Proceedings of Symposia in Applied Mathematics
Volume 48, 1994

Who invented the computer? The debate from the viewpoint of computer architecture

Raúl Rojas

Abstract. Several countries and researchers claim for themselves the distinction of having invented the computer. In this paper we discuss the problem from the viewpoint of computer architecture: a computer should be understood as a universal calculating device. We show which is the minimal structure and the minimal instruction set computing automata should have in order to achieve universality. An analysis and comparison of the first calculating machines provides us with a surprising answer to the question stated in the title of this paper.

1. The problem of universal computation

The discussion about who should be called the true inventor of the computer has never been satisfactorily settled. The debate has been additionally obscured, because priority is claimed only over some part of what we understand today under the name "computer". For anyone interested in this debate, the first point to be clarified is certainly: what do we understand under the name "computer"? The only possible way to arrive at a consensus is to define a computer as a device capable of *universal computation*, that is, capable of implementing all computable functions.

1.1. Minimal structure of a universal computing machine. In the theory of recursive functions, we start from a small number of primitive functions and composition rules in order to define the general recursive functions. These can be implemented in a computing machine with a single accumulator by providing the following primitive instructions: CLR accumulator, INC accumulator, LOAD and STORE. Additionally, function composition and FOR and WHILE loops are required.

It is also necessary to provide some kind of indirect addressing in the machine. Otherwise, it is not possible to access numerical tables of variable length. The FOR and WHILE loops can be implemented in a simpler manner by providing some kind of conditional branch in the machine. An instruction "BR address" which tests the accumulator and branches to "address" if the accumulator is zero can be used to that effect. An unconditional branch can be implemented by executing a CLR instruction followed by a conditional branch.

1991 *Mathematics Subject Classification.* Primary 68M05, 01A65; Secondary 68Q05.
This paper is in final form and no version of it will be submitted for publication elsewhere.

1.2. Self-modifying programs. General-purpose programming languages implement iterative calculations over tables of numbers by providing indexed arrays. At the machine level some kind of indirect addressing is required to retrieve the elements of an array from memory. There are two possible solutions to this problem. The first is to make the LOAD instruction introduced above retrieve its parameter from a register in the processor. The second is cheaper and is in fact the one which was used in early computers: just store the program in memory and let the LOAD instruction work only with a constant argument. Since the program is stored in the memory of the machine, all instructions can be modified by the program being executed. A self-modifying program can store the code for the instruction "LOAD 100" or "LOAD 200" in the required address, and when this address is executed, the effect is the same as if we had provided a register for indirect addressing.

Summarizing: we will be looking for the instructions CLR, INC, LOAD, STORE, conditional branches and indirect addressing (or equivalently self-modifying programs), when trying to decide if a particular machine can be called a universal computer.

2. Early computing machines

2.1. The architecture of the Z1. The first machine whose architecture we consider is the Z1 developed by the German engineer Konrad Zuse from 1936 to 1938 in Berlin. The Z1 was a mechanical computer, but it was not built with rotating parts like the calculators used at the time. It used instead sheets of metal capable of controlling binary switches used to store binary digits and to perform logical operations.

The Z1 was capable of computing the four elementary arithmetic operations as well as the square root of numbers. The program was punched on a film tape and was read sequentially by a tape reader. Besides the arithmetic operations, there was an instruction for reading a number from a keyboard and storing it in any of 64 memory cells, as well as an instruction for showing the contents of the accumulator of the machine in a field of electric lamps. Another interesting feature of the Z1 was its use of a floating-point representation. Each number was stored as a mantissa, an exponent and a sign. The processor computed all arithmetic operations using this coding (called by Zuse *logarithmic representation*) [7]. It would be more than fifteen years before someone built another machine with floating-point registers.

2.2. Atanasoff's machine. John Atanasoff built his electronic computer at the University of Iowa from 1938 to 1942. Whereas all other computing machines developed at the time worked with parallel arithmetic hardware, Atanasoff decided to use a serial approach. Two rotating drums stored up to thirty fixed-point numbers. Each number was stored in one sector of the drum using 50 capacitors, that is, 50 bits. The machine could thus hold at any time just two different vectors in its $30 + 30$ memory cells. The vectors were read in decimal form from punched cards and were transformed to binary numbers before being stored. Atanasoff designed his machine in order to solve systems of linear equations with up to 29 variables. Since the machine worked with Gauss reduction, one of the vectors was used repetitively to reduce the other. The leading coefficient of the other vector was reduced to zero through a combination of additions, subtractions and shifts [5].

Since the drums rotated once every second, all bits of corresponding vector elements could be read sequentially and could be fed into a serial adder whose

result could be written back to one of the drums. The adder could be implemented with a minimum of hardware by recycling the carriage bit. The adder could also subtract one number from an other, if this was desired. A logical shift of a binary number was nothing but a read and write operation with a specific delay. There was an adder for each vector element.

2.3. The architecture of Mark I. Howard Aiken built the Mark I at Harvard University from 1939 to 1944. Like Atanasoff, Aiken was a physicist anxious to create a machine capable of doing the extensive calculations that he needed. The Mark I was an electromechanical machine, a kind of hybrid between the all-mechanical nature of previous computing devices and the electronics available at the time.

Aiken used a decimal representation for storage. Rotating gears were used to store and transmit the numbers in the memory. There was no sharp distinction between processor and memory, since each of the 72 memory cells was an accumulator capable of adding or subtracting a given fixed-point number. Each accumulator could transmit its contents through what we would today call a bus to other accumulators. There were also special units which stored tables of numbers and a multiplier. Sixty storage words could be set manually by the operator and constituted the constants in a program. The program itself was stored on punched paper tape, and two reading units made possible the execution in parallel of two programs or of the same one (for checking purposes). The Mark I used what we now call a three-address format for its instructions. All elementary arithmetic operations were possible, as well as sequencing of instructions [1].

Aiken's Mark I was not a universal computer since it lacked conditional branches. It was possible to repeat a section of code a given number of times, but no conditional execution of loops was possible. Since the program was stored externally, self-modifying codes could not be implemented.

2.4. The ENIAC. The *Electronic Numerical Integrator and Computer* was built at the Moore School of Electrical Engineering of the University of Pennsylvania from May 1943 to 1945. It solved its first problem in December 1945 and was officially presented in February 1946 [3].

From the viewpoint of computer architecture the ENIAC was a *parallel dataflow machine*. There was no separation between memory and the computing sections. Decimal fixed-point numbers were stored in any of twenty accumulators, each one capable of transmitting its contents to any of the other accumulators, or of receiving a number and adding it to its stored content. There was a multiplying and a dividing unit. Since the machine operated with a basic clock rate of 100,000 pulses per second, it was felt by the designers that an external program (in punched tape, for example) would not fully exploit the calculating speed of the ENIAC. Programs were implemented by physically connecting the different modules, that is, by effectively hardwiring the program into the machine.

Sequencing of arithmetic operations was achieved by connecting the arithmetic units in the desired order, even in parallel. The units worked asynchronously and in a self-clocking manner. A unit which was done with its calculation signaled this with a pulse to the next unit in the computation stream. The ENIAC was thus capable of implementing the CLR operation, addition, load and store of numerical data. It was also capable of implementing primitive recursion since a special unit, called the master controller, could restart a given thread of computation a fixed

number of times. This is equivalent to a FOR loop in a high-level language. More important was the ability of the master controller to stop a given computational thread when a given quantity changed its sign. In this way it was also possible to implement WHILE loops and conditional branches in the machine.

2.5. The Analytical Engine. Finally, we must consider the design of the Analytical Engine conceived by Charles Babbage in the nineteenth century. The architecture of Babbage's machine is surprisingly modern. It includes such features as a clean logical separation of numerical storage and processor, a microprogrammed computing unit and even a pipelined design! [2]. Babbage worked on the Analytical Engine from 1834 until his death in 1871.

Although only some fragments of the Analytical Engine were built, and the complete design has only been studied recently, this machine should be considered a serious contender for the distinction of having been the first computer. Each number was stored in a vertical stack of rotating wheels using a decimal representation. The contents of each storage unit could be transported to the processor of the machine, called by Babbage "the mill". The four basic arithmetic operations were implemented in the mill. The program was read from punched cards, similar to the ones used in the Jacquard looms. Another set of cards contained the data for the program. This separation of code and data was a clever way of eluding the problem of indexed addressing. Since Babbage's machine could implement loops, it was possible to repetitively read the code cards at the same time as new data cards were sequentially fed into the machine. Babbage also included the possibility of conditional branching. He was aware of the restrictions imposed by an external program. Because of this, he speculated with the idea of including a card puncher as an output option. The machine could in this way produce its own programs!

3. Architectural comparison

3.1. Summary of characteristics. The following two tables show the most relevant information about the machines discussed in the previous sections. As should be clear from the descriptions given above, no single machine fulfills all the necessary requirements for a universal computer. The Analytical Engine is surprisingly the one which comes closest. Table 1 shows the main features of the machines we analyzed. We also include the Mark 1 machine built at Manchester from 1946 to 1948, because as far as we know this was the first machine to fit our definition of a universal computer. This machine stored its program in random-access digital memory implemented with CRT tubes. All necessary instruction primitives were available (in modified form), and although it lacked indirect addressing, self-modifying programs could be written [4].

3.2. Conclusions. There is no unambiguous answer to the question stated in the title of this paper. Among the earliest machines considered, Babbage's comes closest to a universal computer. Yet this machine was not built and we know about it only through some recent historical research. The ENIAC was the fastest machine of its time and could implement loops, but lacked any kind of soft programming. Atanasoff's machine was the first *electronic* calculating engine. Although the Mark 1 from Manchester was indeed a universal computer, it would be hard to credit its designers with having invented the computer single-handedly, since some of the design ideas were taken from the EDVAC project conceived by the ENIAC creators

Table 1. Comparison of architectural features

Machine	memory and CPU separated?	conditional branching?	soft or hard programming	self-modifying programs?	indirect addressing?
Babbage's	√	√	soft	planned	×
Zuse's Z1	√	×	soft	×	×
Atanasoff's	√	×	hard	×	×
H-Mark I	×	×	soft	×	×
ENIAC	×	partially	hard	×	×
M-Mark 1	√	√	soft	√	×

Table 2. Some additional architectural features

Machine	internal coding	fixed-point or floating-point?	sequential logic?	architecture	technology
Babbage's	decimal	fixed-point	no	pipelined	mechanical
Zuse's Z1	binary	floating-point	no	sequential	mechanical
Atanasoff's	binary	fixed-point	yes	vectorized	electronic
H-Mark I	decimal	fixed-point	no	parallel	electromechanical
ENIAC	decimal	fixed-point	no	data flow	electronic
M-Mark 1	binary	fixed-point	yes	sequential	electronic

and John von Neumann. In fairness, it should be concluded that the invention of the computer spans a period of more than hundred years, in which scientists from at least three different countries provided the theoretical and practical foundations for this enterprise. If there is a technological achievement in which only international cross-fertilization could have led to success, then the invention of the computer is indeed one.

References

[1] H. Aiken and G. Hopper, *The Automatic Sequence Controlled Calculator*, reprinted in [5], 203–222.

[2] A. Bromley, *The evolution of Babbage's calculating engines*, Annals of the History of Computing **9**, no. 2 (1987), 113–136.

[3] A. W. Burks and A. R. Burks, *The ENIAC: First general-purpose electronic computer*, Annals of the History of Computing **3**, no. 4 (1981), 310–399.

[4] S.H. Lavington, *A history of Manchester computers*, NCC Publications, Manchester, 1975.

[5] B. Randell, *The Origins of Digital Computers*, Springer-Verlag, Berlin, 1982.

[6] N. Stern, *From ENIAC to UNIVAC*, Digital Press, Bedford, 1981.

[7] K. Zuse, *Der Computer — Mein Lebenswerk*, Springer-Verlag, Berlin, 1970.

[8] H. H. Goldstine, *The Computer: From Pascal to von Neumann*, Princeton University Press, Princeton, 1993.

MATHEMATICS DEPARTMENT, FREIE UNIVERSITÄT BERLIN, BERLIN 14195
E-mail address: rojas@inf.fu-berlin.de