

# COMP 590: Computer Architecture

## Lecture 1 - Introduction

Amogh Akshintala

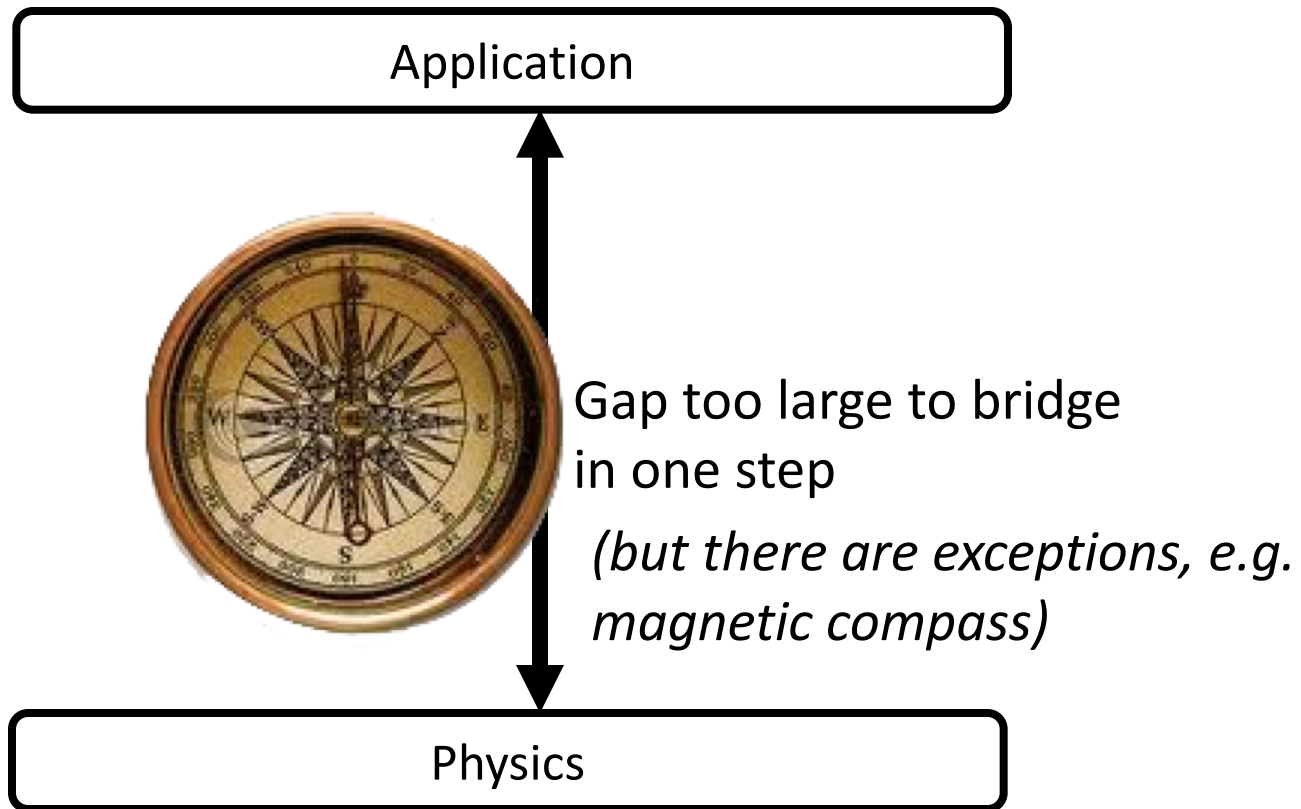
Computer Science

University of North Carolina at Chapel Hill

<https://aakshintala.com/>

<https://aakshintala.com/teaching/comp590-s20>

# What is Computer Architecture?



In its broadest definition, computer architecture is the *design of the abstraction layers* that allow us to implement information processing applications efficiently using available manufacturing technologies.

# Abstraction Layers in Modern Systems



Application

Algorithm

Programming Language

Operating System/Virtual Machines

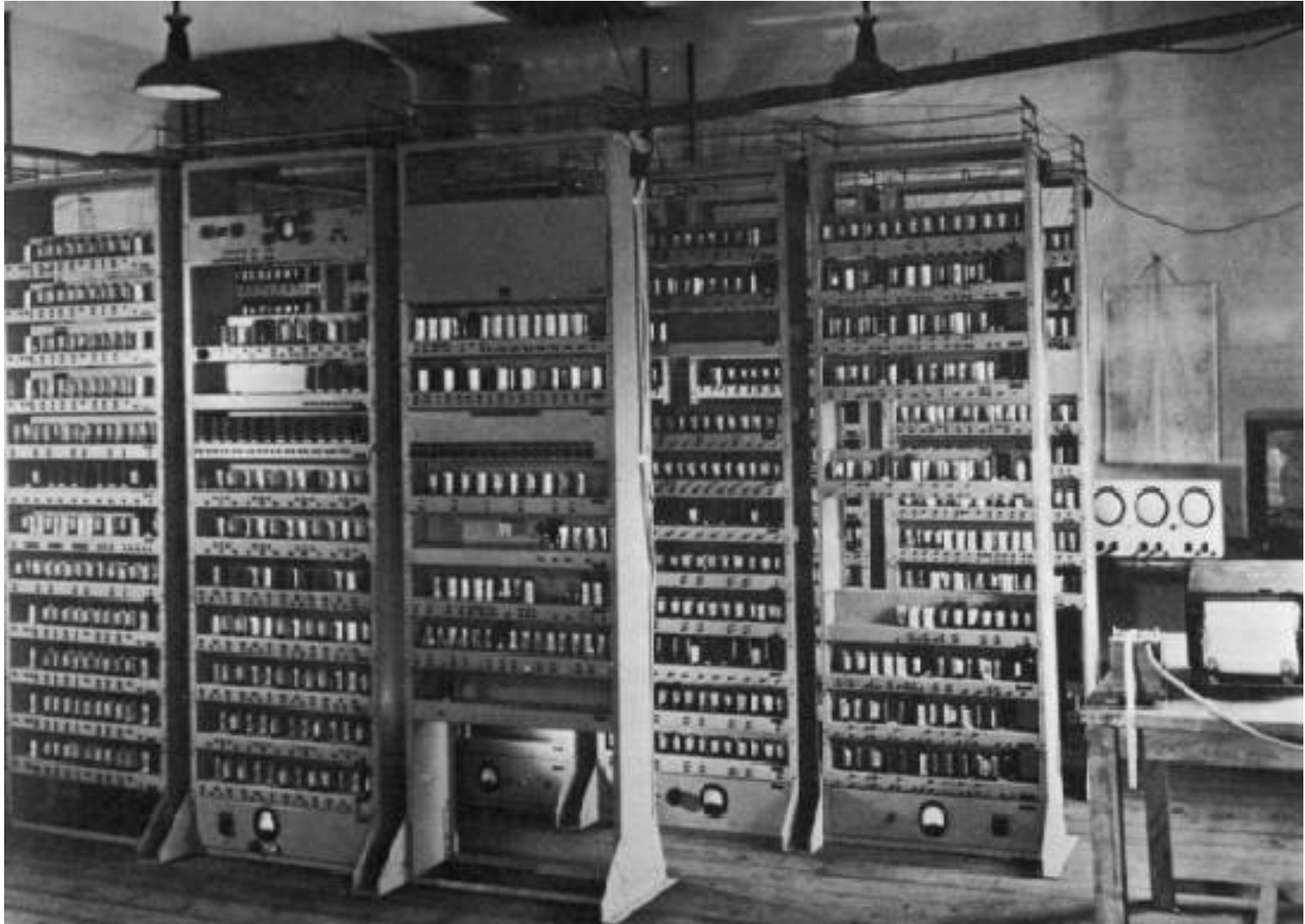
Instruction Set Architecture (ISA)

Microarchitecture

Gates/Circuits

Physics

# Computing Devices Then...



**EDSAC, University of Cambridge, UK, 1949**



# Computing Devices Now



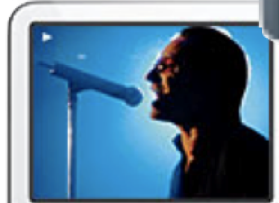
**Cameras**



**Set-top boxes**



**Games**



**Media Players**



**Laptops**



**Servers**



**Robots**



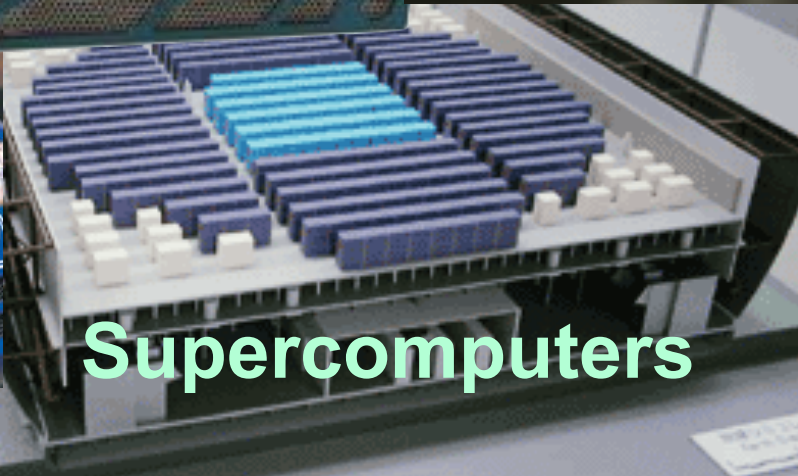
**Smart phones**



**Automobiles**



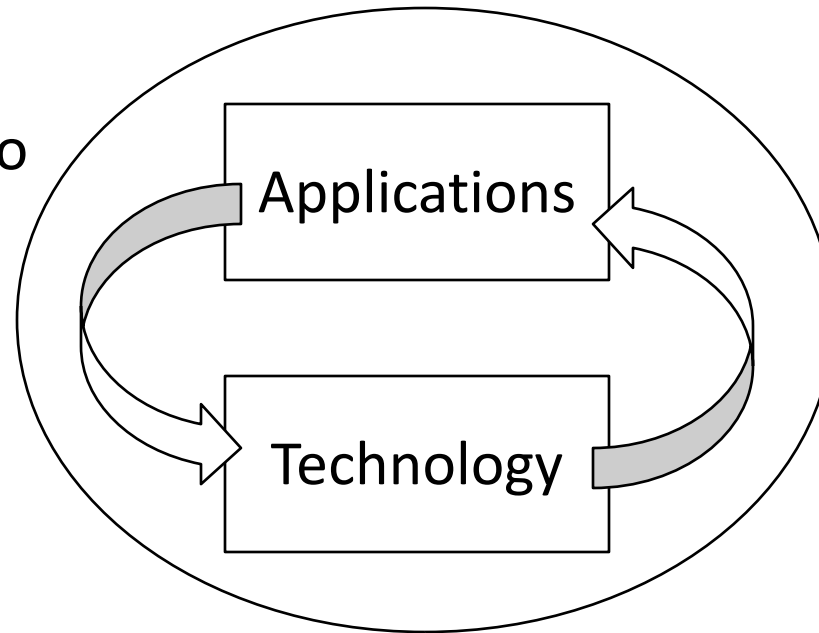
**Routers**



**Supercomputers**

# Architecture continually changing

Applications suggest how to improve technology, provide revenue to fund development



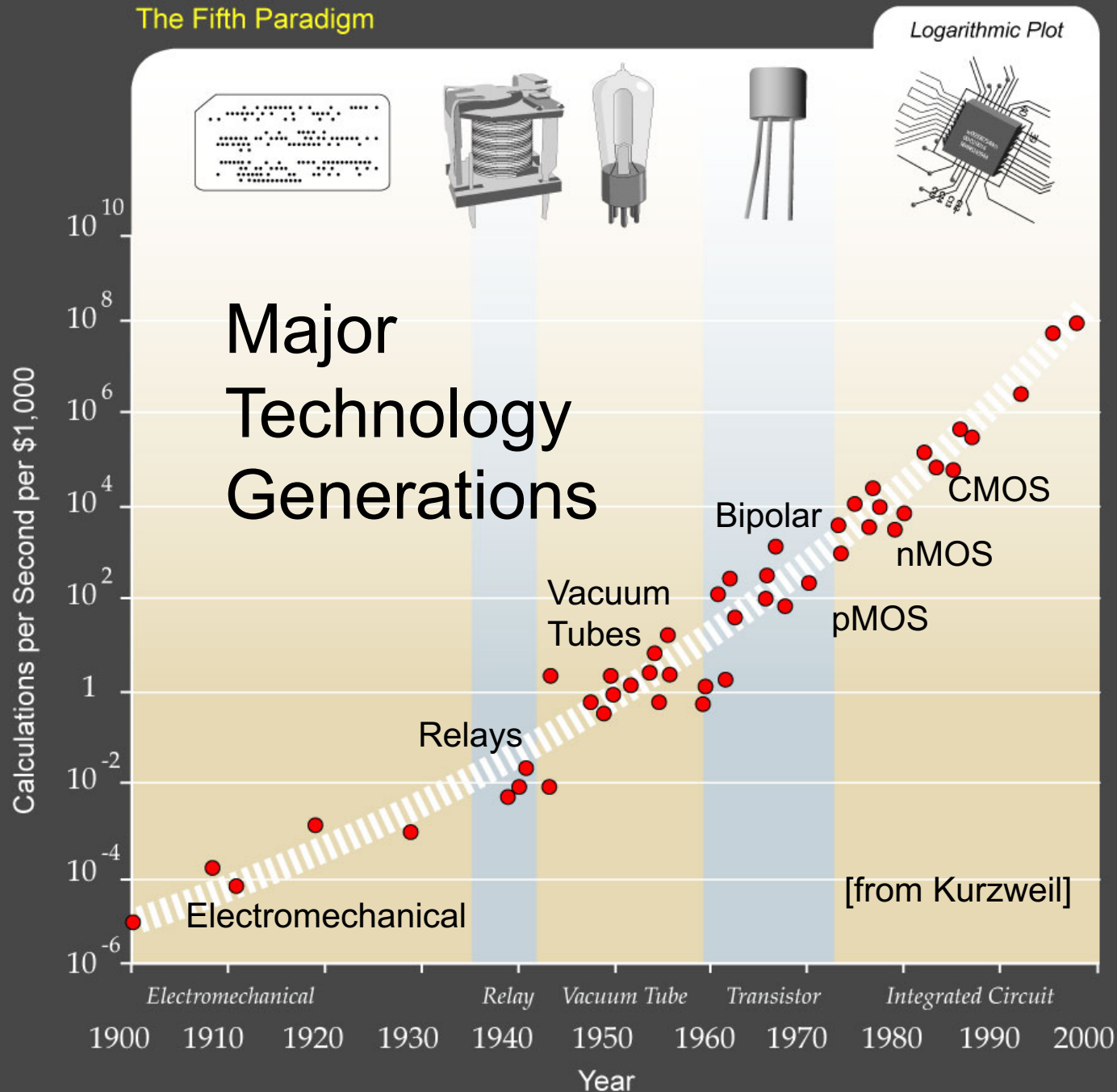
Improved technologies make new applications possible

Compatibility

Cost of software development makes compatibility a major force in market

# Moore's Law

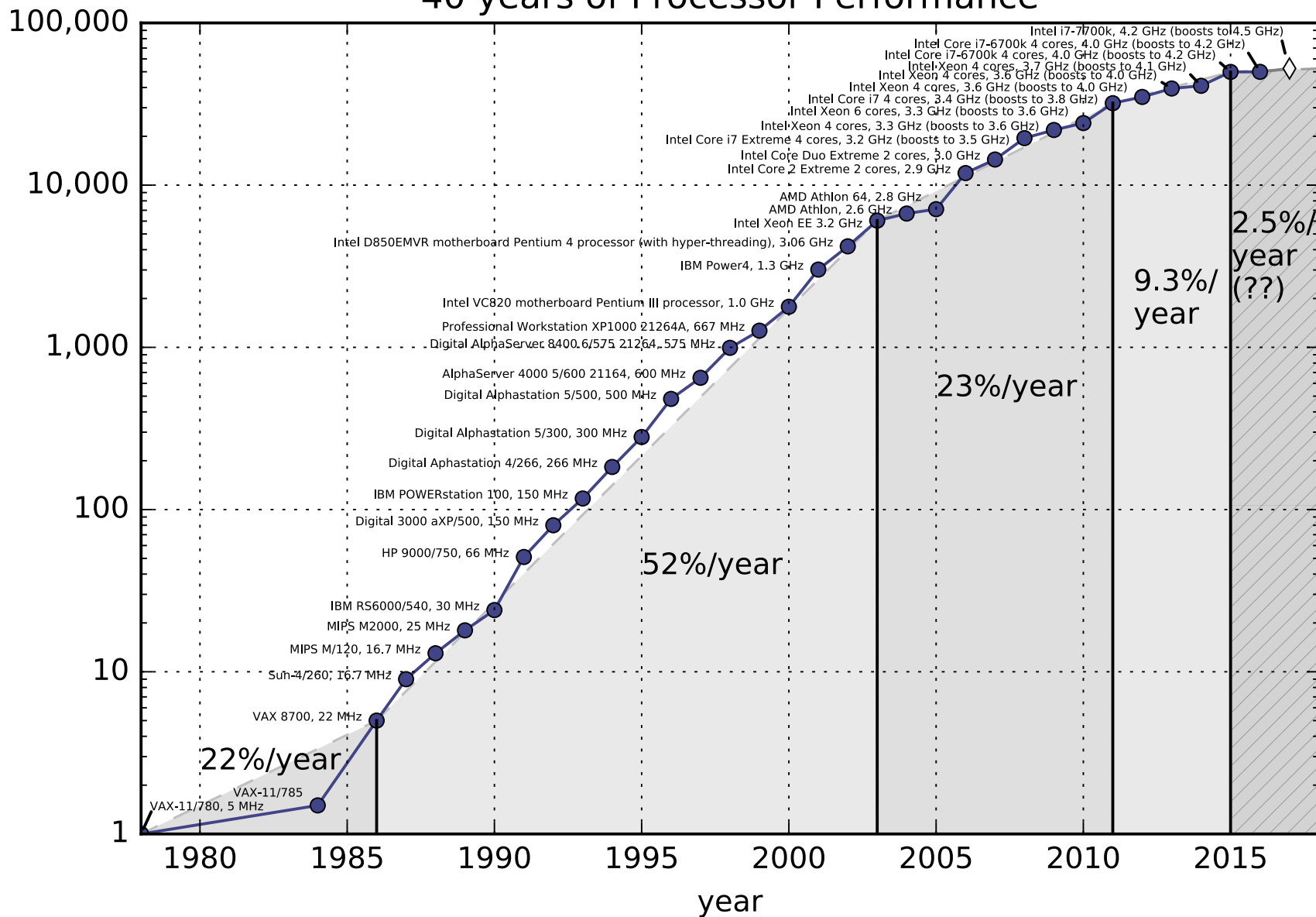
The Fifth Paradigm



# Single-Thread Processor Performance

## 40 years of Processor Performance

Performance vs. VAX-11/780



[ Hennessy & Patterson, 2017 ]

# Upheaval in Computer Design

- Most of last 50 years, Moore's Law ruled
  - Technology scaling allowed continual performance/energy improvements without changing software model
- Last decade, technology scaling slowed/stopped
  - Dennard (voltage) scaling over (supply voltage ~fixed)
  - Moore's Law (cost/transistor) over?
  - No competitive replacement for CMOS anytime soon
  - Energy efficiency constrains everything
- No “free lunch” for software developers, must consider:
  - Parallel systems
  - Heterogeneous systems

# Today's Dominant Target Systems

- Mobile (smartphone/tablet)
  - >1 billion sold/year
  - Market dominated by ARM-ISA-compatible general-purpose processor in system-on-a-chip (SoC)
  - Plus sea of custom accelerators (radio, image, video, graphics, audio, motion, location, security, etc.)
- Warehouse-Scale Computers (WSCs)
  - 100,000's cores per warehouse
  - Market dominated by x86-compatible server chips
  - Dedicated apps, plus cloud hosting of virtual machines
  - Now seeing increasing use of GPUs, FPGAs, custom hardware to accelerate workloads
- Embedded computing
  - Wired/wireless network infrastructure, printers
  - Consumer TV/Music/Games/Automotive/Camera/MP3
  - Internet of Things!



# COMP 590-154 Administrivia

Instructor: Amogh Akshintala, [aakshintala@cs.unc.edu](mailto:aakshintala@cs.unc.edu)

Office: SN 305

Office Hours: Tu/Th. 3:30-5:00 PM or by appointment

Lectures: Tu/Th, 2:00-3:15 PM, SN 011

Text: *Computer Architecture: A Quantitative Approach, Hennessey and Patterson, 6<sup>th</sup> Edition* (2017)

Readings assigned from this edition, some readings available in older editions – see web page.

Web page: <https://aakshintala.com/teaching/comp590-s20>

Lectures available online by noon before class

Piazza: <https://piazza.com/unc/spring2020/comp590154>



# Class policies

## Working in groups

- Permitted on everything except Final
- Groups may be of any size
- Points deducted on group work are multiplied by group size
- Great opportunity or Rope to hang yourself – you pick

## Attendance

- Optional (but highly advised)
- No laptop, tablet, or phone use in class

# Academic Integrity Policy

- You may...
  - Discuss assignment, design, techniques
- You may not...
  - Share code outside your group
  - Use code off the internet
    - » Only exception is for libraries/utilities for project
      - Clear with me first
- You must declare group composition...
  - Explicitly via email to TA and instructor
  - Explicitly for each assignment
  - At most five days after assignment handout

# Course Grading

Score >100 points and you get an A.

All scores below 100 points will be curved.

Total 150 possible points:

- Assignment 1 – 15
- Assignment 2 – 15
- Assignment 3 – 15
- Assignment 4 – 15
- Final Exam – 30 (you **MUST** show up and sign the coversheet even if you leave it blank)
- Project – 50
- Participation – 10

# Questions?

# Computer Architecture: A Little History

Throughout the course we'll use a historical narrative to help understand why certain ideas arose

Why worry about old ideas?

- Helps to illustrate the design process, and explains why certain decisions were taken
- Because future technologies might be as constrained as older ones
- Those who ignore history are doomed to repeat it
  - Every mistake made in mainframe design was also made in minicomputers, then microcomputers, where next?

# Analog Computers

- Analog computer represents problem variables as some physical quantity (e.g., mechanical displacement, voltage on a capacitor) and uses scaled physical behavior to calculate results



*[Marsyas, Creative Commons BY-SA 3.0]*

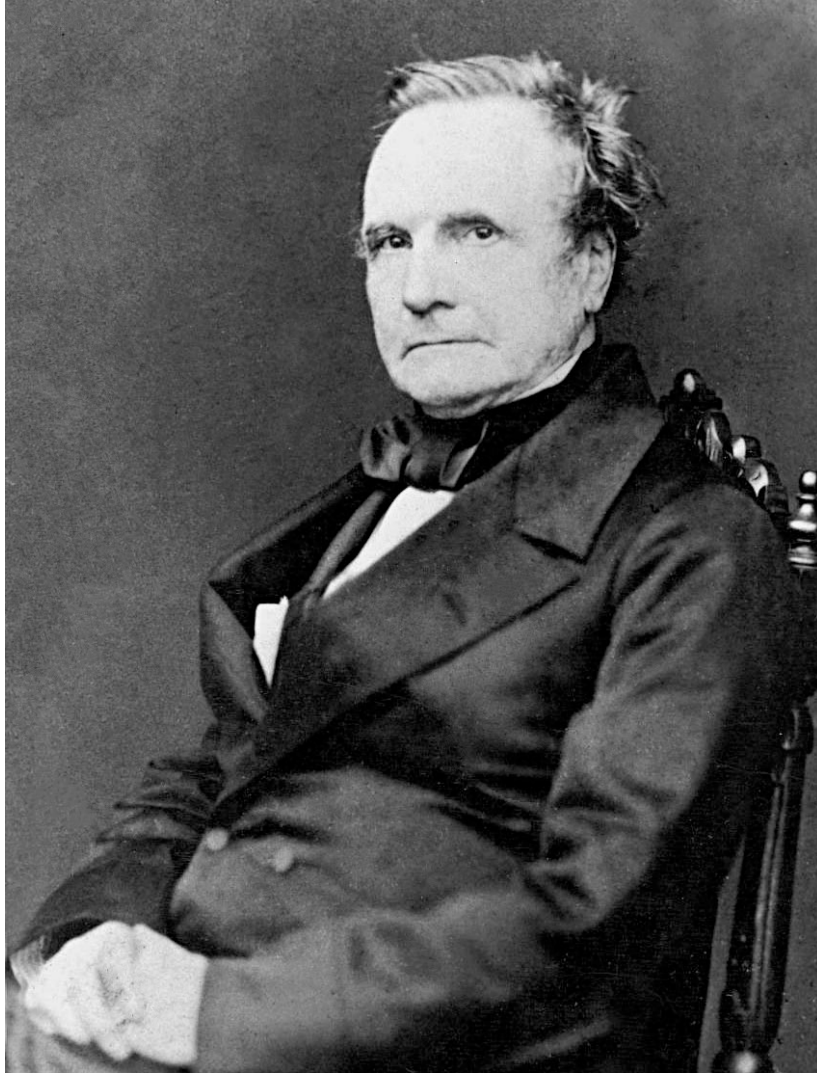
Antikythera mechanism c.100BC

# Digital Computers

- Represent problem variables as numbers encoded using discrete steps
  - Discrete steps provide noise immunity
- Enables accurate and deterministic calculations
  - Same inputs give same outputs exactly
- Not constrained by physically realizable functions
- Programmable digital computers are the focus of this class



# Charles Babbage (1791-1871)



*[Copyright expired and in public domain.  
Image obtained from Wikimedia Commons.]*

- Lucasian Professor of Mathematics, Cambridge University, 1828-1839
- A true “polymath” with interests in many areas
- Frustrated by errors in printed tables, wanted to build machines to evaluate and print accurate tables
- Inspired by earlier work organizing human “computers” to methodically calculate tables by hand

# Difference Engine 1822

- Continuous functions can be approximated by polynomials, which can be computed from difference tables:

$$f(n) = n^2 + n + 41$$

$$d1(n) = f(n) - f(n-1) = 2n$$

$$d2(n) = d1(n) - d1(n-1) = 2$$

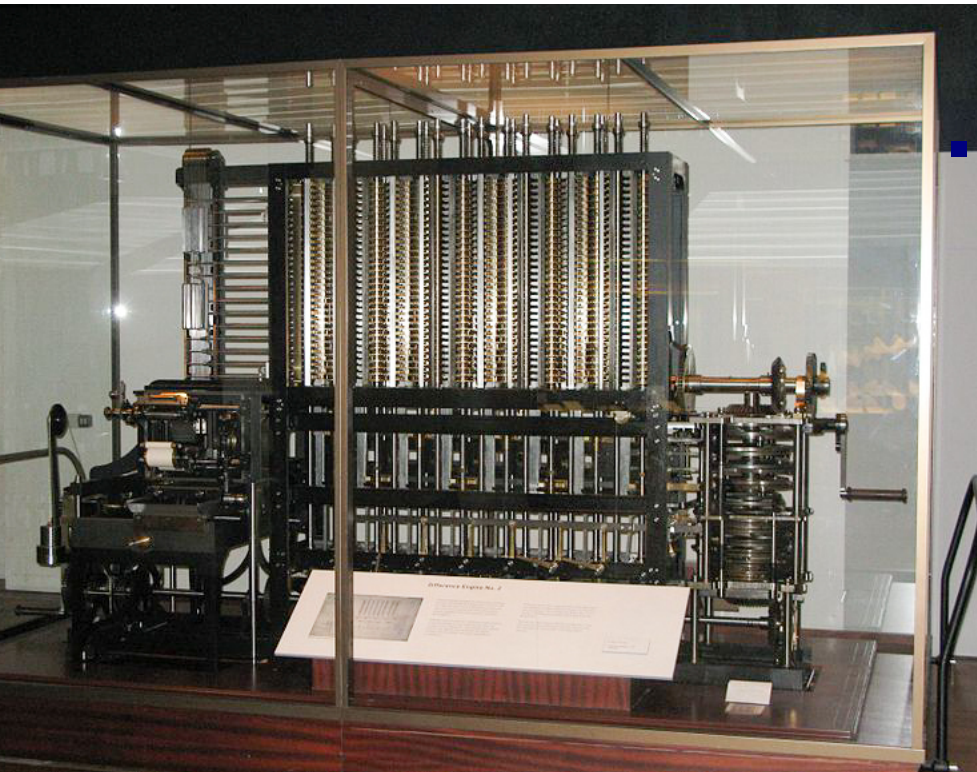
- Can calculate using only a single adder:

n	0	1	2	3	4
d2(n)			2	2	2
d1(n)		2	4	6	8
f(n)	41	43	47	53	61

The diagram illustrates the calculation of  $f(n)$  from  $d1(n)$  and  $d2(n)$  using a single adder. The table shows the values for  $n$  from 0 to 4. The  $d2(n)$  row shows constant values of 2. The  $d1(n)$  row shows values 2, 4, 6, 8. The  $f(n)$  row shows values 41, 43, 47, 53, 61. Red arrows indicate the flow of values:  $d2(n)$  values are added to the previous  $d1(n)$  value to get the next  $d1(n)$  value, which is then added to the previous  $f(n)$  value to get the next  $f(n)$  value.

# Realizing the Difference Engine

- Mechanical calculator, hand-cranked, using decimal digits
- Babbage did not complete the DE, moving on to the Analytical Engine (but used ideas from AE in improved DE 2 plan)
- Schuetz in Sweden completed working version in 1855, sold copy to British Government



- Modern day recreation of DE2, including printer, showed entire design possible using original technology
  - first at British Science Museum
  - copy at Computer History Museum in San Jose

# Analytical Engine 1837

- Recognized as first general-purpose digital computer
  - Many iterations of the design (multiple Analytical Engines)
- Contains the major components of modern computers:
  - “Store”: Main memory where numbers and intermediate results were held (1,000 decimal words, 40-digits each)
  - “Mill”: Arithmetic unit where processing was performed including addition, multiplication, and division
  - Also supported conditional branching and looping, and exceptions on overflow (machine jams and bell rings)
  - Had a form of microcode (the “Barrel”)
- Program, input and output data on punched cards
- Instruction cards hold opcode and address of operands in store
  - 3-address format with two sources and one destination, all in store
- Branches implemented by mechanically changing order cards were inserted into machine
- Only small pieces were ever built

# Analytical Engine Design Choices

- Decimal, because storage on mechanical gears
  - Babbage considered binary and other bases, but no clear advantage over human-friendly decimal
- 40-digit precision (equivalent to  $>133$  bits)
  - To reduce impact of scaling given lack of floating-point hardware
- Used “locking” or mechanical amplification to overcome noise in transferring mechanical motion around machine
  - Similar to non-linear gain in digital electronic circuits
- Had a fast “anticipating” carry
  - Mechanical version of pass-transistor carry propagate used in CMOS adders (and earlier in relay adders)



# Ada Lovelace (1815-1852)



*[By Margaret Sarah Carpenter,  
Copyright expired and in public domain]*

- Translated lectures of Luigi Menabrea who published notes of Babbage's lectures in Italy
- Lovelace considerably embellished notes and described Analytical Engine program to calculate Bernoulli numbers that would have worked if AE was built
  - The first program!
- Imagined many uses of computers beyond calculations of tables
- Was interested in modeling the brain

# Early Programmable Calculators

- Analog computing was popular in first half of 20th century as digital computing was too expensive
- But during late 30s and 40s, several programmable digital calculators were built (date when operational)
  - Atanasoff Linear Equation Solver (1939)
  - Zuse Z3 (1941)
  - Harvard Mark I (1944)
  - ENIAC (1946)



# Atanasoff-Berry Linear Equation Solver (1939)

- Fixed-function calculator for solving up to 29 simultaneous linear equations
- Digital binary arithmetic (50-bit fixed-point words)
- Dynamic memory (rotating drum of capacitors)
- Vacuum tube logic for processing



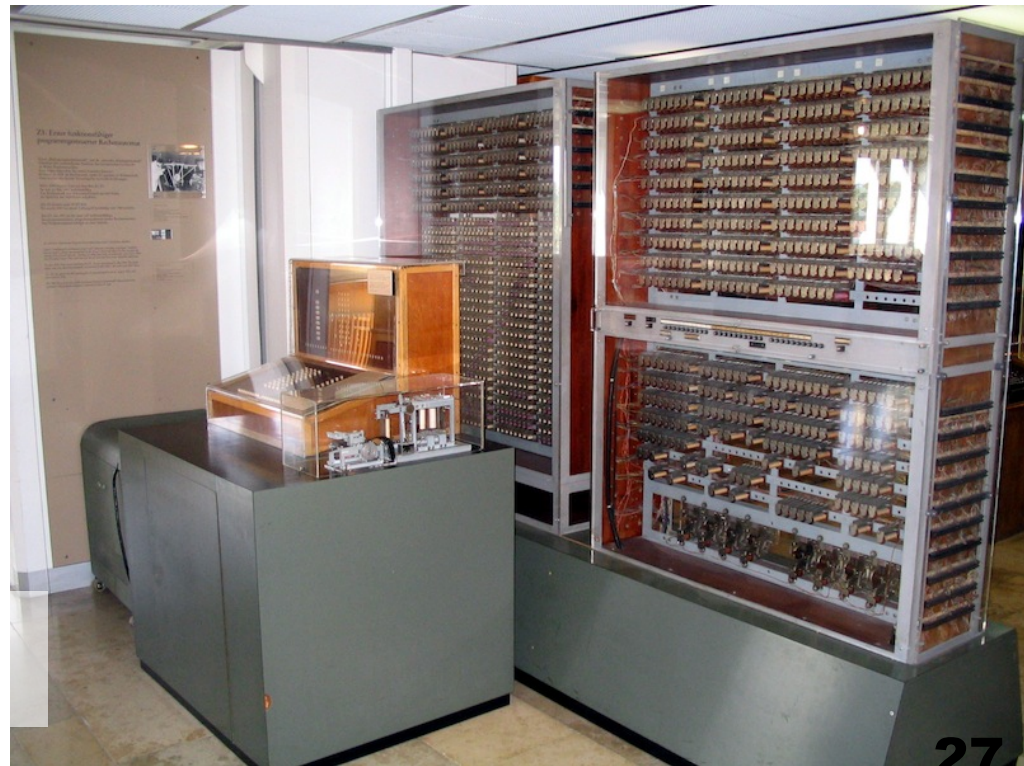
[Manop, Creative Commons BY-SA 3.0]

*In 1973, Atanasoff was credited as inventor of “automatic electronic digital computer” after patent dispute with Eckert and Mauchly (ENIAC)*

# Zuse Z3 (1941)

- Built by Konrad Zuse in wartime Germany using 2000 relays
- Had normalized floating-point arithmetic with hardware handling of exceptional values (+/- infinity, undefined)
  - 1-bit sign, 7-bit exponent, 14-bit significand
- 64 words of memory
- Two-stage pipeline 1) fetch&execute 2) writeback
- No conditional branch
- Programmed via paper tape

Replica of the Zuse Z3 in the  
Deutsches Museum, Munich





# Harvard Mark I (1944)

- Proposed by Howard Aiken at Harvard, and funded and built by IBM
- Mostly mechanical with some electrically controlled relays and gears
- Weighed 5 tons and had 750,000 components
- Stored 72 numbers each of 23 decimal digits
- Speed: adds 0.3s, multiplies 6s, divide 15s, trig >1 minute
- Instructions on paper tape (2-address format)
- Could run long programs automatically
- **Loops by gluing paper tape into loops**
- No conditional branch
- Although mentioned Babbage in proposal, was more limited than analytical engine



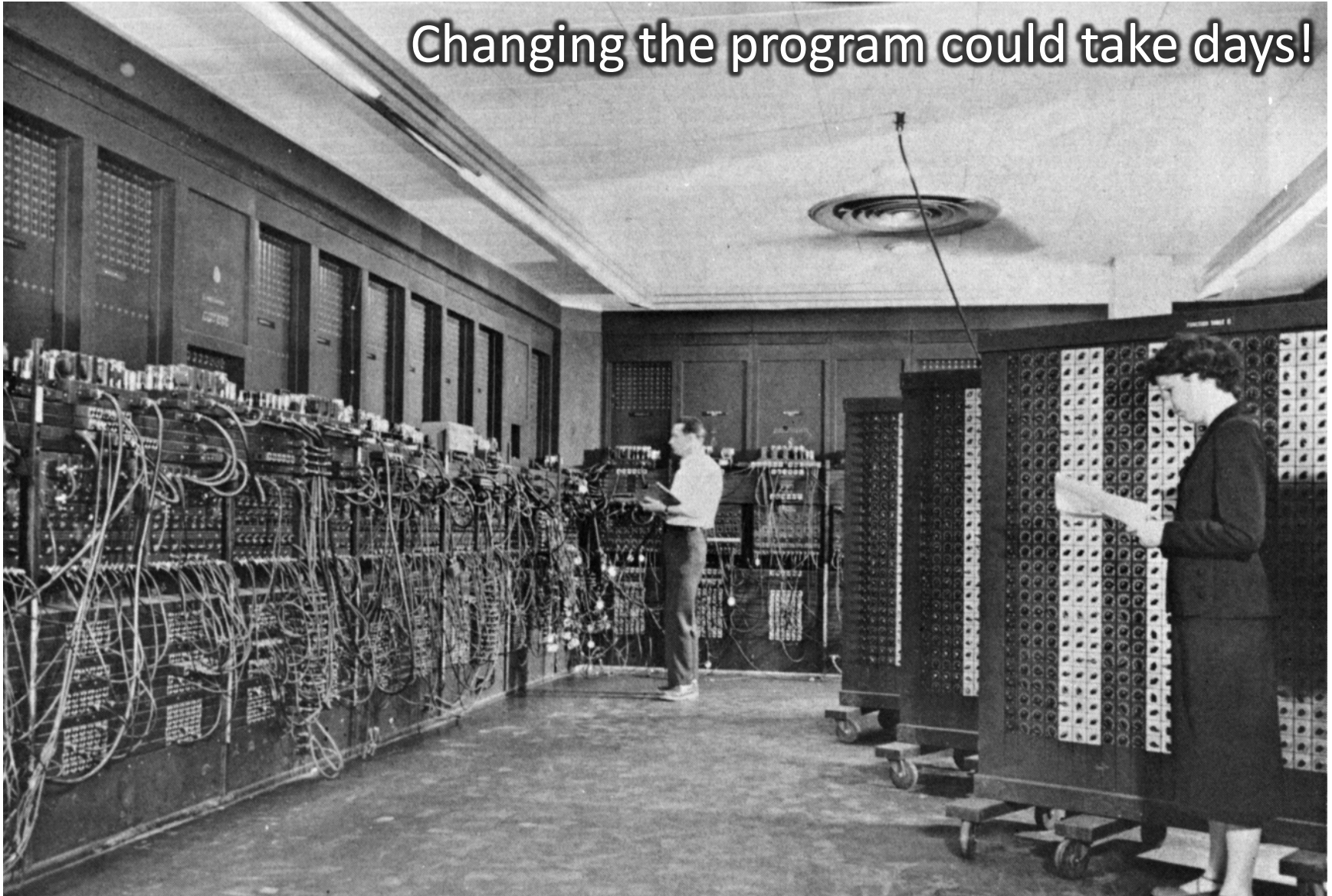
# ENIAC (1946)

- First electronic general-purpose computer
- Construction started in secret at UPenn Moore School of Electrical Engineering during WWII to calculate firing tables for US Army, designed by Eckert and Mauchly
- 17,468 vacuum tubes
- **Weighed 30 tons, occupied 1800 sq ft, power 150kW**
- Twelve 10-decimal-digit accumulators
- Had a conditional branch!
- Programmed by plugboard and switches, time consuming!
- Purely electronic instruction fetch and execution, so fast
  - 10-digit x 10-digit multiply in 2.8ms (2000x faster than Mark-1)
- As a result of speed, it was almost entirely I/O bound
- As a result of large number of tubes, it was often broken (5 days was longest time between failures)



# ENIAC

Changing the program could take days!



*[Public Domain, US Army Photo]*

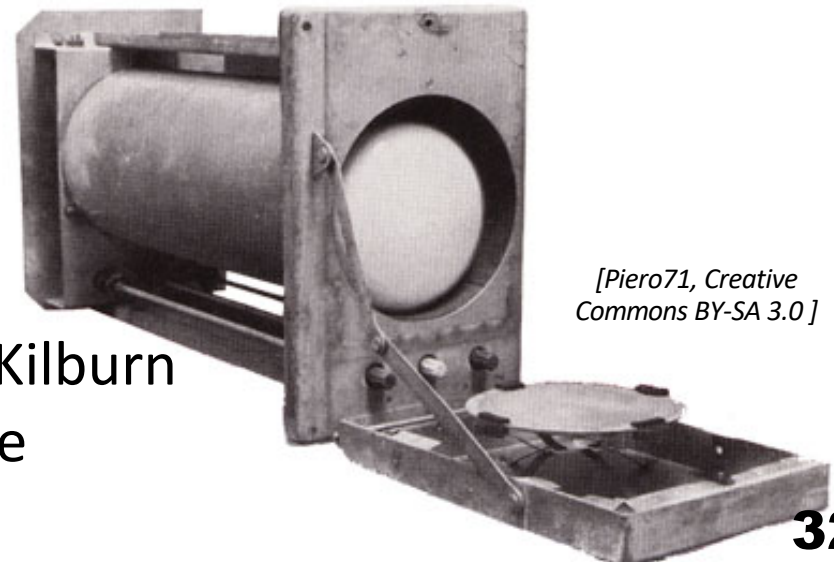
# EDVAC

- ENIAC team started discussing stored-program concept to speed up programming and simplify machine design
- John von Neumann was consulting at UPenn and typed up ideas in “First Draft of a report on EDVAC”
- Herman Goldstine circulated the draft June 1945 to many institutions, igniting interest in the stored-program idea
  - But also, ruined chances of patenting it
  - Report falsely gave sole credit to von Neumann for the ideas
  - Maurice Wilkes was excited by report and decided to come to US workshop on building computers
- Later, in 1948, modifications to ENIAC allowed it to run in stored-program mode, but 6x slower than hardwired
  - Due to I/O limitations, this speed drop was not practically significant and improvement in productivity made it worthwhile
- EDVAC eventually built and (mostly) working in 1951
  - Delayed by patent disputes with university

# Manchester SSEM “Baby” (1948)

- Manchester University group build small-scale experimental machine to demonstrate idea of using cathode-ray tubes (CRTs) for computer memory instead of mercury delay lines
- *Williams-Kilburn Tubes were first random access electronic storage devices*
- 32 words of 32-bits, accumulator, and program counter
- Machine ran world’s first stored-program in June 1948
- Led to later Manchester Mark-1 full-scale machine
  - Mark-1 introduced *index* registers
  - Mark-1 commercialized by Ferranti

Williams-Kilburn  
Tube Store



[Piero71, Creative Commons BY-SA 3.0]



# Cambridge EDSAC (1949)

- Maurice Wilkes came back from workshop in US and set about building a stored-program computer in Cambridge
- EDSAC used mercury-delay line storage to hold up to 1024 words (512 initially) of 17 bits (+1 bit of padding in delay line)
- Two's-complement binary arithmetic
- Accumulator ISA with self-modifying code for indexing
- David Wheeler, who earned the world's first computer science PhD, invented the subroutine ("Wheeler jump") for this machine
  - Users built a large library of useful subroutines
- UK's first commercial computer, LEO-I (Lyons Electronic Office), was based on EDSAC, ran business software in 1951
  - Software for LEO was still running in the 1980s in emulation on ICL mainframes!
- EDSAC-II (1958) was first machine with microprogrammed control unit

# **Commercial computers:**

## **BINAC (1949) and UNIVAC (1951)**

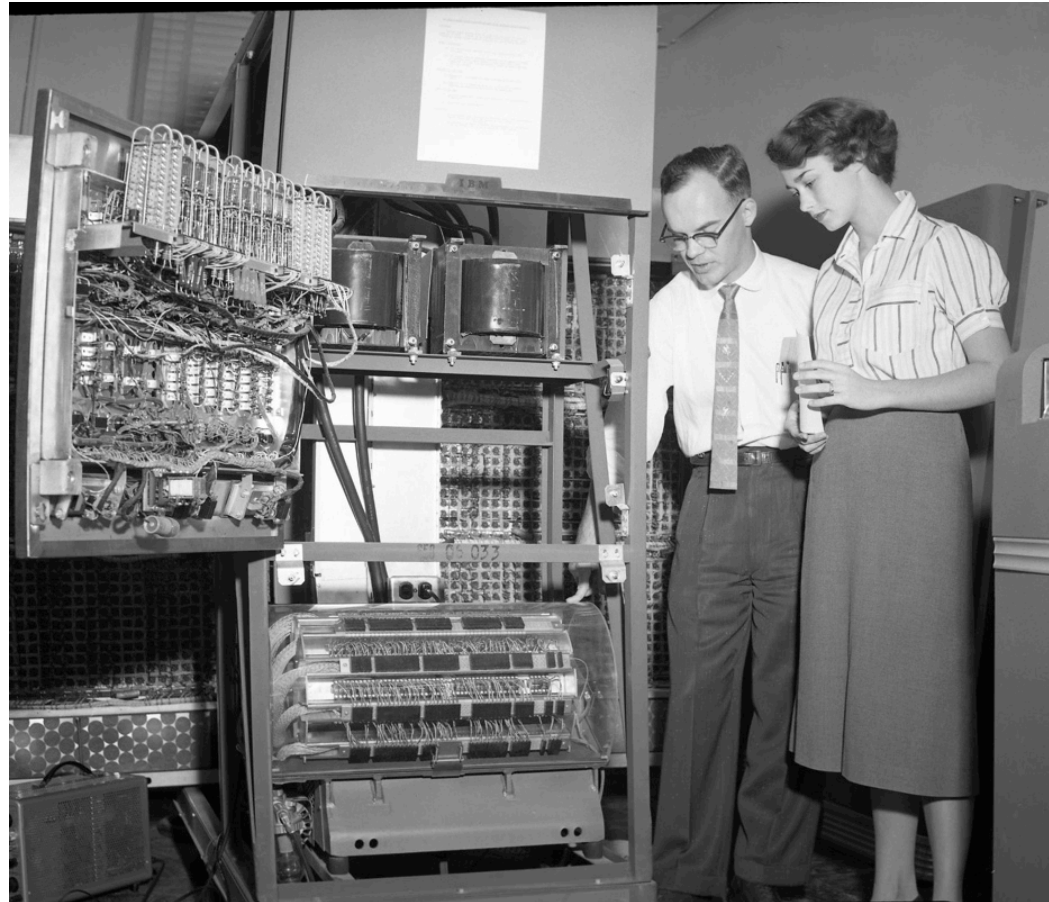
- Eckert and Mauchly left U.Penn after patent rights disputes and formed the Eckert-Mauchly Computer Corporation
- World's first commercial computer was BINAC with two CPUs that checked each other
  - BINAC apparently never worked after shipment to first (only) customer
- Second commercial computer was UNIVAC
  - Used mercury delay-line memory, 1000 words of 12 alpha characters
  - Famously used to predict presidential election in 1952
  - Eventually 46 units sold at >\$1M each
  - Often, mistakingly called the IBM UNIVAC

# IBM 701 (1952)

- IBM's first commercial scientific computer
- Main memory was 72 Williams's Tubes, each 1Kib, for total of 2048 words of 36 bits each
  - Memory cycle time of 12μs
- Accumulator ISA with multiplier/quotient register
- 18-bit/36-bit numbers in sign-magnitude fixed-point
- Misquote from Thomas Watson Sr/Jr:  
***“I think there is a world market for maybe five computers”***
- Actually TWJr said at shareholder meeting:  
***“as a result of our trip [selling the 701], on which we expected to get orders for five machines, we came home with orders for 18.”***

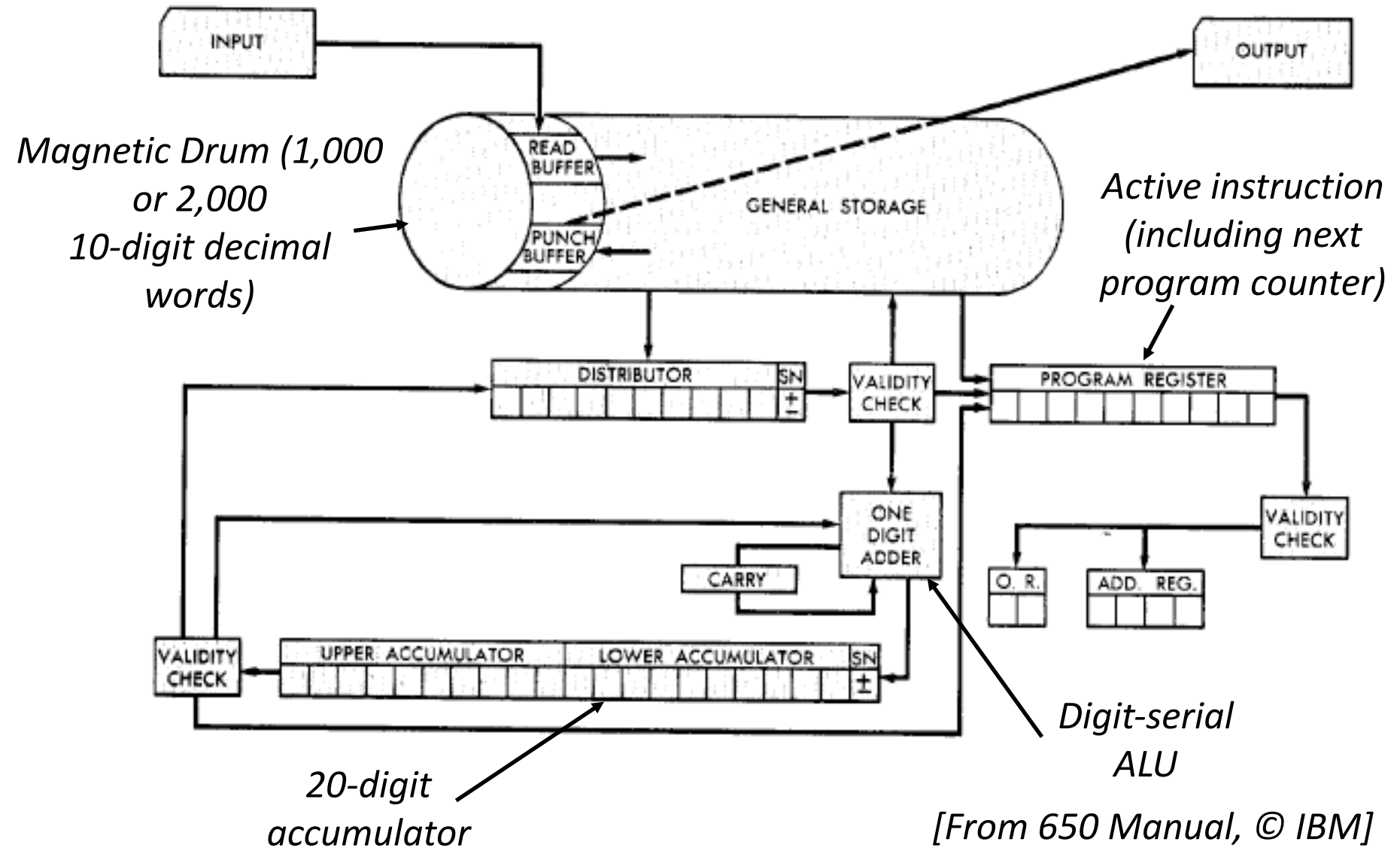
# IBM 650 (1953)

- The first mass-produced computer
- Low-end system with drum-based storage and digit serial ALU
- Almost 2,000 produced



*[Cushing Memorial Library and Archives, Texas A&M,  
Creative Commons Attribution 2.0 Generic]*

# IBM 650 Architecture



# IBM 650 Instruction Set

- Address and data in 10-digit decimal words
- Instructions encode:
  - Two-digit opcode encoded 44 instructions in base instruction set, expandable to 97 instructions with options
  - Four-digit data address
  - Four-digit next instruction address
    - Programmer's arrange code to minimize drum latency!
- Special instructions added to compare value to all words on track

# Early Instruction Sets

- Very simple ISAs, mostly single-address accumulator-style machines, as high-speed circuitry was expensive
  - Based on earlier “calculator” model
- Over time, appreciation of software needs shaped ISA
- Index registers (Kilburn, Mark-1) added to avoid need for self-modifying code to step through array
- Over time, more index registers were added
- And more operations on the index registers
- Eventually, just provide general-purpose registers (GPRs) and orthogonal instruction sets
- But some other options explored...

# **Burrough's B5000 Stack Architecture:**

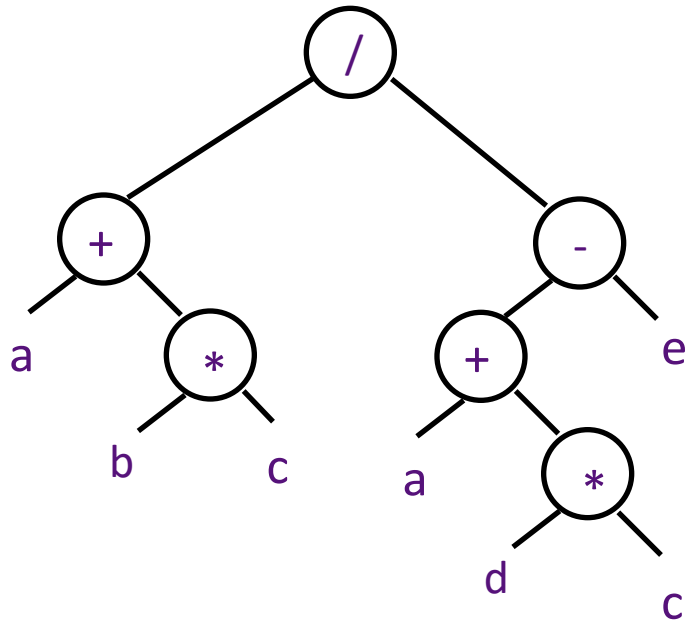
## **Robert Barton, 1960**

- Hide instruction set completely from programmer using high-level language (ALGOL)
- Use stack architecture to simplify compilation, expression evaluation, recursive subroutine calls, interrupt handling,...



# Evaluation of Expressions

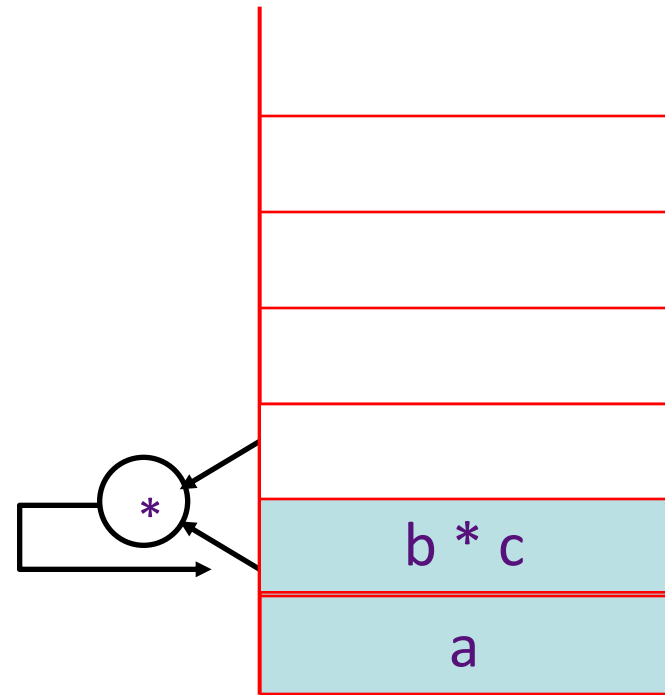
$(a + b * c) / (a + d * c - e)$



Reverse Polish

$a \ b \ c \ * \ + \ a \ d \ c \ * \ + \ e \ - \ /$

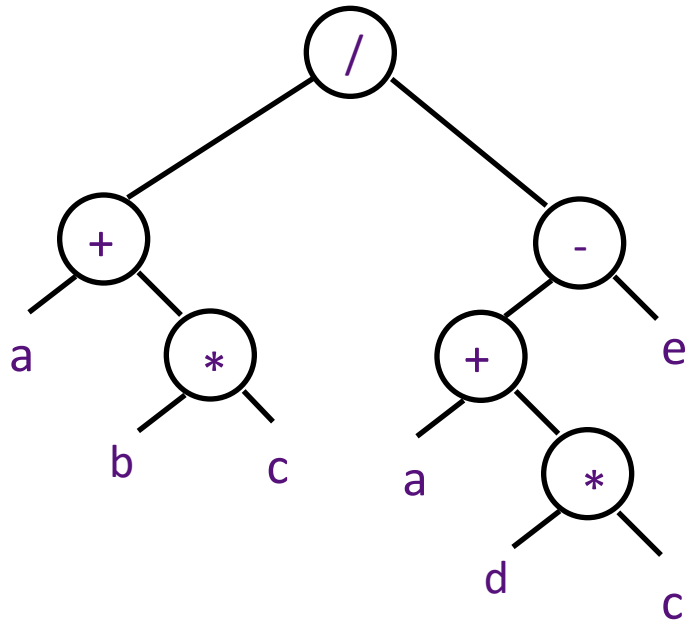
↑ ↑ ↑ ↑  
push push push multiply



Evaluation Stack

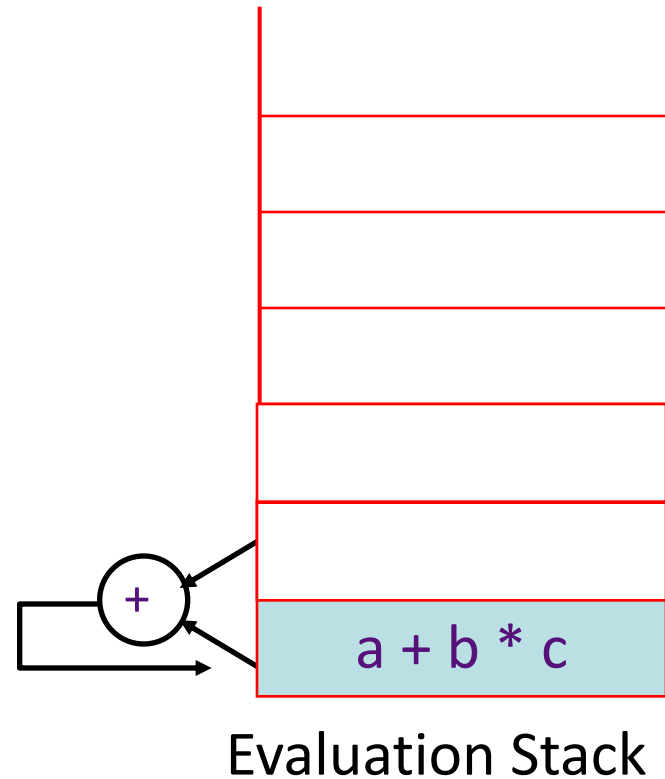
# Evaluation of Expressions

$(a + b * c) / (a + d * c - e)$



Reverse Polish

$a \ b \ c \ * \ + \ a \ d \ c \ * \ + \ e \ - \ /$   
                    ↑  
                  add



# IBM's Big Bet: 360 Architecture

- By early 1960s, IBM had several incompatible families of computer:
  - 701 → 7094
  - 650 → 7074
  - 702 → 7080
  - 1401 → 7010
- Each system had its own
  - Instruction set
  - I/O system and secondary storage (magnetic tapes, drums and disks)
  - assemblers, compilers, libraries,...
  - market niche (business, scientific, real time, ...)

# **IBM 360 : Design Premises**

## **Amdahl, Blaauw and Brooks, 1964**

- The design must lend itself to growth and successor machines
- General method for connecting I/O devices
- Total performance - answers per month rather than bits per microsecond → programming aids
- Machine must be capable of supervising itself without manual intervention
- Built-in hardware fault checking and locating aids to reduce down time
- Simple to assemble systems with redundant I/O devices, memories etc. for fault tolerance
- Some problems required floating-point larger than 36 bits

# Stack versus GPR Organization

*Amdahl, Blaauw and Brooks, 1964*

1. The performance advantage of push-down stack organization is derived from the presence of fast registers and not the way they are used.
2. “Surfacing” of data in stack which are “profitable” is approximately 50% because of constants and common subexpressions.
3. Advantage of instruction density because of implicit addresses is equaled if short addresses to specify registers are allowed.
4. Management of finite-depth stack causes complexity.
5. Recursive subroutine advantage can be realized only with the help of an independent stack for addressing.
6. Fitting variable-length fields into fixed-width word is awkward.

# IBM 360: A General-Purpose Register (GPR) Machine

- Processor State
    - 16 General-Purpose 32-bit Registers
      - may be used as index and base register
      - Register 0 has some special properties
    - 4 Floating Point 64-bit Registers
    - A Program Status Word (PSW)
      - PC, Condition codes, Control flags
  - A 32-bit machine with 24-bit addresses
    - But no instruction contains a 24-bit address!
  - Data Formats
    - 8-bit bytes, 16-bit half-words, 32-bit words, 64-bit double-words
- ↖ The IBM 360 is why bytes are 8-bits long today!  
Change from 6-bit to 8-bit by **Fred Brooks!***

# IBM 360: Initial Implementations

	<i>Model 30</i>	<i>...</i>	<i>Model 70</i>
<i>Storage</i>	8K - 64 KB		256K - 512 KB
<i>Datapath</i>	8-bit		64-bit
<i>Circuit Delay</i>	30 nsec/level		5 nsec/level
<i>Local Store</i>	Main Store		Transistor Registers
<i>Control Store</i>	Read only 1 $\mu$ sec	Conventional circuits	

*IBM 360 instruction set architecture (ISA) completely hid the underlying technological differences between various models.*

*Milestone: The first true ISA designed as portable hardware-software interface!*

*With minor modifications it still survives today!*

# IBM Mainframes survive until today

*[z14, 2017, 14nm technology, 17 layers of metal, 696 sq mm]*

IBM Z

IBM

## z14 processor design summary

### Micro-Architecture

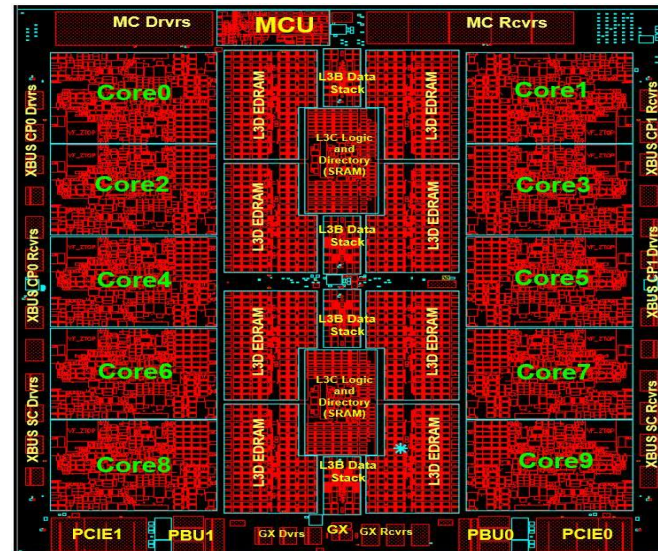
- 10 cores per CP-chip
- 5.2GHz
- Cache Improvements:
  - 128KB I\$ + 128KB D\$
  - 2x larger L2 D\$ (4MB)
  - 2x larger L3 Cache
  - symbol ECC
- New translation & TLB design
  - Logical-tagged L1 directory
  - Pipelined 2<sup>nd</sup> level TLB
  - Multiple translation engines
- Pipeline Optimizations
  - Improved instruction delivery
  - Faster branch wakeup
  - Improved store hazard avoidance
  - 2x double-precision FPU bandwidth
  - Optimized 2<sup>nd</sup> generation SMT2
- Better Branch Prediction
  - 33% Larger BTB1 & BTB2
  - New Perceptron & Simple Call/Return Predictor

### Architecture

- PauseLess Garbage Collection
- Vector Single & Quad precision
- Long-multiply support (RSA, ECC)
- Register-to-register BCD arithmetic

### Accelerators










- Redesigned in-core crypto-accelerator
  - Improved performance
  - New functions (GCM, TRNG, SHA3)
- Optimized in-core compression accelerator
  - Improved start/stop latency
  - Huffman encoding for better compression ratio
  - Order-preserving compression





# Server Market

## Top global server vendors (in percent)

Company	2017	Market share	2018	Revenue growth <sup>†</sup>	Server core processor chip
Dell	17.7		18.8	52.9	Intel, AMD
HPE/New H3C Group	21.4		16.6	11.7	Intel, AMD
IBM*	6.6		7.3	57.0	IBM
Lenovo*	5.3		6.9	85.7	Intel, AMD
Inspur*	3.3		4.8	112.3	Intel, AMD
Cisco*	5.6		4.8	22.4	Intel, AMD
Huawei*	3.5		4.3	77.1	Intel, AMD, ARM Holdings
ODM direct**	22.5		24.4	55.9	Intel, AMD
Others	14.2		12.2	23.1	Intel, AMD

TSMC supplies server core processor chips for AMD, ARM Holdings and soon IBM platforms

\*IDC declares a statistical tie when there is a difference of 1% or less in the share of revenues or shipments among two or more vendors \*\*Original design manufacturer direct includes megadata center orders by Google, Amazon, Facebook, Microsoft to manufacturing partners

<sup>†</sup>As of second quarter

Source: IDC

## And in conclusion ...

- Computer Architecture >> ISAs and RTL
- COMP 590 is about interaction of hardware and software, and design of appropriate abstraction layers
- Computer architecture is shaped by technology and applications
  - History provides lessons for the future
- Computer Science at the crossroads from sequential to parallel computing
  - Salvation requires innovation in many fields, including computer architecture
- Read Chapter 1 & Appendix A for next time!

# Acknowledgements

- These slides contain material developed and copyright by:
  - Arvind (MIT)
  - Krste Asanovic (MIT/UCB)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)
  - Mike Ferdman (SBU)
  - Donald E. Porter (UNC/SBU)
- MIT material derived from course 6.823
- UCB material derived from course CS252