Notes

Dr Greig Russell

05 May, 2019

Contents

List of Figures

1 Introduction

2 The modern computer

In information science, there is a constant dance between the applied and theoretical branches of the discipline. The work of one underpins the next advance in the other. While electronic computers were created to meet human needs when those needs exceeded the capacity of individual humans, information science preceded and enabled their development. The focus for advancing the capacity of computers or their application to solving human problems ever since has been about developing the hardware and software together with theoretical paradigms. To understand information there requires an understanding of computers.

Simply put, a computer is the combination of its hardware and software, and its current state of development is an iterative development from the original concepts developed during and before WW2.

2.1 The hardware

In 1945, Von Neumann [1993] first described the components of what he termed "very high speed automatic digital computing system" (section 1.1), and it has become the standard architecture of the computer. The concept has not changed since, even if the component level details have changed dramatically. The aim of this computer, called the EDVAC, was to solve non-linear partial differential equations (Von Neumann [1993], section 1.2).

The current unified concept of a Central Processing Unit (CPU) served two functions. The first was an arithmetic processor to complete basic mathematical operations. For @Von Neumann [1993] this also included more complex functions such as sin, cos, log and square root functions. (section 2.2). The second component was the logic controller (section 2.3). The logic controller was responsible for the orderly execution of a specific set of instructions. The central logic controller was responsible for the orderly operation of the computer as a whole, including hardware components, regardless of any specific program that it was running. It was this latter feature that made this the architecture for a general computer, not dedicated to a specific task like some of the earlier machines (such as the "Colossus" of Bentley Park and the WW2 code breakers fame).

Key to the new design for the EDVAC was the provision of "considerable" memory (section 2.4). Although hilariously small at 34KB by today's standards, such memory allows for the solution to be broken into components for efficient problem solving within a supervising logic tree. It also allowed for interim states of variables to be available between components to be saved (Von Neumann [1993], section 2.4). Von Neumann [1993] also saw that such generalised computers could be useful for statisticians in regards to the managing of large data samples.

The computer will require input devices (Von Neumann [1993], section 2.6), such as punch cards, and recording this information by use by the computer. Note, for this architecture, the recorded information is not available for the computer. Hence, the computer needs a means of moving the data into the working memory or CPU (section 2.7). Equally, a means of moving the results back to be recorded (section 2.8) and then outputting the results in human-readable forms, such as on a teletype (section 2.9)

2.2 The software

Alan Turing outlined the fundamental functional concepts of a computer in his landmark 1937 paper "On computable numbers, with an application to the Entscheidungsproblem" (Turing [1937]). The EDVAC was supposed to enable such functions to occur within a general computer, as Turing described.

Turing's computer was not the focus of his paper. Instead, it was a thought experiment that served as a vehicle to address the real issue the Entscheidungsproblem or "halting" problem. Yet, this description has served as the archetype for what a computer needs to have to be a complete general computer. The term "Turing complete" has been coined to describe any such computer (or language on modern computers with

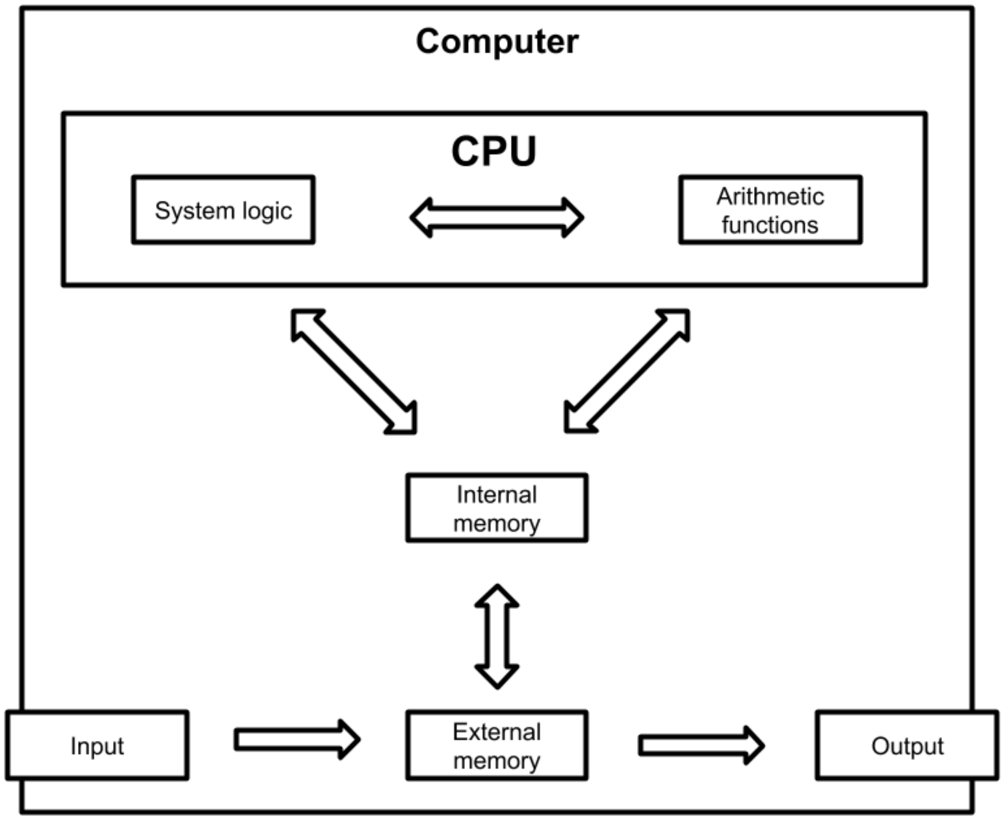

Figure 1: Von Neumann Architecture

the abstraction of the programming logic from the hardware into the software) that can perform all the functions of a Turing machine.

The components of a Turing machine are the machine itself and paper tape. The machine consists of a reader/writer head and a set of instructions that depending on the initial state, describe what will happen next. The paper tape consists of a series of symbols, and in Turing's day this was always numbers but what they are is irrelevant to the operation of Turing's machine. The machine reads a symbol on the paper tape, and then performs the specified action by the instruction set based on that symbol. This action may be to change the symbol on the tape or move to another place on the tape or even do both (Turing [1937]).

The paper tape describes the initial state of such a machine. After the machine holts, the paper tape now describes the output of the machine. Turing's research was about whether such machines will always holt and he was able to prove that they did not (Turing [1937]).

For everyone else, Turing's machine is the blueprint of a computer, which was brought to life on physical hardware following the design based on Von Neumann's architecture.

2.3 The Operating System

Remarkably, this remains the fundamental design of a computer ever since. The only other significant change was in the 1960s IBM is credited with the introduction of the operating system (OS). This innovation overcame the practical problem of the instruction set is unique to every computer chip. Changing the hardware meant rewriting all of the software, which as the scale of instruction sets grew exponentially was utterly impractical.

The OS became an abstraction that took standardised requests from an instruction set and translated them to the system used by the underlying hardware. For software developers, they could write the code once, and it would run across different hardware configurations. For hardware engineers, they could rapidly innovate without breaking all the client's software.

The standardised computer has becoime as shown in figure 2. As an example of such a machine, consider the word processor. A user buys a physical machine, which comes pre-equipped with an OS. From the list of word processors compatible with that OS, the user installs one. Their consequent writing becomes the data or paper tape, and the program is the set of instructions to enable the data to be transformed, but the truth function of this transformation is in the instruction set and not in the data.

On reviewing the code, but not the data, one could see that a particular set of instructions will cause a selected text to be made bold. Regardless of what the truth value of the text is, the truth value of the program is whether the text was made bold or italic.

While the differentiation between the two may seem a pedantic quibble. The assumption of truth that the correct rendering of a blog means that the contents of the blog are equally correct has led to vaccination hesitancy to be included in the WHO top10 threats to world health due to incorrect anti-vaccination propaganda [1]. The truth value of the data or content, is for the user. In the era of Turing and von Neumann, users were expert mathematicians or equivalent.

The rest of computing for the last 50 years has been an arms racing between providers of the hardware, the developers of operating systems, and the developers of software. The capacity and capabilities of individual machines has increased dramatically over time, but the core concepts have not.

What has changed is the falling price of computers so that they are now widely available, meaning users are no longer experts, making interpretation of output more fraught. The rise of the internet or interconnected computers makes determining the truth at scale even more fraught. Some believe that Shannon's law or communication protocols has considered this.

[1] "Ten threats to global health in 2019, from "https://www.who.int/emergencies/ten-threats-to-global-health-in-2019 downloaded 5.May.2019

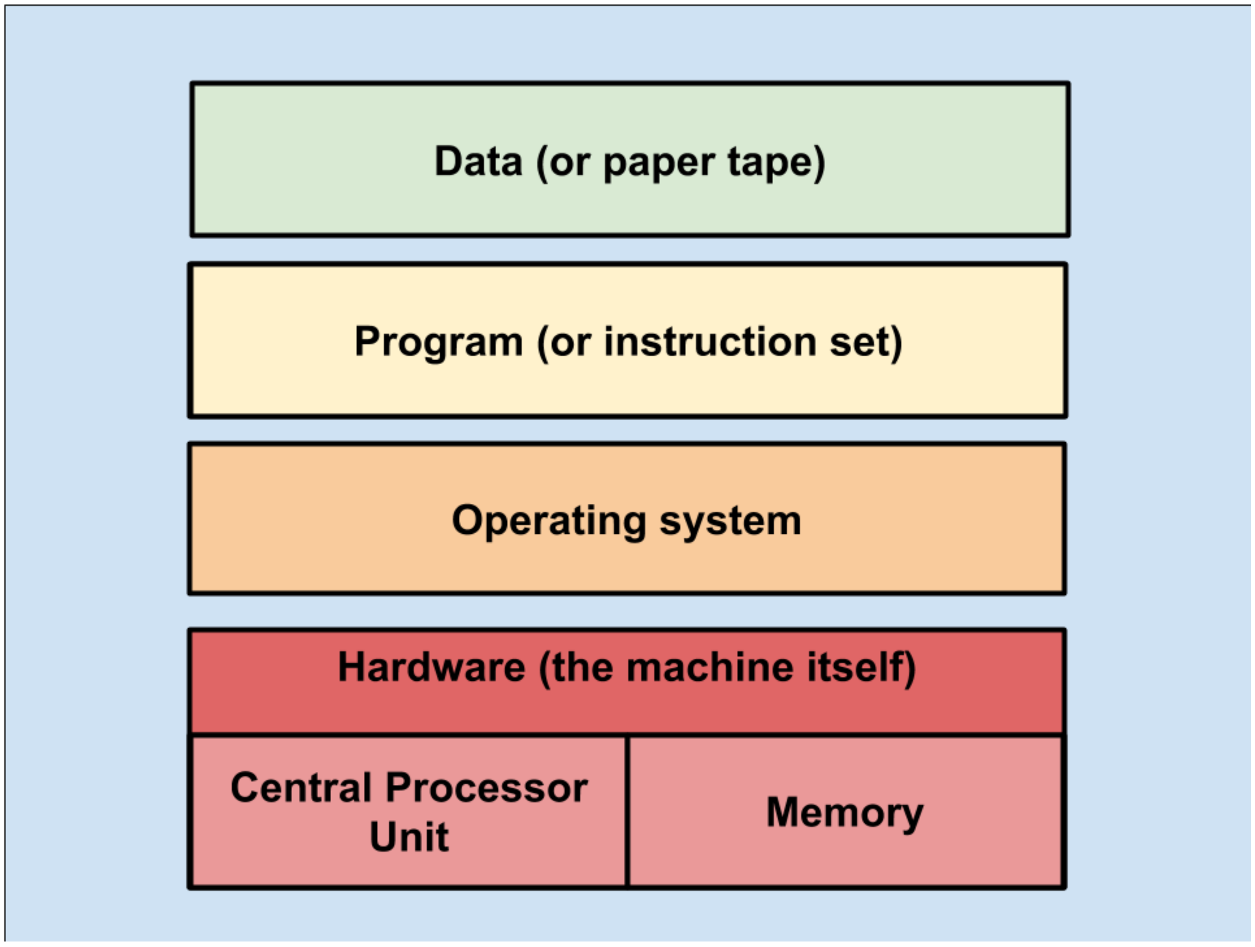

Figure 2: The essential modern computer

3 Shannon’s law

3.1 The algorithm is the truth

3.2 Turing machine

3.3 Lambda calculus & functional programming

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{1}$$

4 The arms race

4.1 Moores law

4.2 Hardware, software and OS evolution

4.3 Data still static

5 Collaboration

5.1 Shannon’s communication theory

5.2 The internet

5.3 Fake news

6 The ontology of information

6.1 Dretske

6.2 Carnap and Bar Hillel

7 The lost counterfactual

7.1 Lewis

8 Conclusion

Bibliography

Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265, 1937.

John Von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4): 27–75, 1993.