HISTORY OF COMPUTING

How do computers work?

The Human CPU

Overview: This lesson is the second of two looking at how the computer works at processor level. Rather than just showing *Video 5 – The Human CPU* it would be much better to model this for real using the students. Although this lesson takes much preparation it is well worth it for students needing to really understand how a computer processes binary data.

The teacher needs to have the confidence that they can explain the Opcode and Operand correctly at the decode and execute stages before attempting this. If you try with Program 1 first and then with other groups you could also attempt Program 2 or Program 3 using the PowerPoint support. The PowerPoint also contains slides showing each program in Assembly Code (a shorthand for the binary instructions) – see the Technical Background section below.

Computing National Curriculum Attainment Target:

- KS3: Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems
- KS3: Understand how instructions are stored and executed within a computer system
- KS4: develop their capability, creativity and knowledge in computer science, digital media and information technology
- KS4: Develop computational thinking skills
- KS5: Understand the Fetch Decode Execute Cycle including the role of CPU, RAM, buses and specific registers
- KS5: Be able to trace the execution of machine code (binary) instructions to be able to investigate what a given program in Binary or Machine code does.

Lesson Objectives:

- Understand the common components inside a computer and the function of each
- Understand the stored program concept
- Understand what the CPU is and what it does during the fetch decode execute cycle
- Understand what the RAM is and what it does during the fetch decode execute cycle

Lesson Outcomes:

- All: Have participated in a simulation of a CPU as a kinaesthetic learning experience
- Most: Be able to explain the stages of the fetch decode execute cycle
- Some: Be able to explain the function of each register and component including RAM and buses. Be able to explain performance issues relating to hardware specifications

Lesson Resources:

- Inside the Computer - **Video 5**
- **Human CPU** – PowerPoints – LGfL site
- A set of draws for the RAM and a whiteboard for the CPU
- **Labels to print** – LGFL site
- **Help To Print** – PowerPoint explaining the 14 bit instructions

Technical Background

Video 5 models the fetch decode cycle which is the foundation for understanding how binary instructions work inside the computer. Each instruction is binary in RAM. When a user runs a program (eg Microsoft Word) it is loaded up in RAM in consecutive memory locations. The Program Counter in the CPU holds the location (memory address label) of the first instruction for MS Word in RAM.

Each binary instruction in RAM consists of an Op Code - what to do (eg ADD / SUBTRACT / LOAD or SAVE) and the Operand - the data itself, which could either be a binary number or an ASCII code (see Video 6) or a memory address.

Each instruction loaded into RAM is accessible whenever called for by the CPU. To do this, the CPU puts its memory location into the MAR and sends it to the RAM along the address bus.

The program we show on Video 5 only has a few lines of instructions and the fetch decode execute cycle is slow as we act it all out. Most programs have 1000s of lines of instructions and the cycle of a typical CPU runs billions of times a second.

There is a straight forward link between grasping the theory for this lesson and going further to make connections about performance issues…

- More RAM would allow more programs to be able to work at once.
- Having multiple cores (eg Quad core = 4 CPUs in one) allows instructions to be processed 4 at a time so could be up to 4 times as fast as a single core CPU.
- The demo uses 14 bit instructions. A modern computer typically uses 64 bit instructions so a greater range of data can processed during each cycle.

The PowerPoint also has each program written out as Assembly Code. This is simplified shorthand for each instruction. The Op code is a 3 letter abbreviation (eg LDA means **L**oa**D** into **A**ccumlator) and the Operand is written as Hexadecimal rather than binary to make it more readable (see Video 6).

Keywords

- Fetch Decode Execute
- Registers | Address bus | Data bus
- RAM | CPU
- CIR – Current Instruction Register | MAR – Memory Address Register | MDR – Memory Data Register | PC – Program Counter | ACC – Accumulator

Lesson Summary: This lesson will concentrate on modelling what happens in the CPU and RAM when a computer is running a program. It works in binary.

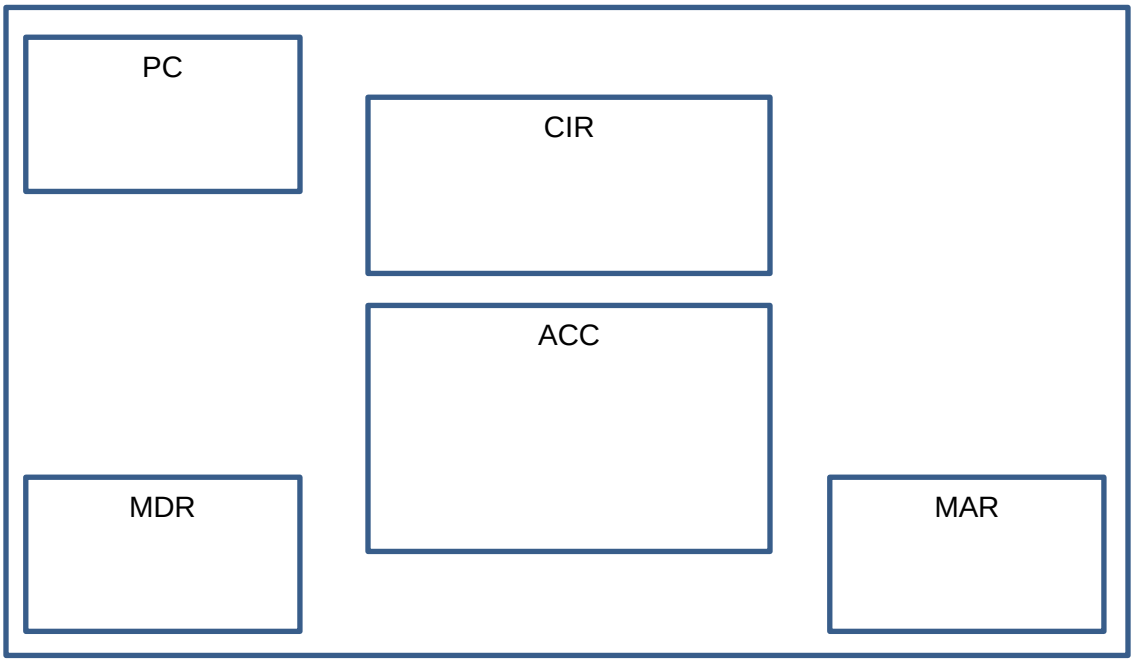
Starter: Show **Video 5 – The Human CPU**

Main/Development:

1. Discuss the video. Test the students understanding of what is happening using the PowerPoint as support.

2. Recap hardware components. What is the RAM doing? What is the CPU doing? What is the address bus used for? What is the data bus used for?

3. Set up the CPU as the whiteboard, ideally located at the opposite end of the room to the RAM. Draw onto the whiteboard the Program Counter (top), Memory Address Register (bottom right), Memory Data Register (bottom left), Current Instruction Register (upper middle) and the Accumulator (lower middle).



The Address bus needs to be set out as a line of chairs between the RAM and the MAR.
The Data bus needs to be set out as a line of chairs between the RAM and the MDR. Use the **labels to print** file to label everything up.

It is recommended that you also display the PowerPoint **Help To Print** (or print it out) for the students involved in decoding the instructions in the CPU.

The RAM should be set up as a set of trays or drawers as shown:

Load the 14 bit instructions for the program you want to run into the correct trays (starting with the first instruction in the top tray and loading each one in the tray below the preceding one.

The Program Counter on the whiteboard needs to have the address (location in memory) of the first instruction to be used in RAM (eg Program 2 would need to read 01010110 before starting).

Before starting two of the programs also need data to be in the right hand trays (marked as Input/Output above).

- Program 1 needs 00110010 in tray 00000001
- Program 3 needs 01000000 in tray 00000010

1. Once set up, ask the students if the memory address locations (the labels on each tray) are sequential or not? Do they go up by one each row? (Answer: yes).

2. Assign each student a role. Individuals need to take control of each register in the CPU. One student can manage the RAM. The rest of the students should be put into two groups, one group makes up the

address bus, the others make up the RAM.

3. Discuss what each register's job is, recalling Video 4. Explain what is going to happen. Use slide 2 of the PowerPoint to help.

4. The group work on fetching and decoding and executing each program in turn. After each program is run, each person can say what their role was. What was their part of the system doing as the program was run?
Plenary: What does each program do?

Program 1
Loads 50 (in binary) from tray 1 into the ACC
Adds binary 2 to the ACC (making 52)
Subtracts 19 from ACC (making 33)
Saves the result in the ACC to RAM, overwriting tray 1's data.

Program 2
Loads 4 (in binary) into the ACC
Adds 4 (in binary) to the ACC
Adds 4 (in binary) to the ACC
Saves the result in the ACC to RAM, overwriting tray 1's data.

Program3
Loads 64 (in binary) into the ACC (from location 2)
Adds 127 to the ACC (making 191).
It then tries to add 69 to the ACC (which would make 260)

However, the operand is only 8 bits and the largest number
you can make with 8 bits is 1111 1111 (255).
The is an example of **OVERFLOW** – an error has occurred as the CPU has attempted to calculate a number too big for it to represent.
Instead of the final calculation going ahead, the status register would indicate that an overflow error has occurred.
Modern CPUs have a 64 bit instructions, so they can represent massively large numbers (and with great accuracy).

Extension tasks:

1) Research the fetch, decode, execute cycle in more detail, looking at;

- Labelled diagrams of the CPU on the Internet
- CPU Cache – what is it and what does it do?
- Bottlenecking issues – what makes a computer slow down?
- Types of RAM – DDR / DDR2 / DDR3 – what's the difference?
- Types of CPU – Dual Core / Quad Core / Octa Core
- What is virtual memory – what happens when a computer runs out of available RAM?