The Anitra Computer

How simple can a basic computer be designed to be using TTL/74-series standard circuits?

http://www.princeton.edu/~ebakke/anitra
by Eirik Bakke

Goal

A computer is a machine that, given enough time and memory, can perform any rigidly defined computational operation. A minimalist computer is a computer that satisfies this requirement with only a minimum level of architectural complexity. Although hypothetical minimalist computers has long been an interesting study in Computer Science, there exists very few actual hardware implementations of such, and the goal of this investigation was to design one from scratch using the standard 74-series TTL-compatible digital logic circuits (ICs).

The Investigation

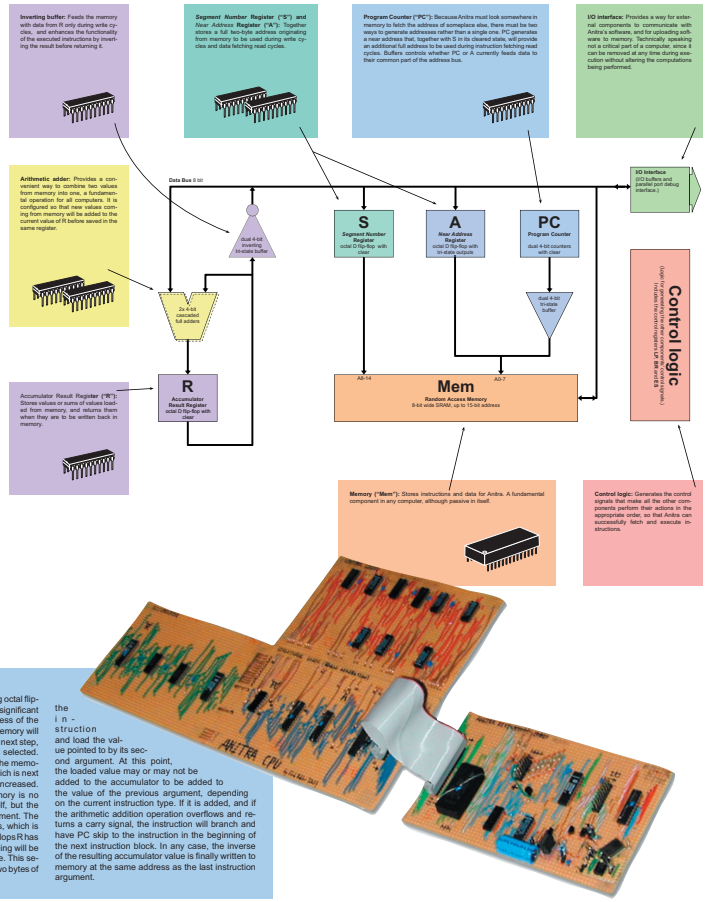
I started my investigation by defining the exact requirements for my computer. Then, as I was aiming to keep its design as simple as possible, I tried to discuss which components would unquestionably have to be included. The resulting set of components included D-flip flops for holding target addresses and values during memory write cycles, and a unit resembling a program counter for simultaneously keeping track of an additional address. An arithmetic adder was needed for the computer to be able to combine two values into one. Using only components presumed to be essential, I could put together a structure that should be capable of fetching and executing instructions in memory. From this, I went on by designing the necessary control logic and defining all circuits in detail on paper. The resulting computer, called Anitra, has two universal instructions: "move with complement" and "add with complement and branch if carry". Branching is done in a special way which does not require program counter to be programmable. In addition, the program counter is limited in size, which restricts the memory area where instructions may reside.

Conclusions

The CPU part of the finished Anitra computer has an 8-bit data bus and consists of three octal D-flip-flops, two octal tri-state buffers, one 8-bit non-programmable counter, two 4-bit arithmetic adders and some control logic (single gates and D-flip-flops only). Given the initial premises, it is difficult to imagine how the structural parts of the computer could have been made with significantly fewer or simpler components than those used; each of its components has been shown to fill a specific need during the execution of operations presumed to be essential.

I have now completed the next great step in this project, which is the construction of a working physical prototype of the Anitra computer. Finally, I am now in the process of writing an operating system for the computer.

The Anitra Computer



Theory of Operation

(Summary)
WARNING: Geeks only!

Anitra is built around a 32-kilobyte SRAM memory chip with an 8-bit data bus. Full 15-bit addresses for the memory are made from a 7-bit segment number and an 8-bit near address. The two octal D flip-flops S and A may both retrieve data from the memory through the data bus. The segment number is always prepared in flip-flops S. The latter may contain data from the memory, or it may be reset to zero. The near address can come from either flip-flops A or the counter PC, depending on which of the two currently has its tri-state outputs enabled. PC's first and last 4 bits may be increased or reset individually. PC's least significant bit is controlled directly by the control logic. A simple accumulator system is connected to the data bus. The flip-flops R may be triggered, which will add the current data from memory to its existing value through an arithmetic adder, or it may be reset to zero. The result may be directed back to memory during write cycles through an inverting tri-state buffer. To execute instructions, the control logic outputs a 7-step sequence of control signals, one for each clock cycle. A set of D flip-flops, called ES, is configured to keep track of the current execution state. Each instruction is 4 bytes long, consisting of two two-byte full addresses. The most significant bit of the segment number of the last address is used to distinguish between two different instruction types.

The execution sequence starts by resetting octal flip-flops R and S, and selecting PC. PC's least significant bit is set to 0. This will generate the address of the current instruction's first byte, which the memory will now load and place on the data bus. In the next step, this byte is clocked into A while PC is still selected. PC's least significant bit is then set to 1. The memory will load the second instruction byte, which is next clocked into S while A is selected. PC is increased. The address that is now passed to memory is no longer the address of the instruction itself, but the pointer that was specified as its first argument. The memory will load the value at this address, which is next clocked to the accumulator. Since flip-flops R has been reset and initially contains zero, nothing will be added to the value as R is clocked this time. This sequence is then repeated to fetch the next two bytes of the instruction and load the value pointed to by its second argument. At this point, the loaded value may or may not be added to the accumulator to be added to the value of the previous argument, depending on the current instruction type. If it is added, and if the arithmetic addition operation overflows and returns a carry signal, the instruction will branch and have PC skip to the instruction in the beginning of the next instruction block. In any case, the inverse of the resulting accumulator value is finally written to memory at the same address as the last instruction argument.

> *"A computer is a machine that, given enough time and memory, can perform any computational operation on a set of data. A minimalist computer is a computer that satisfies this requirement with only a minimum level of architectural complexity."*

Anitra's two universal instructions

Computer work by mechanically executing a sequence of so-called instructions in memory, and even Anitra works this way. An instruction is a signal for the computer to carry out a single low-level computational operation. Such operations may read and alter values in memory, or affect the order instructions are executed themselves. By combining such instructions in the appropriate way, more advanced operations may be synthesised. The Anitra computer has a predefined area in memory where instructions may reside, subdivided into *blocks* of 8 instructions each. These will execute over and over again in an eternal loop. While traditional computers have hundreds of different instructions available for the programmer to use, Anitra is a "minimalist" computer and has only two:

"mov S,Q"	Read the value in memory at address S, and save an inverted copy at address Q.
"add S,Q"	Read the values in memory at addresses S and Q, add them together, and save the result inverted at address Q. If the result is too large to fit into Q, skip the rest of the instructions in the current block. (In the latter case, discard the result's most significant bit and save the rest. If the current instruction is the last one in its block, skip all the instructions in the following block instead.)

An address is a value identifying a single place in memory. In Anitra, an 8-bit computer, each such place may hold a value made up of 8 binary digits. A value is said to be inverted if each of its digits are complemented (1s becomes 0s and vice versa)

Although Anitra's two different instructions may seem fairly primitive, and although the area of memory where instructions are executed can only hold 128 such instructions, there is still in fact no limitation on what software can be written using them. Because the instructions can access any part of the memory, and because some instructions may be used to reprogram other ones, there are plenty of ways to extend Anitra's functionality to a more practical level. Anitra's two instructions are hence truly universal.