Parallel Computing Prehistory

John Burkardt
Information Technology Department
Virginia Tech

.....

FDI Summer Track V:
Using Virginia Tech High Performance Computing
https://people.sc.fsu.edu/~jburkardt/presentations/...
history_2009_vt.pdf

26-28 May 2009

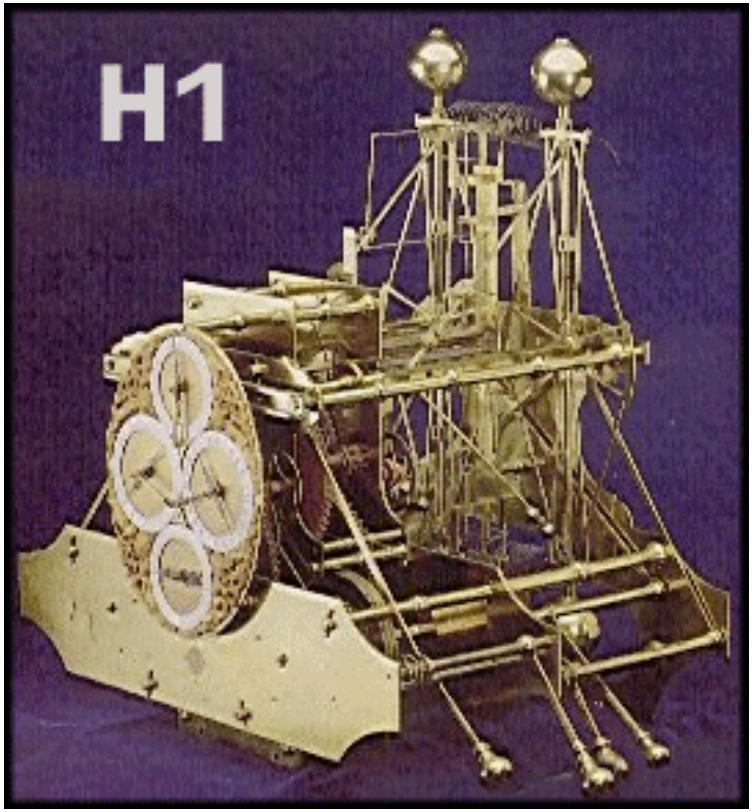
How Did We Get Here? [date uncertain]

I don't know where I am, but I'm making record time!
Last transmission from a Navy pilot, somewhere in the Pacific...

What Happens if You Don't Know Where You Are

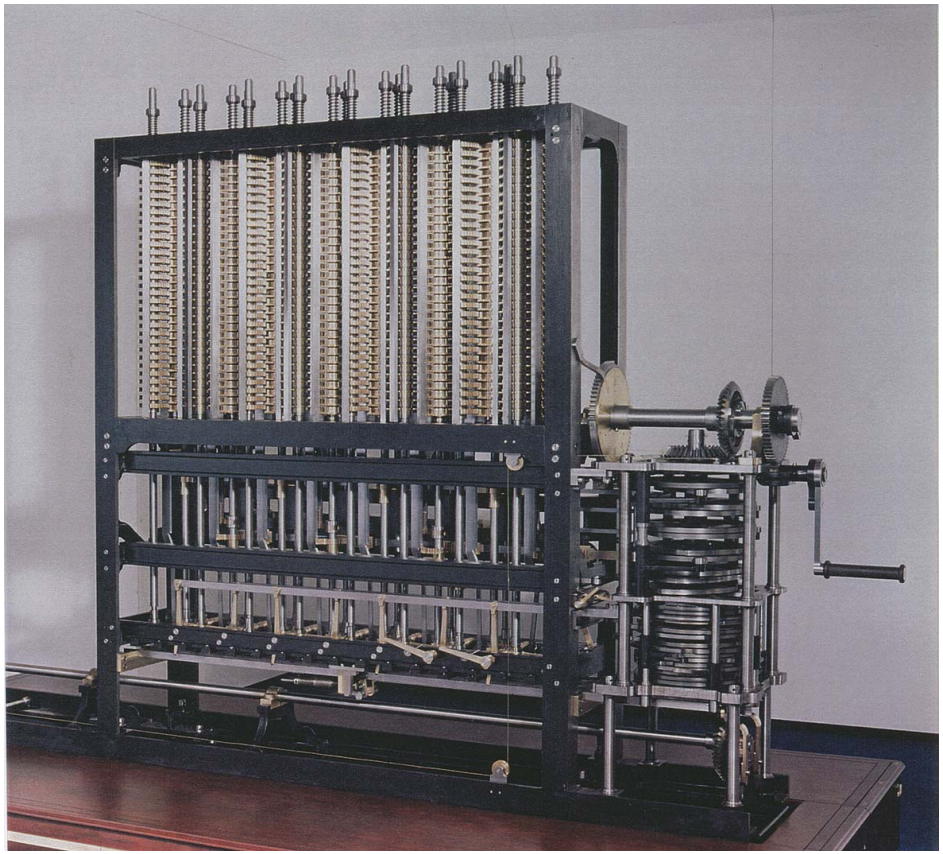
In 1707, Admiral Sir Cloudesly Shovell sank his fleet by sailing directly into the Scilly Islands, which weren't where he thought they were...because he didn't know where he was.

Fine Clockwork Can Tell You Where You Are



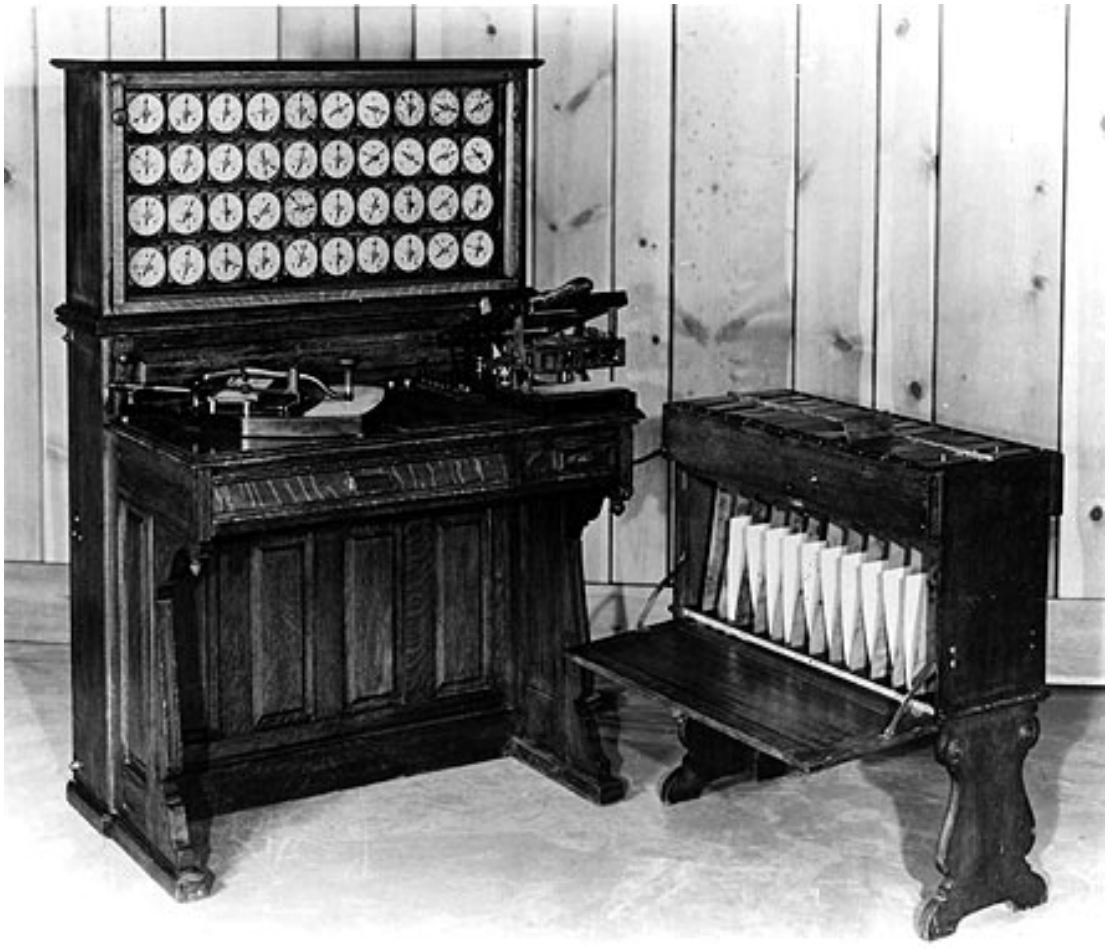
The British Admiralty put out an RFP, seeking a reliable means of determining longitude. John Harrison's "superclock" solved the problem...though of course he had much trouble getting reimbursed!

Babbage Shows Calculation is Like Clockwork



The Navy still had plenty of navigational problems in the 19th century, and (partially) funded Charles Babbage's Difference Engine and his Analytical Engine.

Hollerith Automates Data Processing

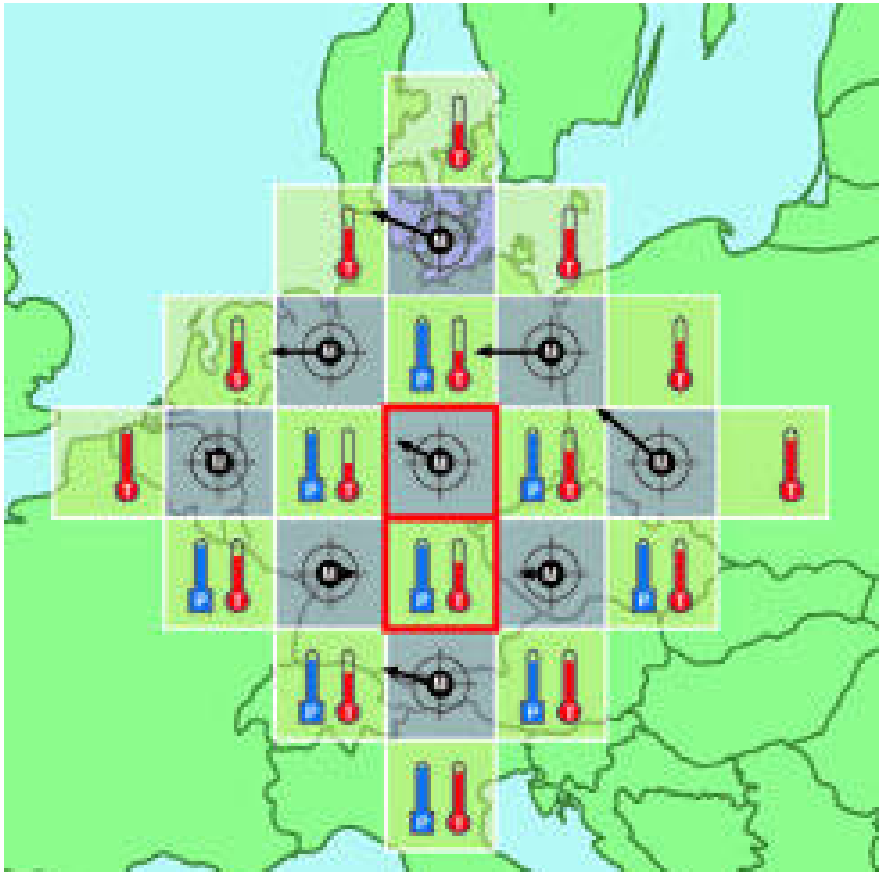


Hermann Hollerith, inspired by an automatic weaving device, developed a machine to tabulate the 1890 census - later a big seller for the International Business Machine Corporation, aka IBM.

Numerical Calculations Were Done by "Computers"

Computing was done by computers, that is, **people**. This is the computer lab for the Harvard Astronomical Observatory in 1890.

Richardson Forecasts the Weather



Lewis Richardson attempted the first weather forecast in 1917. (It was for 20 May 1910; it was wrong.)

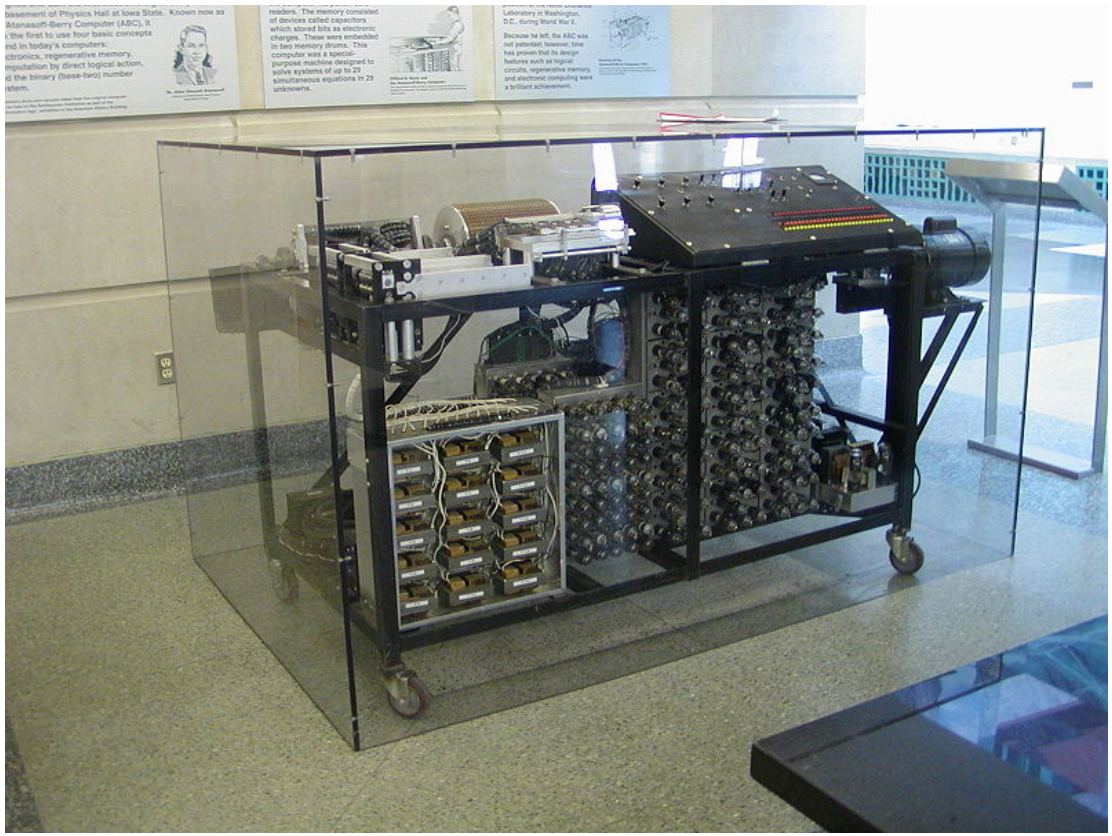
The Atanasoff Computer

John Atanasoff, of Iowa State University, took a midnight drive to the Mississippi and worked out the final details for the "ABC" machine.

It could only solve linear systems, up to order 29, by entering two equations at a time by hand, and eliminating one variable and outputing the revised pair of equations.

As the result of a lawsuit settled in 1973, the ABC machine was declared the first working computer. (Though it never worked)

The Atanasoff Computer



The Atanasoff-Berry Computer or "ABC"

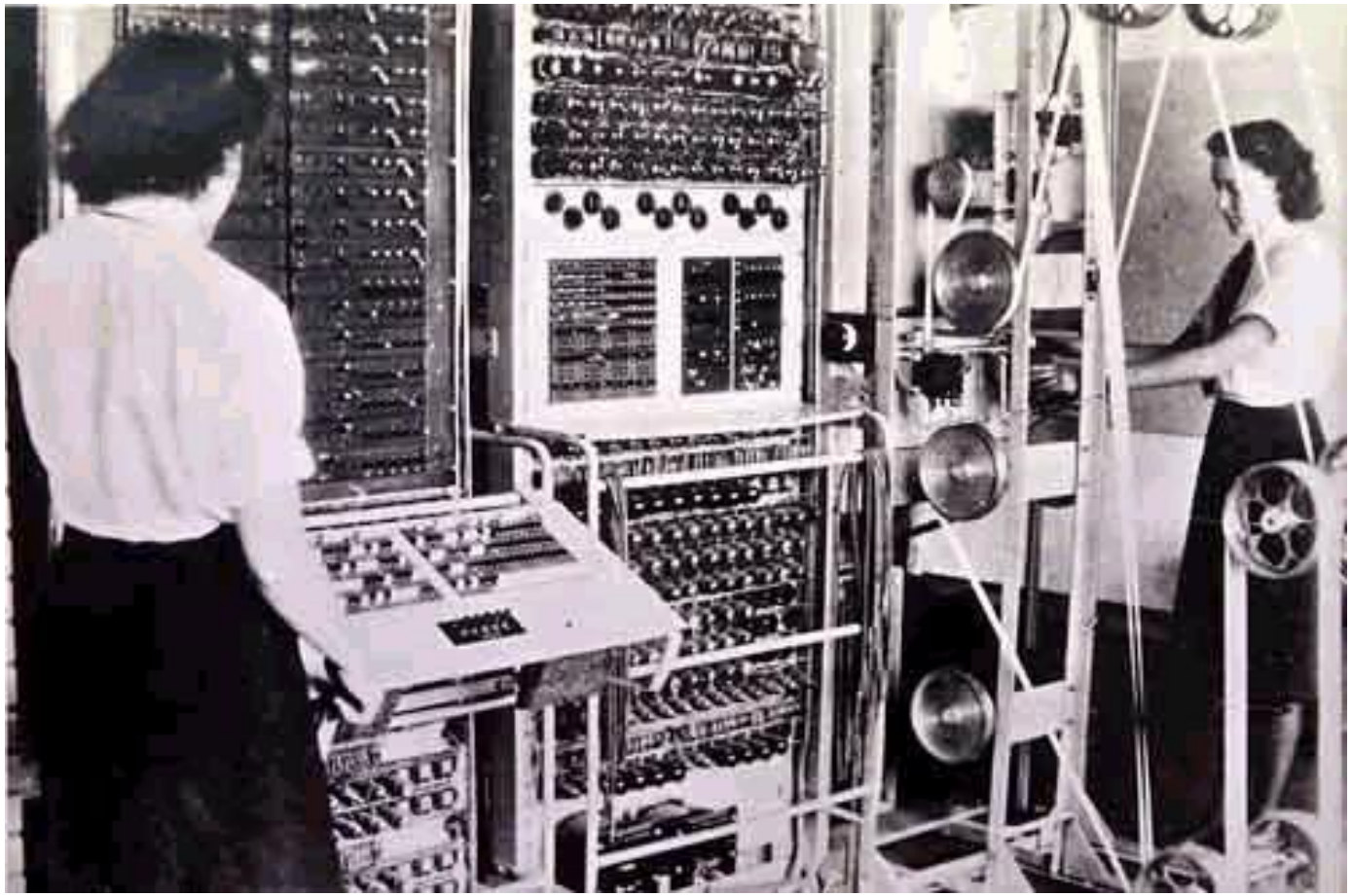
The COLOSSUS Computer

Early computers were "hardwired" for very specific tasks and could not be reprogrammed.

The Colossus computer, built in World War II, read a paper tape punched with a ciphered message, and compared it to another tape containing output from a cipher machine, looking for matches.

(As the tapes were spun back and forth, they tended to stretch, causing errors and delays!)

The COLOSSUS Computer



The Colossus Computer, with which Alan Turing was associated.

The Password was SLOW

Everything about early computers was so slow that there was no obvious reason to speed anything up.

The data was punched onto cards or paper tape, and read, one at a time, as needed.

The "program" might be literally hardwired (the computer could do one thing.)

More advanced computers might have plugs (like old telephone switches) that allowed the operator to modify the computation.

Everything was so slow that there were no obvious inefficiencies!

ENIAC Arrives

World War II showed the power of computing.

ENIAC was the first **general purpose** electronic computer.

Initially intended to compute artillery firing tables, it could be reprogrammed to solve a wide range of mathematical problems.

ENIAC could do about 5000 additions or 400 multiplications per second.

ENIAC Could Store Data Internally

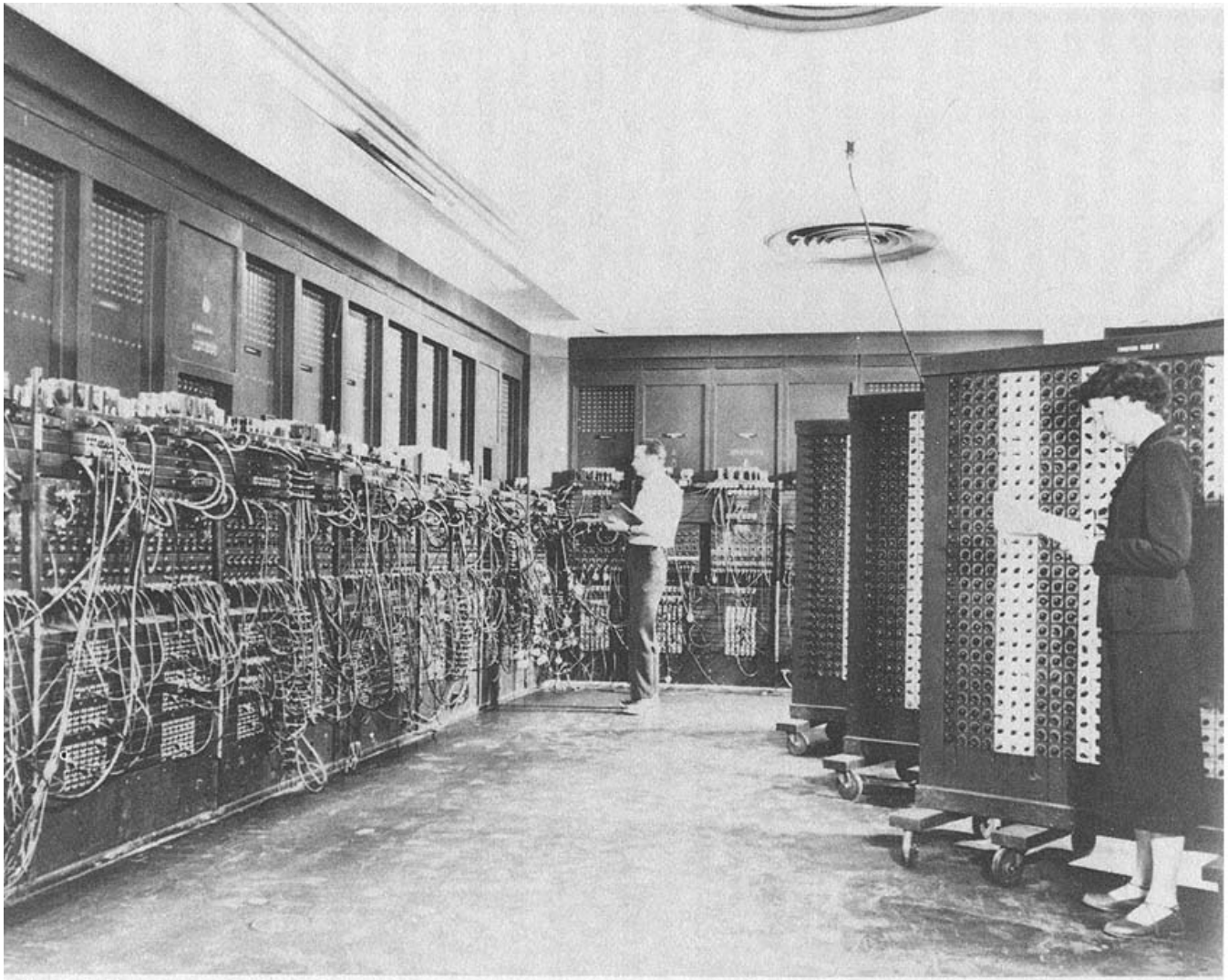
ENIAC could store 20 numbers of 10 decimal digits.

Each decimal digit was stored as a voltage level in a vacuum tube and the set of tubes was known as an **accumulator**.

One number could be added to another by rearranging plugs on a board; this caused the voltage levels in one set of vacuum tubes to be sent to the other, in such a way that the corresponding digits were incremented.

Since each accumulator essentially could perform arithmetic independently of the others, it is sometimes claimed that this was a beginning of parallel processing!

ENIAC Weighed 30 Tons



The ENIAC.
Smithsonian Institution Photo No. 53192.

Some Parts of ENIAC Were Not Slow Enough

The ENIAC processor was so fast that most of the time it was doing nothing, while waiting for new instructions.

The computer's designers realized that the computer would be even faster if the program itself was also stored in memory, along with the data.

EDVAC, the second version of the machine, was one of the first **stored program computers**. It also, thank goodness, switched to binary arithmetic!

Here was an early example where speeding up one part of a computer meant changes in other parts.

ENIAC inspires the von Neumann Architecture

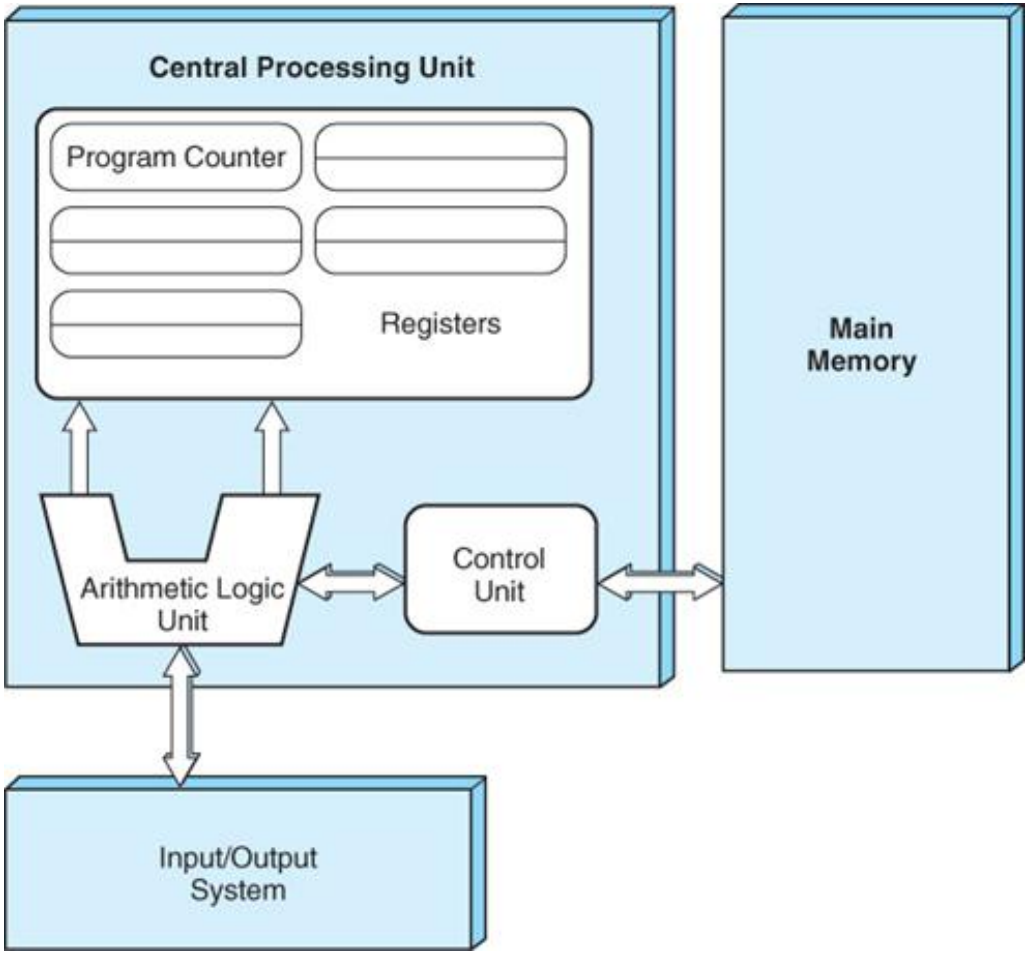
ENIAC was actually useful (!) so it required a user manual.

Princeton mathematician John von Neumann studied the machine and wrote up instructions and a description, now known as *the von Neumann architecture*.

The main feature was the division of the computer into **main memory** and the **central processing unit** or **CPU**.

A controller unit fetched data and instructions from memory.

The von Neumann Architecture



The von Neumannn Architecture

The von Neumann architecture is the standard model of a computer.

New advances required adjusting these "blueprints".

Memories got enormous, so it took more time to find data.

Processors got faster, so they needed data faster.

Result: the **von Neumann bottleneck**

The von Neumann Bottleneck

Faster CPU's + huge memories => "starvation".

The von Neumann bottleneck showed that there was a third important problem in computation:

- memory
- processing power
- **communication**!

Supercomputers

The power of computers created an insatiable demand for bigger and faster computations.

With unlimited money available, very fast (and very expensive) new hardware was developed.

Supercomputers in the late '70s and '80s were able to achieve faster computation by faster communication:

- increasing the connections between memory and CPU
- using faster connections
- moving the data closer to the memory

Supercomputers - Vectorization

Supercomputers also began to find new ways to improve performance by dealing with the von Neumann bottleneck.

When a computation involved carrying out the same operations on many elements of a vector, the computer could achieve far greater speeds than seemed possible.

- the CPU had two input connections to memory, and one output, all of which could be operating independently
- the CPU included several functional units, which could all do useful work independently (computing addresses, doing arithmetic
- the arithmetic functional units had a sort of internal "assembly line"; even though an arithmetic operation might take 5 or 6 clock cycles, the unit could produce a new result every clock cycle.

Light Speed Not Fast Enough

As processor clock cycles speeded up, an interesting problem was discovered:

```
c = speed of light
  =     186,282     miles / second
  = 983,568,960     feet  / second
  =     983,568     feet  / millisecond   (1 kiloHz)
  =         983     feet  / microsecond   (1 megaHz)
  =           0.983 feet  / nanosecond    (1 GigaHz)
```

Light Goes 1000 Feet A MicroSecond

Grace Hopper illustrates how far light can go in a microsecond. She handed out rulers for nanoseconds, and packets of pepper for picoseconds!

Light Goes 1000 Feet A MicroSecond

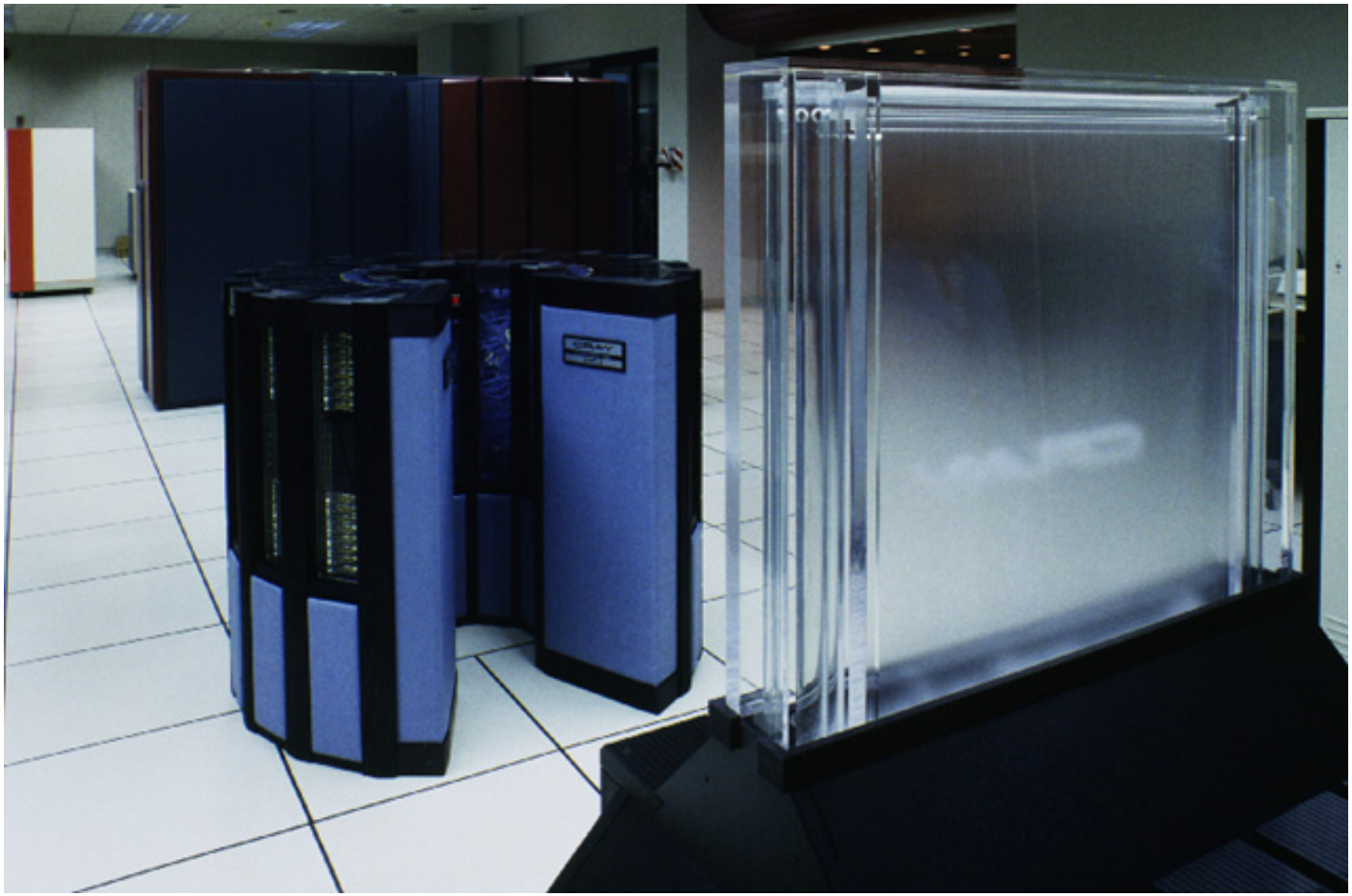
Today's fast computer chips have clock cycles of about 4 GigaHertz.

That means that in a single clock cycle, a beam of light can travel at most 3 inches (optimistic!).

Faster clocks require chip components too small to manufacture, **and** more heat in smaller spaces.

So the computer chips of tomorrow and the forseeable future will also have clock cycles of about 4 GigaHertz.

Powerful Chips + Crowded Memories = BIG HEAT



Shrinking the memory and using superfast processors leads to a new computational problem: **cooling**!

Powerful Laptops Burn Up Too

Supercomputers weren't the only victims of overheating. Laptop design meant that "hot" chips were crowded together.

Physics Calls a Timeout

Two physics problems:

- the speed of light (processor-memory distance)
- heat transfer (tiny chips get too hot)

Two software/hardware solutions:

- moderately fast communication between separate chips
- multiple processors on one chip

Distributed Memory Computing

Instead of very fast, very expensive specialized architecture, try linking massive numbers of cheap processors.
1994: Cluster of 16 personal computers connected by Ethernet.

Distributed Memory Computing

A cluster of 8 Playstation2's or 32 Linux boxes could actually cooperate to solve certain problems.

These problems tended to be "embarassingly parallel" perhaps many copies of the same program, each with a different subset of the data.

The computation was only efficient if communication between computers was very low, as the communication links were quite inefficient.

Distributed Memory Computing - Communication Systems

Results from these early distributed memory systems were so interesting that people began to figure out how to connect processors so they could communicate more quickly.

Distributed memory computing became competitive.

Communication networks include:

- EtherNet
- Infiniband
- Myrinet
- NUMAlink (SGI)
- QSnet

Distributed Memory Computing: Servers+Communication

Orange cables connect servers to Myrinet switch.

Distributed Memory Computing - Communication Issues

Communication Network Issues:

- **bandwidth** - how much can you send at one time?
- **speed** - how fast can you send it?
- **latency** - delay before message can be sent
- **scalability** - can same configuration be used for bigger systems?

Distributed Memory Computing - MPI Standard

The **MPI** standard defines how a computation can be carried out on a distributed memory system.

Each computer gets a copy of the same program.

The program begins on all computers simultaneously;

Each computer knows its own ID, and how many others there are.

Computers are free to send results to each other.

Shared Memory Computing

The second variation of the von Neumann architecture was to place multiple processors on a single chip.

This kept the processors close to the memory, enabled chip designers to get more theoretical output using less power.

Software issue: Who's in charge? Can cores cooperate on a task? How does the user "talk" to the chip?

Each processor maintains a private cache of some of the common data. When a common data item is changed, what mechanism guarantees that cached copies are updated?

Shared Memory Computing

A strong consistent push for multicore architecture began with the graphics and visualization groups, particularly at SGI.

Rendering 2D graphics involved a simple repetitive sequence of operations, easily divided up among a set of stripped-down cores.

Shared Memory Computing

The initial efforts showed that simple tasks could run twice as fast with two cores, four times as fast with four. And cores were getting cheaper.

Soon 3D graphics could be handled in real time, and money from game companies was paying for more research.

The graphical processor units (GPU), originally inflexible and highly specialized, were soon competitive with traditional architectures.

The success of multicore chips in graphics programming showed a way for general purpose programming to develop.

Shared Memory Computing

Manufacturers are rapidly developing multicore architecture.

They are already designing processors with tens and hundreds of cores!

Parallel programs that take advantage of this multicore architecture are called **shared memory** systems.

Shared Memory Computing - the OpenMP Standard

The **OpenMP** interface is a shared memory standard.

OpenMP allows users to write programs in C or Fortran that access the underlying parallel abilities of multicore chips.

OpenMP standard emphasizes:

- ease of use (hide details from user)
- ability to convert programs one step at a time
- ability to run in sequential mode
- ability to adapt to any number of processors

Parallel MATLAB Computing

Virginia Tech has always had a large and devoted group of MATLAB programmers.

But MATLAB has not been available as a parallel programming language.

Recently, the MathWorks has begun adding parallel features to MATLAB, making it possible to run on shared memory systems.

A copy of this parallel MATLAB has been running on an experimental cluster at Virginia Tech. (We will have access to this cluster during the workshop).

In June or July, Virginia Tech will install a substantial new cluster that will enable parallel MATLAB to run with up to 64 processors at a time.

That Was How We Got Here! This is Where We're Going!

In this workshop, we will present some ideas on parallel programming, using parallel MATLAB, and the **OpenMP** and **MPI** interfaces.

Some of what we do can be tested out as well on your laptop or desktop (particularly parallel MATLAB and **OpenMP**).

To use **MPI** requires access to a cluster of computers such as System X.

Access to both good software and good hardware is crucial for users of the supercomputing facilities offered by the VT Advanced Research Computing facility.

This is How We Got Here! This is Where We're Going!

A satisfied user of Virginia Tech's System X.