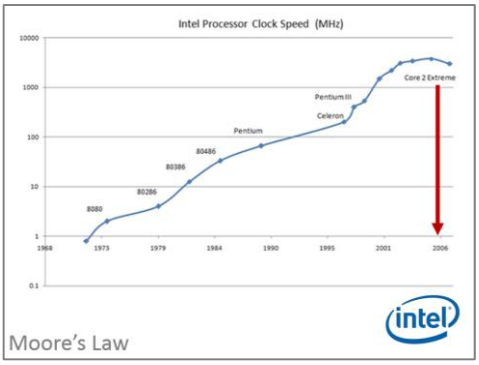
An introduction to Moore's Law and practical activity to demonstrate the principle of parallel processing.

Preparation required:
RI Christmas Lecture Investigation sheet for each person.
One Problem, Many Cores calculation cards prepared with Post-It labels.

Moore's Law

Gordon Moore, co-founder Intel in 1968. Three years earlier, whilst working for Fairchild Semiconductors, he had noted a tendency for the number of transistors packed into a chip to double every 18 months or so. He speculated the trend would continue for at least another ten years. This became known as Moore's Law. As you can see from the graph, which plots the number of transistors in Intel chips, Moore's Law held true for far longer than 10 years. Look carefully at the scale in the left. This is not a linear progression, but an exponential one. In 1974, the Intel 8080 chip had under 5,000 transistors etched into it. In 1984, the 386 had about 300,000 and by 1994, the first Pentium processors had over 3 million.

When most people talk about Moore's Law they don't talk about transistor count, but about processing power. That's because the two are related. By squeezing more transistors, particularly those for caches onto a chip, they can reduce the fetch time for data. And because the fetch time decreases, they can increase the clock speed. So computers have got faster and faster, meaning they can do more and more things. It's very difficult for children to appreciate what these clock speeds mean in real time. Another 3 minute video from the marvellous old Canadian TV series, Bits and Bytes makes the point well: youtu.be/x_DqMUkZHn0.



Ask children if they know what MHz stands for. Hertz are cycles per second, megahertz millions of cycles per second. What sort of clock speeds had been achieved by the turn of the new century? Around 2 billion cycles per second or 2GigaHertz. Note the same logarithmic scale on the axis of the clock speed chart as that used previously for transistors. A similar exponential growth can be seen in clock speeds.

But notice too what has happened since the start of the new millennium. The rate of growth slowed and clock speeds have reached their peak. The reason for this is heat. Transistors are so closely packed that they generate a huge amount of heat. Dissipating this has meant ever larger heat sinks, sitting on top of processors. Children will see these if they take a pc apart. In the first of The Royal Institution Christmas Lectures from 2008, goo.gl/HJyCDg, Chris Bishop explains the challenges well. Around 40 minutes long, this makes a good 'flipped classroom' activity. Included in the resources, is a simple investigation to support the video and encouragement to do a little research to find the specification of a pc / phone.

Back to our clock speed chart. Notice the name of the Intel processor that appeared in 2006. The development of multi-core processors started around 2003. By 2007, 8 cores were being packed on a chip. 2009 saw 16 cores and by 2013, 64 core chips were being produced. At the time of writing (2015), 8 core processors are appearing in home computers. And the major chip manufacturers have all developed processors with 128 or more cores for use in more industrial settings. What we are witnessing is the start of a new exponential! This is one of the exciting trends in computer science that will define the coming era. Welcome to the era of massive parallelism! Two articles from cs4fn (goo.gl/xPWIQM and goo.gl/XLOiF2) provide great introductory reading about Moore's Law and the challenges posed by parallelism.

Has parallelism solved everything? No – it's just passed the buck to the programmers, who need to write programs that make use of the new architecture. Challenge the pupils to think of everyday activities that would benefit from being carried out 'in parallel' … and others that couldn't be. Suggestions overleaf.

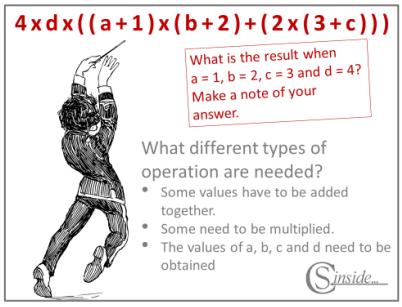
One Problem, Many Cores

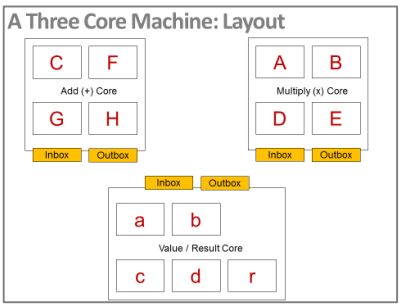
Many human tasks can be broken down into individual parts. Some of those can then be performed simultaneously. An orchestra might be a good example. Other, more mundane tasks might include:

- Giving out exercise books or worksheets,
- Carrying heavy loads in multiple bags, made easier by spreading amongst several people.
- Planting or picking crops.
- Using Bit-torrent to download a file.

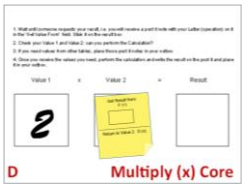
But there is a key challenge with any parallel activity, as the orchestra analogy suggests. Co-ordination is key. We're going to try a class activity where we collectively solve a long mathematical equation. The task was devised by Quintin Cutts and Margaret Brown as part of the CS Inside outreach project. It is just the sort of task that lends itself to 'distributed computing' – sharing out parts of the task to individual processors. In fact, some of the earliest examples of extra processors in computers, were 'maths co-processors' which aimed to take the load of the central processor. When long, heavy calculations needed doing these were hived off to the maths co-processor so the CPU could get on with another task. You see the same thing at work with graphics cards which take the long task of rendering 3D images in games, so the CPU can handle other elements.

So let's consider the long equation. Before doing the simulation, ask pupils to individually calculate the answer when a=1, b=2, c=3 and d=4 and make a note of their answer. If we want to break this calculation up and distribute the processing we need to start by identifying the different types of operations required. Discuss this with the class then reveal the 3 key operations shown. Each of the three operations can be given to a dedicated processor. One to handle values, another to Add, another to Multiply. Once we start and input data, the processors will pass values back and forth, before the result is obtained.

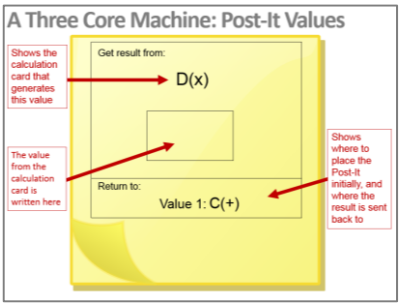




Each three core machine in our exercise will require 3 separate desks, labelled with their function. Each desk will have an inbox and outbox tray. Onto each desk are placed the 'calculation cards' shown. Each is identified by a letter and need to be photocopied in advance. Post-It notes, with incomplete values will need to be stuck onto the cards before they are laid out. There are 13 'calculation cards' in total. Each is identified by a letter, and the Core.

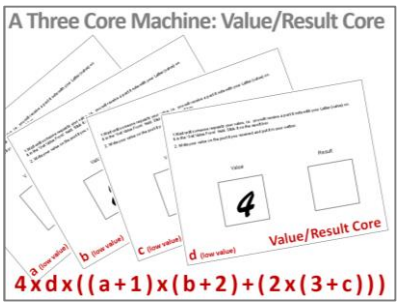


Some cards may already have a value filled in, others will be blank. The respective Post-It Value notes should be placed on the cards before they are given out. There are instructions on each card at the top. Each person must WAIT until someone requests the result of their card. This will come as a Post-It note. If they cannot calculate the result, they too must request a value via a Post-It note placed in their outbox. When they receive the value, they can complete their calculation, write the result on a Post-It and place in the relevant outbox. If that sounds complicated, the presentation provides a walkthrough. Practice it in advance.



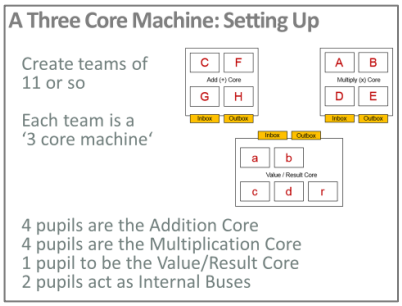
The values generated by the processes are passed around on post-it notes. Labels are included in the resources which can be printed out and stuck directly onto post-it notes. The values can then be filled in on the label. The incomplete Post-It's should be stuck onto the calculation cards before the simulation begins. A series of clicks will walk through a sample calculation. The + and x cores have parts of the calculation. Their results will need to be passed around the network to others. A core might need to request results from other cores BEFORE they can calculate their own!

It is very important to stress that they must ***WAIT*** for a request for their calculation, ***BEFORE*** they request values for themselves. This cannot be stressed too often.

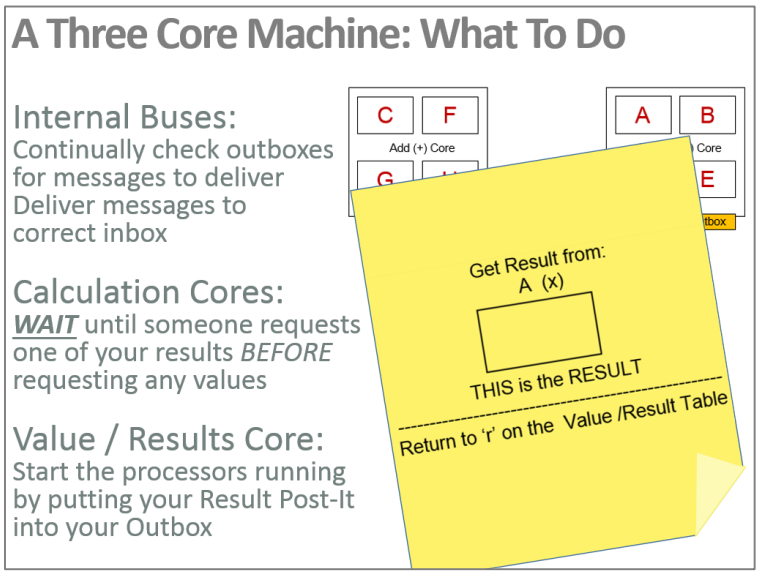


The equation we will solve requires four values (a, b, c and d). Two sets of value cards are supplied – one with low values 1, 2, 3 and 4.The other with higher values 5, 6, 7 and 8, but you can always create more. The lower value set are used in the example. These remain on the Value / Result Core table until requested by one of the other cores. The processors have completed their work when the final value can be posted onto the Result card.

Teams can be varied sizes depending on class size, but you will need at least 2 pupils for the Multiplication and Addition Cores. Between 7 and 11 works best.



Once each 'multi-core team' are in place the final instructions can be read to the class. Each group needs to listen carefully to their role. The slides reveal each of the 3 roles. They are outlined below. Stress that Calculation Cores ***MUST WAIT*** for a request for their value before trying to retrieve any values themselves.

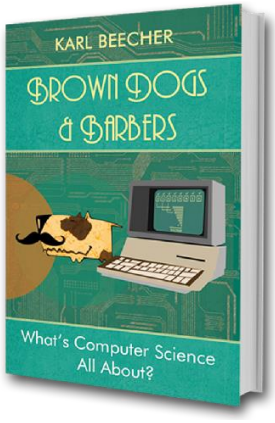


Several runs of the calculation can be made. After a couple of attempts, the teams should get the hang of parallel calculations. It might be worth having one group observe another.

Afterwards draw out the (dis)advantages of solving a long calculation in this way. Some simple tasks may lend themselves to easy distribution, but more complex tasks require careful planning.

Harnessing the potential of massive multi-core processors is a hot topic. And it's not just in one machine. Distributed computing lies behind many network services – modern web services are a very good example of distributed tasks.

We look at network sorting algorithms in another module, but the Network Sort activity devised by CS unplugged is another excellent activity to demonstrate the power of parallel processing.



Developments in parallel processing are just the latest example in the explosive development of computing technology. Along the way, computer scientists have had to grapple with many problems. This fascinating story, and the way key ideas such as abstraction, algorithms and decomposition lay at the heart of solving every problem is well told in this book: Brown Dogs and Barbers.

Each chapter is less than 4 pages – ideal for a busy teacher. The introductory chapter is included in the resources.