WHAT IS AN ELECTRONIC BRAIN?

1 Introduction

Electronic Brains, i.e., high speed electronic digital computers, are producing major and far-reaching changes in our society. This article aims to tell you what an electronic computer is.

First of all, it is not an "electronic brain". The machine does not think. It merely obeys instructions literally. Possible instructions might be: (1) Read a number from a punched card. (2) Store that number away in the memory. (3) Add two numbers (4) Multiply two numbers. (5) Print out a resulting number. For every calculation which we want done, we must first give the machine a detailed, explicit set of such "commands"; this set of commands is called a "programme". The machine reads the entire programme, and stores it away in its memory. We then press the button which says "start", and the machine proceeds to execute these commands, one by one. At each step, the machine does what it is told to do, no more and no less. At no stage does the machine "think". Rather than being an electronic brain, it is an electronic moron.

The advantage of the machine lies not in its (non-existent) intelligence; rather, the advantage is the incredible speed with which this moron obeys commands. Two 10 digit numbers can be added in five millionths of a second, and multiplied in one hundred thousandth of a second! Years of human computing are done by the machine in less than an hour.

2. Components of a Digital Computer.

The basic components of a computer are: Storage (or Memory), Arithmetic Unit, Control Unit, and Input-Output Unit.

Storage consists of a set of numbered cells, or "storage locations", each of which can be used to store a number, or alternatively to store a command to be obeyed by the machine. Modern machines commonly have 32,000 or more of these storage locations. The most common method of fast storage is on "magnetic cores", which are tiny little rings of magnetizable material. The magnetic field lines may go clockwise around the ring, or else counterclockwise, giving two clearly different "states" of the ring. Once the ring has been magnetized, it stays that way for a long while. Since an electric current generates a magnetic field of its own, we may alter the state of magnetization of a ring by passing a wire through the hole in the ring, and sending current through that wire. In this way, the memory of the machine can be altered.

The Arithmetic Unit of the machine is the actual "computer", i.e. the device which carries out additions, subtractions, multiplications, and divisions. The necessary numbers must be transferred from memory to "registers" in the arithmetic unit, and the resulting numbers are generally returned from the registers back to memory. In most machines, the four basic arithmetic operations are the only ones carried out by the machine as such. More complicated operations, such as taking a square root, must be broken down into these elementary operations and a programme must be written to tell the machine what to do at every step.

The Control Unit controls what the machine does at each stage; that is, the control unit takes the next command to be obeyed out of a memory position then "decodes" it, and initiates the electrical signals which cause the rest of the machine to actually execute the command. For example, the control unit may activate the multiplication circuits in the arithmetic unit so as to carry out a multiplication.

Last but not least is the Input-Output Unit which keeps the machine in contact with the rest of the world. Input may be a device to read punched cards, or one that reads punched paper tape; output may be a unit which punches cards, or punches paper tape, or it may be an electric typewriter, or a fast printer. With the exception of the input-output device, all operations in the machine are done by purely electrical or magnetic signals.

3. Numbers in the Computer.

Since the basic memory unit, the magnetized ring, has only two possible states (clockwise and counterclockwise magnetization), it cannot be used to represent a decimal digit; there are ten decimal digits, namely 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. For this reason, it is convenient to represent numbers in the machine in binary form, which we shall now explain.

The decimal number 2347 is an "ordered set" of four decimal digits, namely the decimal digits 2, 3, 4, and 7, in that order. As we know, the desired number is to be reconstructed from this decimal representation as

$$2 \times 1000 + 3 \times 100 + 4 \times 10 + 7 \times 1 = 2 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 7 \times 10^0 .$$

Thus the rightmost digit, 7, which we may think of as the digit in "position 0", multiplies $10^0 = 1$. The next digit, 4, is in "position 1", and multiplies 10^1. The next digit to the left, namely 3, is in "position 2" and multiplies 10^2. Finally, the leftmost digit, 2, is in "position 3" and multiplies 10^3. In general, then, the digit in "position n" is meant to multiply 10^n. The number 10, whose powers occur in this way, is called the base of the decimal number system. The possible decimal digits are all the integers between 0 and 9, i. e. all integers from 0 up to one less than the base 10.

In the binary system, the "base" is 2 rather than 10. The possible digits are all integers from 0 to one less than the base, i. e. from 0 to 1. There are only two digits, therefore, and this is just what we want for our magnetized rings. In the binary system, numbers are again represented by strings of digits; but the possible digits are only 0 and 1, and the digit in position n multiplies 2^n rather than 10^n.

As an example, consider the number 1101_2; we write the base, 2, as a subscript here, to make it explicit that we don't mean the decimal number 1101_{10}. According to our rules above, we have

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$
$$= 13_{10}.$$

Thus the binary number 1101_2 is identical with the decimal number 13_{10}; both represent the same number, in different notation.

Although all numbers in the machine are in binary notation, the programmer need not worry about carrying out the conversion from decimal to binary or from binary to decimal because the machine itself can be programmed to carry out this conversion.

4. Commands in the Computer.

A memory position may be used to store a machine command, instead of a number. In each case, we deal with an ordered sequence of bindary digits ("bits"): whether this sequence is interpreted as number or as a machine command depends entirely on the action of the control unit of the machine. If the control unit transfers the contents of the memory location to an arithmetic register, the contents of this register are treated as a number by the arithmetic unit of the machine. If, instead, the control unit transfers the contents of the memory location to the "command register", the sequence of bits is then "decoded" and obeyed as a machine command. This might be, e.g., add, subtract, clear a memory location to zero, store the contents of an arithmetic register into a memory location, or activate the electric typewriter to print out the contents of a register.

5. Programming.

So far, we have described the components of a digital computer, and the ways in which numbers and machine commands are represented in its memory. How does one actually use a computer? In order to get the machine to carry out a given computation, the user must give it a "programme". To write a programme, one must build up an intricate structure from a large number of simple statements which are used as building blocks. One must be able to "see many moves ahead", to visualize what the machine can (and of course will) do wrong as the result of the contemplated set of instructions, and prevent this from happening. Unfortunately there is no room here to attempt to describe the details of writing a programme. Like any other new art, programming has its pitfalls, as the following example shows.

An aeronautical engineer employed by one of the aircraft companies in California found that he needed to solve a set of 30 linear equations in 30 unknown quantities. So he looked in his engineering handbook, and found that there is a standard method for doing this, called Cramer's rule. It involves the evaluation of some quantities called "determinants". The formula for a determinant was also in the handbook, and a programme could be written for evaluating this formula on a computer. The engineer did this, and started the machine on the actual problem. The machine accepted the problem, and commenced to compute with incredible speed. An hour later, the engineer got worried. 10 hours later, he got very worried. Two days later, the machine was still going merrily on its way, and the engineer called in an expert.

The expert pointed out that a determinant of order 30 contains $30! = 1x2x3x4x5x6x7x8x\ldots x29x30$ different terms, each involving 29 multiplications. At the rate of one multiplication in 10^{-5} seconds, the evaluation of the 31 determinants required for Cramer's rule would take approximately 10^{28} seconds, or well over 10^{20} years!

Furthermore, even if our engineer were prepared to wait all this time, and if his company were willing to pay for 10^{20} years of machine time for him, even then the answer would be absolutely useless. The machine carries only a finite and limited number of decimal (or binary) digits. Thus the numbers in the machine are rarely precise, but usually have a "round-off error" attached to them. The round-off errors accumulate at the various steps of the computation, until the final result contains an accumulated round-off error which may be much larger than the unwary programmer suspects. The expert pointed out to the engineer that long before his 10^{20} years were up, the round-off errors would have accumulated to the point where the results meant nothing at all. Not even a single significant figure would be left after all this computing!

Does this mean that 30 linear equations in 30 unknowns are insoluble on a computer? Far from it. The method given in the textbooks of algebra (i. e., Cramer's rule) is unsuitable for practical computation. Alternative methods can be, and have been, devised, and these permit solution of such a problem, not in 10^{20} years, but in a few minutes, with as much accuracy as the accuracy of the input data warrants.

The techniques most useful in numerical work are quite different from the standard methods of analysis. Numerical Analysis is a field of mathematics in its own right, and by no means a fully finished, explored field. It is an art, not a series of cookbook recipes for doing standard tasks in standard ways. Numerical analysis and programming have an appeal of their own, and programming, in particular, tends to appeal to people who like combinatorial games such as chess.

(John M. Blatt).