INTRODUCTION TO COMPUTERS

9TH CENTURY BAGHDAD

Polymath al-Khwarizmi writes the book 'al-Jabr'

'Algorithm' translit. of al-Khwarizmi

'al-Jabr' via Latin: 'algebra'

$$x^2 + px + q = 0$$

$$\Rightarrow \text{”lägg till 0”}$$

$$x^2 + px + (\frac{p}{2})^2 = (\frac{p}{2})^2 - q$$

$$\Rightarrow \textit{förenkla}$$

$$(x + \frac{p}{2})^2 = (\frac{p}{2})^2 - q$$

$$\Rightarrow \textit{förenkla}$$

$$x + (\frac{p}{2}) = \pm\sqrt{(\frac{p}{2})^2 - q}$$

$$\Rightarrow \textit{balansera}$$

$$x = -(\frac{p}{2}) \pm \sqrt{(\frac{p}{2})^2 - q}$$

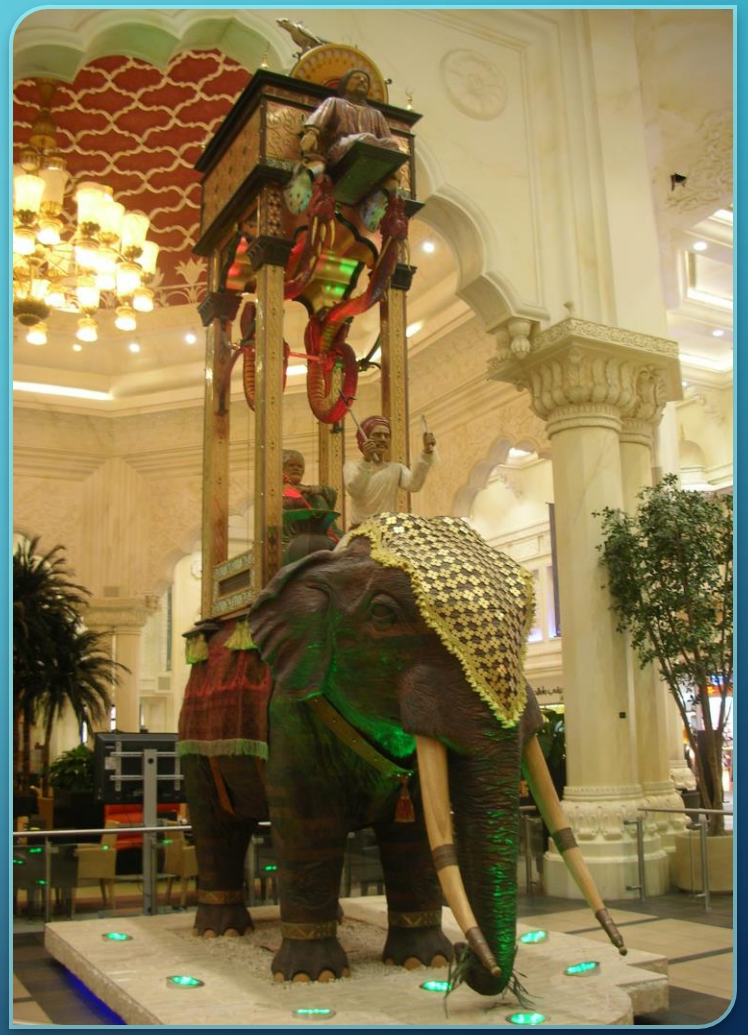
13TH CENTURY BAGHDAD

"Father of robotics": al-Jazari

Synchronized clocks

Programmable music robot band

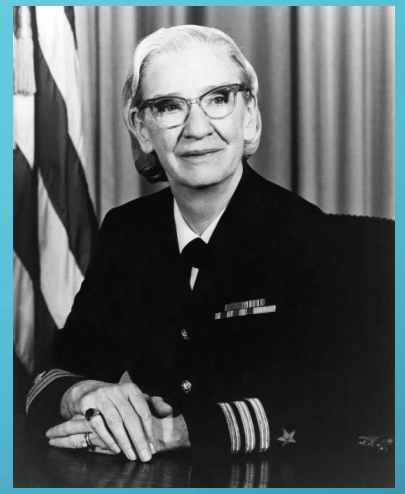
Mechanical algorithms



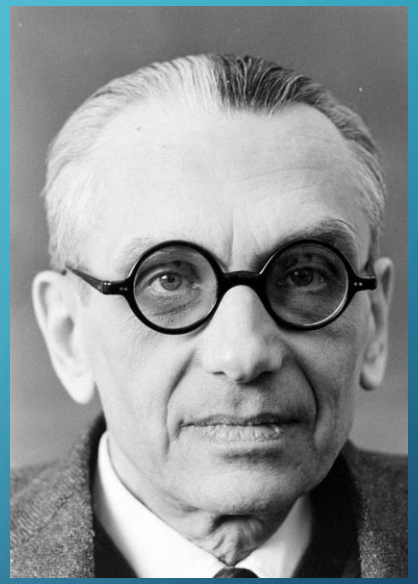
1820-2020 ADA, KURT AND THE GANG

Ada Byron

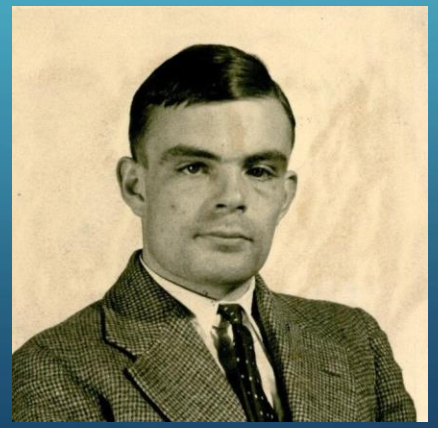
Alonzo Church



Grace Hopper

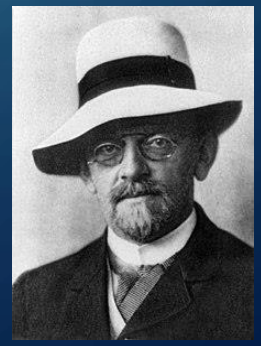


Kurt Gödel



Alan Turing

Robin Milner



David Hilbert

THE RIGHT HONOURABLE AUGUSTA ADA BYRON, COUNTESS OF LOVELACE (1815 –1852)

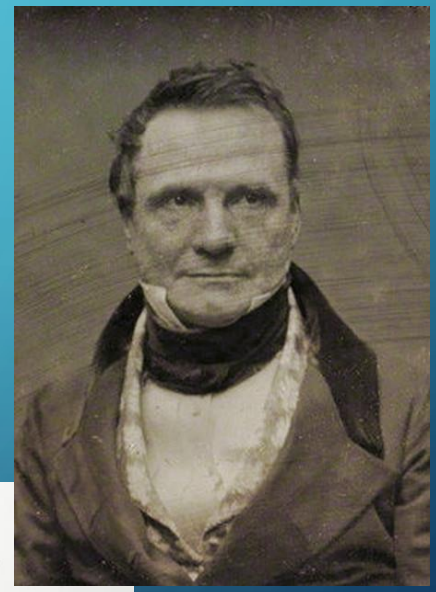
"The science of symbols and operations"
- what we today would call *computation*

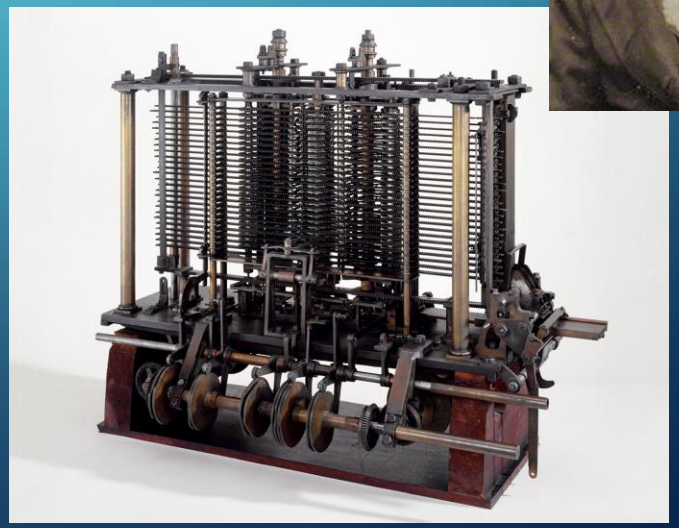
"It may be desirable to explain, that by the word *operation*, we mean *any process which alters the mutual relation of two or more things*, be this relation of what kind it may. This is the most general definition, and would include all subjects in the universe."

Ada was the first to recognize that the machine had applications beyond pure calculation.

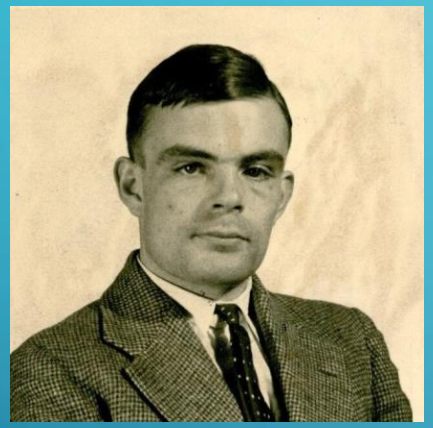
Lady Lovelace's Objection: "a machine [AI] can't originate anything."

Charles Babbage





ALAN TURING (1912-1954)



Perhaps most known for cryptography and the eponymous "Turing Test".

The Turing Machine is probably his most important contribution, leading to the concept of a *register machine*.

In short, a Turing Machine is an abstract and theoretic machine that can *implement all computable functions*.

From S. Wolfram:
"A register machine is an idealized computing machine consisting of a fixed set of data registers and set of instructions for operating on them"

THE CHURCH-TURING THESIS (1936)

- A Turing machine, carries out computations from inputs by manipulating symbols on an infinite tape. It describes what it means for a machine to perform an algorithm. This is basically Ada's vision of computers. Turing had read her "Notes" and coined the term “Lady Lovelace's Objection” in his 1950 Turing Test paper.
- Church created the λ-calculus, a pure language of operations. Using a clever representation of symbols as (constant) operators, this describes what it means for an algorithm to be computable.
- Gödel, formalized the definition of the class of general recursive functions, these describe mathematically what computable means in mathematical logics.

Church and Turing proved that all these mean the same thing – a property now known as "Turing completeness" and the result is known as the Church-Turing Thesis. It proves that machines *can* implement all computable functions – however, much to Hilbert's chagrin this also means that they are fundamentally *incomplete*. That is a consequence of Gödel's Incompleteness Theorems. They rather famously proved that Hilbert's program was impossible.

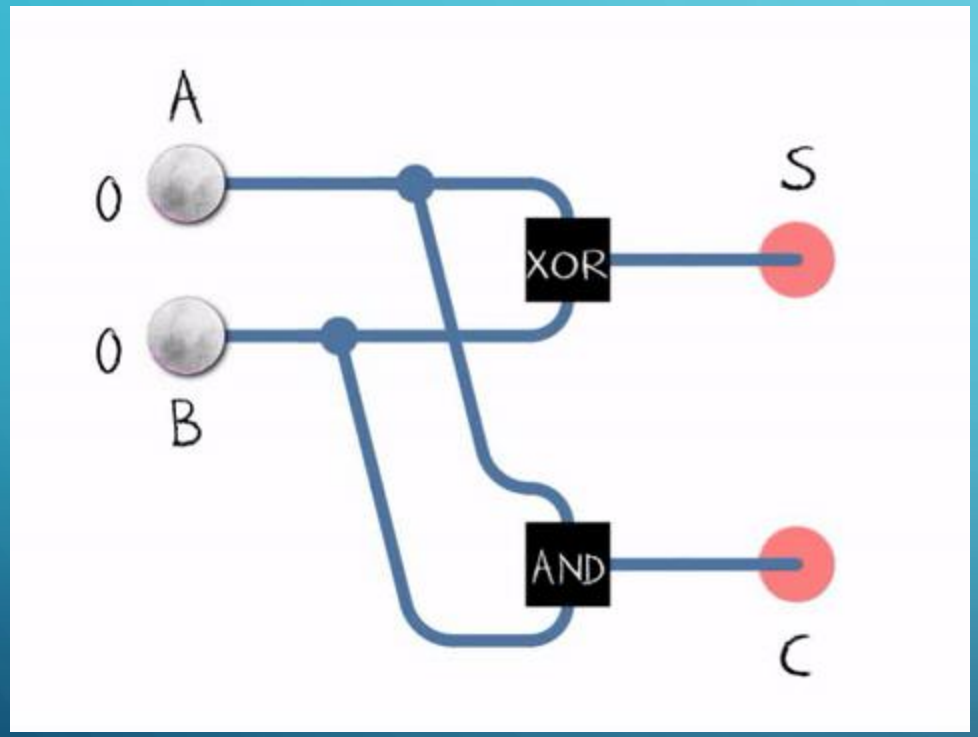
However, since these machines, calculi and logics are incomplete, that means they all have loops/recursion! This is a must for any useful programming language. In other words, if Hilbert's program would be possible, general computing would not be possible. That's kind of strange!

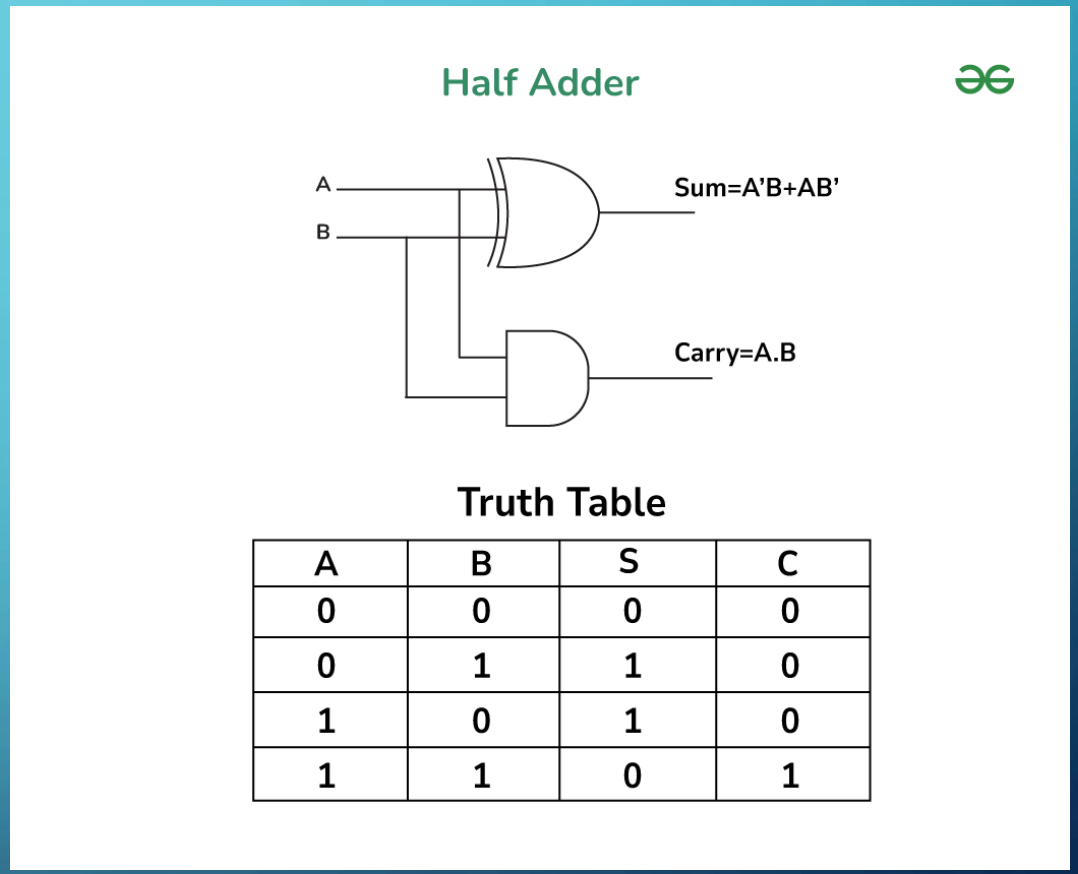
REAL COMPUTERS: STACK, HEAP, REGISTER MACHINES

- A *register* is an integrated circuit consisting of multiple *flip-flops*, each carrying 1 bit. In other words, a register contains a binary number. A 64-bit processor typically has 64-bit wide registers, although it can usually split them into several smaller registers when needed.
- The *heap* is an unordered region of memory. Memory is essentially many huge registers that have addresses. They are much slower and cheaper than those in the processor.
- The *stack*, which is a Last-In-First-Out data structure in memory that is associated with each running program. Special handling makes this much faster than heap memory.
- An *instruction queue* is a special circuit that contains the next instructions to perform and is updated as the program runs. However, size is limited, so there's also a register-based cache to speed up instruction/program loading.
- Programs are read from memory and the instructions are placed in the queue.
- Data can be loaded from the heap onto the stack or directly into registers.
- Every time a function or procedure is called in a program, a new *stack frame* is allocated.
- When a procedure is complete, a stack pointer (a memory address) is updated to point to the previous stack frame.
- Values on the stack are loaded into registers before the processor can use them. Exactly how that is handled is known as a *calling convention*.

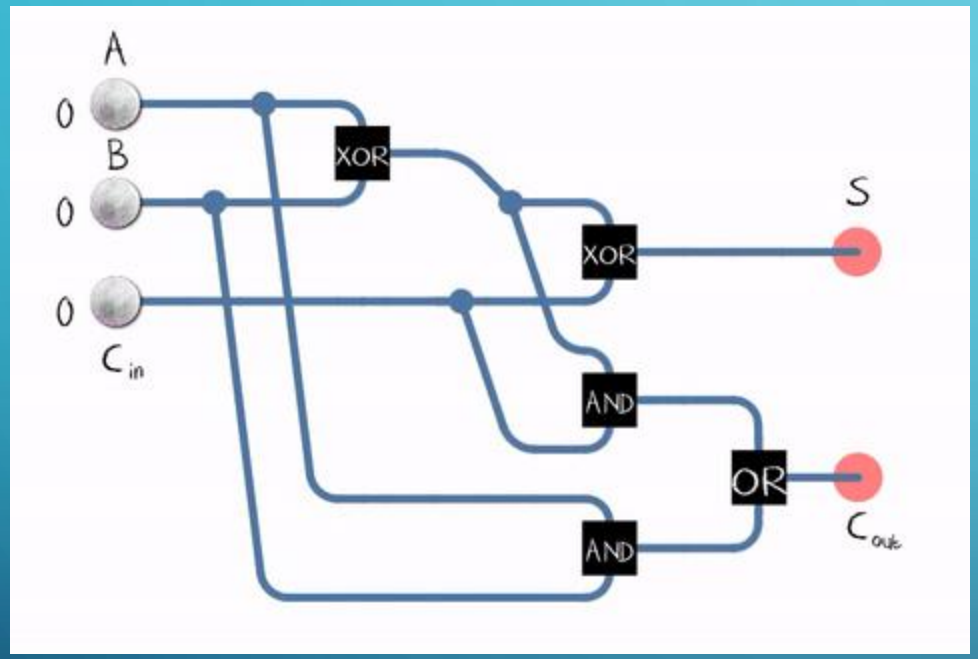
Such a machine can compute all functions that fit in memory. Despite this theoretical limitation, we still call such a machine Turing Complete.

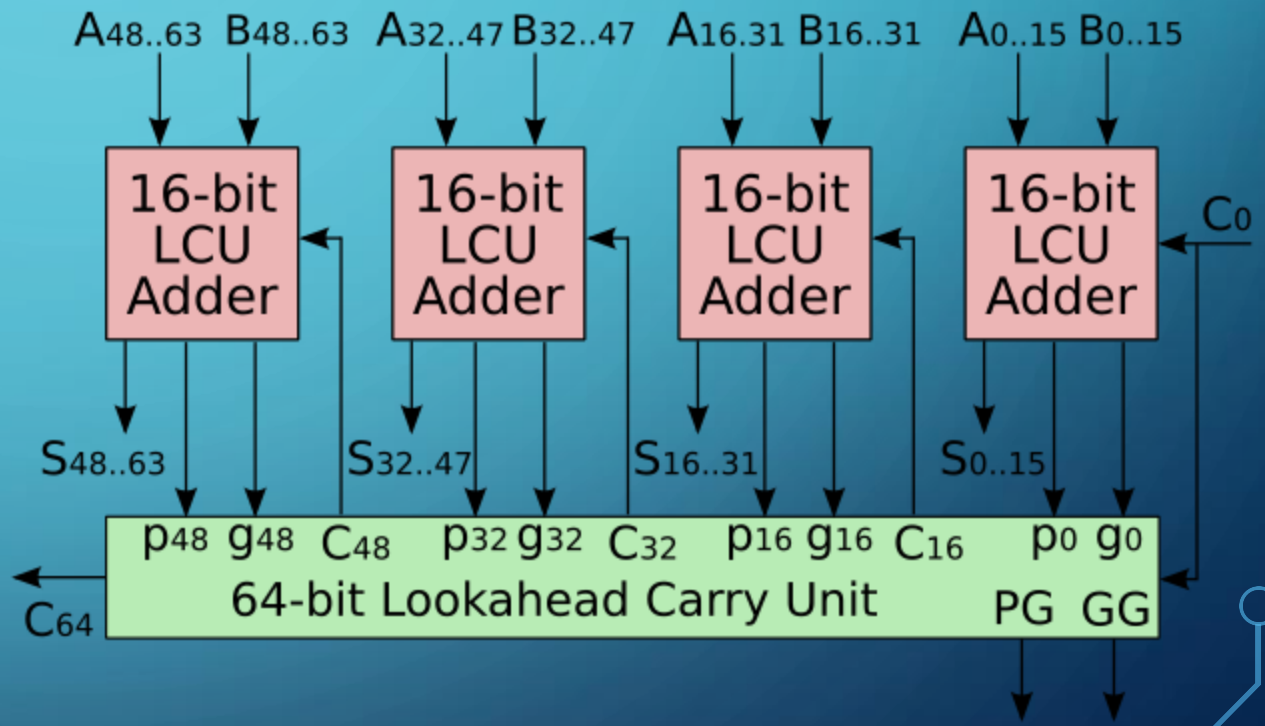
1-BIT HALF ADDER



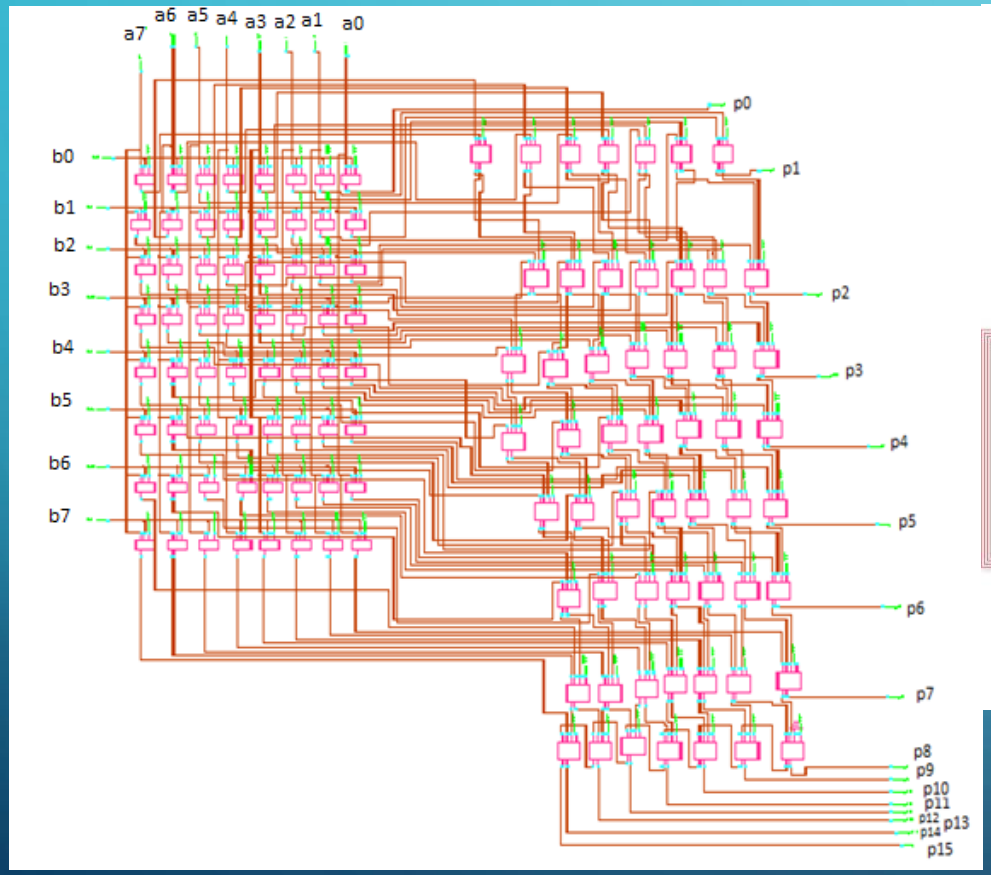


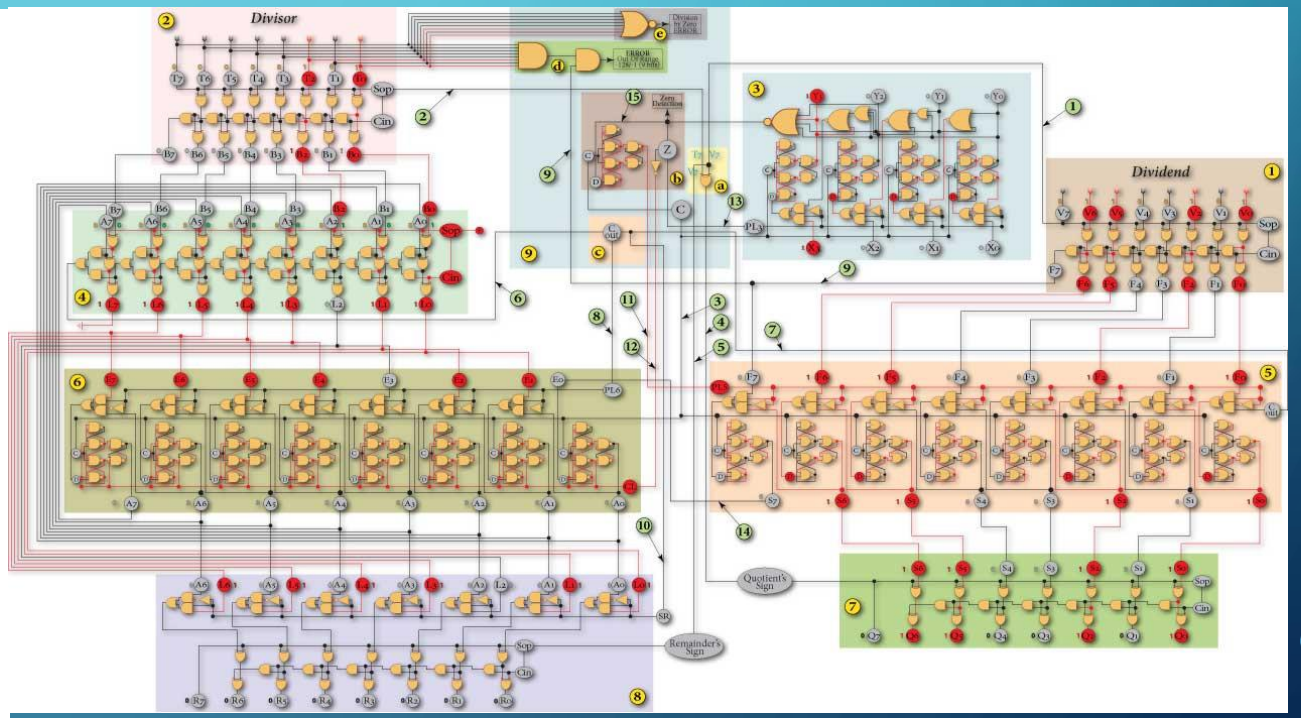
FULL ADDER, CARRY ADDER



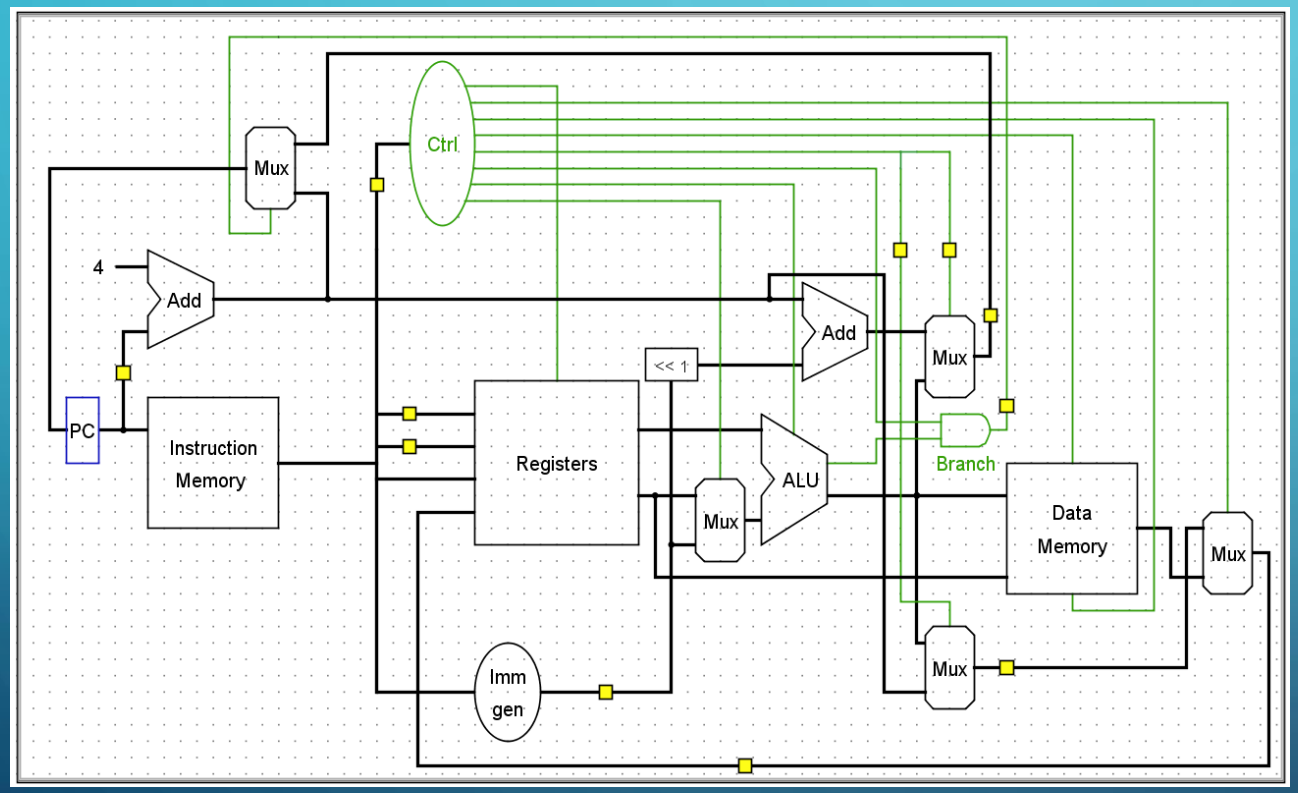


8-BIT MULTIPLICATION, DIVISION

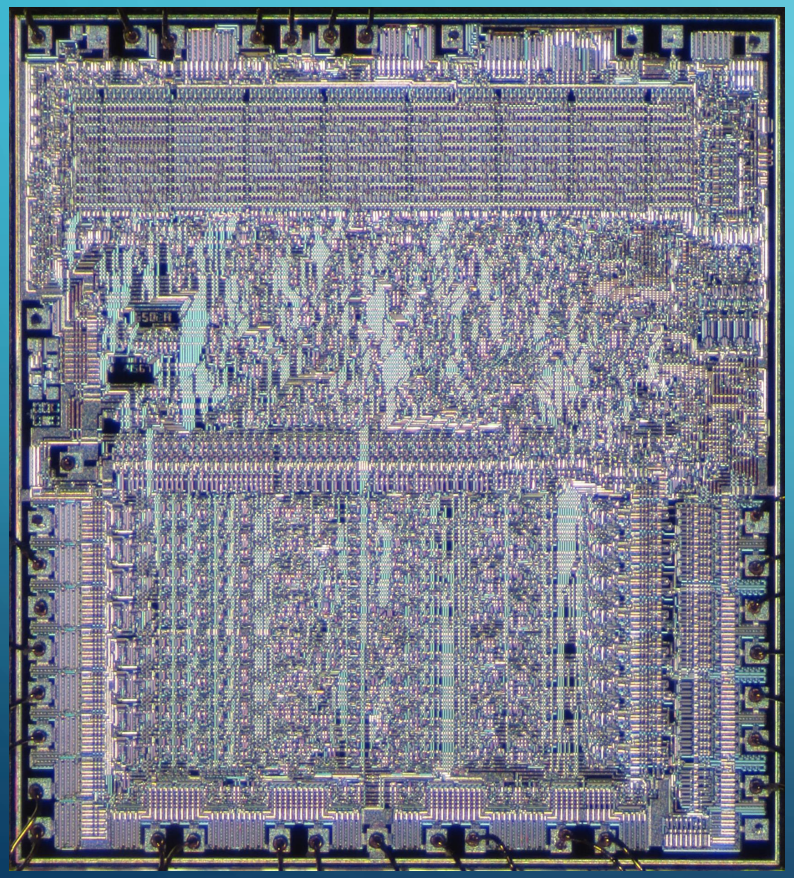




JUMP (LOOPS, RECURSION)



MOS TECHNOLOGY 6502 (1975)



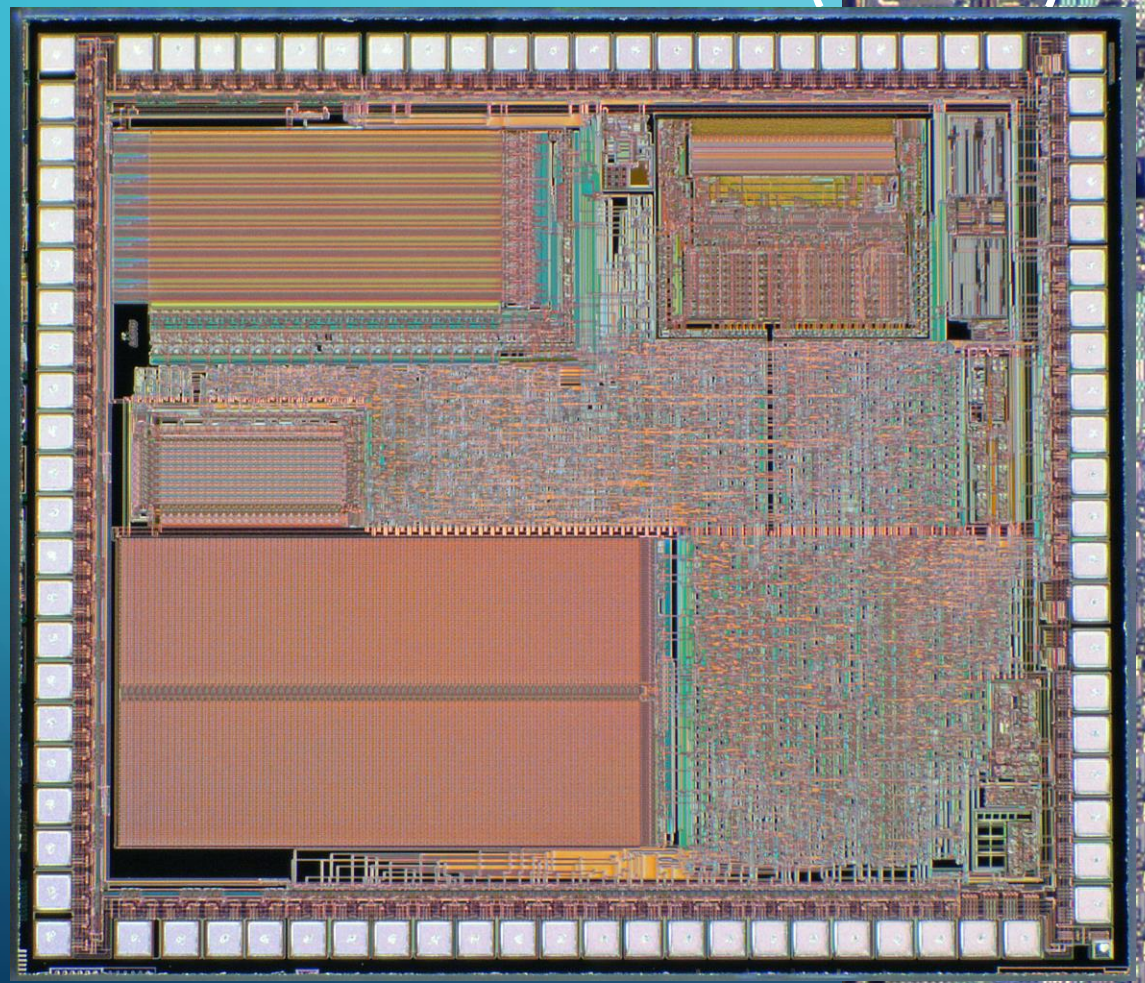
~ 4x4 mm

This part is the instruction decoder

The middle seemingly jumbled part is the control logic (eg. jump instructions).

Inside the 'box' is the ALU (Arithmetic Logic Unit). The registers are on the right, data bus on left and bottom.

SITRONIX ST2064B (2007)

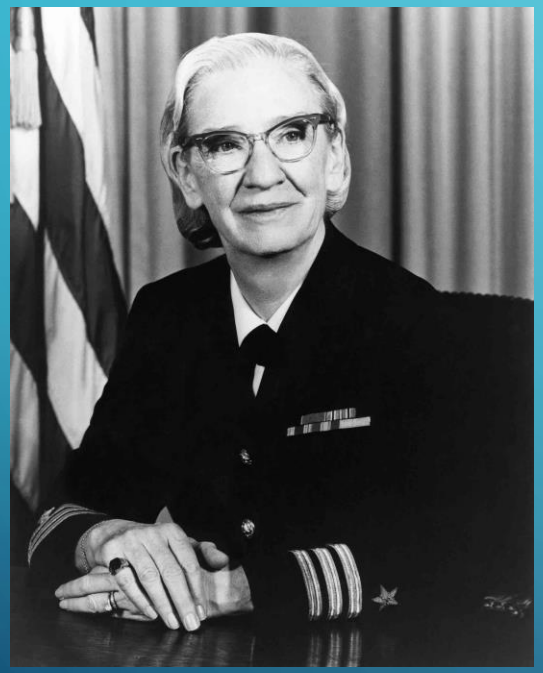


~ 2.5x2.5 mm

Layout no longer done by a human. The jumbled parts are gone.

Last "hand-designed" processor was the DEC Alpha in 1992!

REAR ADMIRAL GRACE HOPPER (1906 – 1992)



COBOL

Coined the term "bug".

Was talking about the "value of data" already in the 80s.

Developed the concept and implementation of machine independent programming languages, that make use of a *compiler*.

Lead to interpreted languages in the 60s (LISP) and 70s (BASIC). Later Java ('95) and Python ('00).

MACHINE INTELLIGENCE

In this course, the perspective is that the history of computers is the history of machine intelligence. "Artificial intelligence" usually refers to specific tasks, however:

- Automation
- Expert systems, Virtual Assistants
- Machine learning / deep learning
- Autonomous systems
- Evolutionary computation
- Alife (*human-like* intelligence is not required here)
- … in popular culture more like synthetic people?