Embedded Systems lectures

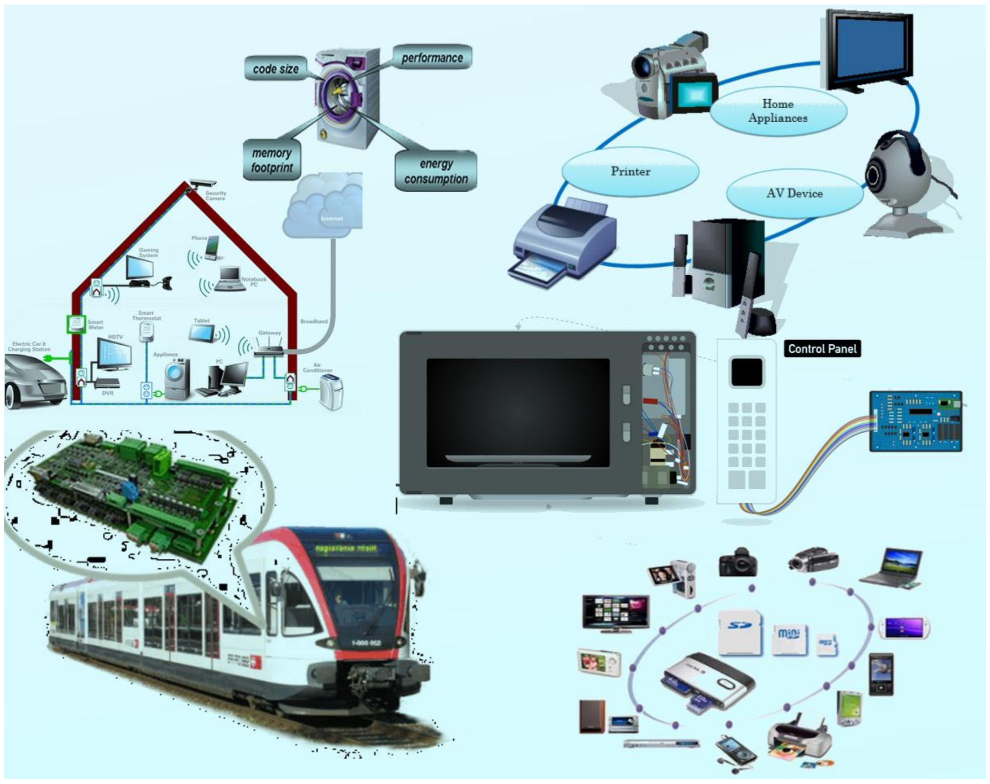
By Dr.Oday A.L.A Ridha
University of Baghdad-2018

References:

1) Han-Way Huang,” PIC microcontroller an introduction to software and hardware interfacing”
2) David Benson, “PIC microcontroller application guide”
3) Martin Bates, “PIC microcontrollers: an introduction to microelectronics”
4) Tim Wilmshurst, “ Designing embedded systems with PIC microcontrollers: principles and applications”
5) Microchip official site: http://www.microchip.com

6) http://www.eeherald.com

Embedded Systems is a special purpose computer system/board, which encapsulates all the devices such as processor, memory, interface and control in single package or board to perform only a specific application tasks.

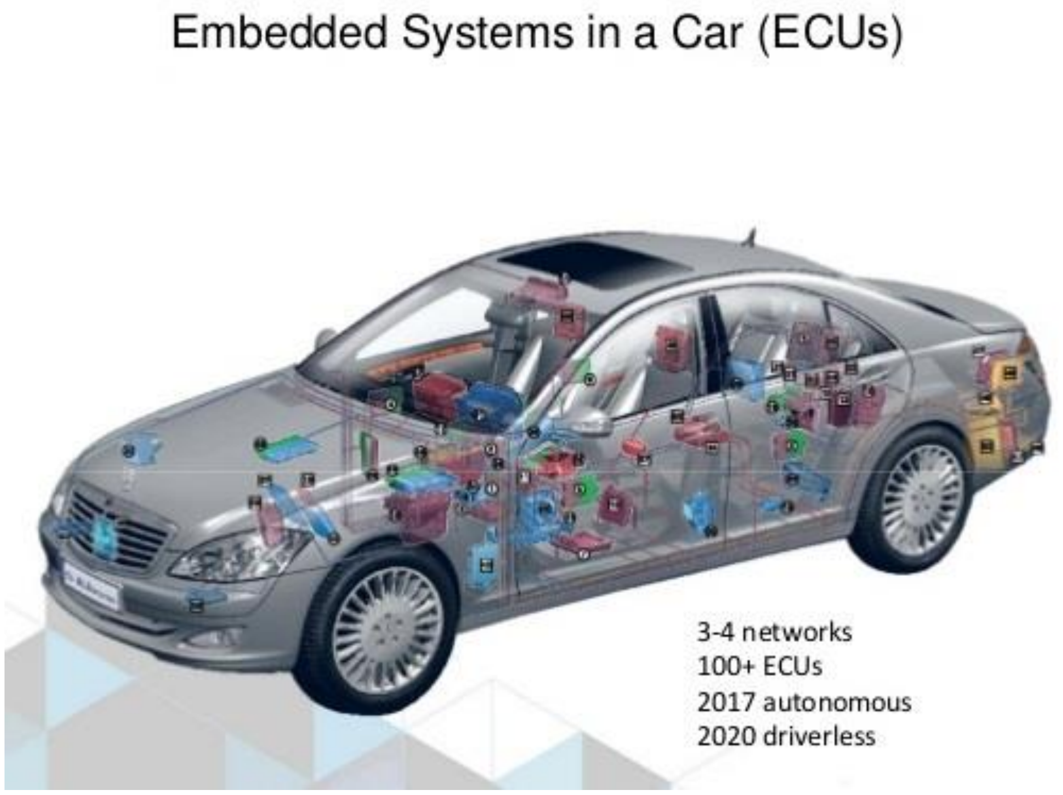


The most common examples are,

Cell-phones
Automatic Teller Machine
The Digital Interfaced Gasoline Station
Airborne Flight Control System
Automotive Engine Health Monitoring System
Home Security Systems
Modern Air-conditioners
Washing Machines
Medical Equipment
DVD Players
Printers
Medical Equipment
The list goes on……….
Wherever the microcontroller is used it's embedded computer.

The leading applications of embedded market are,

Communication
Computer Peripherals
Industrial Control and Automotive
Consumer Electronics
Test and Measurement
Medical
Military/Aerospace

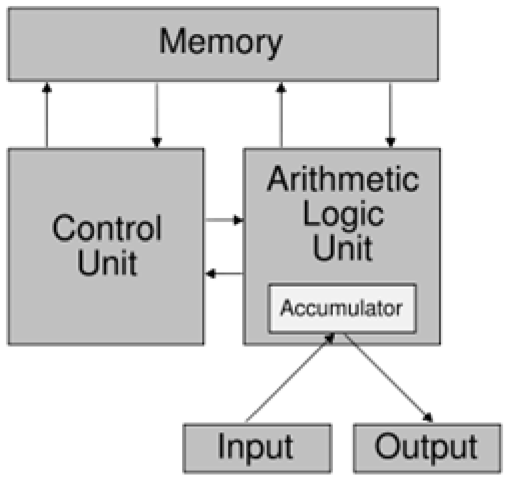


Background

A *computer* is made up of hardware and software. The hardware of a computer consists of five types of components:

1_ *Processor (CPU)*. A processor is also called the *central processing unit* (CPU) is responsible for performing all of the computational operations and the coordination of the usage of resources of a computer. A computer system may consist of one or multiple processors. A processor may perform general-purpose computations or special-purpose computations, such as graphical rendering, printing, or network processing. The processor consists of at least the following three components:

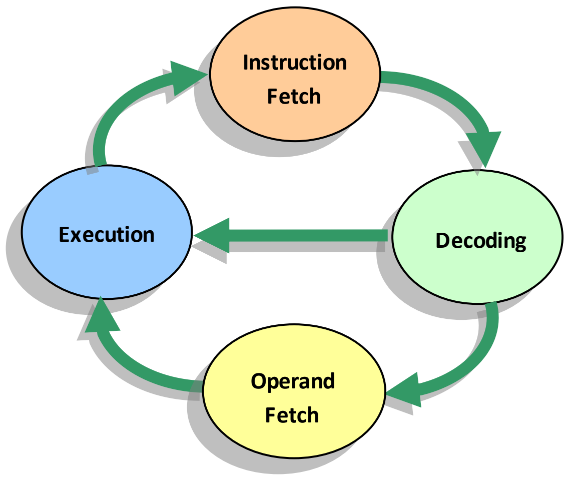
Registers. A register is a storage location inside the processor. It is used to hold data and/or a memory address during the execution of an instruction. Because the register is very close to the processor, it can provide fast access to operands for program execution. The number of registers varies greatly from processor to processor.



Arithmetic logic unit (ALU). The ALU performs all the numerical computations and logical evaluations for the processor. The ALU receives data from the memory, performs the operations, and, if necessary, writes the result back to the memory. Today's supercomputer can perform trillions of operations per second. The ALU and registers together are referred to as the *datapath* of the processor.

Control unit. The control unit contains the hardware instruction logic. The control unit decodes and monitors the execution of instructions. The control unit also acts as an arbiter as various portions of the computer system compete for the resources of the CPU. The activities of the CPU are synchronized by the system clock. The clock rates of modern microprocessors have exceeded 3.0 GHz at the time of this writing. The control unit also maintains a register called the *program counter* (PC) that keeps track of the address of the next instruction to be executed. During the execution of an instruction, the occurrence of an overflow, an addition carry, a subtraction borrow, and so forth are flagged by the system and stored in another register called a *status register*. The resultant flags are then used by the programmer for program flow control and decision making.

At any time, the processor state is one of four states: instruction fetch, instruction decode, operand fetch, or execution.



Processor states

Instruction sets – CISC and RISC

Any processor or CPU has a set of instructions that it recognizes and responds to; all programs are built up in one way or another from this instruction set. We want computers to execute code as fast as possible, but how to achieve this aim is not always an obvious matter. One approach is to build sophisticated CPUs with exotic instruction sets, with an instruction ready for every foreseeable operation. This leads to the CISC, the *Complex Instruction Set Computer*. A CISC has many instructions and considerable sophistication. Yet the complexity of the design needed to achieve this tends to lead to slow operation.

Another approach is to keep the CPU very simple and have a limited instruction set. This leads to the RISC approach – the *Reduced Instruction Set Computer*. The instruction set, and hence overall design, is kept simple. This leads to fast operation. One characteristic of the RISC approach is that each instruction is contained within a single binary word. That word must hold all information necessary, including the instruction code itself, as well as any address or data information also needed. A further characteristic, an outcome of the simplicity of the approach, is that every instruction normally takes the same amount of time to execute.

2_ *Input devices*. A computer is designed to execute programs that manipulate certain data. Input devices are needed to enter the program to be executed and data to be processed into the computer. There are a wide variety of input devices: keyboards, keypads, scanners, bar code readers, sensors, and so on.

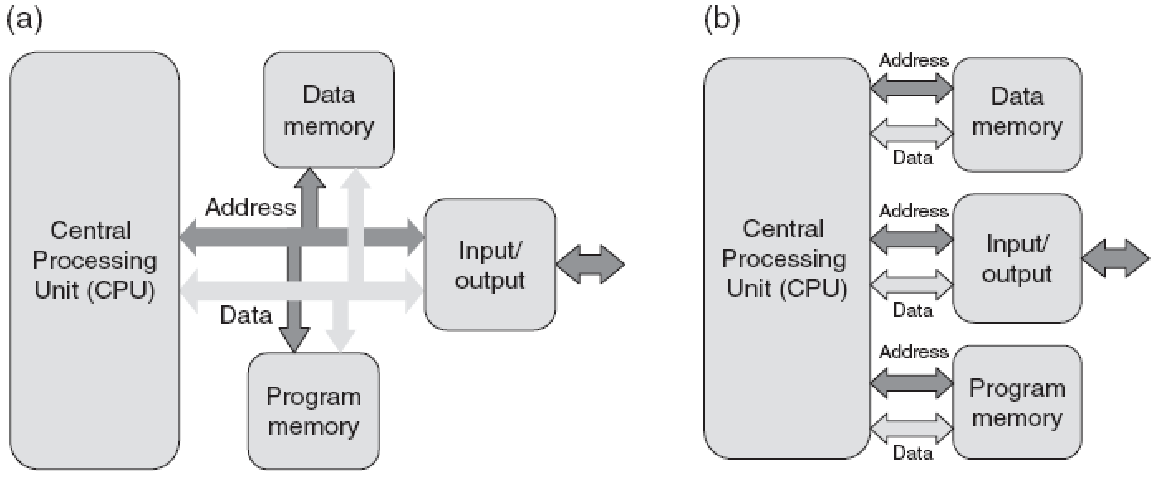
3_ *Output devices*. No matter if the user uses the computer to do certain computation or to find information from the Internet or a database, the end results must be displayed or printed on paper so that the user can see them. There are many media and devices that can be used to present the information: CRT displays, flat-panel displays, seven-segment displays, printers, light-emitting diodes (LEDs), and so on.

4_ *Memory devices.* Programs to be executed and data to be processed must be stored in memory devices so that the processor can readily access them.

5_ *Glue logic* (such as address decoder and buffer chips) is required to interface with the memory chips.

Memory and Processor Architectures

a) ***Von Neumann*** structure or architecture. The computer has just one address bus and one data bus, and
the same address and data buses serve both program and data memories. The input/output may also be interconnected in this way and made to behave like memory as far as the CPU is concerned.
b) ***Harvard* structure** is an alternative to the Von Neumann structure. Every memory area gets its *own* address bus and its *own d*ata bus.

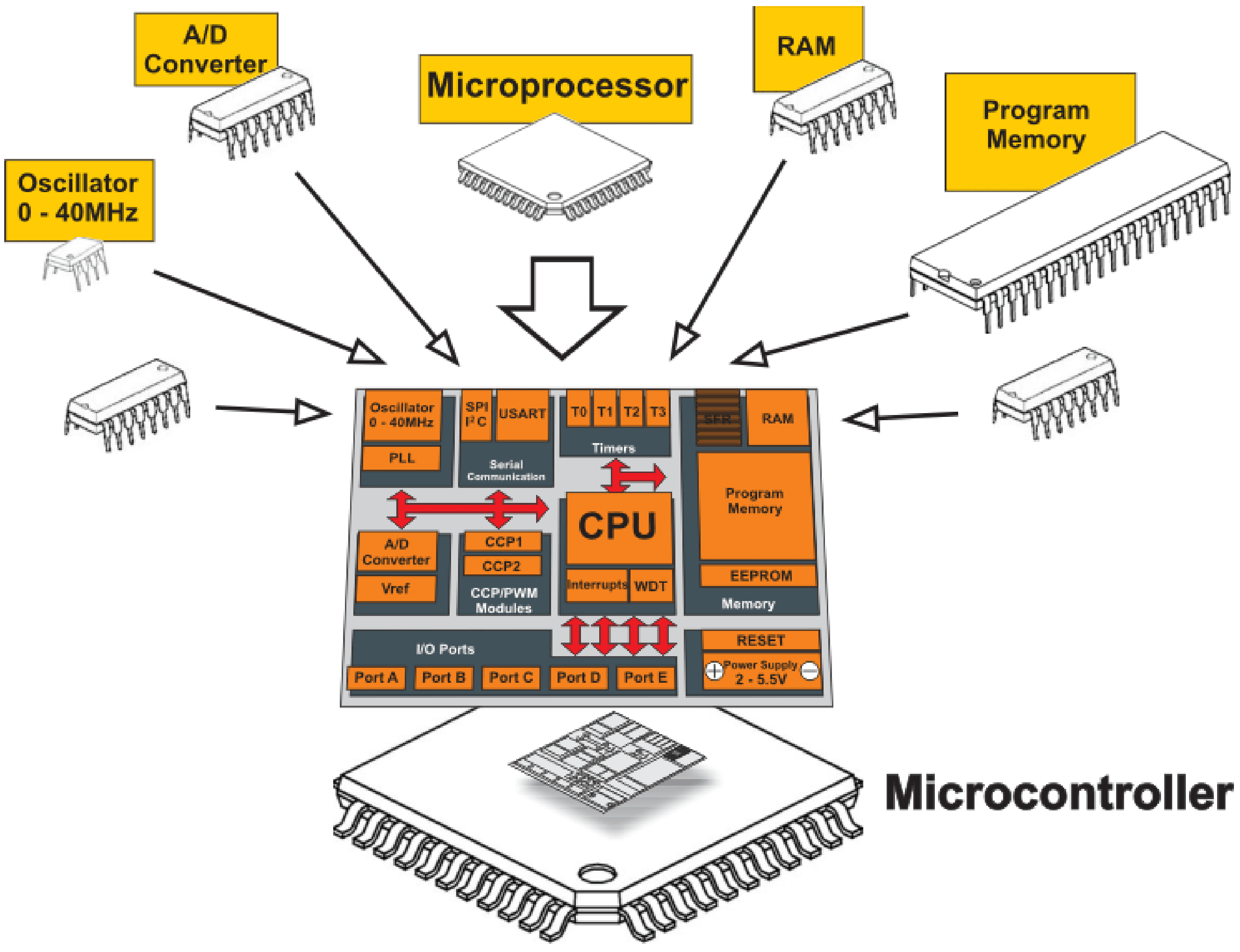


Organising memory access. (a) The Von Neumann way. (b) The Harvard way

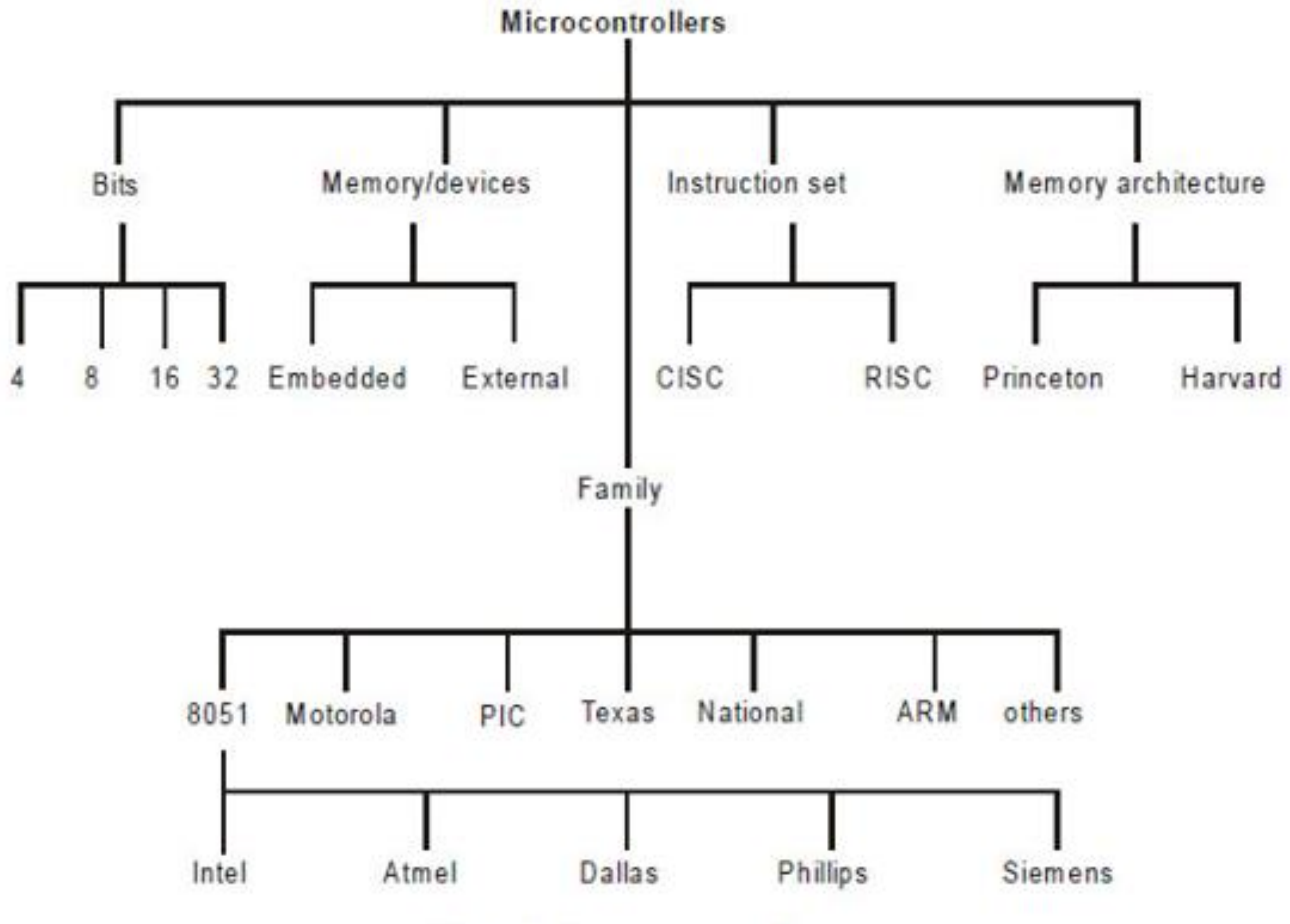
The Von Neumann structure is simple and logical, and gives a certain type of flexibility. The addressable memory area can be divided up in any way between program memory and data memory. However, it suffers from two disadvantages. One is that it is a ‘one size fits all’ approach. It’s the same data bus for all areas of memory, even if one area wants to deal with large words and another wants to deal with small. It also has the problem of all things that are shared. If one person is using it, another can’t. Therefore, if the CPU is accessing program memory, then data memory must be idle and vice versa. In the Harvard approach we get greater flexibility in bus size, but pay for it with a little more complexity. With program memory and data memory each having their own address and data buses, each can be a different size, appropriate to its need, *and* data and program can be accessed simultaneously. On the minus side, Harvard reinforces the distinction between program and data memory, even when this distinction is not wanted. This disadvantage may be experienced, for example, when data is stored in program memory as a table, but is actually needed in the data domain.

Microcontrollers

A microcontroller, or MCU, is a computer implemented on a single very large scale integrated (VLSI) circuit. In addition to those components contained in a microprocessor, an MCU also contains some of the following peripheral components:
_ Memory
_ Timers, including event counting, input capture, output compare, real-time interrupt, and watchdog timer
_ Pulse-width modulation (PWM)
_ Analog-to-digital converter (ADC)
_ Digital-to-analog converter (DAC)
_ Parallel I/O interface
_ Asynchronous serial communication interface (UART)
_ Synchronous serial communication interfaces (SPI, I2C, and CAN)
_ Direct memory access (DMA) controller
_ Memory component interface circuitry
_ Software debug support hardware



Since their introduction, MCUs have been used in almost every application that requires certain amount of intelligence. They are used as controllers for displays, printers, keyboards, modems, charge card phones, palm-top computers, and home appliances, such as refrigerators, washing machines, and microwave ovens. They are also used to control the operation of engines and machines in factories. One of the most important applications of MCUs is probably the automobile control. Today, a luxurious car may use more than 100 MCUs. Today, most homes have one or more MCU-controlled consumer electronics appliances. In these applications, people care about only the functionality of the end product rather than the MCUs being used to perform the control function. Products of this nature are often called *embedded systems*.



Types of microcontrollers

Some example of embedded systems Boards