PHYD57 – Advanced Computations in Physical Sci.

LECTURE 1

- The 3rd Science: Computing. 2 Examples: astro & neural
- Structure of the course
- History of early computing in a few slides
- History of modern scientific computing
- Supercomputing today (incl. at UTSC)
- Languages of High Perf. Computing

Literature

1. Paul Ceruzzi "A history of modern computing", 2nd ed., MIT Press 2003
2. http://computerhistory.org - Computer History Museum online
3. Norbert Schorghofer "Lessons in Scientific Computing" (2019)
5. Joshua Izaac & Jingbo Wang, "Computational Quant. Mech.", Springer (2018) – chapters on Python, Fortran

Is that a fact?

ARCHIMEDES and the first computer

In 3rd century BC, the most famous scientist of antiquity **Archimedes (Ἀρχῑμήδης, 287-212 BC)** worked in the Greek colony of Syracuse, in eastern Sicily (Italy).

He was a famous mathematician and a physicist, also applied physicist (engineer) tasked by kings Hiero and Hieronymus to defend Syracuse with his mechanical contraptions during the Roman siege of Syracuse. This ended with Archimedes being killed, while he was drawing circles in the sand, according to a believable legend. Soldier of general Marcellus did not follow orders to capture him alive.

ARCHIMEDES PALIMSEST

palimpsest = parchment (calves skin) which has been rubbed/erased to make space for a new text (e.g., Middle Age prayer book)

Archimedes created the first laws of mechanics, such as the law of the lever, which also governs pulleys, buoyancy etc
In mathematics, until recently he was celebrated for his calculation of the area of circle, area under the parabola, area and volume of the sphere and the ratio of volumes of a cylinder and an inscribed sphere.

In a most interesting recent investigation, we found out from a previously lost book (Codex C or Archimedes' Palimpsest) that he understood and used infinity and infinitesimals, rediscovered after 1000+ years. Essentially, Archimedes was using integral calculus to derive areas and volumes.

ARCHIMEDES' PALIMSEST

- before 213 BCE: Archimedes writes the treatise
- in 10th century CE, monks copy his original treatise
- in 1229 someone erases and recycles it as valuable parchment
- Palimpsest gets deposited in Metochion, Greek library in Constantinople (now Ankara, Turkey) for many centuries
- in 1880s, a catalog mentions a math text in Constantinople
- in 1906, Danish historian Johan L. Heiberg examines the 174-page book after suspecting correctly it's a treasure. The book is in decent shape.
- in 1920s, the book disappears from Ankara, and re-surfaces as private possession in Paris

ARCHIMEDES' PALIMSEST

- the book disappears again, gets "augmented"
- by falsified drawings to increase its value (during WWII)
- stored inappropriately, it decays from the relatively good state it was in 1906, due to mold etc.
- Palimpsest is sold at Sotheby's auction in London in 1998 for $2 million to an anonymous buyer, who lets Walters Art Gallery staff (Baltimore, Md) and international experts restore and decipher *The Method of Mechanical Theorems* and other treatises, previously assumed lost.
- One lost book *On the Spheres* probably describes the first computer.
- A 20th century finding is not that computer, but it's Archimedes-compatible

ANTIKYTHERA MECHANISM = ancient analog/digital computer

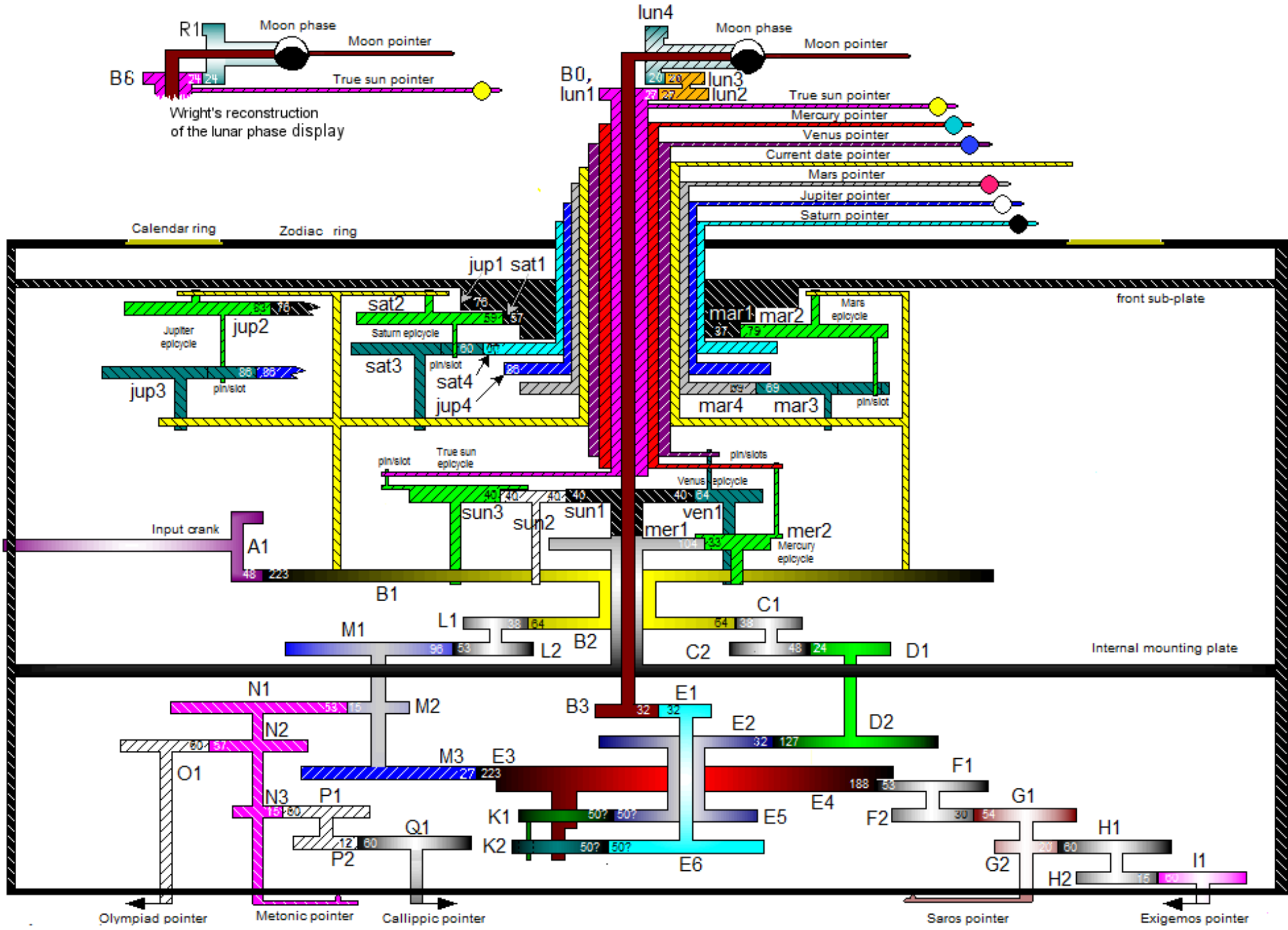
ANTIKYTHERA MECHANISM

wiki page https://en.wikipedia.org/wiki/Antikythera_mechanism
and a video on youtube (below) detail the finding of a mysterious mechanism in an ancient ship wreck on Greek island of Antikythera, as well as the international investigation that found by 3D X-ray scans technique how it functioned.

https://www.youtube.com/watch?v=jSQNEPbQOiI
(see around 17:20, 20:00, 40:00 min.)

The **Antikythera mechanism –** short video of reconstructed device
https://youtu.be/ZrfMFhrgOFc?t=163

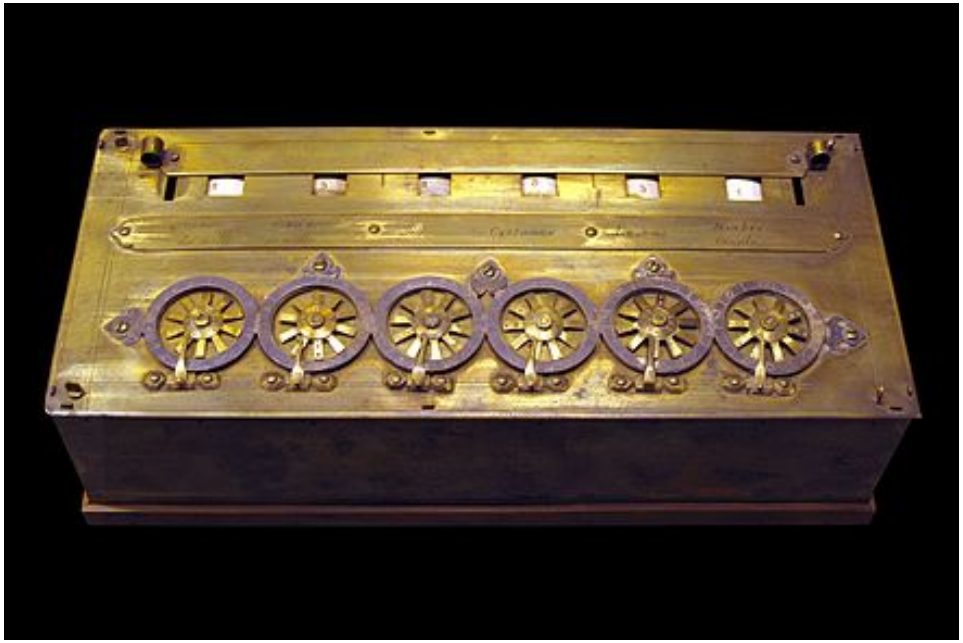
The mechanism is partly digital (whole numbers of teeth on wheels reproduce rational numbers exactly). Eccentric mounting of some parts reproduces in an analog form (analog value can be arbitrary real number, as opposed to integer aor integer ratio) the non-uniform motion of the Moon). This distinction has to do with precision of calculations but is not essential.

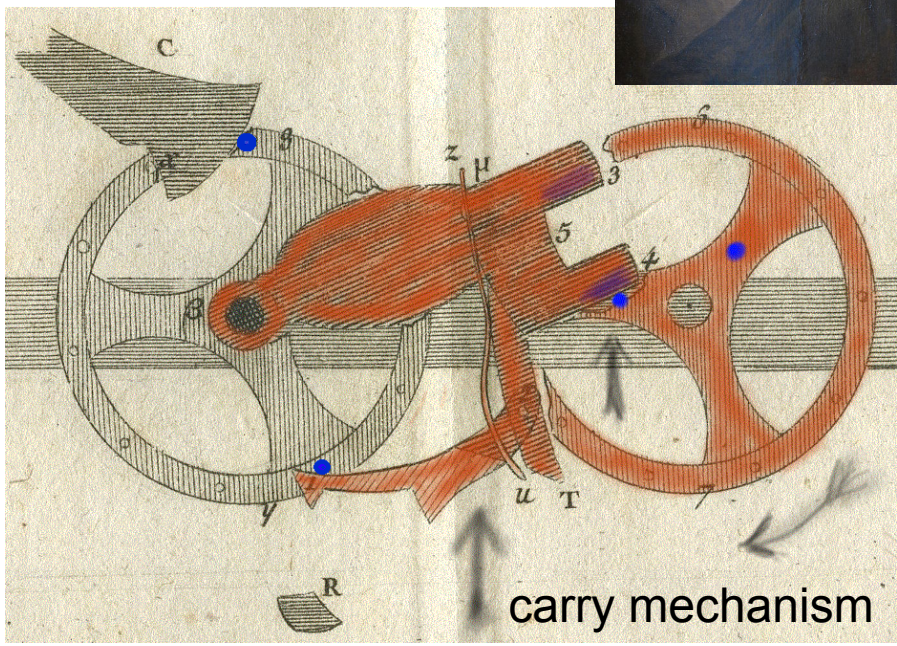

R1
Moon phase
Moon pointer
B6
True sun pointer
Wright's reconstruction
of the lunar phase display
lun4
Moon phase
Moon pointer
B0,
lun1
lun3
lun2
True sun pointer
Mercury pointer
Venus pointer
Current date pointer
Mars pointer
Jupiter pointer
Saturn pointer
Calendar ring
Zodiac ring
jup1 sat1
front sub-plate
sat2
Saturn epicycle
Jupiter
epicycle
jup2
sat3
pin/slot
sat4
jup3
pin/slot
jup4
mar1
mar2
Mars
epicycle
mar4
mar3
pin/slot
pin/slot
True sun
epicycle
pin/slots
Venus epicycle
sun3
sun2
sun1
ven1
mer1
mer2
Mercury
epicycle
Input crank
A1
B1
C1
M1
L1
L2
B2
C2
D1
Internal mounting plate
N1
M2
B3
E1
N2
E2
D2
M3
E3
O1
F1
E4
N3
P1
G1
K1
E5
F2
Q1
H1
K2
E6
G2
I1
H2
P2
Olympiad pointer
Metonic pointer
Callippic pointer
Saros pointer
Exigemos pointer

Willhelm Schickard - astronomer

in 1623 – writes to J. Kepler about his
4-op. calculator using 6 digits.
It is thought that larger number of wheels
was prevented by mechanical friction problem.
Motivation: astronomical table calculation.

Blaise Pascal (1623-1662) - physics, math prodigy.
adders/subtractors (calculators doing + and - operations)
Motivation: to mechanize tax collector's job.

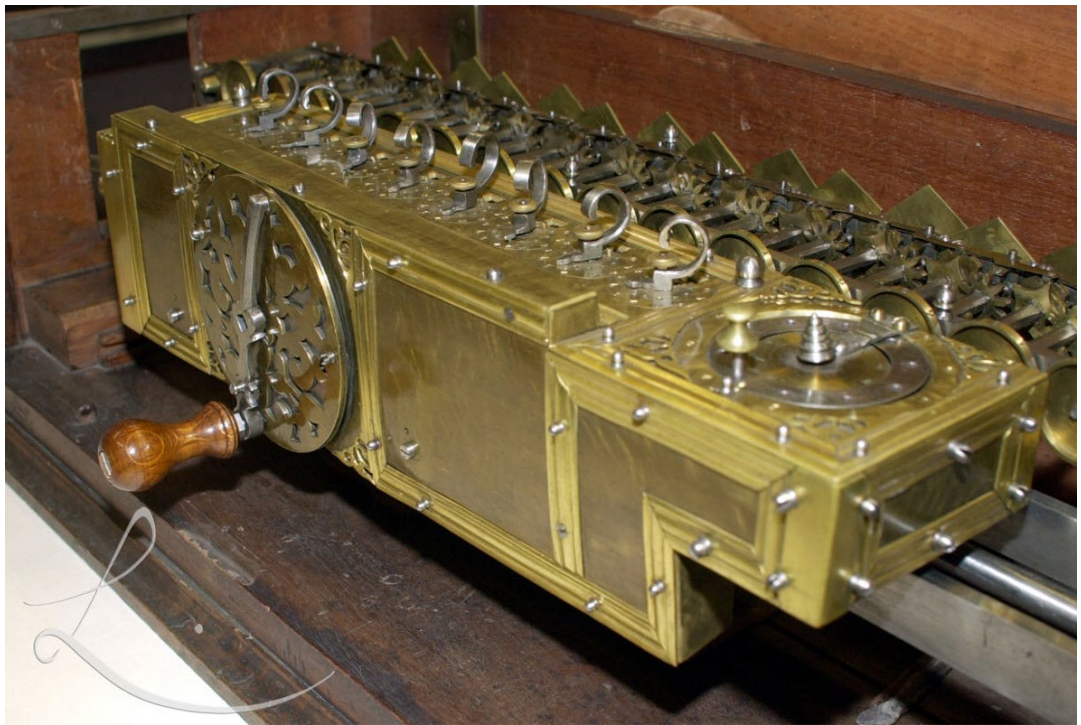




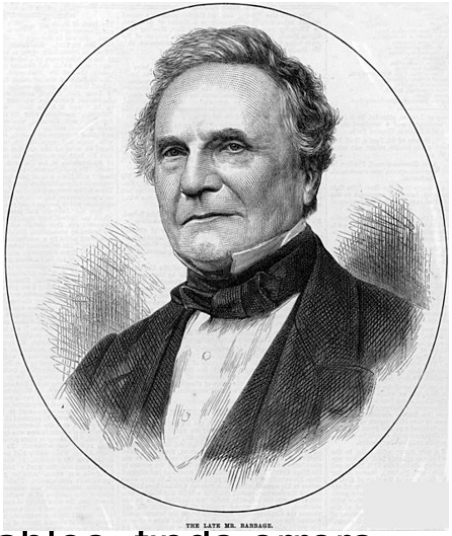
Gottfried Willhelm Leibnitz (1646-1716)

invented calculus independently of Newton & published it first.
For example, *dy/dt* is Leibnitz' not Newton's notation!
One of his results:
$\pi = 4\ (1 - 1/3 + 1/5 - 1/7 + \ldots)$

Leibnitz developed the properties and use of the binary numbers (base-2 repres.)
In 1675 Leibnitz built a digital mechanical calculator called "stepped reckoner", the first 4-op. calculator (+ - * /)



Charles Babbage (1791- 1871)



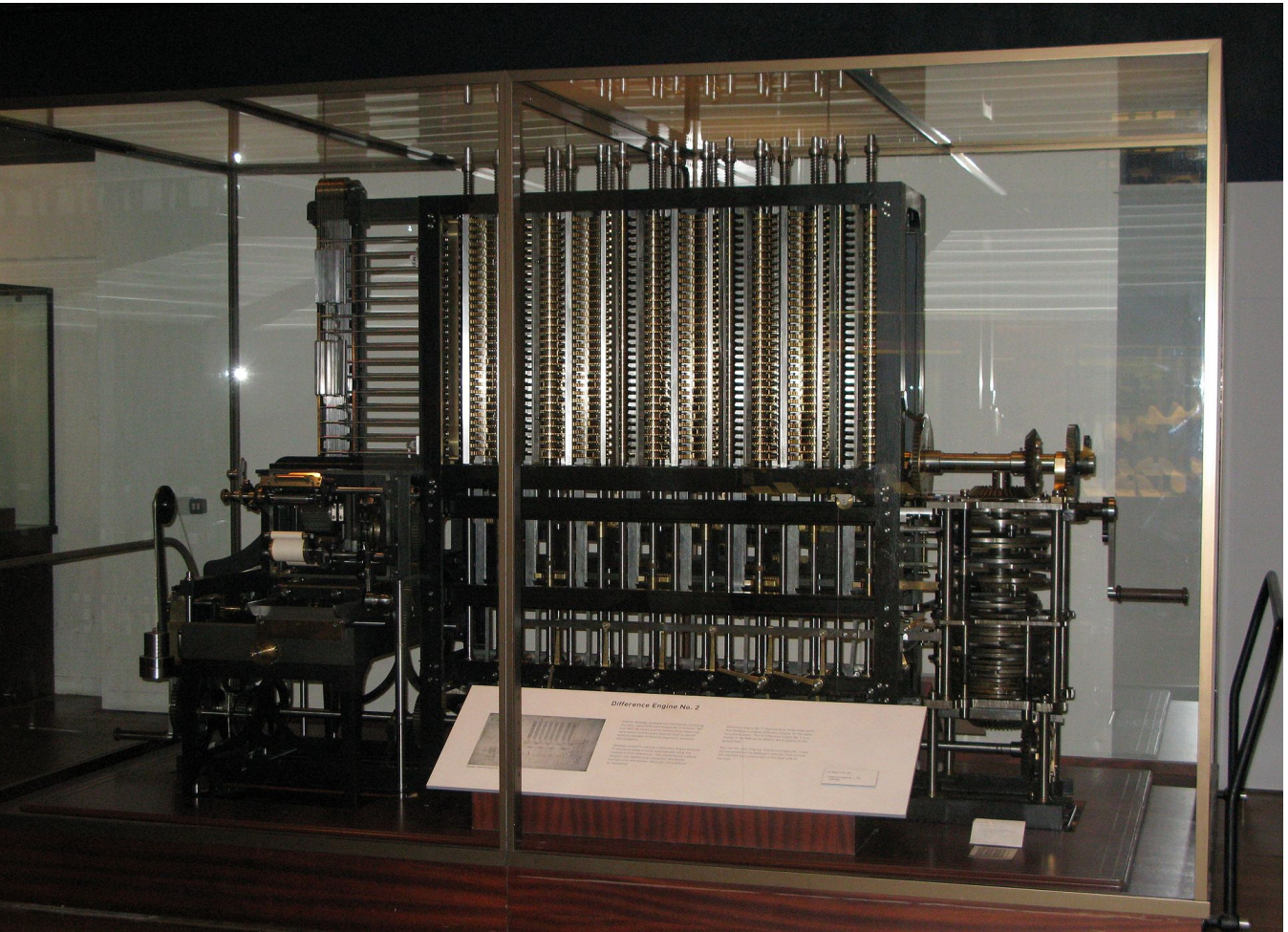
- co-founded the Royal Astronomical Society
- 1812 - with Herschel recalculates The Nautical Almanac tables, finds errors.
- becomes Lucasian Prof. of Mathematics at Cambridge, but not good at teaching
- did operational research (division of labor etc.), cryptography, mechanical patents
- inspired by French gov. project where mathematicians coordinated the effort of 80 human computers trained for specific sub-tasks
- becomes convinced that a machine computation would be better than human

- Motivation: calculating mathematical and astronomical tables
- in 1823 obtained a grant to do mechanical calculation of tables, i.e. for the mechanical *computer* [until then and often for another 100 years, computer meant a person doing calculation]
- called the project "Differential Engine"
- method: finite differences to produce tables of polynomials of order up to 6 or 7

Charles Babbage (cont.)

- 1824 – medal of this Society for invention of “an engine to calculate mathematical and astronomical tables”,
- https://www.youtube.com/watch?v=be1EM3gQkAY (2012 video on reconstruction)
- project partially successful, but unfinished.

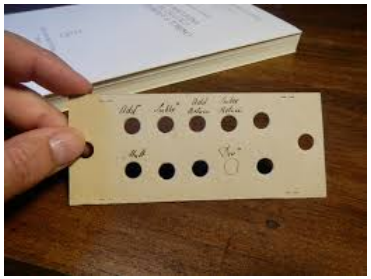
8000 parts,
5 tons, hand-cranked



Difference Engine no.2
(2008 reconstruction
from unrealized plans)

Charles Babbage (cont.)

- proposed a more ambitious project: machine-driven Universal Calculator
- 1846 – gives up the idea of finishing difference engine no. 1 and produces drawings for Difference Engine no. 2, never built in his time
- 1847-1849 obtains gov. funds for Analytical Engine, a general purpose calculator driven by steam engine, that would be programmable, using punched cards
- never builds a working prototype
- project criticized by Astronomer Royal G. Airy, funds withdrawn



Swedish engineer **Georg Scheutz** (1785-1873) built his own difference engine with his son **Edvard** in 1843, following Babbage's ideas.

- began selling them after initially predicting practically no market for computers. (e.g., one machine in 1857 bought by Dudley Obs., Albany, NY)
- the father and son died bankrupt

Ada Lovelace (1815-1852) was introduced to computing by Babbage, in 1843 published translation of a book with own notes on **algorithms** for Analytical Engine. Based on that some consider her the 1st programmer, which is a bit unfounded, but certainly she has first written on applications of computers outside mathematics.

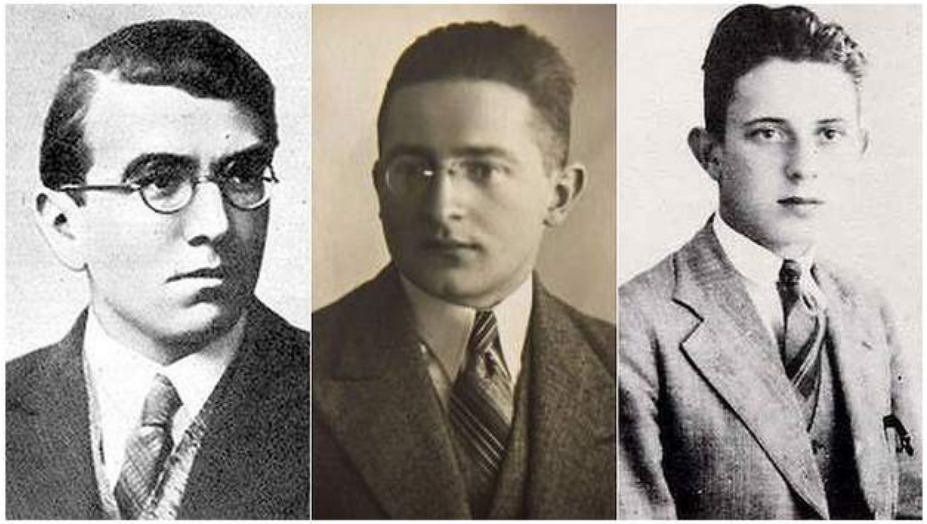
Many groups and individuals undertook the planning and construction of specialized and general, mechanical, electromechanical and electronic, programmable computers in 1930s and 1940s.

The necessity to automate computation of possible combinations of keys in secret cryptographic codes while spying on the potential enemy came up in the years preceding WWII.

https://en.wikipedia.org/wiki/Bomba_(cryptography)

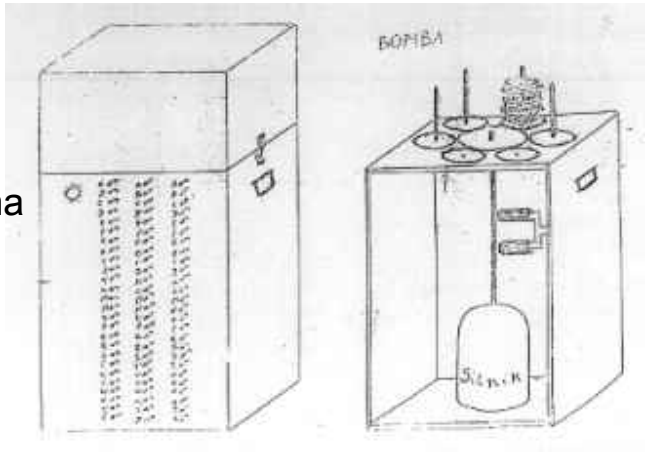
- Unique group-theory breakthroughs and a special-purpose electromechanical computer called Bomba were created by Polish Cipher Bureau mathematicians **Henryk Zygalski, Marian Rejewski,** and **Jerzy Różycki** in 1938.
- Purpose: breaking the changing codes on ENIGMA coding machines of the German military. Their work started already in 1932 but was kept top secret even to the allies France and England.

Enigma
ca. 1938



Ideal location for intercepting German radio traffic was Poznań

Bomba kryptologiczna



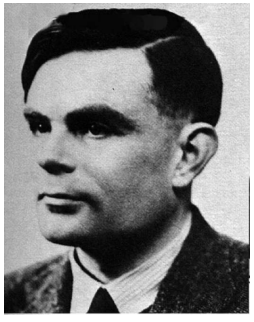
The leader of the effort was Rejewski. Having correctly deduced the internal wiring of the Enigma machine, thanks in part to the help of French military cryptologist Gustave Bertrand, who passed manuals for ENIGMA sold to him by a German spy. Rejewski constructed the so-called cyclometer, consisting of two Enigma copies connected together, with the position of the last coding wheel offset by three position, which allowed a streamlined brute-force search for the combinations of all 4 coding wheels (increased from 3 in the commercial version of the machine that was easier to break)

- 5 weeks before the inevitable war, Polish cryptologists presented the surprised French and British counterintelligence with copies of Enigma machines and methods to decipher the messages.

- Historians agree that breaking of Enigma code shortened the war by 2 years +-0.5
- The decryption of ENIGMA became crucial for the fate of England during the WW II

- 5 weeks before the then inevitable war, Polish military presented the surprised French and British counterintelligence with copies of Enigma machines and methods to decipher the messages.
- The intense hunt for the cryptologists by secret state police of Adolf Hitler called Gestapo ensued once Germans overran Poland, and later also France to which the group escaped via Rumania, but Rejewski & Zygalski survived.

- Decryption of more complicated ENIGMA coding continued at Bletchley Park group led by **Alan Turing**, using more powerful electromechanical computer ("Bombe") built there. The progress was quantitative more than qualitative.



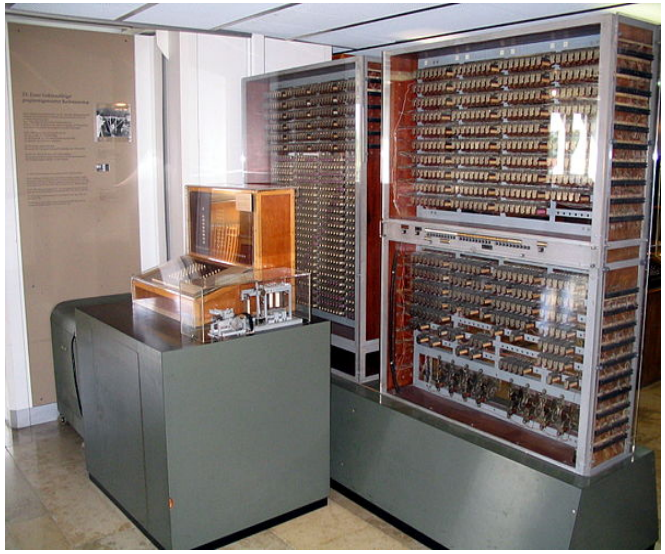
A large number of Enigma-copies were built by Turing's team because the Germans occasionally upgraded their Enigmas to introduce complicated & secure patterns of coding.

Bletchley Park, site of Government Code and Cypher School

In the flying fortress B-29 (first pressurized, high-altitude strategic bomber of WWII period, built in 1944)
5 General Electric **analog computers** allowed coordinated, remotely steered firing of defensive machine guns in electrically powered turrets. Corrections for lead and aircraft speed, altitude, humidity were taken into account, and one gunner could steer up to 3 turrets simultaneously. Analog computers were working pretty efficiently!

The first successful attempts to build a computer were sometimes classified because of war effort and national security, therefore it is difficult to clearly say “who was first”. However, the two early projects below are today considered most important.

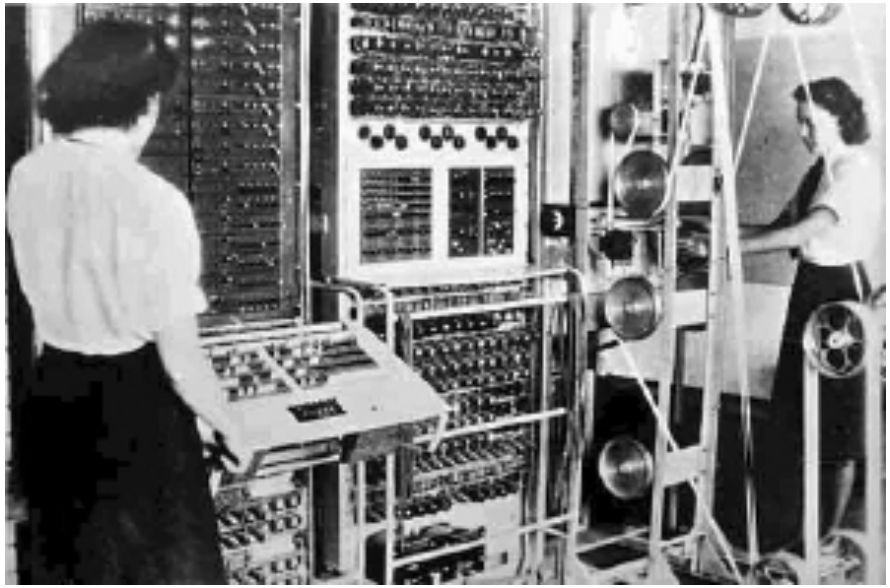
- **Konrad Zuse** (1910-1995) single-handedly built Z3 electromechanical programmable computer in 1941
- Z3 contained 2600 relays operated on 22-bit words
- clock frequency 4-5 Hz. Code stored on punched film
- no conditional branching (no “if”)
- applied to build fully-electronic version, turned down due to war.



- prof. of Iowa State U. **John V. Atanasoff** and
- gradstudent **C. Berry**
- built in 1942 ‘ABC’ for solving systems of linear equations.
- It was non-programmable, but
- fully electronic (~300 vacuum tubes)
- weight 320 kg
- used binary arithmetic, 50-bit fixed-point numbers
- had drum memory based on attached capacitors
- speed of computations ~30 FLOPS (floating point operations per second)

First programmable electronic computer for decryption of messages of Lorenz SZ-40 machine called **Colossus** was built by a team of British engineers working with GC&CS (Government Code and Cypher School, Bletchley Park near London) at Post Office Research Station. Delivered in January 1944, it decoded first message on 5 Feb 1944. Colossus had 1800 vacuum tubes. The information about Colossus was classified & became partially public first in 1970s. It was designed by **Tommy Flowers.** About a hundred machines were built. To keep the new technology secret, Churchill ordered all of them destroyed after war.
https://www.cryptomuseum.com/crypto/colossus/index.htm
The special importance of Lorentz SZ-40 derived form it being used for telex (teleprinter) messages from High Commando of Hitler's army, i.e. for strategic information.

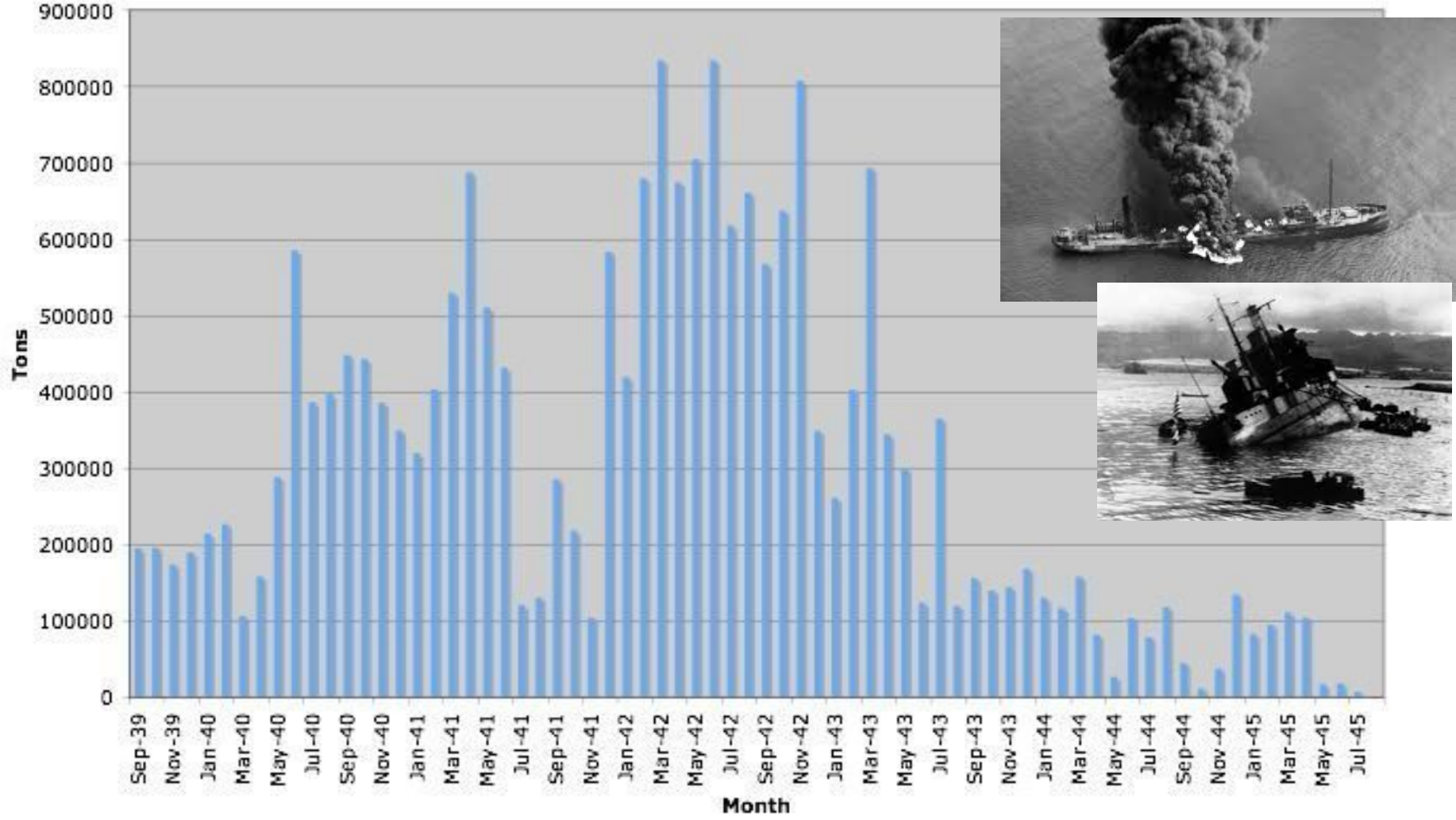
Early version of Colossus



Lorentz SZ-40

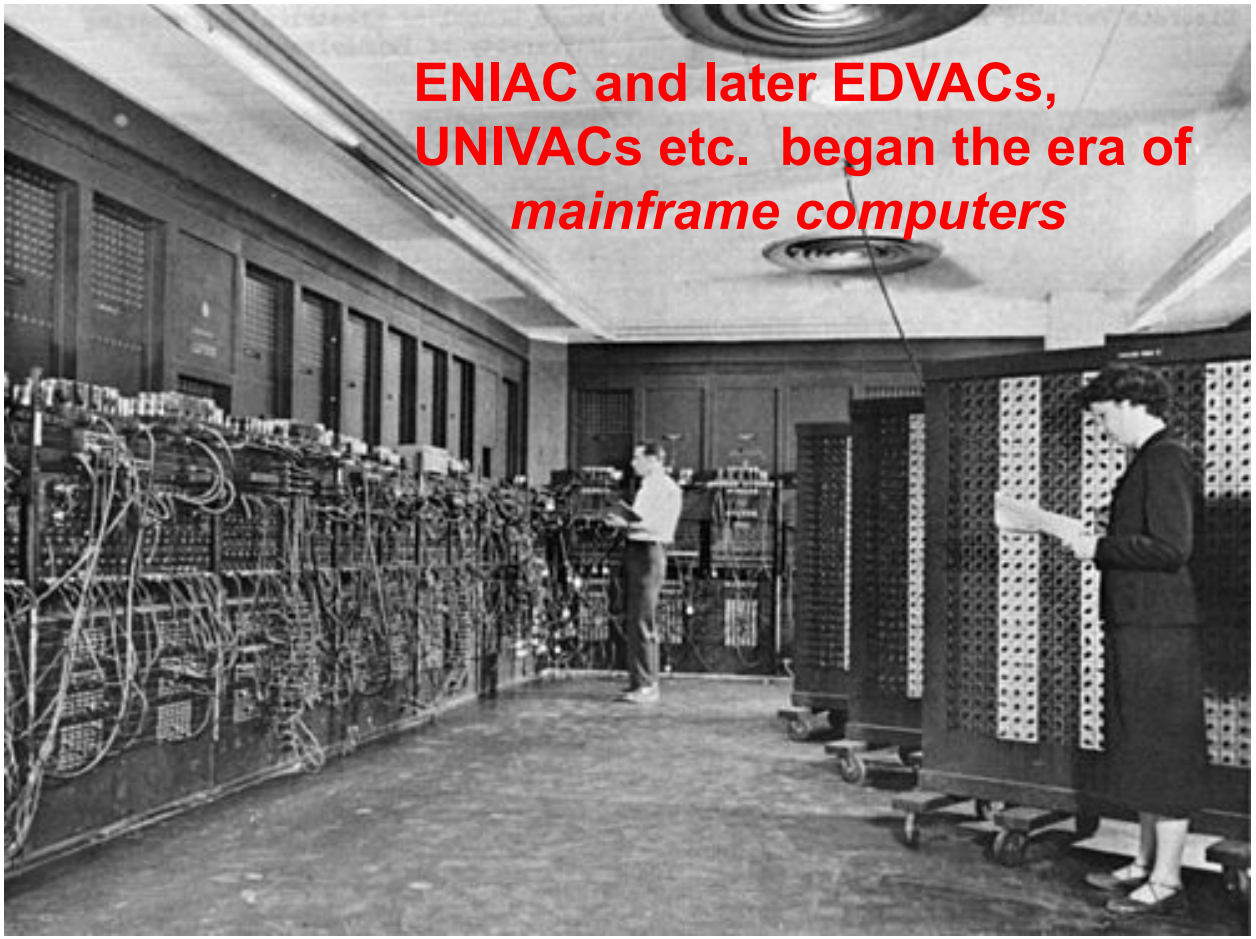
Colossus had an advanced reader + correlator of two punched paper tapes, running 32 ft/s = 5000 characters per sec. Its arithmetic speed was ~100 times larger(!).

Tonnage of British, Allied and Neutral Merchant Shipping sunk by enemy action



Before the information spread about ABComputer, Z3, 'Bombe' and Colossus of the British intelligence, it was often mistakenly stated that the first electronic computer was the American **ENIAC = Electronic Numerical Integrator and Computer**, built in Dec. 1946. It was used by US Army for ballistic calculations, solving differential equations of projectiles moving in air, and in secrecy to evaluate designs of thermonuclear bombs.

Chief designers: **John Mauchley, J. Presper Eckert,** of Moore School of EE, Univ. of Pennsylvania.



ENIAC was modular, had bus architecture, data buffers, and programs with conditional branching.

20k vacuum tubes
7k crystal diodes
10k capacitors
1.5k relays.
Weight 30 t, < 300 FLOPS
it operated on decimal numbers (not binary) & consumed 150 kW power

The first commercially successful mainframe (or supercomputer as we would say today) was **UNIVAC (I)**, produced by Eckert-Maulchey Computer Corporation around 1950 (later Remington Rand Co.).

- it was used by U.S. Census Bureau in 1951
- for CBC TV station it predicted the surprising landslide win by pres. Eisenhauer in 1952 election based on a sample of 1% voters; CBS pretended at first that Univac broke down, because the prediction sounded impossible.
- had less accuracy in numeric representation than liked by scientists
- was directed toward wealthy business companies
- migration to magnetic tape from cards was tough at first

- IBM (International Business Machines Co.) at that time produced mostly the punched-card calculators and office equipment, not computers; that of course was to change later.

UNIVAC operator's console:

1950s to 1960s

At the end of 1950s and beginning of 1960s many advances gradually appeared:

- magnetic tape as mass storage of von Neumann architecture:
- separate operating and mass storage, processor, data and program in the same memory
- magnetic drums playing the role of hard disks, developed later by IBM
- binary digits won with decimal digits
- from among different data formats, 8-bit bytes started to emerge as winners
- they simply were handy for character (text) processing
- instead of setting up the computer by hand or from punched paper media, the programming was done from magnetic media, and the code was no longer the low-level machine code (fairly elementary operation commands understood by the processor.
- instead, *compilers* of *high-level programming languages* appeared

- *compiler* = program that checks and translated your program into the set of low-level instructions for processor (still readable by humans, but program in such assembly language was ~20 times longer than the high-level code in language similar to English+math symbols). Finally the so-called linker (part of compiler in modern times) was translating the asembly language to binary executable containing 00011100110101001000101010 11... which looks like a mess on a screen.

1950s to 1960s

- Hardware and software, previously inextricably bound, now became independent:
- The first widely use computer language which was hardware platform-independent was **FORTRAN** (1957) = Formula Translator, created by John Werner Backus, who worked at IBM.

- This language was loved by academia and the first computer science departments (CS = computer science, just trying to emerge and define itself in opposition to EE). Fortran resembled ***meta-code*** and English language. It produced executable programs running as fast as hand-coded ***assembly code****, the program written in terms of processor's instructions (such as shift value from register A to B, add register B to register C, storing result in register C, write register C in memory location stored in register D etc.)*

- Then **COBOL** was created by the U.S. government committee & mandated on all machines it was purchasing. (COBOL = Common Business Oriented Language.) This forced a compatibility/uniformity across different machines. It was good for databases, payrolls, and some airline reservation systems. It was also supposed to encourage long variable names for better clarity of programs (to maybe replace comments) but it failed at that, which was realized when the software engineers looked at some ancient codes, fearing the so-called Year 2000 Problem (Y2K). Very few COBOL programs are in use. The language was not a general programming language, and was never popular in academia.

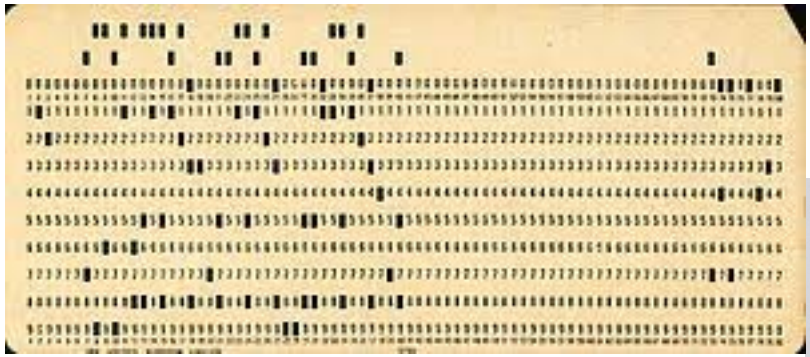
Looking ahead to present times... & making the long story short

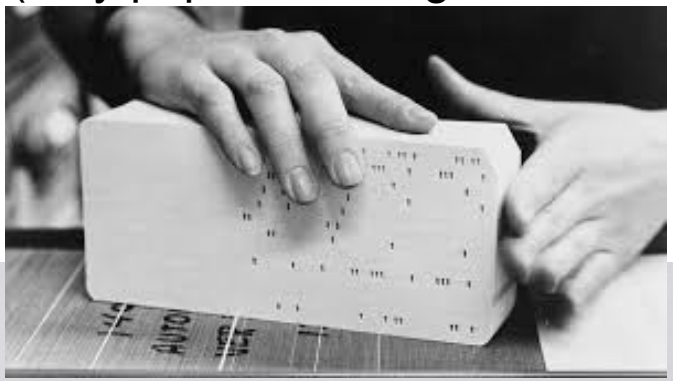
While COBOL is now as good as dead, Fortran (fortran 90, 95, 2003, 2015 versions) thrives in academia, though is now almost unknown in businesses and Computer Science departments, which have switched to **C, C++, Java** and now **Python**. In 1980s CS gurus worked hard at giving Fortran a bad name while promoting new languages trying to make programming easier and more reliable. CS was quickly creating them and then dropping. Even today, scores of programming languages are created every year. There are thousands of them, some useful for certain purposes and some just very amusing. The programming paradigms such as *object-oriented* and more recent *functional programming* do not aim at producing fast code, as a priority. Therefore most are not crunching numbers well (C++ can do it despite, not thanks to being OO).

- Python is a scripting language, *an interpreter*, and on its own will always produce a relatively slow-running code (i.e. slow executables, or run-time codes). But writing and debugging a program in Python is often simpler than in the other, more powerful, languages. One reason is the huge user base, allowing inexperienced programmers to understand & fix errors by using search engines.
- Pre-packaged modules for Python tap into the strength of more powerful languages
- Guess in what language the crucial modules *Matplotlib* and *Numpy* of now-dominant Python are written? Fortran, the language explicitly created to perform numerical calculations, sometimes called *number crunching*.
- The only language other than Fortran, starting from the same goal of top-speed, large-scale scientific computation (including parallelism), is **Julia** created at MIT. However it is not widespread yet, or demonstrably superior to Fortran or C(++).

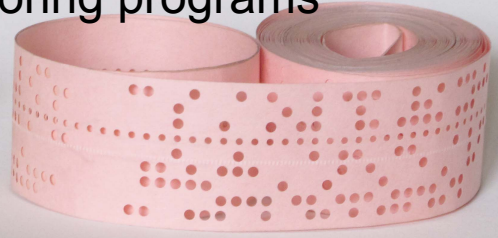
Lest go back to mid-1960s and 1970s

- IBM grew. It transferred from decks of punched cards (very popular among businesses





and perforated tape storing programs



- to magnetic media: tape, drums, disks

1950s to 1960s

At the end of 1950s and beginning of 1960s many advances gradually appeared:

- magnetic tape as mass storage of von Neumann architecture:
- separate operating and mass storage, processor, data and program in the same memory
- magnetic drums playing the role of hard disks, developed later by IBM
- binary digits won with decimal digits
- from among different data formats, 8-bit bytes started to emerge as winners
- they simply were handy for character (text) processing
- instead of setting up the computer by hand or from punched paper media, the programming was done from magnetic media, and the code was no longer the low-level machine code (fairly elementary operation commands understood by the processor.
- instead, *compilers* of *high-level programming languages* appeared

- *compiler* = program that checks and translated your program into the set of low-level instructions for processor (still readable by humans, but program in such assembly language was ~20 times longer than the high-level code in language similar to English+math symbols). Finally the so-called linker (part of compiler in modern times) was translating the assembly language to binary executable containing 00011100110101001000101010 11...

```
# mark_description "Intel(R) Fortran Intel(R) 64 Compiler XE for applications
running on Intel(R) 64, Version 15.0.3.187 Build 2";
        .file "fortran+om-laplace-sp.f90"
(...)
 movsd     %xmm1, data_mp_t1_(%rip)                      #113.5
        movl      1950048+data_mp_grid_(%rip), %r12d     #114.2
        jne       ..B1.18       # Prob 50%               #115.10
                                # LOE rbx r14 r15 r12d r13d xmm1
..B1.13:                        # Preds ..B1.12
        movl      $-1, %esi                              #115.14
        lea       32(%rsp), %rdi                         #115.14
        movq      $0x1208384ff00, %rdx                   #115.14
        movl      $__STRLITPACK_9.0.3, %ecx              #115.14
        xorl      %eax, %eax                             #115.14
        lea       232(%rsp), %r8                         #115.14
        movq      $0, (%rdi)                             #115.14
        movq      $15, 232(%rsp)                         #115.14
        movq      $__STRLITPACK_8, 240(%rsp)             #115.14
        movsd     %xmm1, 288(%rsp)                       #115.14
call      for_write_seq_lis                              #115.14
                                # LOE rbx r14 r15 r12d r13d
        movsd     288(%rsp), %xmm1
#pxor      %xmm0, %xmm0                                  #115.14
        cvtsd2ss  %xmm1, %xmm0                           #115.14
```

Assembler code