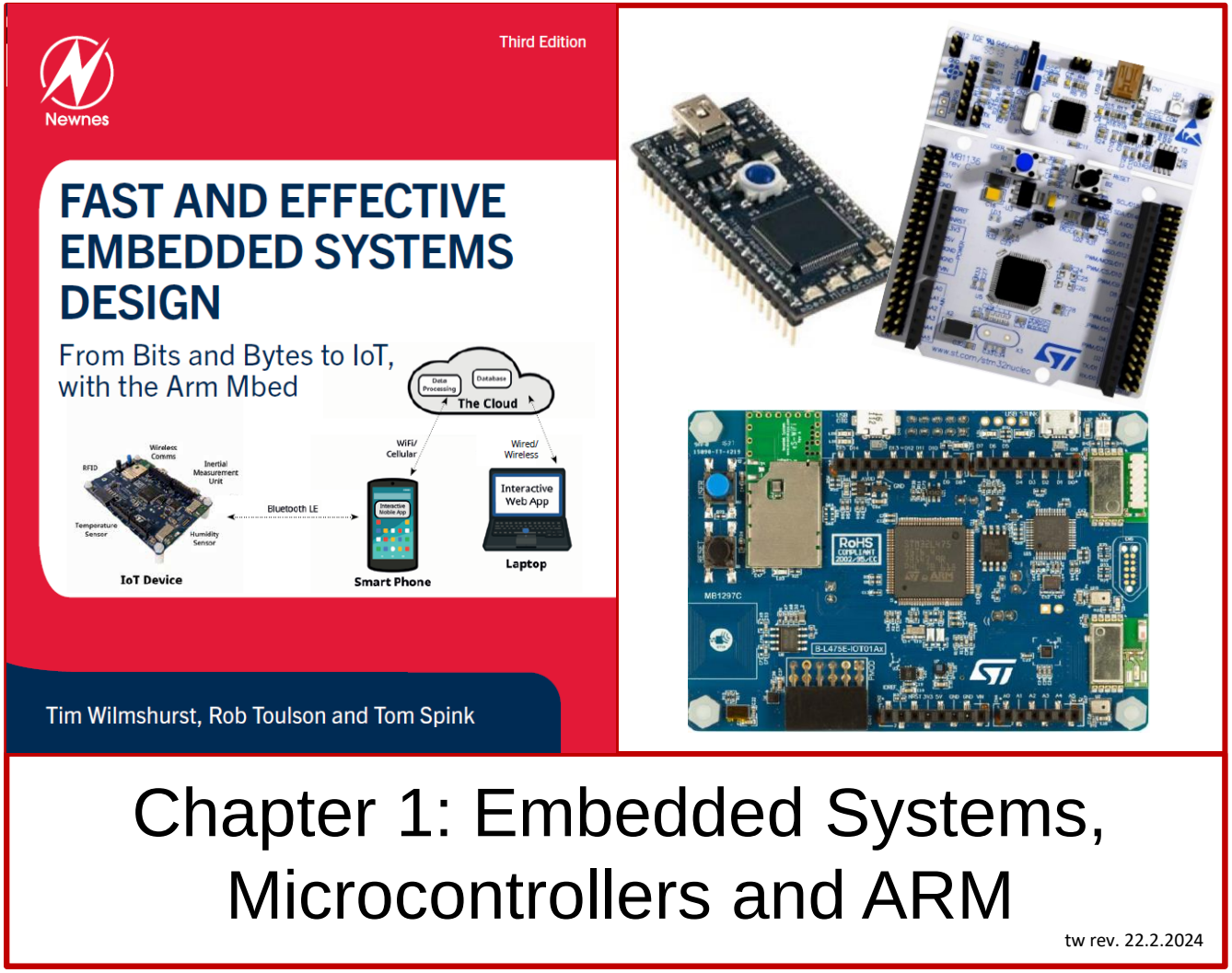


Chapter 1: Embedded Systems, Microcontrollers and ARM

tw rev. 22.2.2024

If you use or reference these slides or the associated textbook, please cite the original authors' work as follows:

Wilmshurst, T., Spink, T. & Toulson, R. (2024). Fast and Effective Embedded Systems Design – From Bits and Bytes to IoT, with the ARM Mbed (3rd edition), Newnes, Oxford, ISBN: 978-0323951975.

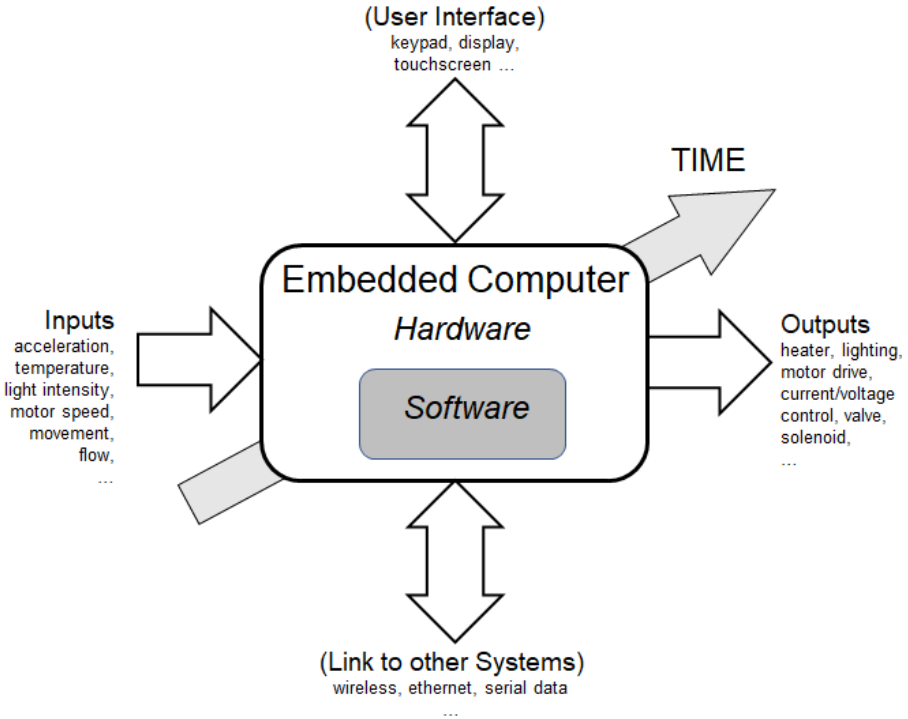
www.embedded-knowhow.co.uk

What is an Embedded System?

An embedded system is a product which is controlled by a computer inside it, but whose overall function is not computing. Examples are endless, and include washing machines, office equipment, and car engine management.



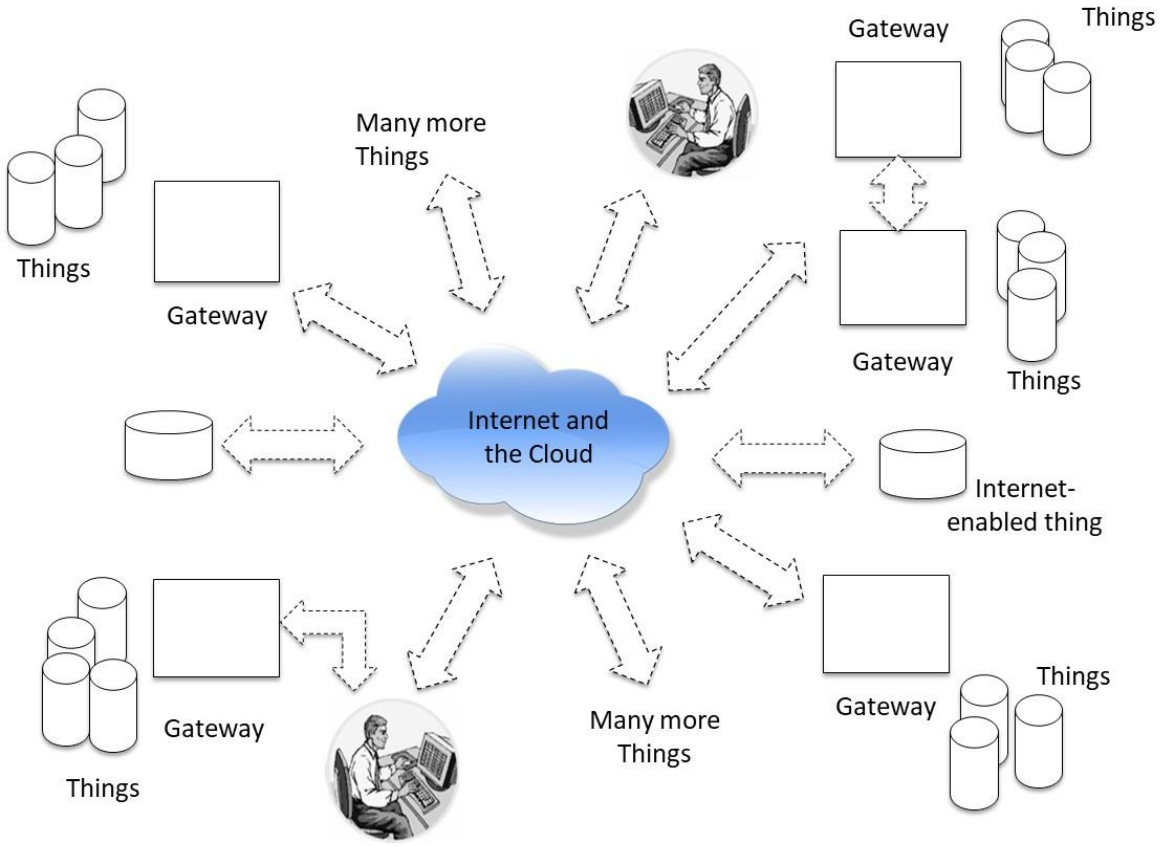
Many embedded systems are networked together, thus each forms a sub-systems of a larger system.





The Internet of Things

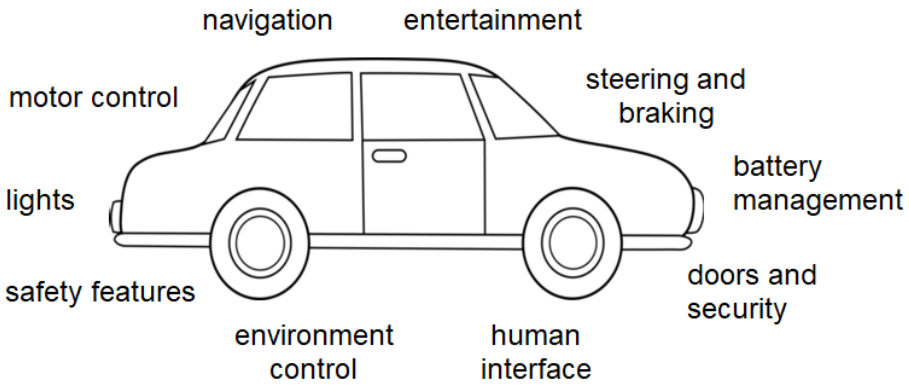
- Embedded systems have become increasingly connected to the internet.
- The Internet of Things (IoT) is the name given to the resulting worldwide network of devices. Each “Thing” is an embedded system.
- These allow remote access to information that can be used to control and enhance diverse activities.
- The IoT includes billions of sensors and databases used for monitoring through the internet, from travel and transport, to environmental, medical, and industrial.

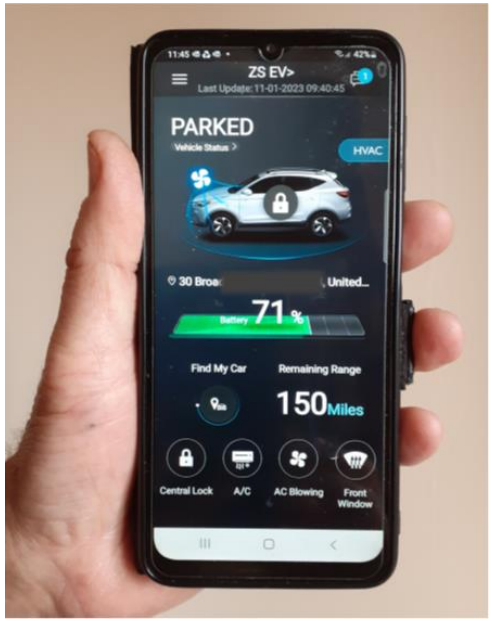


Example Embedded System Application: the Electric Car

- Multiple embedded systems are used in the modern electric vehicle (EV), with possible categories of use shown.
- Each category contains numerous embedded systems, some controlling one or few variables, others managing a sophisticated array.
- Some of these embedded systems link with others with no human interface.
- Others are user-accessible; for example the app shown is used to monitor vehicle status, and to control things like battery charging and pre-drive heating.







Microprocessors and Microcontrollers

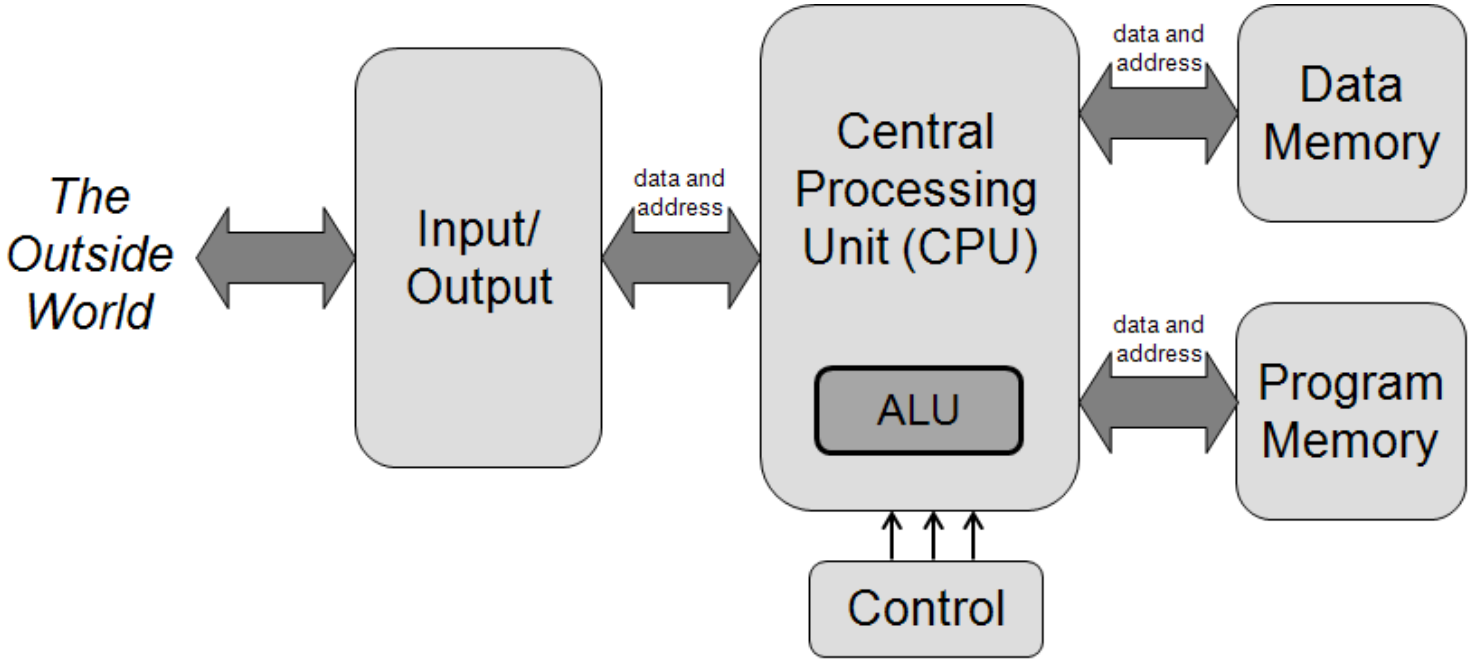
The Figure shows the essential elements of a computer system.

A computer can perform arithmetic or logical calculations. It does this in a digital electronic circuit called the ALU, or *Arithmetic Logic Unit*.

The ALU is placed within a larger circuit, called the *Central Processing Unit* (CPU), which provides some of the supporting features that it needs.

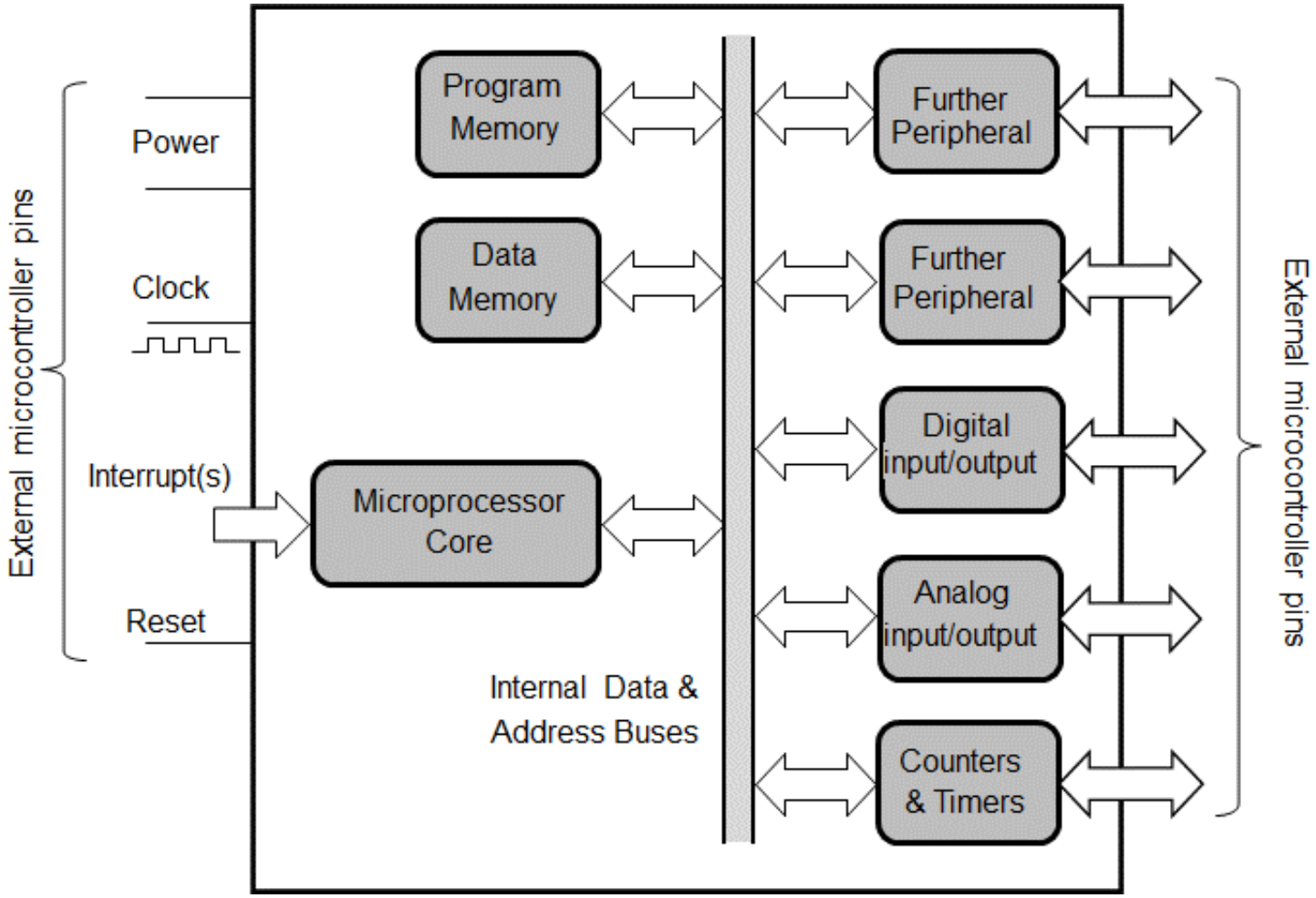
Which arithmetic or logical calculation the ALU does depends on a digital code which is fed to it, called an instruction.

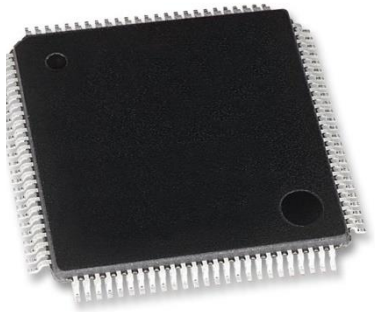
If the ALU is kept busy, by feeding it a sequence of instructions, and also passing it the data it needs to work on, then we have a simple computer.



The Microcontroller

A microcontroller takes the essential features of the computer, and adds to these the features that are needed for it to perform its control functions. It's useful to think of it as being made up of three parts: ***core, memory and peripherals***,



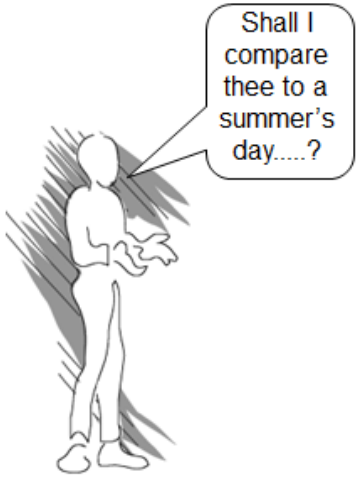


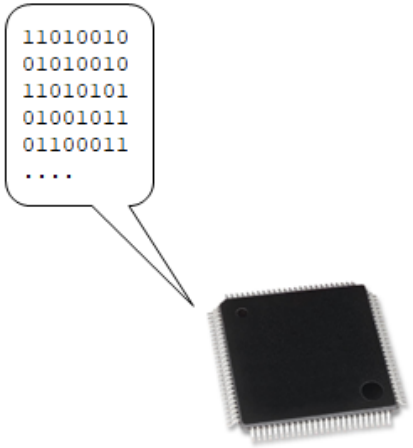
Code Development - what's so Special about C/C++?

As programmers, it is our goal to produce a program which makes the microcontroller do what is needed. That program must be a listing of instructions, in binary, drawn from the instruction set. We sometimes call the program in this raw binary form *machine code.*

It's nearly impossible for us as humans to work with the binary numbers which form the instruction set. A step to programming sanity is to use a *High Level Language* (HLL), which generates the machine code through a *compiler.*

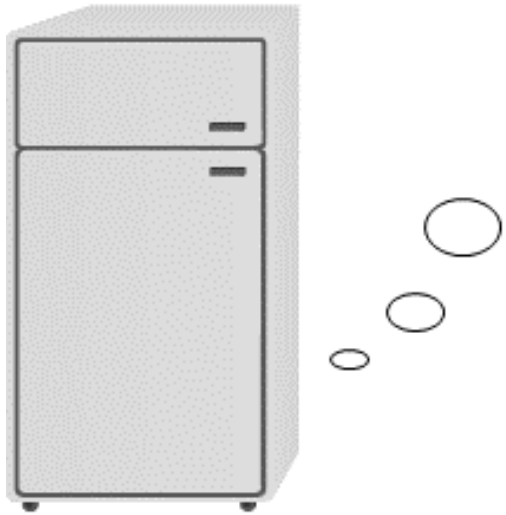
C is a HLL with clear advantages. This is because it is simple, and has features which allow access to the hardware when needed. A step up from C is C++, which is also widely used for more advanced embedded applications.

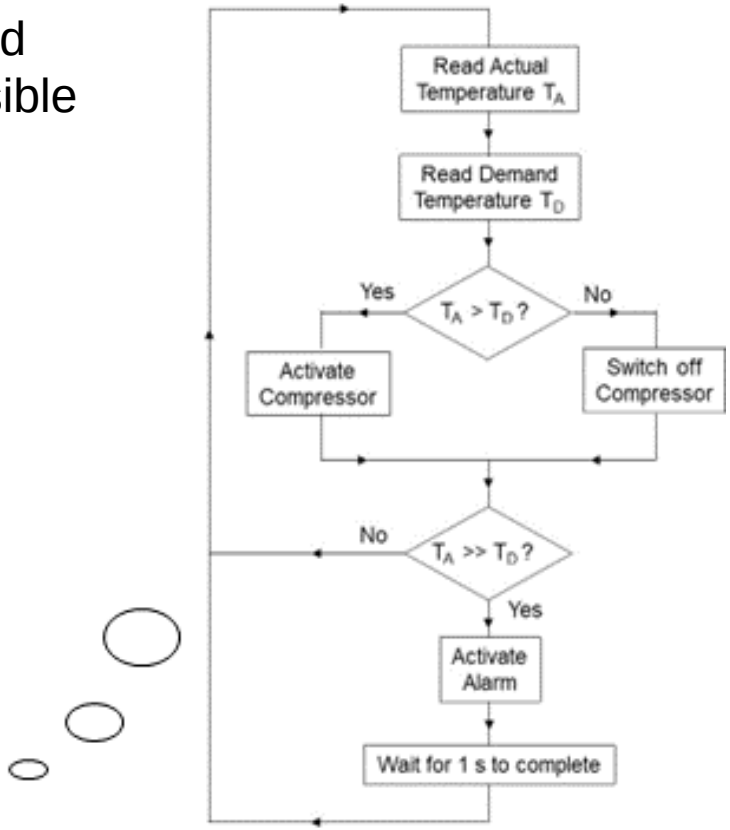




Flow Diagrams, and Simple Sequential Programming

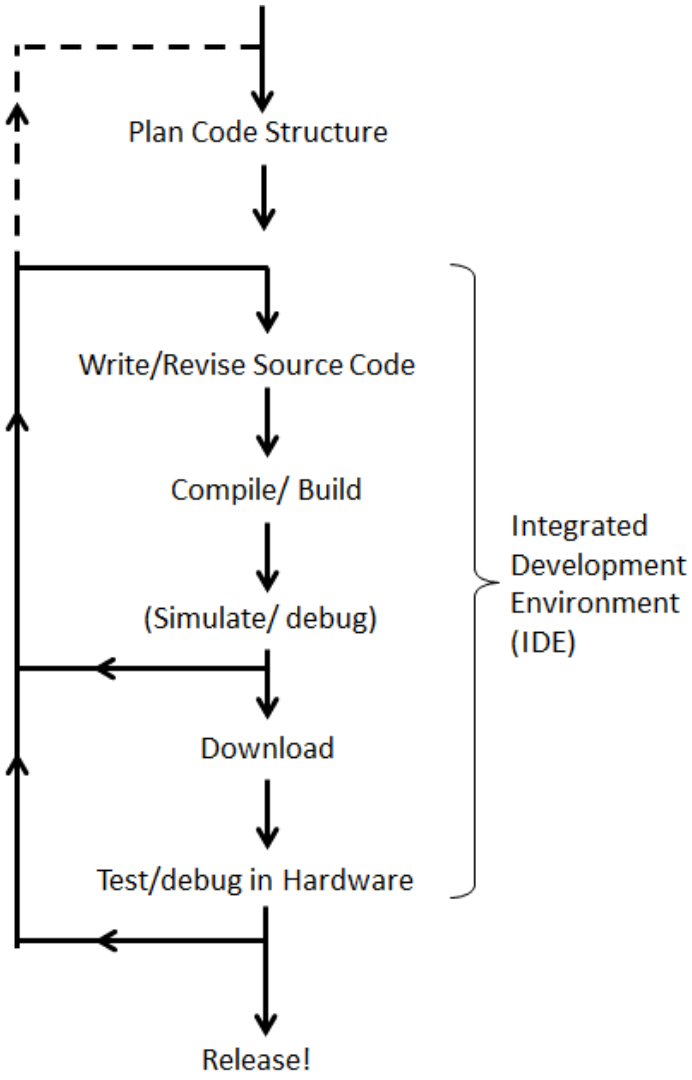
- We will be writing programs which read in data, make computations and decisions, and generate outputs.
- A simple example is shown, a refrigerator controller, where a flow diagram represents the program structure.
- Here we use only two symbols: a box for any sort of action, and a diamond for decision. The diamond contains within it a yes/no choice, with two possible exit points and one entry point.
- As with many programs in the embedded world, which just go on and on running, the program is structured as a continuous loop.





The Code Development Cycle

- Most development stages are made within an *Integrated Development Environment* (IDE), a software package running on the *host computer*.
- The source code, writing (in our case) in C/C++, is written in a text editor in the IDE. As well as original code, the programmer can use libraries of code already available.
- The source code is converted into the binary machine code by the compiler, running as part of the IDE.
- Program download to the *target system* requires temporary connection to the host computer.



The World of ARM - a Little History



In 1981 the Acorn computer became a popular machine in schools and universities in the UK. It used an 8-bit processor made by MOS Technology.

Around the same time IBM produced its very first PC, based on a more powerful Intel 16-bit microprocessor.

Throughout the 1980s the influence of the IBM PC grew, and its smaller competitors began to fade.

Designers at Acorn realised:

- that they had the capability to design the microprocessor itself and didn't need to buy it in from elsewhere.
- that their future may not lie in selling the completed computer.
- that you don't need to manufacture silicon to be a successful designer, what matters is the ideas inside the design. These can be sold as IP, intellectual property.

A group of them formed *Advanced RISC Machines Ltd,* later simply called ARM. They applied the above principles.

What does RISC Mean?

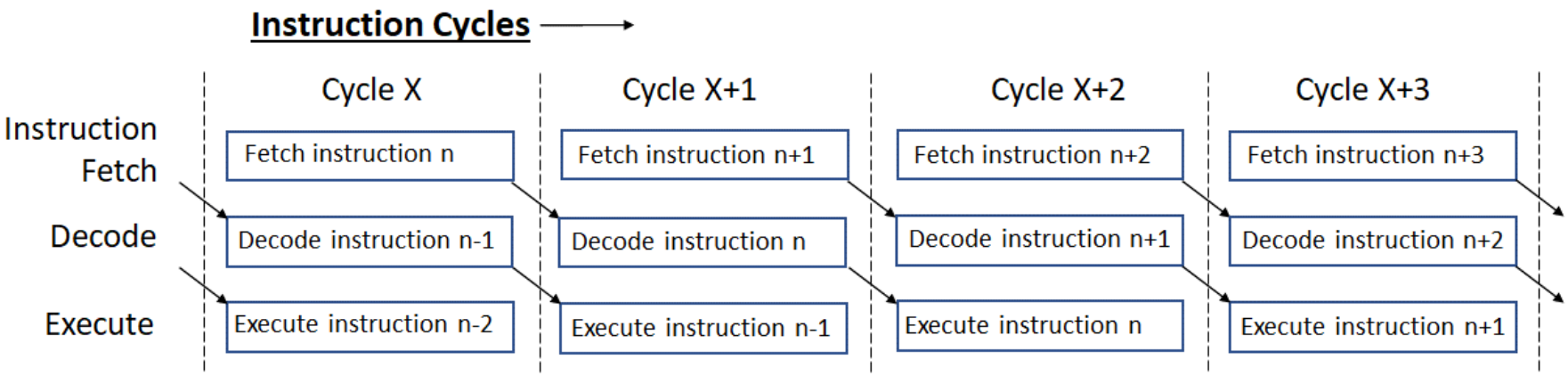
One approach to CPU design is to insist on keeping things simple, and have a limited instruction set. This leads to the RISC approach – the *Reduced Instruction Set Computer*. This differs from the *Complex Instruction Set Computer* (CISC).

The RISC approach looks like a "back to basics" move. A simple RISC CPU can execute code fast, but it may need to execute more instructions to complete a given task, compared to its CISC cousin. Characteristic of the RISC approach include:

- each instruction is contained within a single binary word.
- every instruction normally takes the same amount of time to execute.
- Other useful design features can then be implemented, for example pipelining - an approach by which as one instruction is being executed, the next is already being fetched from memory.

Pipelining

- Because every instruction normally takes the same amount of time to execute, *pipelining* can be implemented.
- As one instruction is being executed, the next is already being fetched from memory.
- This is illustrated for a 3-stage pipeline: the first stage fetches the instruction, the next decodes it, and the third executes. The diagram shows four cycles of operation.
- Pipelining can dramatically increase the speed of operation of a CPU, because things are made to happen simultaneously, rather than in turn.



The world of ARM - The Cortex core

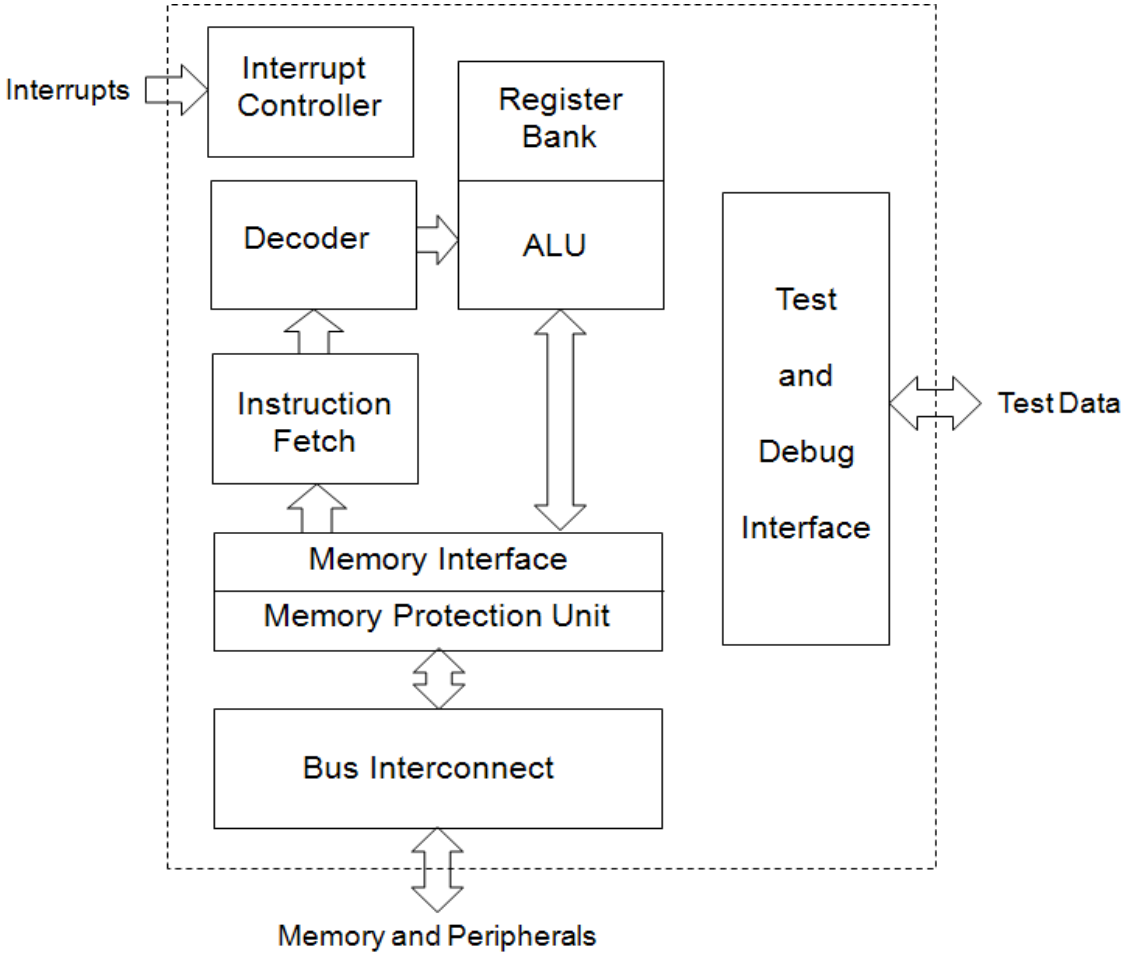
- The Cortex microprocessor family was introduced by Arm in 2004. Cortex-A and Cortex-R were introduced for higher level applications, while Cortex-M was intended for smaller, microcontroller-oriented uses.
- These implement Thumb-1 and Thumb-2 instruction sets, sub-sets of the original ARM instruction set.

Core	Main Characteristics	Instruction Set
Cortex-M0	With its minimum size and power consumption, the M0 is of special interest for low-cost, low power, or small size embedded systems. 3-stage pipeline. Up to 32 interrupts.	56 Thumb-1 and Thumb-2 instructions. Restricted multiply or divide instructions.
Cortex-M0+	Similar to Cortex-M0, but further reduced power consumption, partly due to 2-stage pipeline. Up to 32 interrupts.	56 Thumb-1 and Thumb-2 instructions. Restricted multiply or divide instructions.
Cortex-M1	Similar to Cortex-M0, but designed as a “soft core” which can be dropped into an FPGA (Field Programmable Gate Array) design. 3-stage pipeline. Up to 32 interrupts.	56 Thumb-1 and Thumb-2 instructions. Restricted multiply or divide instructions.
Cortex-M3	The first Cortex-M device to be launched, so often viewed as a point of reference for the others. Targeted at common embedded applications, including automotive and industrial. 3-stage pipeline with branch speculation. Up to 240 interrupts.	74 instructions including all Thumb-1 and Thumb-2. Multiply and divide instructions included.
Cortex-M4	All the features of Cortex-M3, with additional digital signal processing (DSP) instructions, and optional hardware floating point unit (FPU). 3-stage pipeline with branch speculation. Up to 240 interrupts.	137 instructions, plus 32 for the FPU. Multiply, divide and DSP instructions included.
Cortex-M7	A high-performance core with significant advances over the Cortex-M4. Includes a 6-stage pipeline with branch speculation and optional FPU. Up to 240 interrupts.	Multiply, divide, FPU and DSP instructions available.
Cortex-M23 and upwards	Ongoing developments and advances.	

The World of ARM - The Cortex Core

A simplified block diagram of the Cortex-M3 core is shown, incorporating computer features already seen, e.g. the ALU and pipelining.

The core also includes an *interrupt interface*. Interrupts are external inputs used to force the CPU to divert from its current activity, and jump to another section of code.



Chapter Review

- An embedded system contains one or more tiny computers, which control it, and give it a sense of intelligence.
- The embedded computer usually takes the form of a microcontroller, which combines microprocessor core, memory and peripherals, all on the same IC.
- Embedded system design combines hardware (electronic, electrical and electro-mechanical) and software (program) design.
- The embedded microcontroller has an instruction set. The code developed by the programmer must, through one means or another, be converted to instructions from this instruction set.
- Most embedded system programming is done in a high level language, with C or C++ often being preferred.
- A compiler is used to convert the program into the binary code, drawn from the microcontroller instruction set.
- Arm has developed a range of effective microprocessor and microcontroller designs, widely applied in embedded systems.
- The Arm Cortex-M series microcontroller cores are intended specifically for microcontroller applications, with different versions targeted at different use scenarios.