

# Bits AND Chips

Boolean logic and machine architecture

# Trainer's Notes



## Background

Breaking open a computer, or other digital device to identify the main components can help to reinforce the basic input / process / output / storage model most students will be familiar with. However the appearance of the components reveals very little about how they work. Answering the ‘How do they do that?’ question that inevitably springs from a curious mind requires an appreciation of the layers of hardware abstraction involved in this model. At its lowest level, this requires us to demonstrate that a few basic operations, namely adding, shifting and inverting bit patterns can be achieved through simple mechanisms, usually implemented as electrical circuits. At the heart of these are the fundamental ‘gates’ that perform the logical operations of AND, OR and NOT. These logical operations appear in many other contexts, but exploring machine architecture adds concrete detail to the abstract computer model they are familiar with.

Establishing these ‘first principles’ allows teachers to appreciate that a computer is built from combinations of small circuits, encapsulated into ever larger ‘chips’, the detail of which is hidden to designers. These are further combined to provide the functionality of the core components in a computer. For children, familiarity with these operations allows them to express logical conditions to trigger events, as well as acting as an exemplar for the concept of abstraction; a means of understanding and coping with complexity. At a higher level of abstraction, our computer model identifies the key components such as the memory, control unit and ALU required to run a program. By simulating the operations required to fetch, decode and execute a program stored in memory we can make a bridge between the software written and the hardware.

Children will be familiar with the idea of a program as a set of instructions, followed in sequence. So they can be executed on a computer, the program instructions we write are translated into bit patterns, just like those used to represent data. This process is hidden, so children need to be encouraged to consider how, if a computer can only understand bit patterns, the code they write can be translated into these digital instructions. Again, here is an opportunity to appreciate the layers of software abstraction involved when coding in something like, say Scratch, or Python.

Investigating the translation process throws up insights into the nature of how statements in formal languages can be evaluated and syntax errors, for example, detected. Children often have the capacity to articulate things in very small steps and some simple low level programming, using child friendly resources should be within their grasp. Practical activities help provide another different programming environment in which to develop their computational thinking capacity. Moreover, it helps them appreciate the many steps required to implement even simple programming structures at a low level, and the sort of tasks that might be programmed at this level.

In recent years multi core architecture, using several processors has developed rapidly. Now that they are familiar with machine architecture, considering the new challenges required to execute a program on this sort of ‘parallel’ hardware again makes the link between the layers of hardware and software abstraction. The challenge of harnessing the potential of massive, multi-core processors is a hot topic in Computer Science. And it’s not just parallel programming in one machine. Distributed computing lies behind many network services – in fact, modern web services are a very good example of distributed tasks.

Although we don’t address this in the unit, one final consideration. As already noted, children will already be familiar with the idea of a program as a set of instructions, followed in sequence. However, their experience of a computer does not match this. Most of the time, a pc will appear to be doing many things at once, so into this model of the operation of computer hardware we need to introduce the notion of multi-tasking and consider how a computer can schedule competing demands on its processing power.

# The aim of the day

The aim of the unit is summarised in the learning objectives for teachers. The primary aim is to **educate teachers** and illustrate the breadth and depth of Computer Science. The specific outcomes **for teachers** from this unit, are to:

- Become familiar with Boolean logic and machine architecture.
- Be able to express combinational logic circuits as truth tables.
- Make connections with exemplar circuits in processors (adders).
- Investigate exercises exploring / coding bitwise operations.
- Become familiar with the fetch execute cycle, Von Neumann bottleneck and the potential of parallel processing.

Ultimately, we hope they will leave inspired to introduce some of these key ideas at KS3

The purpose of this Tenderfoot session is to equip **trainers** with material and ideas to meet these outcomes and broaden the outlook of teachers new to Computing. It hopes to provide a buffet of resources on which teachers can draw, to enrich their Key Stage 3 lessons, at the same time as meeting the key aim: providing greater depth of knowledge for **teachers** themselves. Developing **teachers** is the focus, not providing activities for pupils.

Throughout the material there are references to famous computer scientists and lots of pointers to other material. The aim is to encourage teachers to delve deeper and take the ideas further.

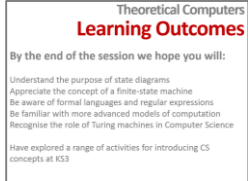
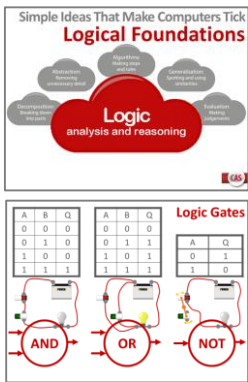

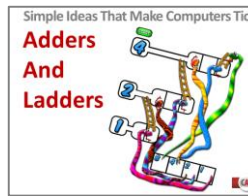
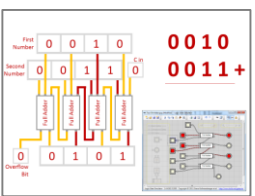
Before delivering the unit, please check you are comfortable with the narrative and references to other material. Some of the sessions are complex and require preparation beforehand. There are teacher's notes which include a summary of each activity. Ensure you rehearse the delivery to familiarise yourself with transitions and animations. Where it is required, the slides include further detailed notes.


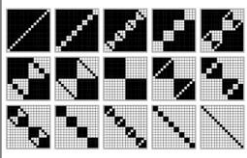
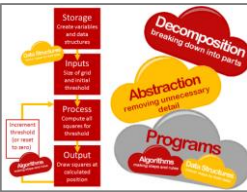

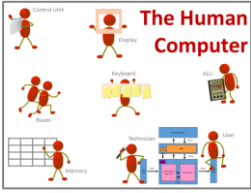


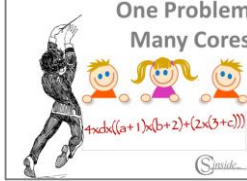

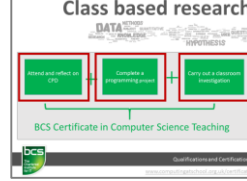
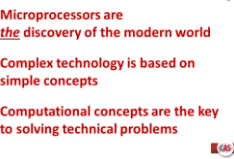
Keep in mind the main purpose of the session – to engage experienced teachers with some of the deeper ideas in Computer Science they may not be familiar with. This sort of background knowledge is broadly ‘A Level’ standard. In time, all Computing teachers should have this grounding, so the aim is to empower the experienced teachers, providing them with material they can deliver and use with less experienced colleagues in shorter training sessions.

There are lots of exercises and supplementary material. The pace should be fast, with the assumption that the audience are experienced teachers, probably already teaching to GCSE level, with some familiarity with the concepts of Computer Science. As such they will not need to work through every activity in full. Sometimes it will be sufficient to part complete an activity so teachers ‘get it’ and can discuss how it might be used. Judgement is required and the timings below are indicative, to help with planning. Always be flexible and encourage discussion and engagement. Details of each activity are given in the teachers notes. Further guidance on the narrative, slide transitions and animation can be found in the slide notes.

## Indicative Timetable

The trainer’s presentation is broken down into 5 sections, with several formal inputs and 5 main practical activities outlined below:

 <p><b>Theoretical Computers Learning Outcomes</b></p> <p>By the end of the session we hope you will:</p> <ul style="list-style-type: none"> <li>Understand the purpose of state diagrams</li> <li>Appreciate the concept of a finite-state machine</li> <li>Be aware of formal languages and regular expressions</li> <li>Be familiar with more advanced models of computation</li> <li>Recognise the role of Turing machines in Computer Science</li> </ul> <p>Have explored a range of activities for introducing CS concepts at KS3</p> <p><b>10 minutes</b></p>	<p>Establishes key outcomes from the day for teachers (5 mins).</p> <p>Sets the session in the context of the key educational goal: developing computational thinking and associated concepts.</p> <p>Emphasises that all the concepts collectively involve logical thinking. (3 mins).</p>
 <p><b>Simple Ideas That Make Computers Tick Logical Foundations</b></p> <p>Logic analysis and reasoning</p> <p><b>Logic Gates</b></p> <p>AND OR NOT</p> <p><b>60 minutes</b></p>	<p>Introducing fundamentals of combinational logic / truth tables through circuits - practical activity need not be done. (20 mins)</p> <p>Human Logic Gates activity (20 mins)</p> <p>Taught input introducing NAND, NOR, XOR, algebraic representation, generalising from circuits, distinguishing between interface and implementation. Proving any logic can be created from combinations of AND, OR and NOT. (20 mins)</p> 
 <p><b>Simple Ideas That Make Computers Tick Adders And Ladders</b></p> <p><b>75 minutes</b></p>	<p>Adders and ladders activity (20 minutes)</p> <p>Taught input - binary addition and half adder (10 mins)</p> <p>Practical lead session building 4 bit adder using logic gate simulator. (45 mins)</p> 

<p>Simple Ideas That Make Computers Tick <b>Munching Squares</b></p>  <p>75 minutes</p>	<p>Introducing Munching Squares and bitwise operations. (15 mins)</p> <p>Presenter led session analysing the problem and designing the solution. Introduces 2D arrays and emphasises development of algorithms and data structures independent of language implementation. (30 mins)</p> <p>Evaluation of programming languages and implementation using Game Maker. (30 mins)</p>	 
<p>Simple Ideas That Make Computers Tick <b>Inside the Machine</b></p>  <p>60 minutes</p>	<p>Taught input on a conceptual model of a computer. (10 mins)</p> <p>Human Computer activity. (30 mins)</p> <p>Introducing the Little Man Computer simulator and the Fetch Execute Cycle – potential to extend as practical challenges (20 mins)</p>	 
<p>Simple Ideas That Make Computers Tick <b>The Need For Speed</b></p>  <p>60 minutes</p>	<p>Taught input on Moore's Law, Von Neumann bottleneck and potential for parallelism. (15 mins)</p> <p>One Problem, Many Cores activity. (45 mins)</p>	
<p><b>Class based research</b></p> <p>Why?</p>  <p>10 minutes</p>	<p>A short discussion to promote classroom research and encourage reflective practice.</p> <p>Draw out suggestions for potential research areas and mention possible techniques.</p> <p>Ends with a quick promotion of the BCS Certificate in Computer Science Teaching.</p>	
<p><b>Summary</b></p> <p>Microprocessors are <u>the</u> discovery of the modern world</p> <p>Complex technology is based on simple concepts</p> <p>Computational concepts are the key to solving technical problems</p>  <p>5 minutes</p>	<p>Final summary emphasising connection between the development of technology, key concepts and computational thinking. (5 mins).</p>	

Above all else, remember that the aim is to empower attendees to offer similar sessions to colleagues. It should be inclusive, enjoyable and embody the CAS ethos of collegiality: There is no 'them', only us!

When someone books to attend the training session, send a prompt acknowledgment informing them when final confirmation and further details will be sent. Set a cut-off date, at which point you decide if there are enough bookings to make a viable session.

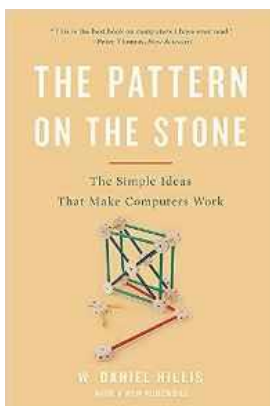
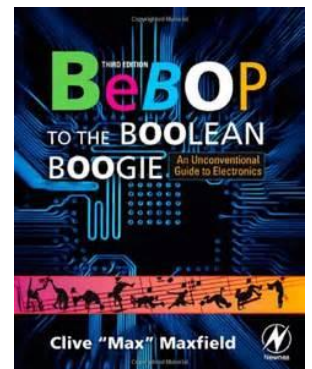
Once you have enough people booked, contact them again with brief details and suggested prior reading. Although not essential, by suggesting some prior reading you are indicating that this is in depth CPD which requires some commitment on the part of the attendees. It also gives you a chance to establish some dialogue with attendees prior to the event. With a week to go, you could mail a reminder and enquire about the reading and whether it would be useful for teaching. This helps keep the attendees focused on the event.

## Prior Reading

It is likely that most attendees will have some familiarity with basic Boolean logic. A gentle introduction to the idea of Moore's Law and the potential offered by parallelism is provided in the first RI Christmas Lecture 2008 from Chris Bishops. 'The Need For Speed' is suggested for pupils in the last session of this unit, so prior familiarisation would be advantageous for teachers. It can be found at [goo.gl/HJyCDg](http://goo.gl/HJyCDg). The materials produced by Cork University to celebrate Boole's bicentenary, Boole2School, are excellent. Their introductory puzzles would be another 'light' starter, particularly for colleagues who lack any background in Boolean logic: [goo.gl/Vsw3WM](http://goo.gl/Vsw3WM). All teachers would be well advised to look at all the materials available on their site.

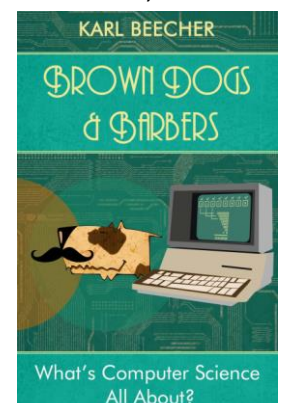
## Further Reading

If attendees wish to explore the ideas of Boolean logic further the most comprehensive book to point them towards is 'Beebop To The Boolean Boogie', written by Clive 'Max' Maxfield. An unconventional guide to electronics written in an entertaining and quirky style, the first section (13 chapters) provides a very thorough background to the fundamental ideas that goes well beyond that required to teach A-Level in little more than 150 pages. Moreover, within those pages it makes clear the links to other key ideas looked at in other Tenderfoot units such as Binary, Analogue to Digital conversion and Finite State Machines. Primarily an electronics primer it also makes clear how logic functions can be implemented in arrangements of transistors.



Two general books are also worth highlighting, and both are mentioned during the session. Although long out of print, The Pattern On The Stone, written by W. Daniel Hillis is an excellent introduction to 'the simple ideas that make computers work'. Chapter 1 in particular is a well articulated explanation of the fundamentals, illustrating how abstraction can separate the idea from the implementation, leaving readers free to worry about the 'difference that makes the difference'. For a very succinct, but comprehensive introduction to how this fits into the bigger picture, the first 3 chapters cannot be bettered.

Finally Karl Beecher's Brown Dogs and Barbers is a wonderful, accessible entry level book that all teachers should read. With no chapter more than 4 pages long, it is easy to read and provides a thorough introduction to Computer Science, rooting the technological developments in the cerebral challenges posed by its explosive development. A sample chapter is included in the resources.







Well before the session is due to take place ensure you have considered computer access. Check whether attendees will be logging on to institution machines or bringing their own laptops. If BYOD, ensure that is made clear in any prior publicity. Check that the venue has a projector and speakers.

An essential piece of software required is the logic gate simulator created by Steve Kollmansberger. Included in the resources, it could be distributed in advance, or you could suggest downloading from [goo.gl/wTTVEV](http://goo.gl/wTTVEV). To complete the implementation of Munching Squares, Game Maker is also required. Check for the latest free version at [goo.gl/55gvfn](http://goo.gl/55gvfn).

Ensure there is enough space for the Human Computer and Human Logic activities, and enough spare tables to set up the One Problem, Many Cores activity in advance.

Ensure you have the following general material:

- Facilities for taking notes (paper and pens)
- A3 Computational Thinking Posters
- CAS Publicity: Copies of SwitchedON, BCS Certificate flyers and any local information

## Attendee / Trainers Materials and Resources

Attendee materials are in black, requiring one copy per person. Supplementary trainer's materials for demonstration purposes are indicated in red (single copy only required). Continues overleaf.

Activity	Materials (Per Attendee)	
Logical Foundations	Class sets of batteries, switches, bulbs and wires	<input type="checkbox"/>
	Combinational Logic Exercises	<input type="checkbox"/>
	Human logic circuit resources plus sticky tape	<input type="checkbox"/>
	Dotsy Scorecards and 5 dice	<input type="checkbox"/>
	Teachers Notes	<input type="checkbox"/>
Adders and Ladders	Adders and ladders board + 6 counters for each group.	<input type="checkbox"/>
	Adders and ladders handout (each)	<input type="checkbox"/>
	Logic Gate Simulator available	<input type="checkbox"/>
	Logic Circuit Challenges	<input type="checkbox"/>
	Teachers Notes	<input type="checkbox"/>
Munching Squares	Munching Squares templates	<input type="checkbox"/>
	Munching Squares implementation handout	<input type="checkbox"/>
	Game Maker available	<input type="checkbox"/>
	Teachers Notes	<input type="checkbox"/>
Inside The Machine	Hiccup resources	<input type="checkbox"/>
	Hiccup Program listings	<input type="checkbox"/>
	LMC simulator available	<input type="checkbox"/>
	LMC Instruction Set and Program Challenges	<input type="checkbox"/>
	Teachers Notes	<input type="checkbox"/>

The Need For Speed	RI Christmas Lecture Investigation sheet for each person. One Problem, Many Cores calculation cards prepared with Post-It labels Brown Dogs and Barbers sample chapter (or book) Teachers Notes	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Reflective Practitioner	BCS Certificate Flyers Trainers Notes	<input type="checkbox"/> <input type="checkbox"/>

The unit presentation is designed to support a full one day session, delivered to CAS Master Teachers and other curriculum champions. It will likely be fast paced, delivered to experienced teachers.

It is envisaged that those attendees will take the material and deliver shorter sessions, either as half day, twilight or CAS Hub inputs. It is expected these will take longer to cover each activity as the material will be unfamiliar to teachers new to Computer Science. Please find time to discuss with attendees possible ways to use the material and encourage them to offer further sessions in their locality.

## Resources

All supporting material is available for download, corresponding to each session in the Unit.

The material includes

- a full presentation to support all the activities covered in the Unit
- a set of Teachers Notes explaining the material for each session
- separate activity handouts

If you intend to use the material at shorter sessions, simply hide the slides not used.

If you wish to combine material in a different order please consider adding slides to introduce the 'big picture' at the start and to discuss being a reflective practitioner at the end. Please try to stick to the CAS House Style which is outlined on the opening introduction slide.

## Half Day / Twilight CPD Sessions

It is suggested the material could be delivered as 5 separate shorter sessions, as indicated by the sessions.

- **Logical Foundations:** Introducing logic gates, truth tables and combinational logic circuits.
- **Adders and Ladders:** Binary addition, half and full adders, building a 4 bit adder circuit.
- **Munching Squares:** Bitwise operations, analysing a problem and planning a solution, Game Maker scripting.
- **Inside The Machine:** A conceptual model of a computer, a Human Computer exercise, low level language programs and the fetch execute cycle.
- **The Need For Speed:** Moore's Law, von Neumann bottleneck and parallelism. One problem, many cores activity.

Of course, Master Teachers and other trainers can combine sessions and activities as they feel best fit the local circumstances.

Introduction

Any CPD session should be topped and tailed with the Introduction and Conclusion slides as the exemplar (left) demonstrates.

Bitwise operations

The five suggestions listed above are just a few of the possible combinations which allow time to explore some of the more formal exercises in more depth.

Analysing a problem

Many activities are short enough to introduce at CAS Hubs or worked through in a school departmental meeting. Whatever ways you choose, the aim is to develop teacher appreciation of CS concepts, not just demonstrate an activity.

Game Maker scripting

Reflection / Conclusion

We hope they are useful.