Computers are not what you think

(Condensed from Theodor H. Nelson, *The Computer,* to be published)

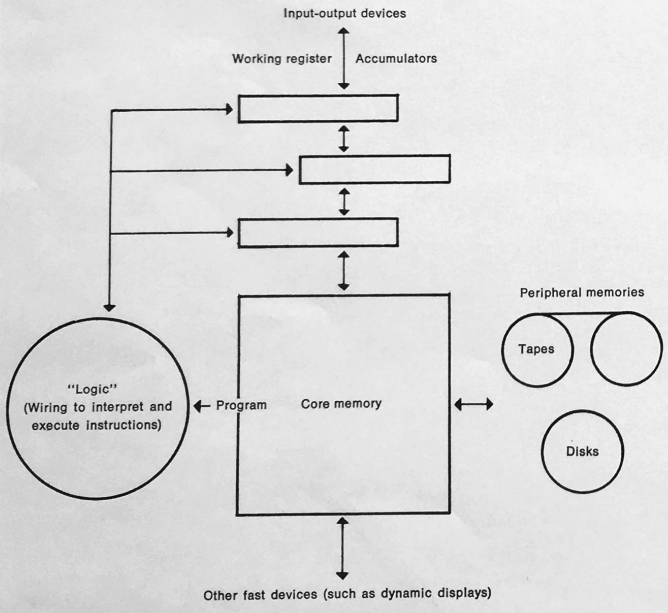
What is a digital computer?

A Digital Computer has the wrong name to begin with. John von Neumann, at the very beginning (late forties), called it the All-Purpose Machine, and that was the right name, but it got mislaid somewhere. A digital computer is a device which can be programmed to move information, shuffle information, receive information, send information, test information, make decisions (on the basis of its tests), quit what it's doing and start something else; perform long chains of activities made up of the above; and carry out necklaces of activities, each bead of which is one of the above chains.

People who understand how to select a computer's commands, combining and interrelating the machine's corresponding actions, are called *programmers.*

(People who use computers in some way, through programs with whose details they are not involved, are *users.* Not everyone who sits at a computer-connected device, such as a display, is a programmer. Soon most of us will be users, in one way or another.)

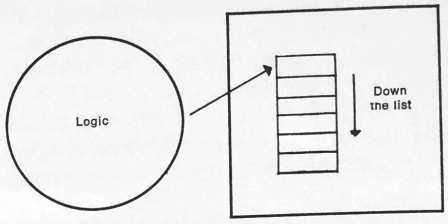
Computers can also count, and do arithmetic. But this numerical activity of computers has been vastly overemphasized. For it is their ability to vary and combine programs, test the outside world and modify it (through input and output), and handle the general diddling of vast quantities of information—such as storing it and printing it, as well as doing arithmetic on it—that makes it, in some very important sense, the ultimate machine.



A computer system ordinarily has four kinds of parts: *logic, registers, memories,* and *I-O devices.*

Logic is the wiring that causes the computer to respond to its instructions (or other events in the system).
A *register* is where something happens to information.
A *memory* is where nothing happens to information. (It stays there unchanged till requested.)
An *I-O device* is something which sends information to the computer (I is for Input) or gets information from the computer (O is for Output), or both.

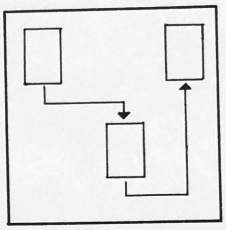
A *program* is ordinarily made of a series of patterns temporarily stored in the computer's main memory, or core memory. These patterns are *commands* to which the computer is wired to respond. They are in lists consecutively located in core memory.



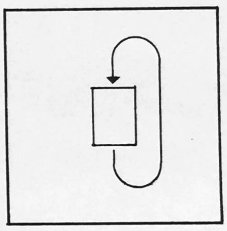
One at a time the commands are drawn from the program list into the specifies.
For instance:

test whether a register contains the same pattern as a certain location in memory.
send the contents of a register to a particular I-O device.
receive a message from an I-O device into a register.
move the contents of a particular memory location into a register.
move the contents of a register into a memory location.
add one to the contents of a register. And so on and on.

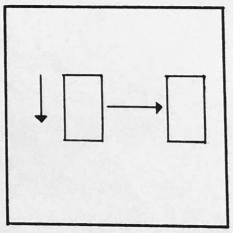
Programs stored in different parts of memory can be chained together,



or repeat themselves, as incantations,



or operate on blocks of data stored elsewhere in memory:



and much more. For instance, computer programs can summon other computer programs, which in turn—ah, never mind. Programming is creating the garlands, diadems and crosswords of instructions that make things happen.

A computer *language* is usually a set of artificial words (usually gritty-looking ones like MUL, ZONK, SNR, GLB) which a programmer may combine by some set of rules into a long spell (a program). This spell is then given in some way to another program in the computer, the "language processor", which turns the relatively few instructions in the computer language into the many ittybitty instructions needed by the computer itself. If the spell is properly cast, the system does what the programmer wants, either directly or through *more* programs ground out by the language processor.

There are thousands of languages, with different but overlapping purposes, some for arithmetic, some for text handling, some for pictures, but the majority indescribable in everyday terms. Well-known languages include FORTRAN, COBOL and ALGOL, but there are also

SNOBOL, TRAC, SIMSCRIPT, JOVIAL AND LISP. All of these languages have their jealous creators, guardians and partisans, for each language represents not just a kind of work to be done, but most likely a philosophy, an outlook, a way of handling problems in general.

Programming is creating and arranging instructions for machines and languages that already exist. Computer architecture is designing registers, instructions and logic. Software design means making languages and giant programs. Systems analysis is working out good ways to do things, with or without computers. Computer science is all these things.

The reader may observe that the possible complications are endless, but they are not unruly, they are quite fascinating, and they are not necessarily hard for ordinary people to understand. The technicalities are unavoidable; and the world is already sharply divided between those who know them and those who do not. The latter are at a disadvantage, and will remain so.

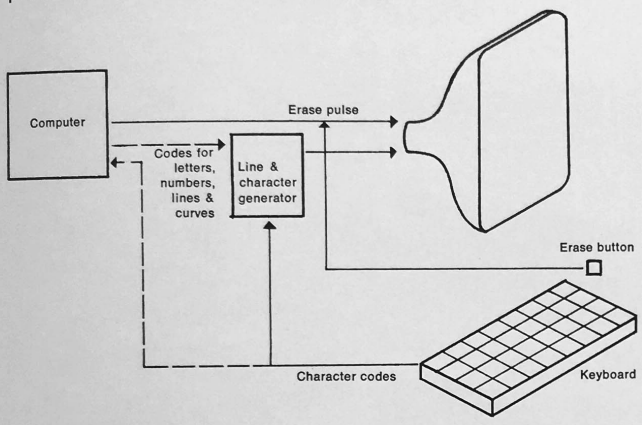
What is a computer display?

Static computer displays

1. Storage keyscope

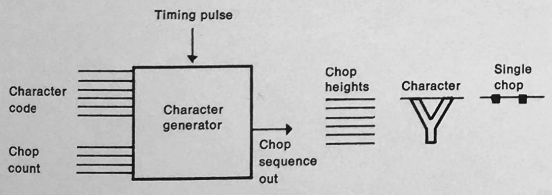
A presentation is sent slowly to this display from the main computer.

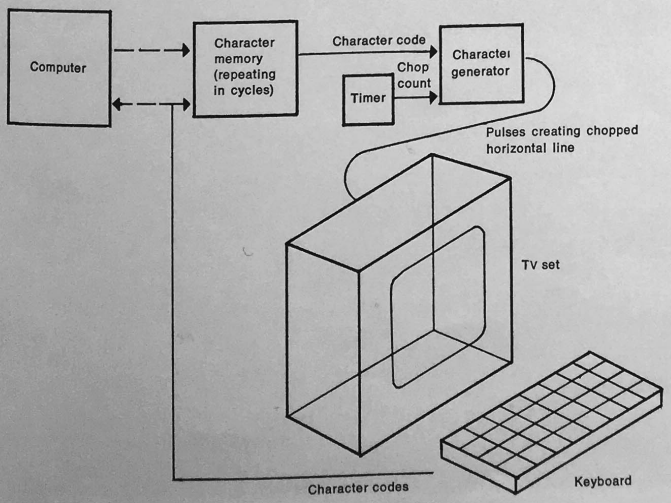
The storage tube holds the presentation until it is erased; a new presentation must be put on it from scratch.



2. TV keyscope

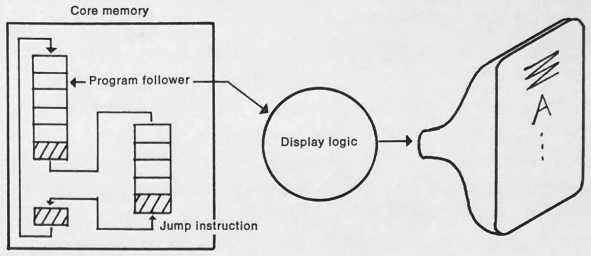
In this display, the contents of the screen are held in a local memory. In sync with the incessant television scanning beam, each character code goes to a TV character generator, which then sends out the appropriate blips to "chop" a single character at the beam's present height. Then on to the next, and so on; at the end of a line advance the chop count.





Dynamic computer display

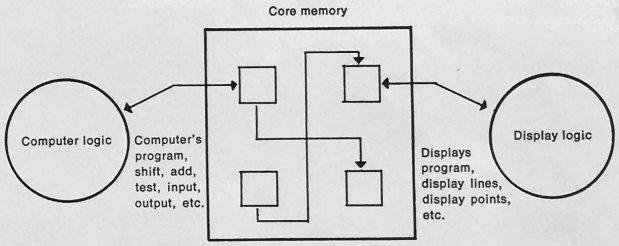
In this display, display logic (like the computer's) pulls display commands, one at a time, from a display program in a core memory. The commands in the sequence are interpreted by the display logic as successive moves of a tiny point of light on a screen, the tip of an electron beam being switched, rapier-like, around the screen's surface. When it finishes it starts over. (If the display program gets too long, it starts to flicker.)



Display program (simplified and translated)

Draw line
Draw line
Draw line
(Move beam)
Plot point
Plot point
(Move beam)
Write letter
Write letter
Write letter
(Move beam)

A high-power display ordinarily, these days, uses the core memory of a small general-purpose computer. Both can use the same memory intermittently, each program unbeknownst to the other.



However, customarily this is wired up so that the display can summon the computer when something has to be done that the display's own program and logic cannot handle. The computer then sets aside its current program, and goes to its help-the-display program. This may be needed just to keep the display going, to help the user interact with the display in some way that is prearranged in the program (to recognize a user's request to draw a line, for instance), or to actually make some numerical calculation related to the display (for instance, a program for the computer to calculate the area of a rectangle showing on the screen.)

In display programming, we use the combined abilities of both the display logic and the computer logic. (How they interact will depend on the particular equipment.) But we can create programs, for instance, to create a stick figure, and by repeating this program at different locations, fill the screen with paper dolls.

Ways to program a display system to respond to user actions, or to time a presentation and show "movies", must here be left to the reader's imagination.

(Note: a good article on computer displays by Ivan Sutherland will be found in the June 1970 issue of the *Scientific American*).