LECTURE 3

HISTORY OF COMPUTERS - HARDWARE

The history of computing probably began with primitive man using pebbles to compute the sum of two amounts. Marshack (of Harvard) found that what had been believed to be mere scratches on old bones from cave man days were in fact carefully scribed lines apparently connected with the moon's phases. The famous Stonehenge on the Salisbury plain in England had three building stages, 1900-1700, 1700-1500, and 1500-1400 B.C., and were apparently closely connected with astronomical observations, indicating considerable astronomical sophistication. Work in archeoastronomy has revealed that many primitive peoples had considerable knowledge about astronomical events. China, India, and Mexico were prominent in this matter, and we still have their structures that we call observatories, though we have too little understanding of how they were used. Our western plains have many traces of astronomical observatories that were used by the Indians.

The sand pan and the abacus are instruments more closely connected with computing, and the arrival of the Arabic numerals from India meant a great step forward in the area of pure computing. Great resistance to the adoption of the Arabic numerals (not in their original Arabic form) was encountered from officialdom, even to the extent of making them illegal, but in time (the 1400's) the practicalities and economic advantages triumphed over the more clumsy Roman (and earlier Greek) use of letters of the alphabet as symbols for the numbers.

The invention of logarithms by Napier (1550-1617) was the next great step. From it came the slide rule, which has the numbers on the parts as lengths proportional to the logs of the numbers, hence adding two lengths means multiplying the two numbers. This analog device, the slide rule, was another significant step forward, but in the area of analog not digital computers. I once used a very elaborate slide rule in the form of a (6"-8") diameter cylinder and about two feet long, with many, many suitable scales on both the outer and inner cylinders, and equipped with a magnifying glass to make the reading of the scales more accurate.

Slide rules in the 30's and 40's were standard equipment of the engineer, usually carried in a leather case fastened to the belt as a badge of one's group on the campus. The standard engineer's slide rule was a "10 inch loglog decitrig slide rule" meaning that the scales were 10" long, included loglog scales, square and cubing scales, as well as numerous trigonometric scales in decimal parts of the degree. They are no longer manufactured!

Continuing along the analog path, the next important step was the differential analyzer, which at first had mechanical integrators of the analog form. The earliest successful ones were made around 1930 by Vanevar Bush of MIT. The later RDA #2, while still analog and basically mechanical, had a great deal of electronic interconnections. I used it for some time (1947-8) in computing Nike guided missile trajectories in the earliest design stages.

During WWII the electronic analog computers came into the military field use. They used condensers as integrators in place of the earlier mechanical wheels and balls (hence they could only integrate with respect to time). They meant a large, practical step forward, and I used one such machine at Bell Telephone Laboratories for many years. It was constructed from parts of some old M9 gun directors. Indeed, we used parts of some later condemned M9's to built a second computer to be used either independently or with the first one to expand its capacity to do larger problems.

Returning to digital computing Napier also designed "Napier's bones" which were typically ivory rods with numbers that enabled one to multiply numbers easily; these are digital and not to be confused with the analog slide rule.

From the Napier bones probably came the more modern desk calculators. Schickert wrote (Dec. 20, 1623) to Kepler (of astronomical fame) that a fire in his lab burned up the machine he was building for Kepler. An examination of his records and sketches indicates that it would do the four basic operations of arithmetic - provided you have some charity as to just what multiplication and division are in such a machine. Pascal (1623-1662) who was born that year is often credited with the invention of the desk computer, but his would only add and subtract - only those operations were needed to aid his tax collecting father. Leibnitz (of calculus fame) also tinkered with computers and included multiplication and division, though his machines were not reliable.

Babbage (1791-1871) is the next great name in the digital field, and he is often considered as the father of modern computing. His first design was the difference engine, based on the simple idea that a polynomial can be evaluated at successive, equally spaced, values by using only a sequence of additions and subtractions, and since locally most functions can be represented by a suitable polynomial this could provide "machine made tables", (Babbage insisted that the printing be done by the machine to prevent any human errors creeping in). The English Government gave him financial support, but he never completed one. A Norwegian father and son (Scheutz) did make several that worked and Babbage congratulated them on their success. One of their machines was sold to the Albany Observatory, New York, and was used to make some astronomical tables.

As has happened so often in the field of computing, Babbage had not finished with the difference engine before he conceived of the much more powerful analytical engine, which is not far from the current von Neumann design of a computer. He never got it to work; a group in England constructed (1992) a machine from his working drawings and successfully operated it as he had designed it to work!

The next major practical stage was the Comptometer which was merely an adding device, but by repeated additions, along with shifting, this is equivalent to multiplication, and was very widely used for many, many years.

From this came a sequence of more modern desk calculators, the Millionaire, then the Marchant, the Friden, and the Monroe. At first they were hand controlled and hand powered, but gradually some of the control was built in, mainly by mechanical levers. Beginning around 1937 they gradually acquired electric motors to do much of the power part of the computing. Before 1944 at least one had the operation of square root incorporated into the machine (still mechanical levers intricately organized). Such hand machines were the basis of computing groups of people running them to provide computing power. For example, when I came to the Bell Telephone Laboratories in 1946 there were four such groups in the Labs, typically about six to ten girls in a group; a small group in the math dept, a larger one in network department, one in switching, and one in quality control.

Punched card computing began because one far seeing person saw that the Federal census, that by law must be done every 10 years, was taking so much time that the next one (1890) would not be done before the following one started unless they turned to machine methods. Hollerith, took on the job and constructed the first punched card machines, and with succeeding censuses he built more powerful machines to keep up with both the increased population and the increased number of questions asked on the census. In 1928 IBM began to use cards with rectangular holes so that electric brushes could easily detect the presence or absence of a hole on a card at a given place. Powers, who also left the census group, kept the card form with round holes that was designed to be detected by mechanical rods as "fingers".

Around 1935 the IBM built the 601 mechanical punch which did multiplications, and could include two additions to the product at the same time. It became one of the mainstays of computing - there were about 1500 of them on rental and they averaged perhaps a multiplication per 2 or 3 seconds. These, along with some special triple product and division machines, were used at Los Alamos to compute the designs for the first atomic bombs.

In the mechanical, meaning relay, area George Stibitz built (1939) the complex number computer and exhibited it at Dartmouth (1940) when the main frame was in New York, thus an early remote terminal machine, and since it normally had three input stations in different locations in the Labs it was, if you are kind, a "time shared machine".

Konrad Zuse in Germany, and Howard Aitken at Harvard, like

Stibitz, each produced a series of relay computers of increasing complexity. Stibitz's Model 5 had two computers in the same machine and could share a job when necessary, a multiprocessor machine if you wish. Of the three men probably Zuse was the greatest, considering both the difficulties he had to contend with and his later contributions to the software side of computing.

It is usually claimed that the electronic computer age began with the ENIAC built for the U.S. Army and delivered in 1946. It had about 18,000 vacuum tubes, was physically huge, and as originally designed it was wired much like the IBM plug boards, but its interconnections to describe any particular problem ran around the entire machine room! So long as it was used, as it was originally intended, to compute ballistic trajectories, this defect was not serious. Ultimately, like the later IBM CPC, it was cleverly rearranged by the users to act as if it were programmed from instructions (numbers on the ballistic tables) rather than from wiring the interconnections.

Mauchly and Eckert, who built the ENIAC, found, just as Babbage had, that before the completion of their first machine they already envisioned a larger, internally programmed, machine, the EDVAC. Von Neumann, as a consultant to the project, wrote up the report, and as a consequence internal programming is often credited to him, though so far as I know he never either claimed or denied that attribution. In the summer of 1946 Mauchly and Eckert gave a course, <u>open to all</u>, on how to design and build electronic computers, and as a result many of the attendees went off to build their own; Wilkes, of Cambridge, England, being the first to get one running usefully, the EDSAC.

At first each machine was a one-of-a-kind, though many were copied from the Institute for Advanced Studies machine under von Neumann's direction, but the engineering of that machine was apparently held up. As a result, many of the so-called copies, like the MANIAC-I (1952), (which was named to get rid of the idiotic naming of machines), and built under the direction of N. C. Metropolis, was finished before the Institute machine. It, and the Maniac-II (1955), were built at Los Alamos, while the Maniac-III (1959) was built at the University of Chicago. The Federal government, especially through the military, supported most of the early machines, and great credit is due to them for helping start the Computer Revolution.

The first commercial production of electronic computers was under Mauchly and Eckert again, and since the company they formed was merged with another, their machines were finally called UNIVACS. Especially noted was the one for the Census Bureau. IBM came in a bit late with 18 (20 if you count secret cryptographic users) IBM 701's. I well recall that a group of us, after a session on the IBM 701 at a meeting where they talked about the proposed 18 machines, all believed that this would saturate the market for many years! Our error was simply that we thought only of the kinds of things we were currently doing, and did not think in the directions of entirely new applications of machines. The best experts at the time were flatly wrong! And not by a small amount either! Nor for the last time!

Let me turn to some comparisons:

Hand calculators	1/20 ops. per sec.
Relay machines	1 op. per sec. typically
Magnetic drum machines	15-1000 depending somewhat on fixed or floating point
701 type	1000 ops. per sec.
Current (1990)	10^9 (around the fastest of the von Neumann type)

The changes in speed, and corresponding storage capacities, that I have had to live through should give you some idea as to what you will have to endure in your careers. Even for von Neumann type machines there is probably another factor of speed of around 100 before reaching the saturation speed.

Since such numbers are actually beyond most human experience I need to introduce a human dimension to the speeds you will hear about. First notation (the parens. contain the standard symbol)

milli (m)	10^{-3}	kilo (K)	10^3
micro (μ)	10^{-6}	mega (M)	10^6
nano (n)	10^{-9}	giga (G)	10^9
pico (p)	10^{-12}	terra (T)	10^{12}
femto (f)	10^{-15}		
atto (a)	10^{-18}		

Now to the human dimensions. In one day there are 60x60x24 = 86,400 seconds. In one year there are close to $3.1\text{x}10^7$ seconds, and in 100 years, probably greater than your lifetime, there are about $3.1\text{x}10^9$ seconds. Thus in 3 seconds a machine doing 10^9 floating point operations per second (flops) will do more operations than there are seconds in your whole lifetime, and almost certainly get them all correct!

For another approach to human dimensions, the velocity of light in a vacuum is about $3\text{x}10^{10}$ cm/sec, (along a wire it is about 7/10 as fast). Thus in a nanosecond light goes 30 cm, about one foot. At a picosecond the distance is, of course, about 1/100 of an inch. These represent the distances that a signal can go (at best) in an IC. Thus at some of the pulse rates we now use the parts must be very close to each other - close in human dimensions - or else much of the potential speed will be lost in going between parts. We can also no longer used lumped circuit analysis.

How about natural dimensions of length instead of human dimensions? Well, atoms come in various sizes running generally around 1 to 3 angstroms (an angstrom is 10^{-8} cm.) and in a crystal are spaced around 10 angstroms apart, typically, though there are exceptions. In 1 femtosecond light can go across about 300 atoms. Therefore the parts in a very fast computer must be small and very close together!

If you think of a transistor using impurities, and that the impurities run around 1 in a million typically, then you would probably not believe a transistor with 1 impure atom, but maybe, if you lower the temperature to reduce background noise, 1000 impurities is within your imagination - thus making the solid state device of at least around 1000 atoms on a side. With interconnections at times running at least 10 device distances you see why you feel that getting below 100,000 atoms distance between some interconnected devices is really pushing things, (3 pico seconds).

Then there is heat dissipation. While there has been talk of thermodynamically reversible computers, so far it has only been talk and published papers, and heat still matters. The more parts per unit area, and the faster the rate of state change, the more the heat generated in a small area that must be gotten rid of before things melt. To partially compensate we have been going to lower, and lower voltages, and are now going to 2 1/2 or 3 volts operating the IC. The possibility that the base of the chip have a diamond layer is currently being examined since diamond is a very good heat conductor, much better than copper. There is now a reasonable possibility for a similar, possibly less expensive, crystal structure with very good heat conduction properties.

To speed up computers we have gone to 2, to 4, and even more, arithmetic units in the same computer, and have also devised pipelines and cache memories. These are all small steps towards highly parallel computers.

Thus you see the handwriting on the wall for the single processor machine - we must be approaching saturation. Hence the fascination with highly parallel machines. Unfortunately there is as yet no single general structure to them, but rather many, many competing designs, all generally requiring different strategies to exploit their potential speeds and having different advantages and disadvantages. It is not likely that a single design will emerge for a standard parallel computer architecture, hence there will be trouble and dissipation in efforts to pursue the various promising directions.

From a chart drawn up long ago by Los Alamos (LANL) using the data of the fastest current computer on the market at a given time they found that the equation for the number of operations per second was

$$n(t) = \exp\{22(1 - e^{-t/20})\}$$

and it fitted the data fairly well. Here time begins at 1943. In 1987 the extrapolated value predicted (by about 20 years!) was about 3×10^8 and was on target. The limiting asymptote is 3.576×10^9 for the von Neumann type computer with a single processor.

Here, in the history of the growth of computers, you see a realization of the "S" type growth curve; the very slow start, the rapid rise, the long stretch of almost linear growth in the rate, and then the facing of the inevitable saturation.

Again, to reduce things to human size. When I first got digital computing really going inside Bell Telephone Laboratories I began by renting computers outside for so many hours that the head of the math dept figured out for himself that it would be cheaper to get me one inside - a deliberate plot on my part to avoid arguing with him as I thought it useless and would only produce <u>more resistance</u> on his part to digital computers. Once a boss says "no!" it is very hard to get a different decision, so don't let them say "No!' to a proposal. I found in my early years that I was doubling the number of computations per year about every 15 months. Some years later I was reduced to doubling the amount about every 18 months. The department head kept telling me that I could not go on at that rate forever, and my polite reply was always, "You are right, of course, but you just watch me double the amount of computing every 18-20 months! It was the fact that the machines available kept up the corresponding rate that enabled me, and my successors, to do it for many years. We lived on the almost straight line part of the "S" curve all those years.

However, let me observe in all honesty to the Dept. head, it was remarks by him that made me realize that it was not the number of operations done that mattered, it was, as it were, the number of micro Nobel prizes I computed that mattered. Thus the motto of a book I published in 1961:

THE PURPOSE OF COMPUTING IS INSIGHT, NOT NUMBERS.

A good friend of mine revised it to:

The purpose of computing numbers is not yet in sight.

It is necessary now to turn to some of the details of how for many years computers were constructed. The smallest parts we will examine are two state devices for storing bits of information, and for gates that either let a signal go through or block it. Both are binary devices, and in the current state of knowledge they provide the easiest, fastest methods of computing that we know.

From such parts we construct combinations that enable us to store longer arrays of bits; these arrays are often called number registers. The logical control is just a combination of storage units including gates. We build an adder out of such devices, as well as every larger unit of a computer.

Going to the still larger units we have the machine consisting of: (1) a storage device, (2) a central control, (3) an ALU unit, meaning arithmetic and logic unit. There is in the central control a single register which we will call the current address register (CAR). It holds the address of where the next instruction is to be found, Figure 3-1.

The cycle of the computer is:

1. Get the address of the next instruction from the CAR

2. Go to that address in storage and get that instruction

3. Decode and obey that instruction

4. Add 1 to the CAR address, and start in again.

We see that the machine does not know where it has been, nor where it is going to go; it has at best only a myopic view of simply repeating the same cycle endlessly. Below this level the individual gates and two way storage devices do not know any meaning - they simply react to what they are supposed to do. They too have no global knowledge of what is going on, nor any meaning to attach to any bit, whether storage or gating.

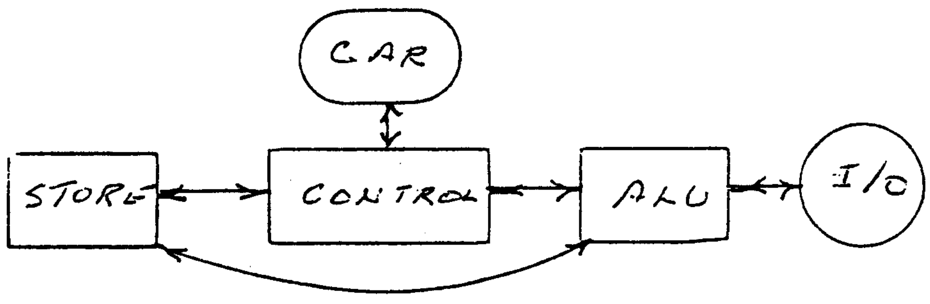
There are some instructions which, depending on some state of the machine, put the address of their instruction into the CAR, (and 1 is not added in such cases), and then the machine, in starting its cycle, simply finds an address that is not the immediate successor in storage of the previous instruction, but the location inserted into the CAR.

I am reviewing this so you will be clear that the machine processes bits of information according other bits, and that as far as the machine is concerned there is no meaning to anything that happens, - it is we who attach meaning to the bits. The machine is a "machine" in the classical sense; it does what it does and nothing else (unless it malfunctions). There are, of course, real time interrupts, and other ways that new bits get into the machine, but to the machine they are only bits.

But before we leave the topic, recall that in ancient Greece Democritus (460?-362?) observed that, "All is atoms and void." He thus expressed the view of many physicists today, that the world, including you and me, is made of molecules, and we exist in a radiant energy field. There is nothing more! Are we machines? Many of you do not wish to settle for this, but feel that there is more to you than just a lot of molecules banging against one another mindlessly, which we see is one view of a computer. We will examine this point in Lectures 6-8 under the title of Artificial Intelligence (AI).

There is value in the machine view of a computer, that it is just a collection of storage devices and gates processing bits, and nothing more. This view is useful, at times, when debugging (finding errors) in a program; indeed that is what you must assume when you try to debug. You assume that the machine obeys the instructions one at a time, and does nothing more - it has no "free will" or any of the other attributes such as the self-awareness and self-consciousness we often associate with humans.

How different are we in practice from the machines? We would all like to think that we are different from machines, but are we essentially? It is a touchy point for most people, and the emotional and religious aspects tend to dominate most arguments. We will return to this point in the Lectures 6-8 on AI when we have more background to discuss it reasonably.



Computer Structure

Figure 3-1