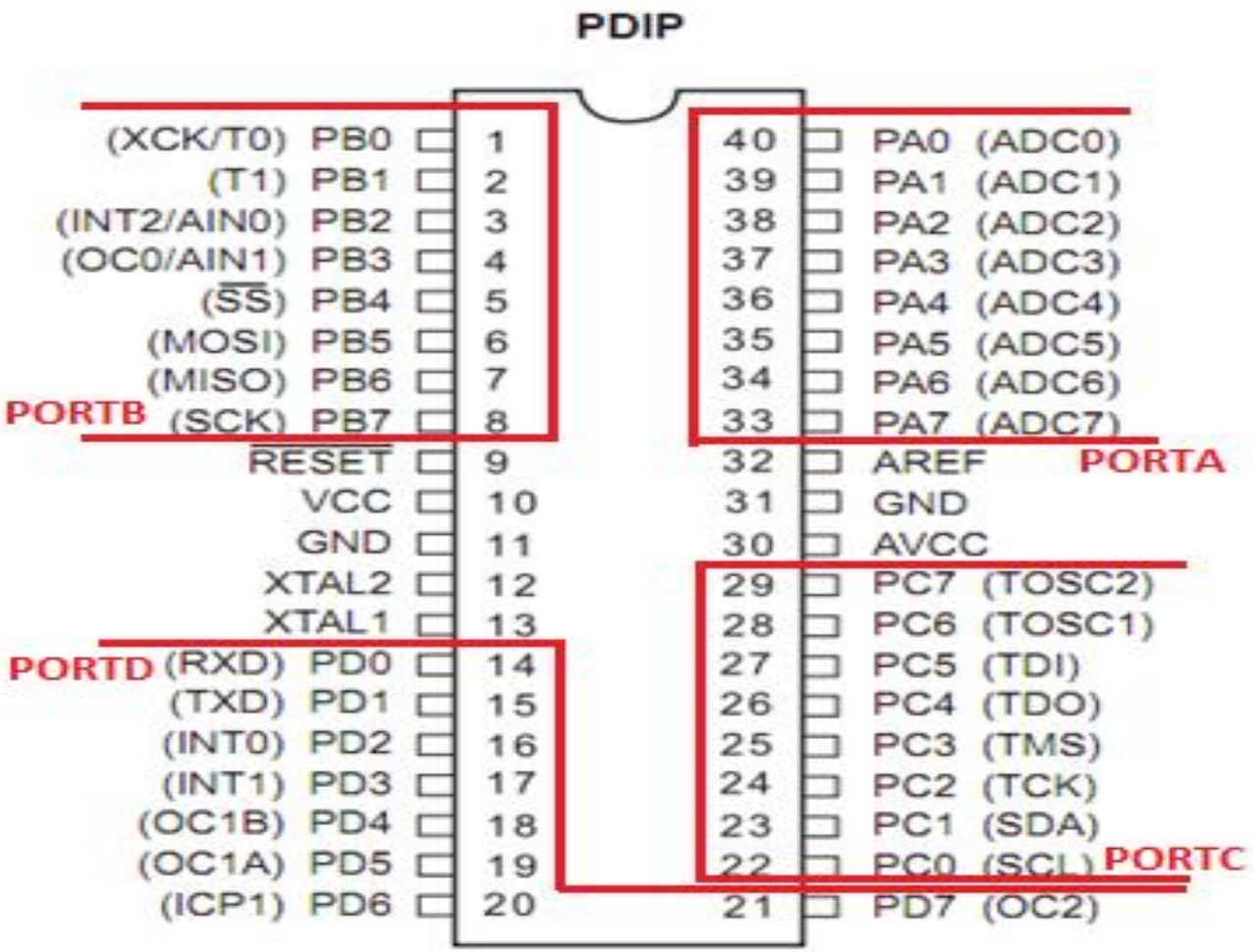


3.I/O Ports

- The microcontroller has to be connected to additional external electronics and peripherals.
- For that reason, each microcontroller has one or more registers (called “port” in this case) to which it’s connected.
- These ports are bidirectional could be output or input.
- Suppose you want your device to turn on and off three signal LEDs and simultaneously monitor logic state of five sensors or push buttons to the microcontroller pins.
- low-current consumption for Micro-Controller Pin (10-20 mA).
- Each I/O port is under control of another SFR, which means that each bit of that register determines state of the corresponding microcontroller pin.

3.I/O Ports

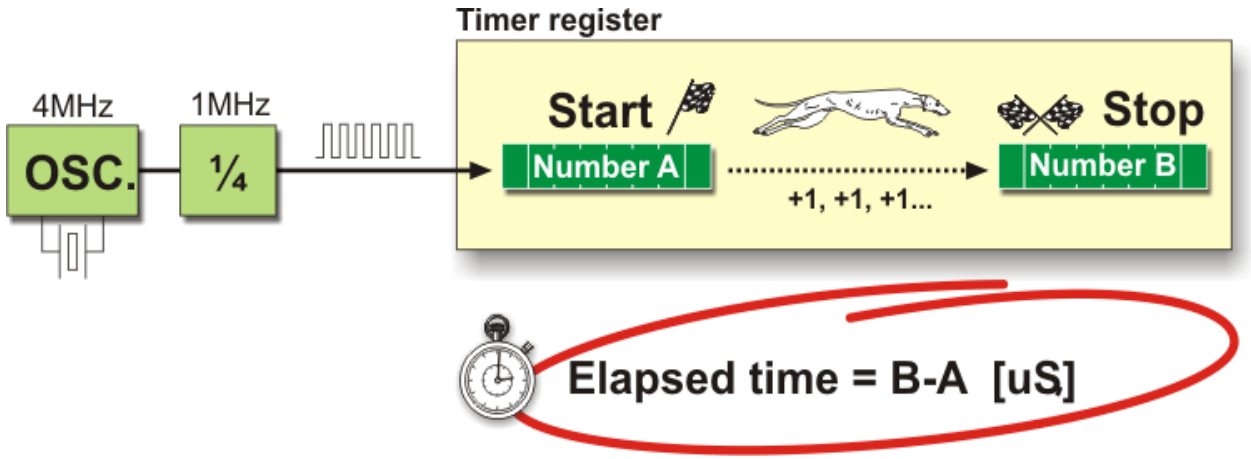


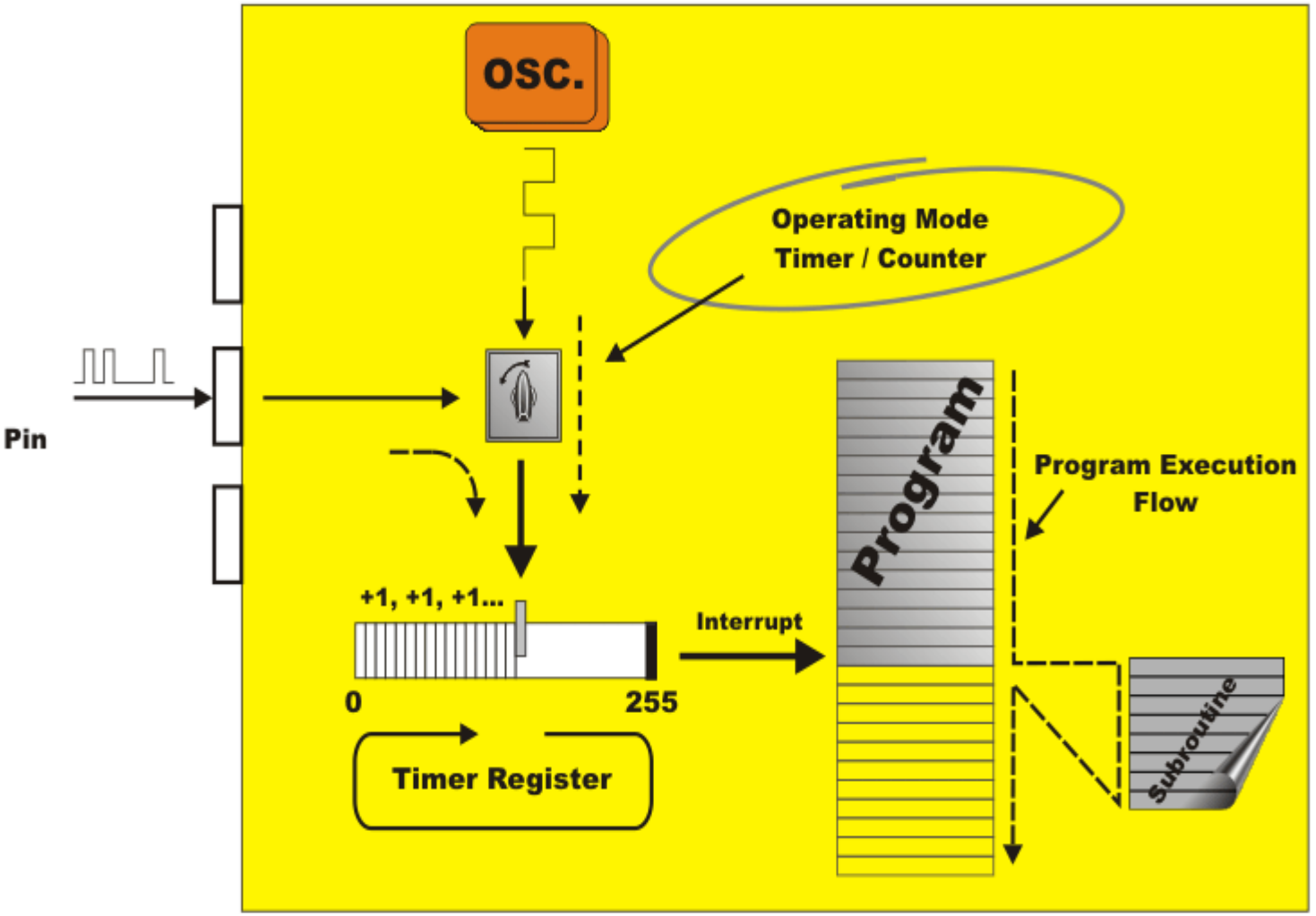


Atmega16 µC Pins

4.Timer

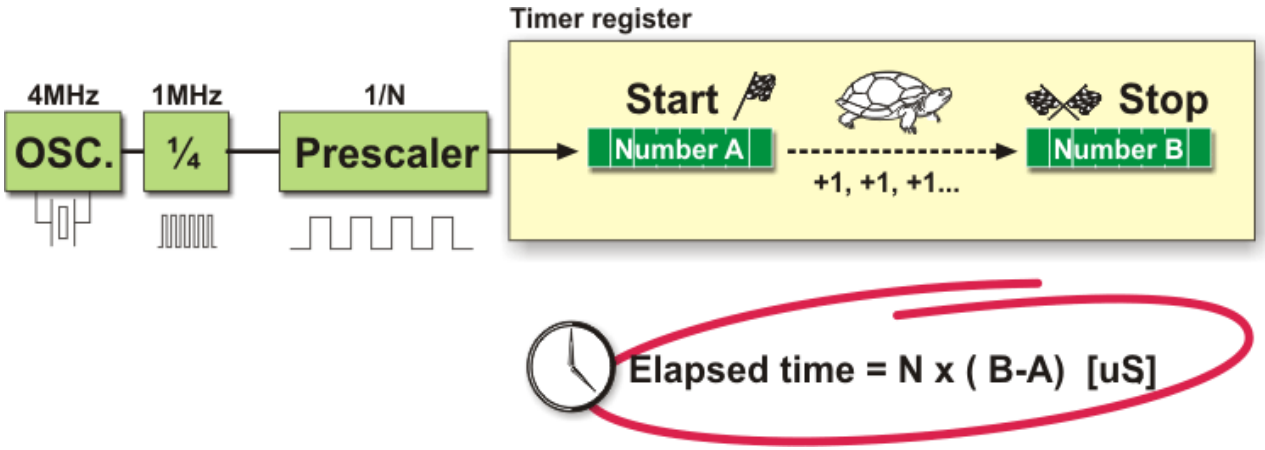
- There are commonly 8-bit or 16-bit timer registers and their content is automatically incremented by each coming clock cycle.
- Most microcontrollers have at least one timer/counter peripheral. Timer/Counter modules are used to perform timing or counting operations in the controller. These include counting events, time passed between two events, generate interrupt after a certain time and generate PWM signals.





Timer Overflow Interrupt

- A **prescaler** is an electronic device used to reduce frequency by a predetermined factor. In order to generate one pulse on its output, it is necessary to bring 1, 2 , 4 or more pulses on its input. Most microcontrollers have one or more prescalers built in and their division rate may be changed from within the program. The prescaler is used when it is necessary to measure longer periods of time. If one prescaler is shared by timer and watchdog timer, it cannot be used by both of them simultaneously.



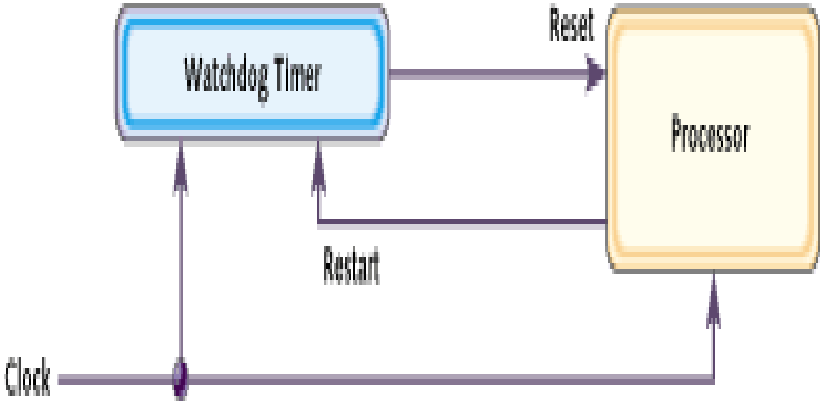
5.Watch Dog timer

- Watch Dog is an essential peripheral in all safety critical embedded systems, it is needed to monitor system status and take corrective actions in case of system failure.

- Special kind of timers that triggers a system reset or other corrective action if the main program, due to some fault condition, such as a hang.

- Usually it's a count down timer.

- The intention is to bring the system back from the unresponsive state into normal operation.



5.Watch Dog timer

- During normal operation, the CPU regularly restarts the watchdog timer to prevent it from elapsing, or "timing out". If due to a hardware fault or program error, the computer fails to restart the watchdog, the timer will elapse and generate a reset signal to reset the CPU.

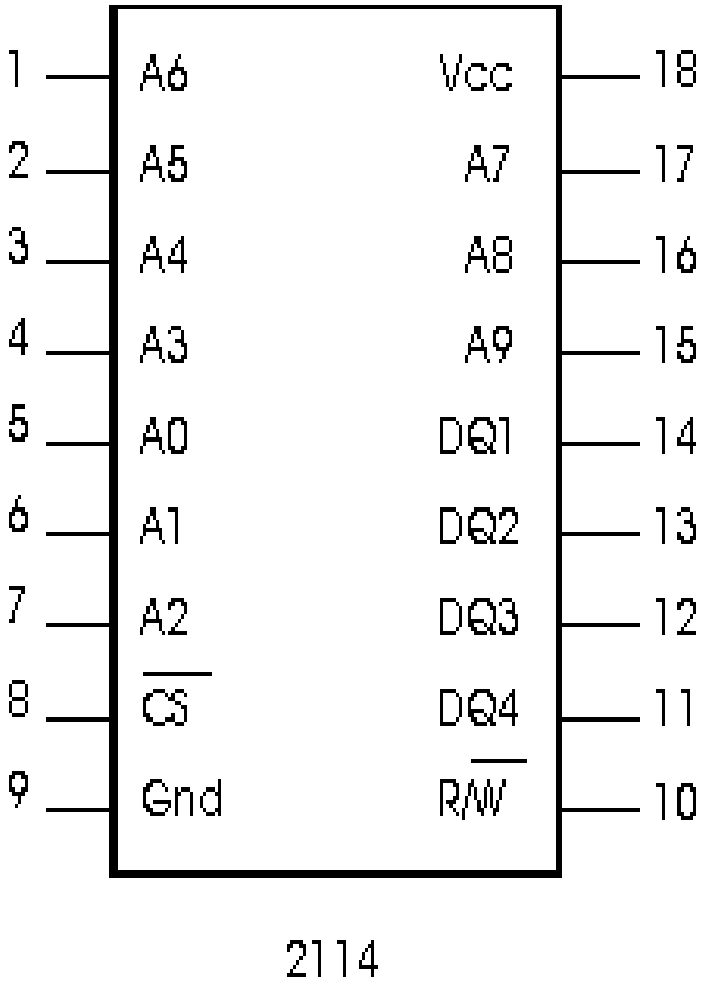
6.Buses

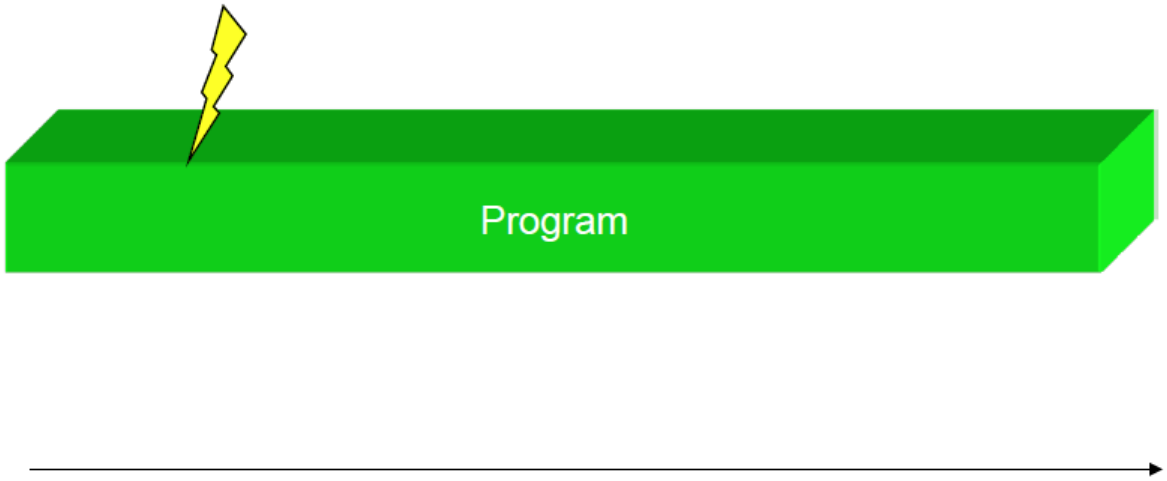
- Physically, the bus consists of 8, 16 or more wire.
- There are two types of buses:
  1. **Address bus**:
     - unidirectional and carries the addresses of memory locations indicating where the data is stored into memory or read from memory.
     - The number of wires in the address bus determines the total number of memory locations.

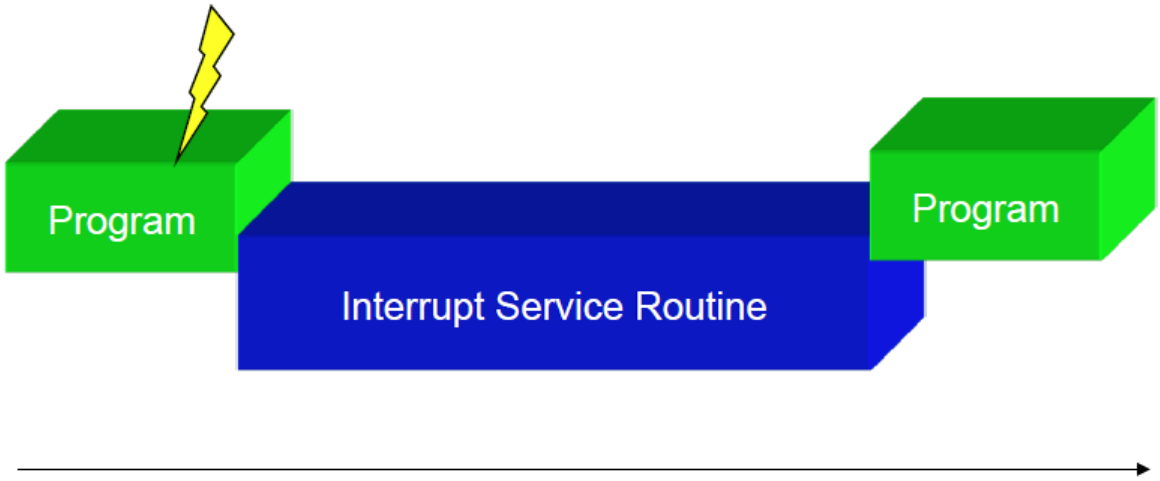
2. **Data Bus**:
   - bi-directional and carries information between the CPU and memory or I/O devices.
   - Computers are often classified according to the size of their data bus. The term “8-bit microcontroller” refers to a microcontroller with 8 lines on its data bus.
   - The number of wires in the data bus determines the number of bits that can be stored in each memory locations.
3. **control bus**
   - Carries commands from the CPU and returns status signals from the devices.
   - For Example: Memory chip READ/WRITE (R/W) lines and Chip Select (CS) lines

Example for 512 Bytes Memory

- Data Path Width = 4
- Address Path Width = 10
- Control lines
  1. R/W
  2. CS







Interrupt Definition

- Basically events that require immediate attention by the microcontroller.

- An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

- When an interrupt event occurs the microcontroller pause its current task and attend to the interrupt by executing an **Interrupt Service Routine (ISR)** at the end of the ISR the microcontroller returns to the task it had pause and continue its normal operations.

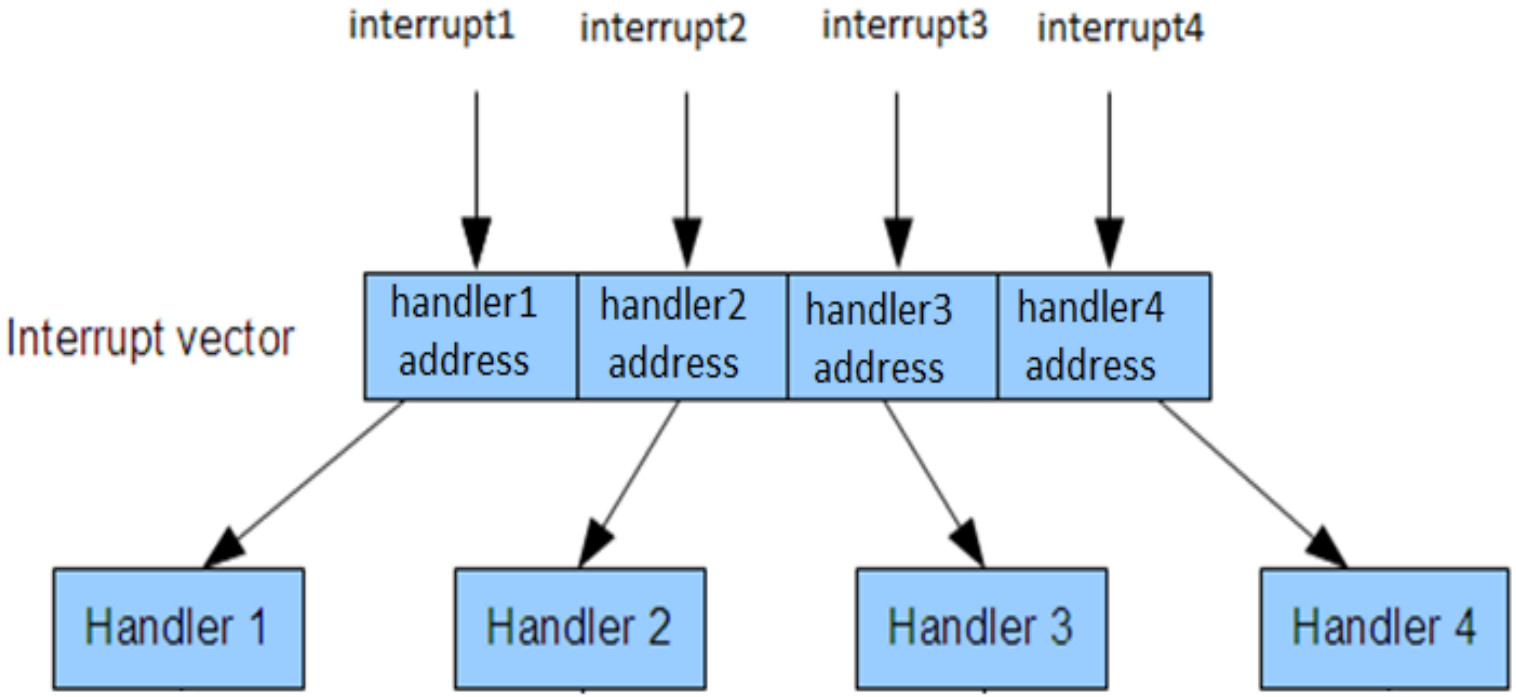
Interrupt Definition

- Such an event may be:
  - An peripheral device has finished the last task it was given and is now ready for another. For example ADC may be generate interrupt after finishing the conversion from analog to digital.
  - External Interrupt event on I/O Pin.
  - An error from a peripheral.

- You can think of it as a hardware-generated function call.

- **Interrupt Service Routine (ISR) or Interrupt Handler**
  - Piece of code that should be execute when an interrupt is triggered. Usually each interrupt has its own ISR. Its address in ROM is usually save in interrupt vector table.
  - should be deterministic and short as possible, as it usually set a flag to a specific task indicating a certain event is happened, or save small data in buffer.

- **Interrupt Vector Table(IVT)**
  - Constant table in ROM.
  - Special addresses with respect to CPU.
  - Each interrupt has specific address in the interrupt vector table for it’s ISR.
  - This specific address should be programmed to have the address of ISR of the corresponding interrupt.



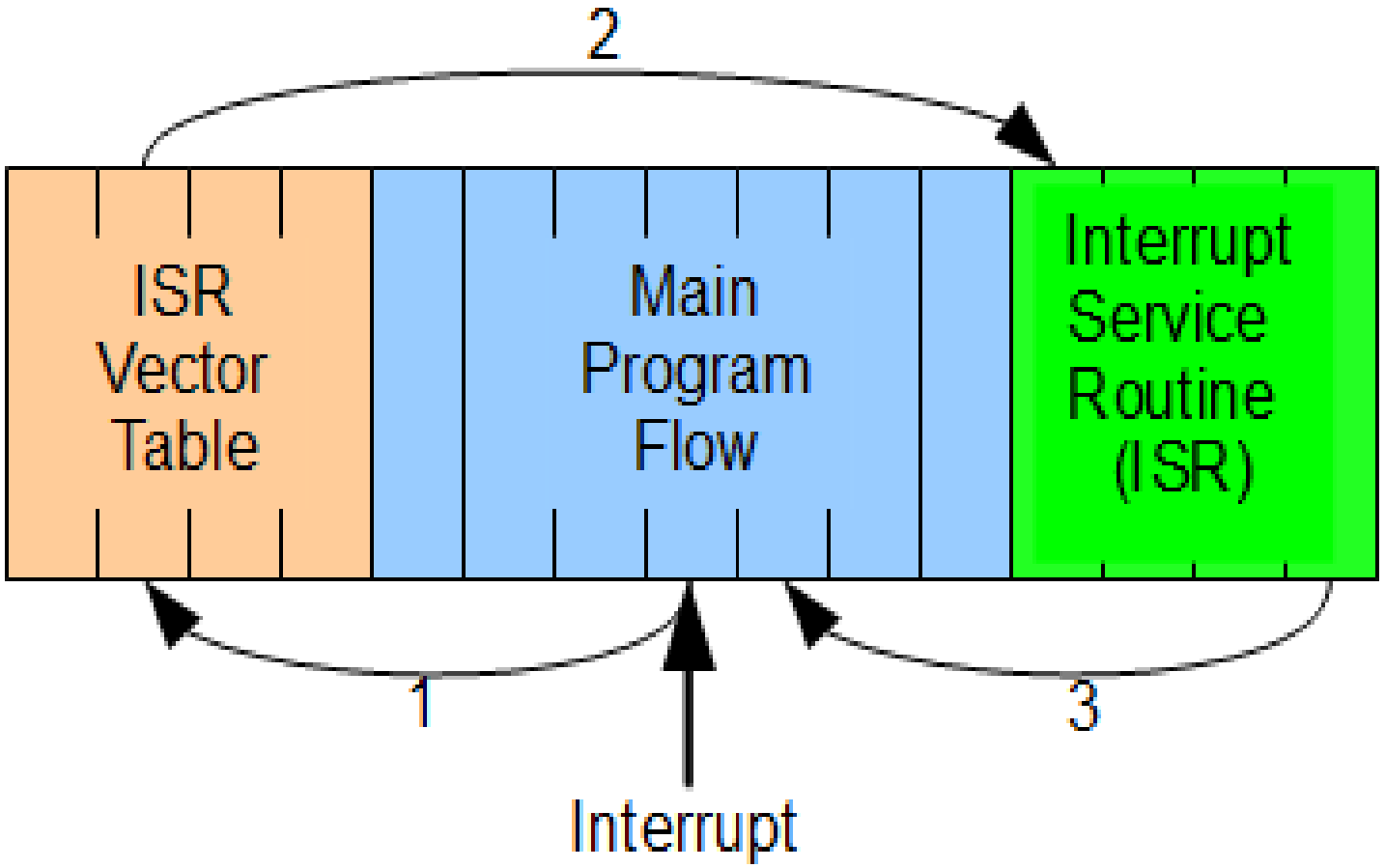
Note : Handler = ISR = Code

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

Interrupt Vector Table for ATmega16 µC

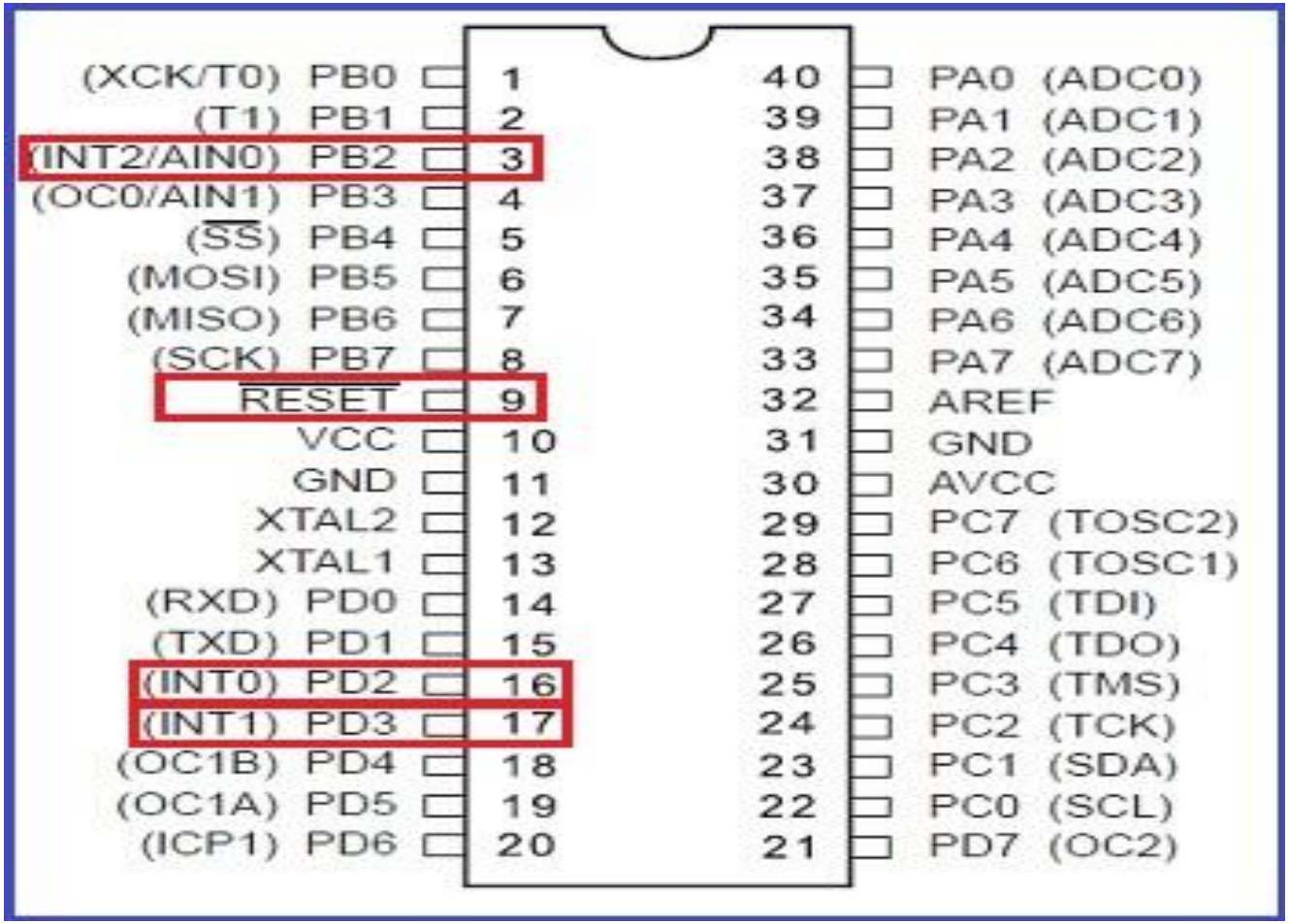
- **What happens when an interrupt occurs?**
  1. The microcontroller CPU completes the execution of the current instruction and stores the address of the next instruction that should have been executed (the contents of the **PC**), status register and the CPU registers are pushed onto the stack.
  2. Microcontroller CPU jump to **Interrupt Vector Table** to get the **ISR** address of the triggered interrupt.
  3. The interrupt vector of the triggered interrupt (ISR start address of this interrupt) is then loaded in the **PC(program counter)** from the interrupt vector table and the CPU starts execution from that point up until reaches the end of the **ISR**.

4. The address that was stored on the stack in **step 1** is reloaded in the **PC**, status register and all the CPU registers are popped from the stack.
5. The CPU then continue executing the **main** program.



Interrupt Sources

- **Hardware Interrupts**
  - Signal to a PIN like reset interrupt or external interrupt.
  - Timer Overflow Interrupt.
  - Serial Port Interrupt.
  - ADC Interrupt.
- **Software Interrupts**
  - interrupts are commands given by the programmer, such as the **SWI** instruction for the Motorola µCs.



External interrupt pins in Atmega16 µC

Interrupt Types

- **Maskable Interrupts**

  maskable interrupt can be disabled and enabled Depends on global interrupt enable in processor status register. And it May be :

  - External interrupt from external pin.
  - Internal interrupt from peripheral.

- **Non-Maskable Interrupts**

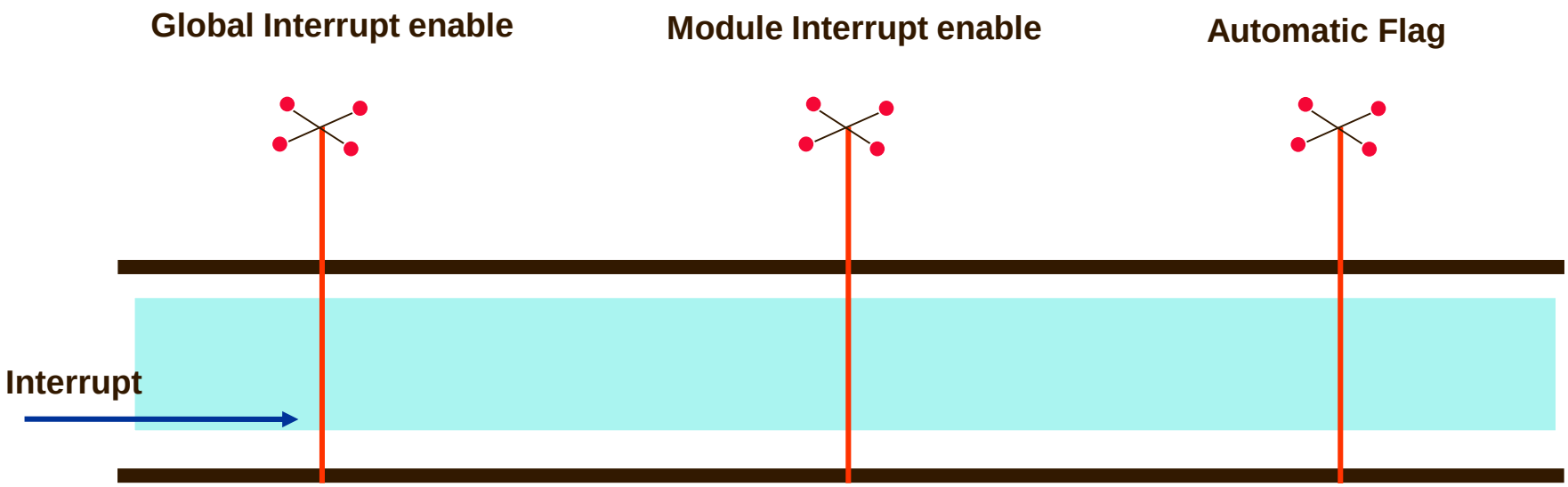
  - non-maskable interrupts can not be disabled and are always enabled.
  - Usually it's external interrupt like: **reset** interrupt.

- **Status Register**
  - Contains flags represent the status of the last operation to control the following instructions
  - Flags as:
    - Overflow flag
    - Negative flag
    - Zero flag
    - Carry flag
    - Half-carry flag
    - **Global Interrupt mask**

AVR CPU Interrupts

- The AVR provides several different interrupt sources.
- These interrupts and the separate reset vector each have a separate program vector in the program memory space.
- The lower the address the higher is the priority level. **RESET** has the highest priority, and next is INT0 - the External Interrupt Request 0.
- All interrupts are assigned individual enable bits to which must be written a logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt.
- Any Interrupt request is triggered by an event that sets its corresponding interrupt flag.

- **When ISR should be executed in AVR CPU?**

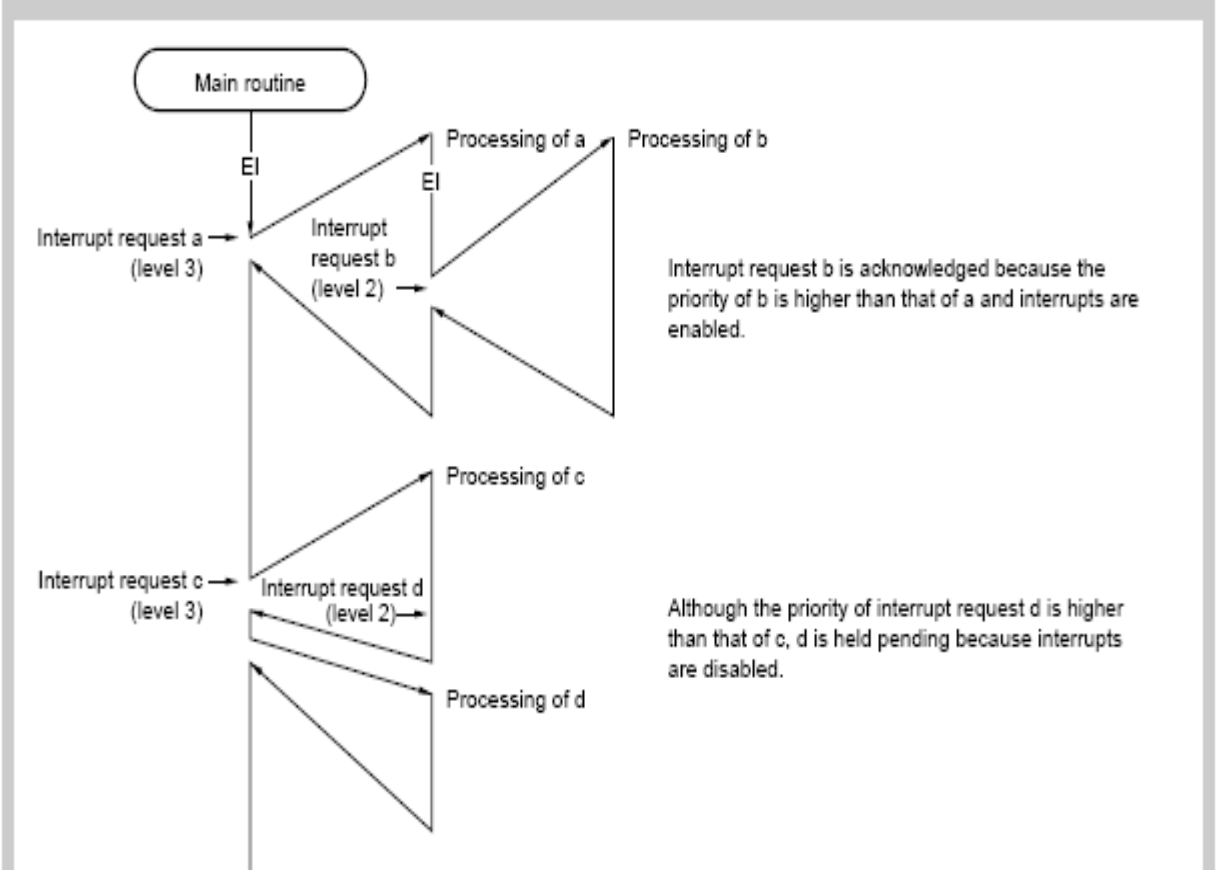


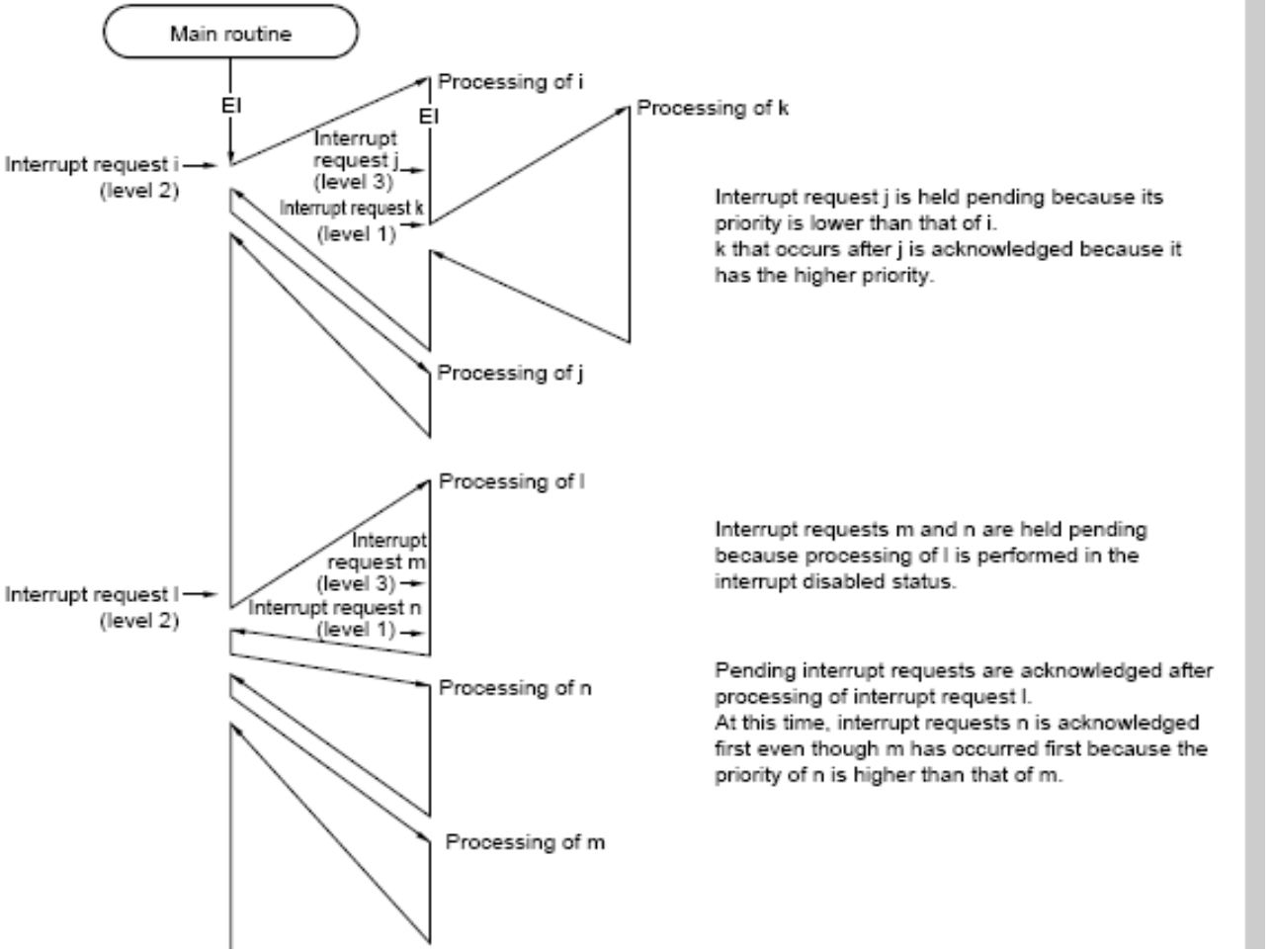
Interrupts Priorities

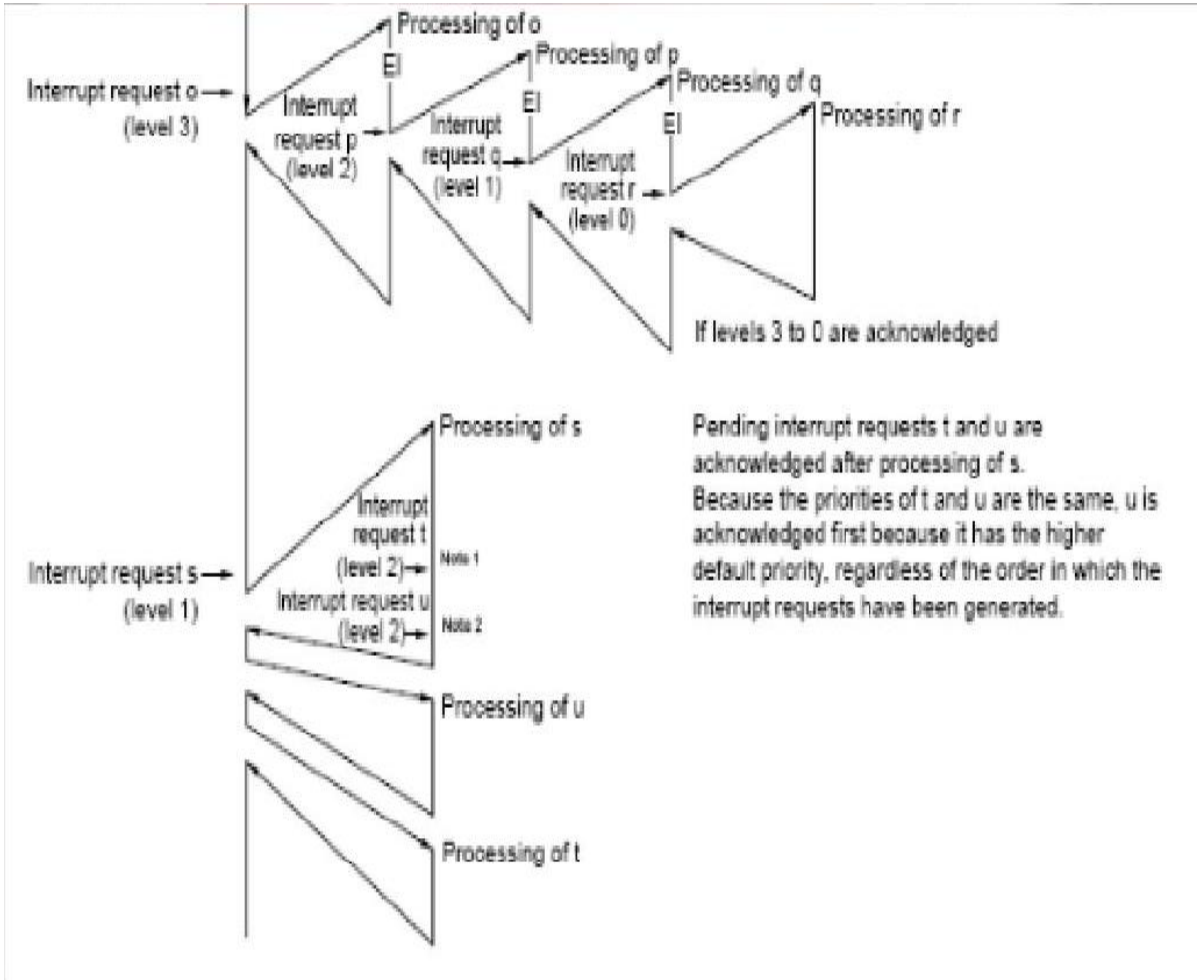
- Each interrupt has default priority by its location in interrupt vector table.
- Some controllers provide more intelligent interrupt controller which give interrupt priority level for each interrupt.
- If two interrupts have a same priority level then the rule to return to the default priority according to the interrupt location in IVT.
- In **AVR CPU** the interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

Interrupt Nesting

- Ability to leave the current interrupt and serve another interrupt.
- Usually done if global interrupt is enabled and this interrupt has higher priority.
- In **AVR CPU** when an interrupt occurs, the Global Interrupt Enable **I-bit** is **cleared** and all interrupts are disabled. The user can write logic one to the I-bit on software to enable interrupt nesting. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction.



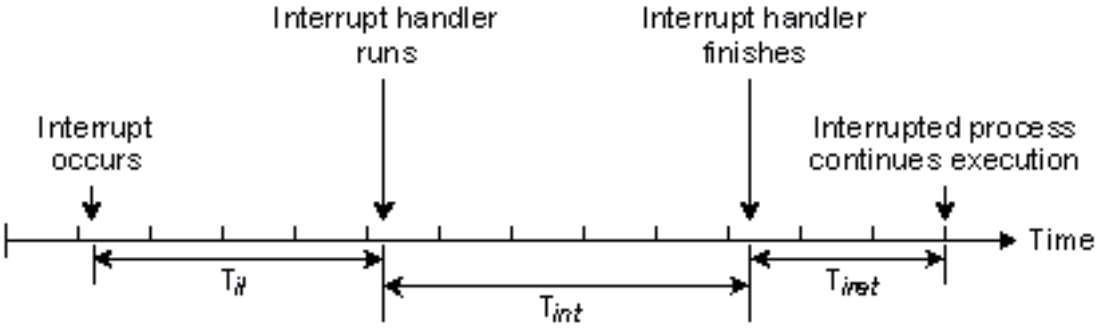




Interrupt processing context:

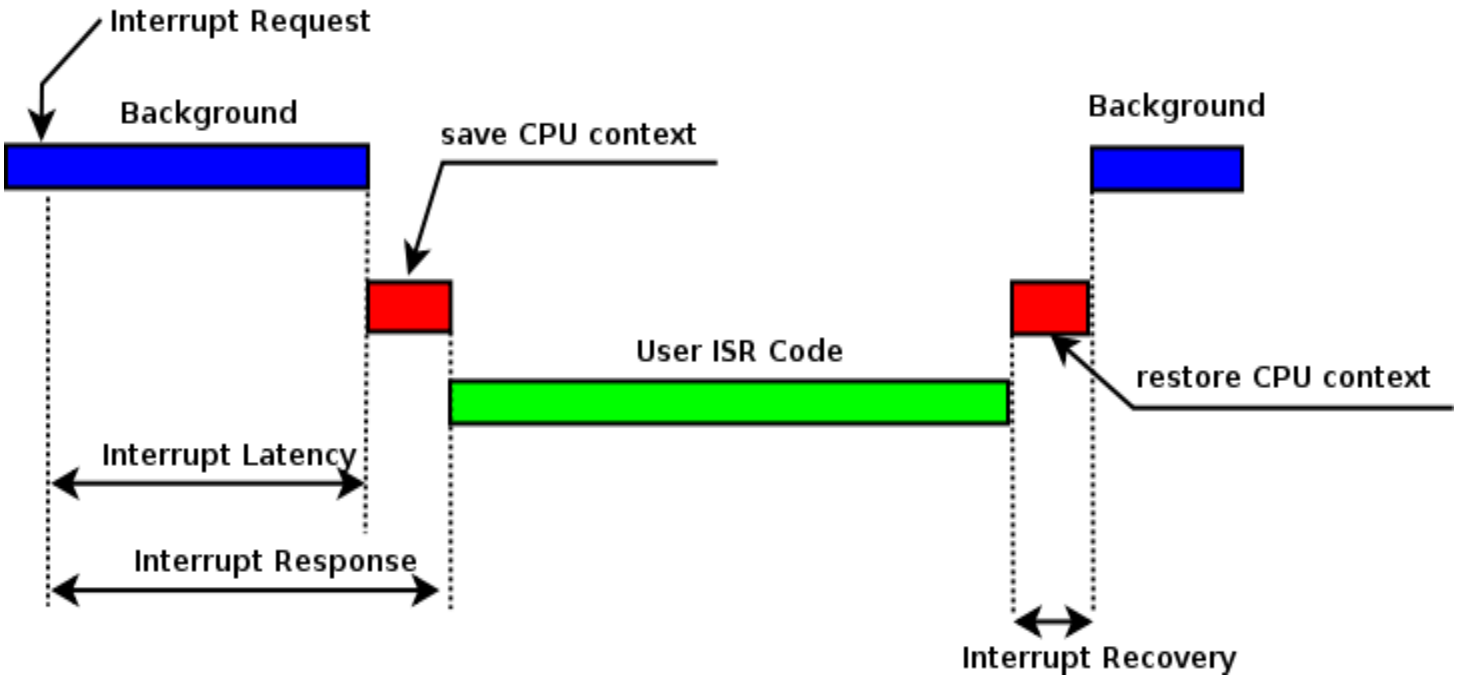
1. If it's maskable interrupt :
    - Check if global interrupt is enabled.
    - Check if this module interrupt mask is enabled.
    - Check if the automatic flag is set.
2. If another ISR is currently in service check the priority level.
3. Save important context registers in special registers or stack, like the following:
    - PC
    - Processor status register
    - Accumulators
    - General purpose register
4. Jump to ISR using vector table

- **Interrupt latency** is the time between the generation of an interrupt by a device and the servicing of the device which generated the interrupt.
- **Interrupt latency** may be affected by:
  - Interrupt controllers.
  - Interrupt masking.
  - OS interrupt handling methods.



T_{il} interrupt latency
T_{int} interrupt processing time
T_{iret} interrupt termination time

- **Interrupt Response**: Interrupt latency + Time to save the CPU context.
- **Interrupt Recovery**: Is defined as the time required for the processor to return to the interrupted code.



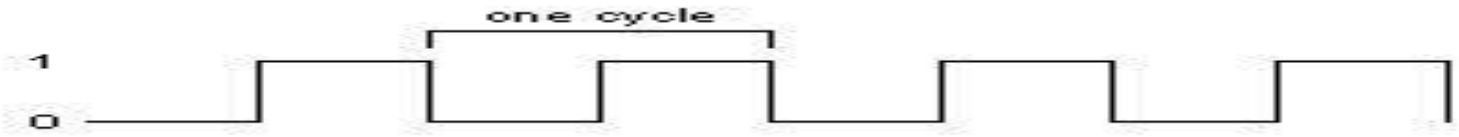
Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - **Micro-controller other components.**
- Embedded Systems Constrains.
- Embedded Systems Market.

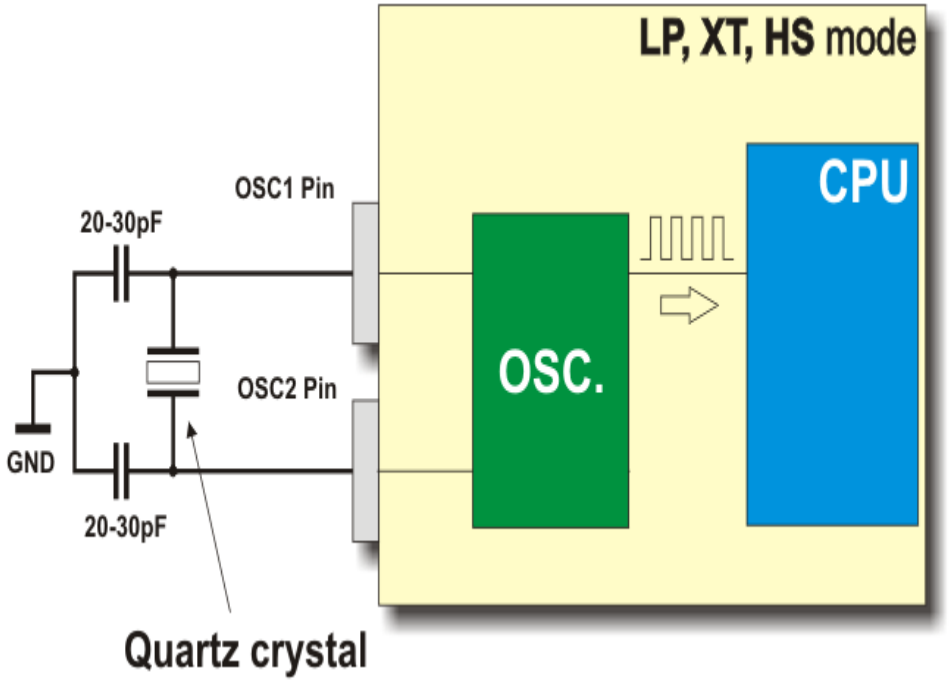
- **Microcontroller may contain these Peripherals:**
  - Analog Digital Converter(ADC).
  - Analog Comparator.
  - Universal Synchronous/Asynchronous Receiver Transmitter(USART).
  - Serial Peripheral Interface(SPI).
  - Inter Integrated Circuits(I2C).
  - Local Interconnect Network(LIN).
  - Controller Area Network(CAN).
  - EEPROM Memory.
  - Direct Memory Access(DMA).
- Will be discussed later in Interfacing Course.

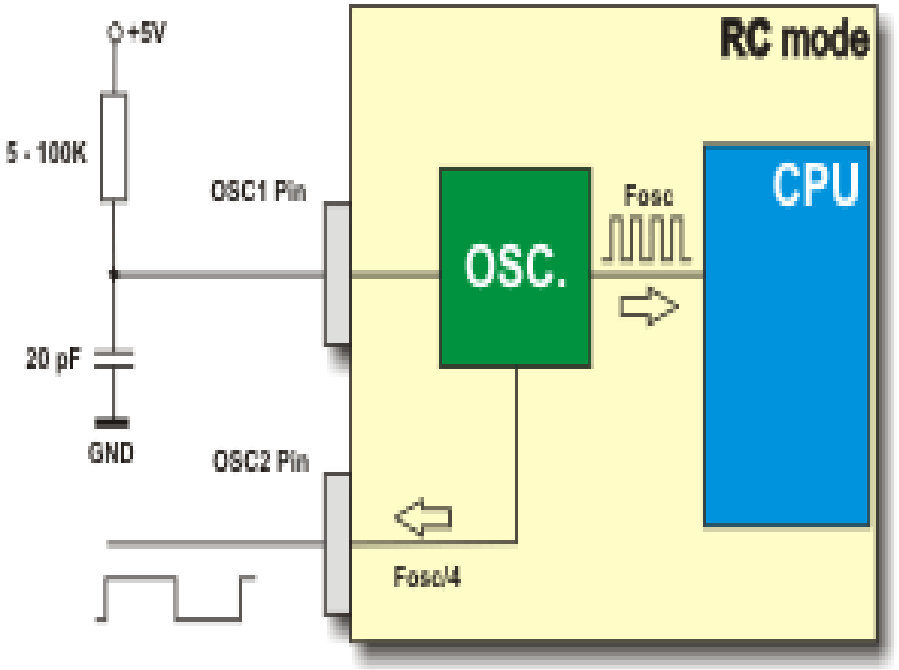
Micro-controller clock

- Clock signal is a particular type of signal that oscillates between a high and a low state, Circuits use the clock signal for synchronization may become active at either the rising edge, falling edge, or, in the case of double data rate, both in the rising and in the falling edges of the clock cycle.
- Microcontroller clock generator source could be **internal** or **external**.
- Microcontroller **external** clock sources could be **RC** oscillator or **Crystal** oscillator but **internal** clock source could by **RC** oscillator only.



Micro-controller clock(External sources)





Crystal oscillator

RC oscillator

- **Power Management features**

  A majority of microcontrollers usually support an operation of 3 - 5.5 V. As consumer goods become trendier, compact and lighter, the focus is on microcontrollers to ensure that products with less power usage are efficiently built and then used by end-users.

- **How do we minimize power ?**

  1. Turn off unnecessary logic
  2. Reduce memory accesses
  3. Use sleep modes in unused controllers

Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- **Embedded Systems Constrains.**
- Embedded Systems Market.

- **Constraints could be after that classified as following :**
  - I/O constraints.
  - Communication constraints.
  - Operating Constraints.
  - Memory Consumption.
  - Microcontroller resources.

- **I/O constrains:**
  any specification related to Digital inputs/outputs or Analog inputs example:
  1. Maximum and Minimum I/O values.
  2. Rate of change.
  3. Sampling rate.
  4. Bit streams coding.
  5. Baud rate.

- **Communication constrains:**

  Constrains related to communication between Microcontrollers and others blocks on boards Example:

  - Communication protocol: SPI, UART, I^2C, CAN.
  - Specification of this protocol.
  - May be specific Interface communication like MMC Interface, SD memory Interface.

- **Operating systems:**
  - Existence.
  - Performance.
  - Preemption.
  - Static/dynamic task allocation.
- **Maximum Execution time:**
  - Interrupts.
  - Critical section.

- **Memory consumption:**
  - RAM size
  - ROM size
  - Stack size
- **Microcontroller constrains:**
  - Frequency
  - Power modes
  - CPU load

How to select a Micro-Controller

- Number of I/O ports.
- Peripherals required (Timers, ADC, USART,SPI and EEPROM).
- Memory requirements.
- Maximum clock speed.
- Real-Time considerations.
- Number of interrupts required
- Power Consumption / sleep modes.
- Availability and Cost.
- Vendor Support.
- Integrated Development Environment.

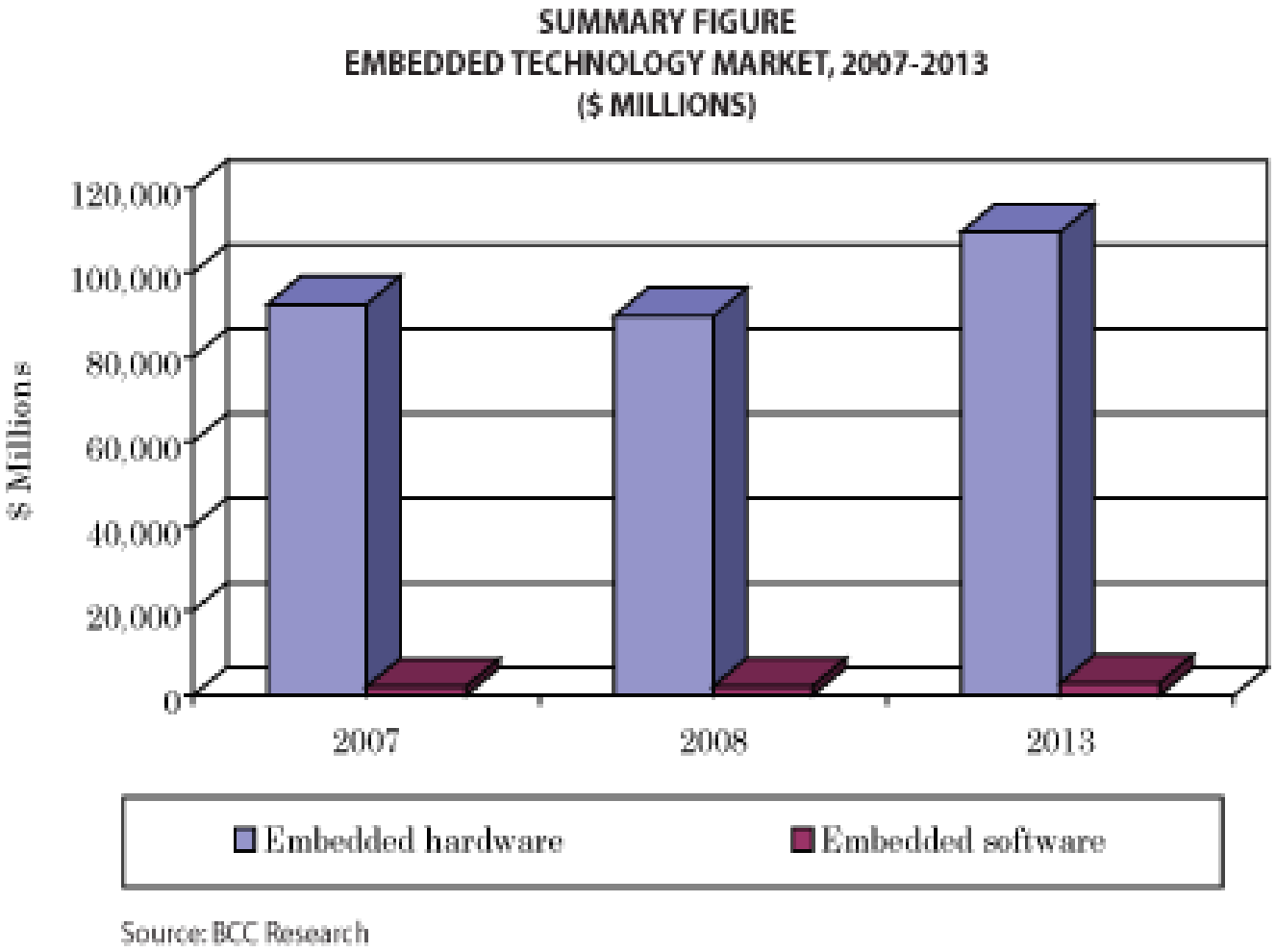
- **Microcontroller Vendor is the manufacture of the microcontroller chip it self.**

Vendor	Families	Derivative
ATMEL	AVR	ATmega16
infineon	TriCore	TC297
MICROCHIP	PIC	PIC16F877A
RENESAS	RH850	RH850E2M
freescale™	HC12	MC9S12XEP100
Texas Instruments	TM4C	TM4C123GH6PM

Agenda

- Embedded Systems Definition.
- Embedded Systems Characteristics.
- Embedded Systems Applications.
- Embedded Systems Design.
- Embedded HW
  - Processing Engines.
  - Micro-processor vs. Micro-controller.
  - Micro-controller main components.
  - Micro-controller other components.
- Embedded Systems Constrains.
- **Embedded Systems Market.**

Embedded Systems Market



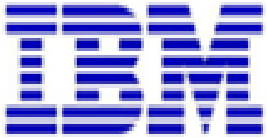
Embedded Systems Market

- The global market for embedded systems is expected to increase from ***$92.0 billion*** in ***2008*** to an estimated ***$112.5 billion*** by the end of ***2013***,
- Embedded hardware was worth ***$89.8 billion*** in ***2008*** and is expected to grow reach ***$109.6 billion*** in **2013**
- Embedded software generated ***$2.2 billion*** in ***2008,***this should increase to ***$2.9 billion*** in ***2013***

Embedded Systems Market

- Embedded market segmented the into **HARDWARE** and **SOFTWARE.**
- **The hardware segment consists of :**
  - Processor IP: MPU/MCU.
  - DSP, ASIC, FPGA.
  - Embedded boards.
- **The software segment consists of:**
  - Operating Systems.
  - Software Applications.
  - Software development and testing tools.
  - Linux-based operating systems and tools.

Embedded Systems Market in Egypt



My Full Embedded Diploma Contents :

http://www.mediafire.com/file/bqqfit41s8n7x8p/Embedded+Diploma+Contents.pdf

My Facebook Group :

https://www.facebook.com/groups/Embedded.Systems.Programming

My Linkedin Account :

https://www.linkedin.com/in/mohamed-tarek-2237a457/

Contact Details

Eng. Mohamed Tarek.
Senior Embedded Software Engineer
Mob: 01115154316
mtarek.2013@gmail.com