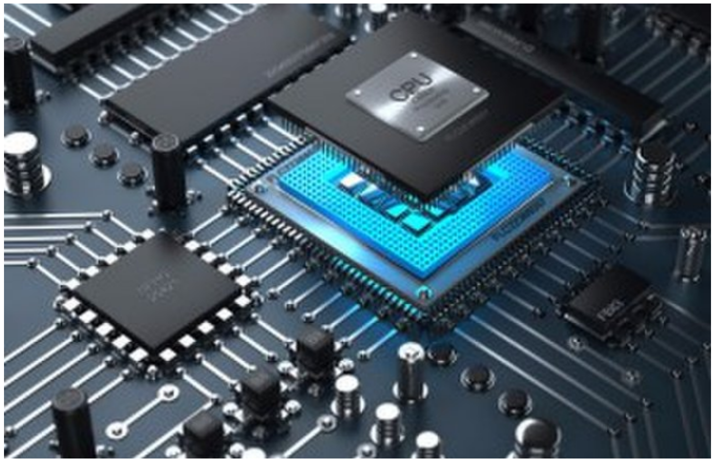


Computer Technology Performance Metrics

CS 154: Computer Architecture

Lecture #3

Winter 2020

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

- Lab 01 – how did Friday go?
- Gradescope account?
- Piazza account?
- **Remember**: due date is Wednesday on Gradescope!

Job/Help Opportunity

Disabled Students Program Notetaker Needed
CMPSC 154 MW 12:30

$25 per unit (of the class)

(prorated based on the number of weeks for which they are selected)

Questions can be sent to DSP Notetaking
Email: notes@sa.ucsb.edu

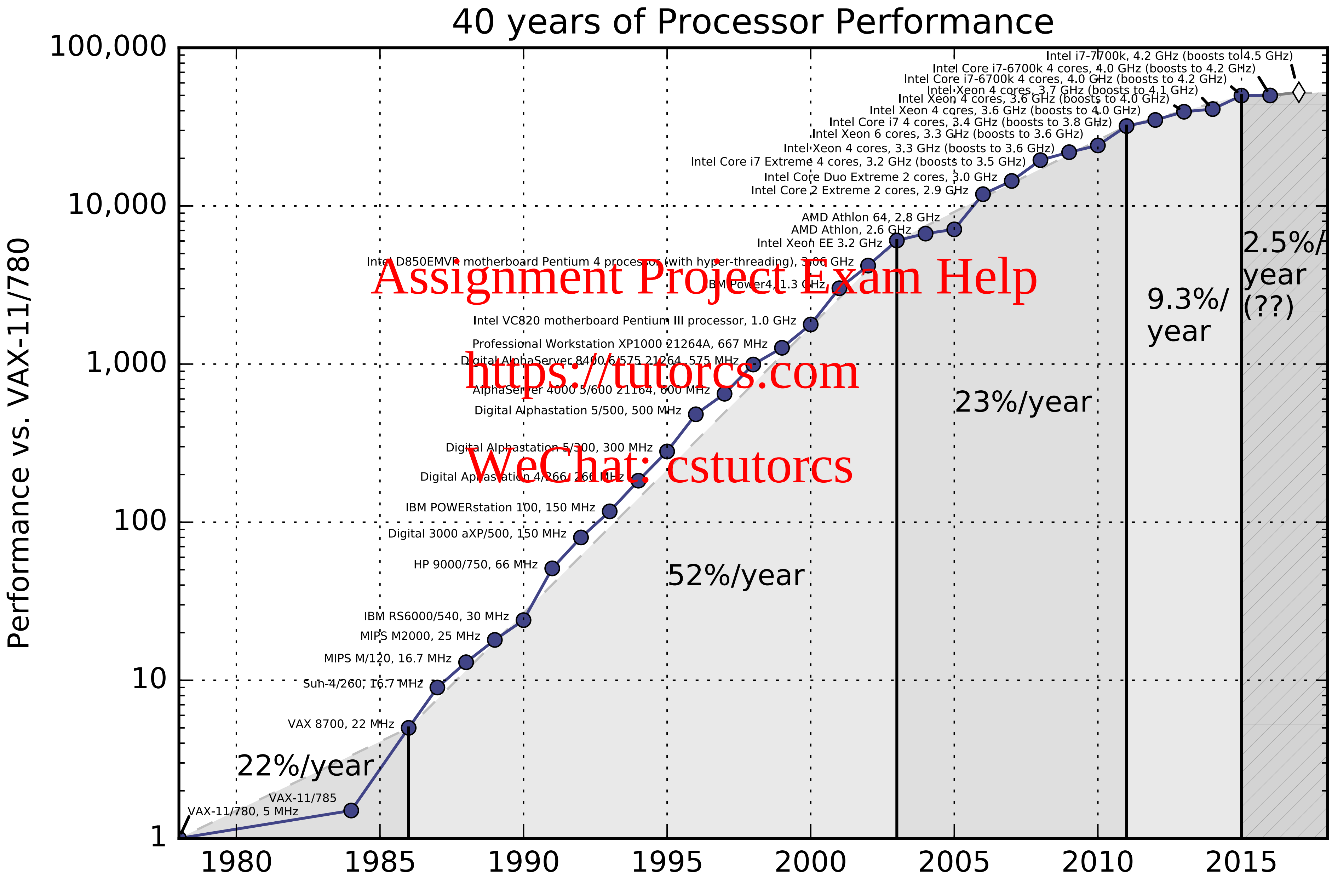
Potential Notetakers can apply online at http://dsp.sa.ucsb.edu/services

Lecture Outline

- Tech Details
  - Trends
  - Historical context
  - The manufacturing process of ICs

- Important Performance Measures
  - CPU time
  - CPI
  - Other factors (power, multiprocessors)
  - Pitfalls

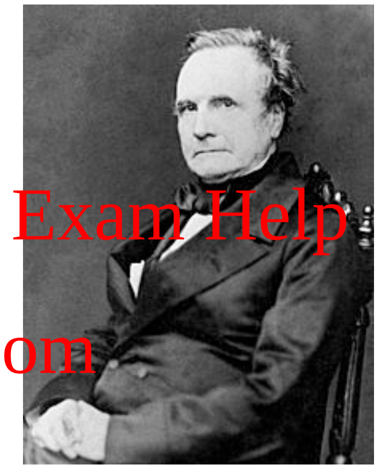
Single-Thread Processor Performance



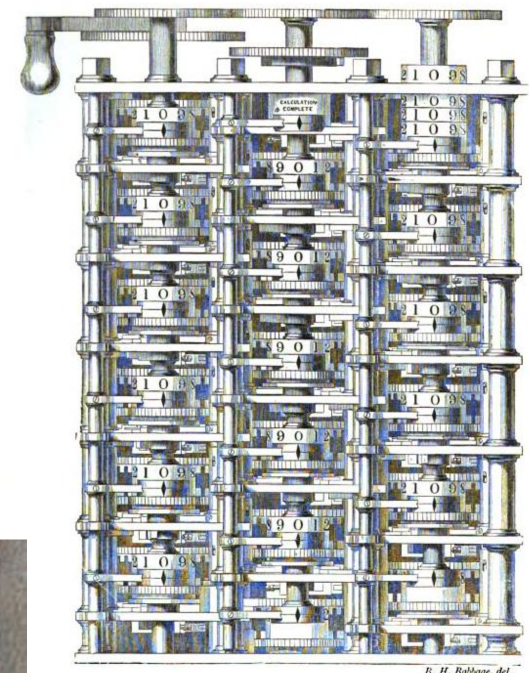
[Hennessy & Patterson, 2017]

Computing Devices for General Purposes

- **Charles Babbage (UK)**
  - *Analytical Engine* could calculate polynomial functions and differentials
  - Inspired by older generation of calculating machines made by Blaise Pascal (1623-1662, France)
  - Calculated results, but also *stored intermediate findings* (i.e. precursor to computer memory)
  - “Father of Computer Engineering”



C. Babbage (1791 – 1871)



Part of Babbage’s Analytical Engine

- **Ada Byron Lovelace (UK)**
  - Worked with Babbage and foresaw computers doing much more than calculating numbers
  - Loops and Conditional Branching
  - “Mother of Computer Programming”

A. Byron Lovelace (1815 – 1852)

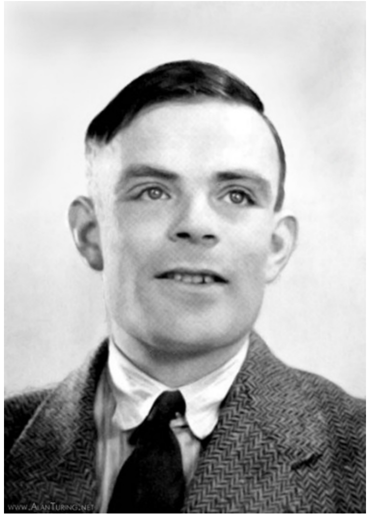
Images from Wikimedia.org

The Modern Digital Computer

- Calculating machines kept being produced in the early 20th century (IBM was established in the US in 1911)
- Instructions were very simple, which made hardware implementation easier, but this hindered the creation of complex programs.

Alan Turing (UK)

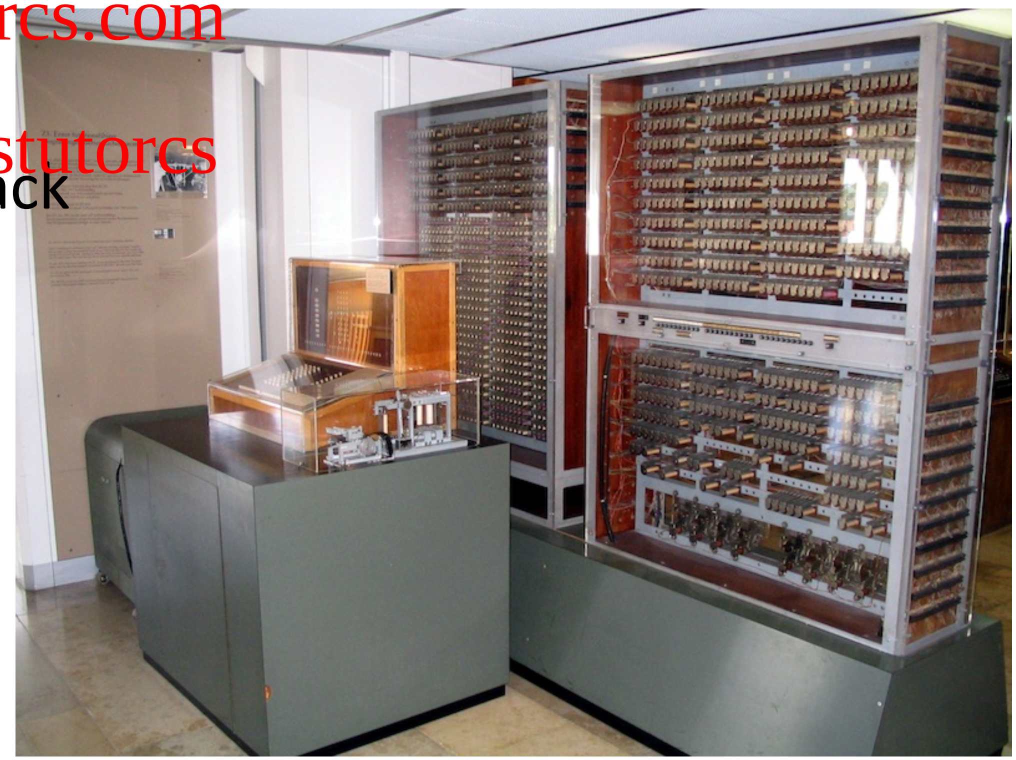
- Theorized the possibility of computing machines capable of performing *any* conceivable mathematical computation as long as this was representable as an *algorithm*
  - Called "*Turing Machines*" (1936) – ideas live on today…
  - Lead the effort to create a machine to successfully decipher the German "Enigma Code" during World War II



A. Turing (1912 – 1954)

Zuse Z3 (1941)

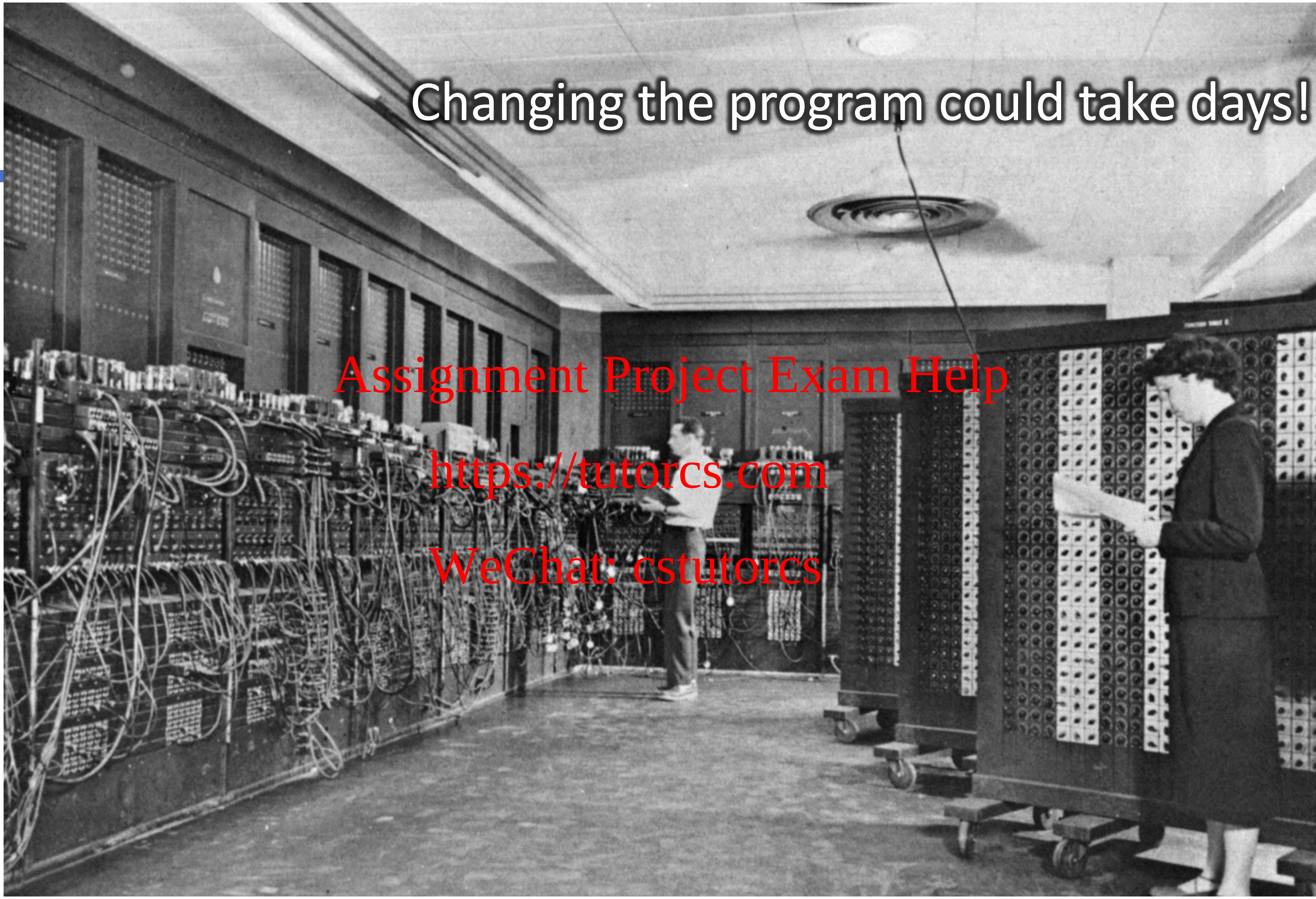
- Built by Konrad Zuse in wartime Germany using 2000 relays
- Could do *floating-point* arithmetic with hardware
- 22-bit word length; clock frequency of about 4–5 Hz!!
- 64 words of memory!!!
- Two-stage pipeline
  1) fetch & execute, 2) writeback
- No conditional branch
- Programmed via paper tape



Replica of the Zuse Z3 in the Deutsches Museum, Munich

ENIAC (1946)

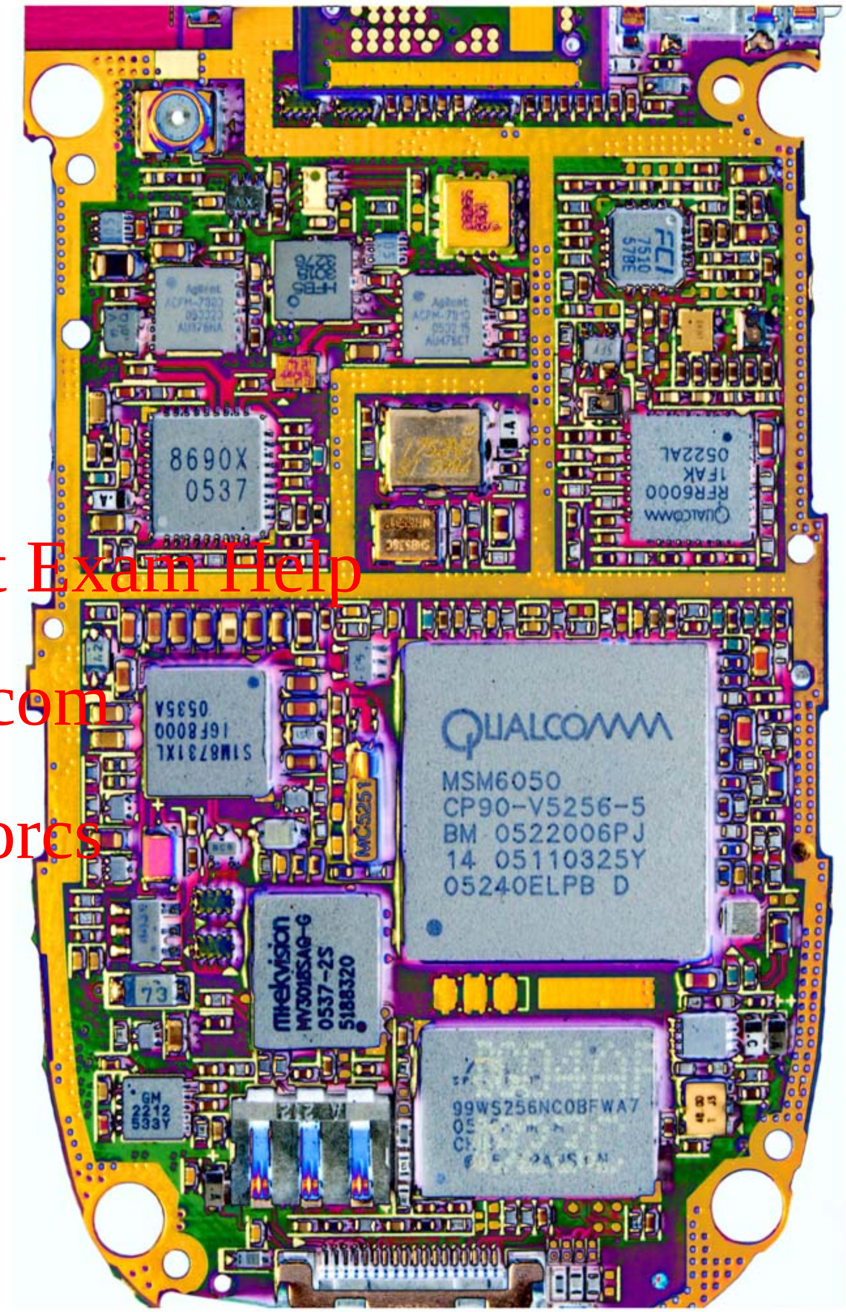
- First electronic general-purpose computer
- Constructed during WWII to calculate firing tables for US Army
  - Trajectories (for bombs) computed in 30 seconds instead of 40 hours
  - Was very fast for its time – started to replace human "computers"
- Used vacuum tubes (transistors hadn't been invented yet)
- Weighed **30 tons**, occupied **1800 sq ft**
- It used **160 kW** of power (about 3000 light bulbs worth)
- It cost **$6.3 million** in today's money to build.
- Programmed by plugboard and switches, time consuming!
- As a result of large number of tubes, it was often broken (5 days was longest time between failures!)



[Public Domain, US Army Photo]

Comparing today's cell phones (with dual CPUs), with ENIAC, we see they

cost 17,000X less
are 40,000,000X smaller
use 400,000X less power
are 120,000X lighter
AND…
are 1,300X more powerful.



EDVAC (1951)

- ENIAC team started discussing ***stored-program concept*** to speed up programming and simplify machine design

- Based on ideas by John von Nuemann & Herman Goldstine

- Still the basis for our general CPU architecture today

Commercial computers: BINAC (1949) and UNIVAC (1951) at **EMC**

- Eckert and Mauchly left academia and formed the Eckert-Mauchly Computer Corporation (EMC)

- World's first commercial computer was BINAC which didn't work…

- Second commercial computer was UNIVAC
  - Famously used to predict presidential election in 1952
  - Eventually 46 units sold at >$1M each

IBM 650 (1953)

- The first mass-produced computer
- Low-end system aimed at businesses rather than scientific enterprises
- Almost 2,000 produced

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

Improvements in C.A.

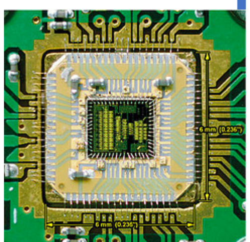
- IBM 650's instruction set architecture (ISA)
  - 44 instructions in base instruction set, expandable to 97 instructions

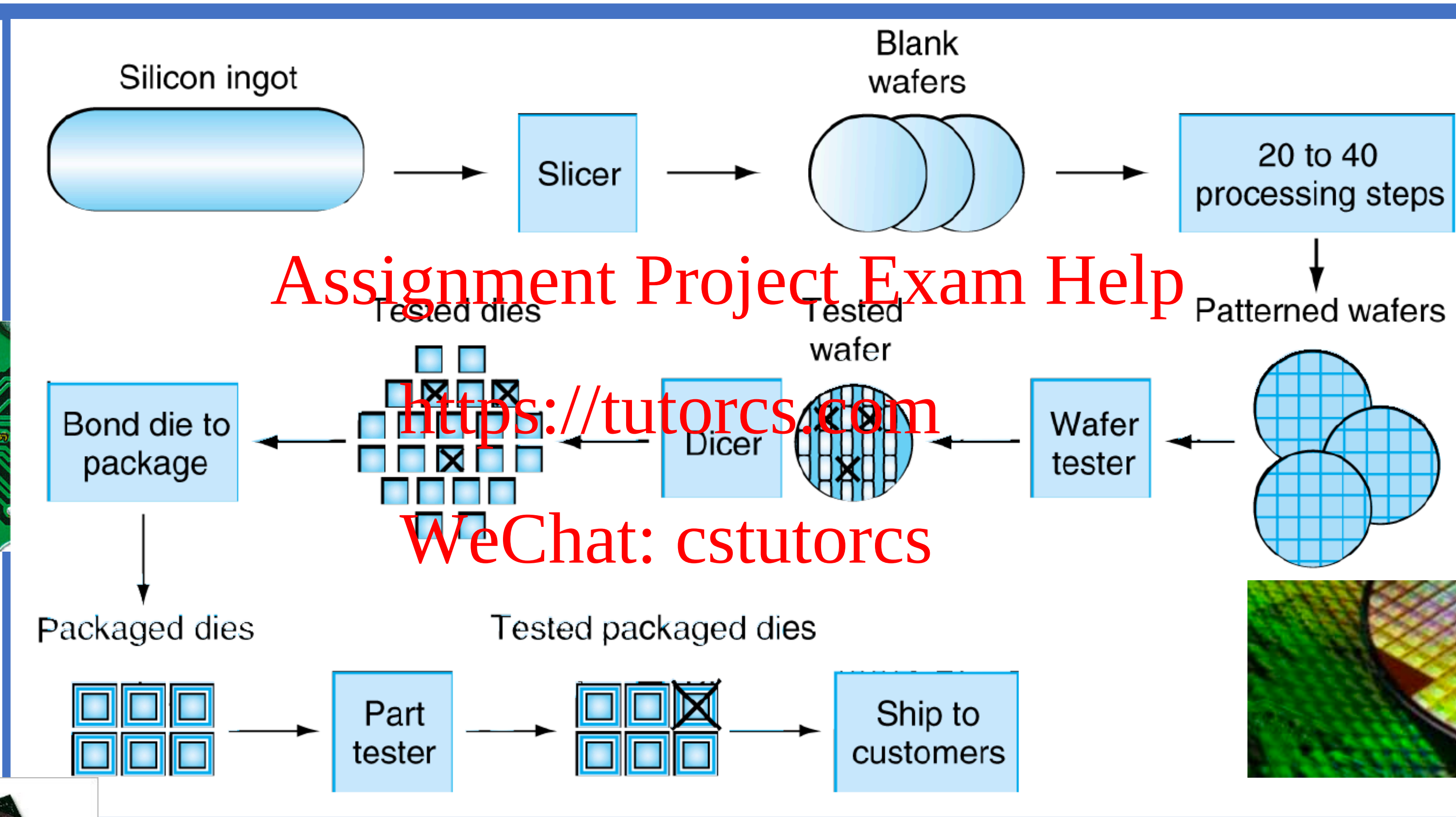
- Hiding instruction set completely from programmer using the concept of ***high-level languages*** like Fortran (1956), ALGOL (1958) and COBOL (1959)
  - Allowed the use of stack architecture, nested loops, recursive calls, interrupt handling, etc…

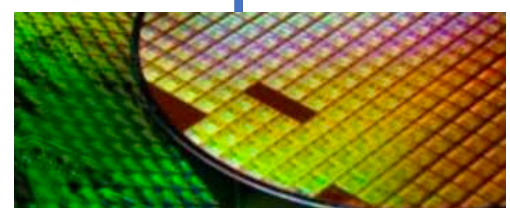
Adm. Grace Hopper (1906 – 1992), inventor of several High-level language concepts

[Public Domain, wikimedia]

Manufacturing ICs



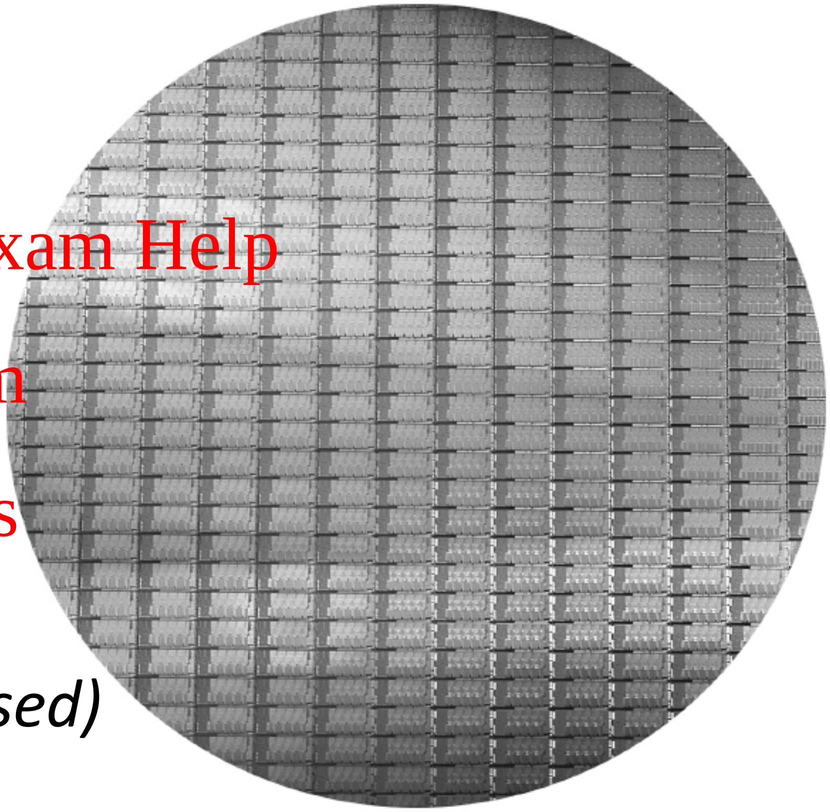




Yield: the proportion of working dies per wafer; often expressed as a number between 0 and 1

Example: Intel Core i7 Wafer

- 300mm (diameter) wafer
- 280 chips
- Each chip is 20.7 mm x 10.5 mm
- 32nm CMOS technology
  (the size of the smallest piece of logic and the type of Silicon semiconductor used)



Costs of Manufacturing ICs

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area}/\text{Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

$$Y = \frac{N_{\text{good}}}{N_{\text{total}}}$$

- Wafer cost and area are fixed
- Defect rate determined by manufacturing process
- Die area determined by architecture and circuit design

Examples

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area/Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area/2}))^2}$$

$$Y = \frac{N_{\text{good}}}{N_{\text{total}}}$$

A 300 mm wafer of silicon has 500 die on it, of which 100 are not working or malfunctioning. What is the yield of this wafer?

- $Y = N_{good}/N_{total} = 400/500 = 80\%$

If the wafer costs \$200, what is the cost per die?

- Cost per die = (\$200)/(500 * 0.8) = \$200/400 = \$0.50

A 300 mm wafer of silicon has N dies that are 0.5 mm x 1 mm each. What is N?

- Area of wafer/Area of each die

  $= (\pi * (300/2 * 10^{-3})^2) / (0.5 * 1 * 10^{-6}) = 141{,}370.605$

So, N = 141,370 (round down)

Response Time and Throughput

- Response time (aka Latency)
  - How long it takes to do a ***fixed task***

- Throughput

  - Total work done per a ***fixed time***

    e.g., tasks/transactions/… per hour

- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?

Latency vs. Throughput
Which is more important?

- They are different.
- It depends on what your goals are…
  - Scientific program? Latency
  - Web server? Throughput

- Example: Move people 10 miles
  - Via car: capacity = 5, speed = 60 mph
  - Via bus: capacity = 60, speed = 20 mph

  - Latency: **car = 10 minutes**, bus = 30 minutes
  - Throughput: car = 15 PPH, **bus = 60 PPH** *(consider round-trips)*

Performance Measures

- Execution Time: Total response time, including EVERYTHING
  - CPU time (processing), I/O use, OS overhead, any idle time
  - This determines **system performance**

- CPU time:
  - Time spent just processing a given job
    (discounts I/O time, OS time, etc…)
  - CPU time = *user* CPU time + *system* CPU time

- Define Performance = 1/Execution Time
- Relative performance
  - The performance of system A vs performance of system B, ie. $\mathbf{P_A / P_B}$

CPU Clocking

- Most digital hardware today operates to a **constant-rate clock**

- Clock **period**: *duration* of a clock cycle
  - e.g. 250 ps = 0.25 ns = 250 x 10^{-12} s

- Clock **frequency**: clock *rate* or *cycles per second*
  - e.g. 4.0 GHz = 4000 MHz = 4.0 x 10^{9} Hz

- Hertz (Hz) is "cycles per second", so

clock freq. = 1 / clock period

Useful Prefixes (Multipliers) to Know

Prefix	Symbol	Multiplier	In words…	Scientific Notation
Kilo	k	1,000	thousand	10^3
Mega	M	1,000,000	million	10^6
Giga	G	1,000,000,000	billion	10^9
Tera	T	1,000,000,000,000	trillion	10^{12}
Peta	P	1,000,000,000,000,000	quadrillion	10^{15}

Prefix	Symbol	Multiplier	In words…	Scientific Notation
milli	m	0.001	thousandth	10^{-3}
micro	μ	0.000001	millionth	10^{-6}
nano	n	0.000000001	billionth	10^{-9}
pico	p	0.000000000001	trillionth	10^{-12}

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$
$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance can be improved (i.e. make CPU Time **less**) by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

- Example: it took the CPU 1000 cycles to run the program. The clock cycle time (i.e. period) is 10 ns, so the CPU time is:
  1000 x 10 ns = 10000 ns = 10 μs, or **10 x 10^{-6} s**

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- **Instruction Count** for a program
  - Determined by program, ISA and compiler

- Average cycles per instruction (**CPI**)
  - Determined by CPU hardware
  - If different instructions have different CPI, then *Average CPI* is affected by instruction mix

- Example: *next slide*

CPI Example

- Computer A: Cycle Time = 250 ps, CPI = 2.0
- Computer B: Cycle Time = 500 ps, CPI = 1.2
- Same Instruction Set Architecture (ISA)

- **Which is faster?**

  - CPU Time = Instruction Count x CPI x Cycle Time
  - CPU_Time_A = NI x 2.0 x 250 x 10^{-12} s = **NI x 500 x 10^{-12} s**

  - CPU_Time_B = NI x 1.2 x 500 x 10^{-12} s = **NI x 600 x 10^{-12} s**
  - **So, CPU A is faster than CPU B**

- **By how much is it faster?**
  - Relative Performance = **NI x 600 x 10^{-12} s / NI x 500 x 10^{-12} s = 1.2**
  - **So, CPU A is 1.2 times faster than B (or you could say it's 20% faster)**

CPI Example using Weighted Classes

- An instruction class = instruction type
  - e.g. arithmetic type vs. branching type vs. jump type, etc…
- A CPU compiles code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5, so Clock Cycles = 2x1 + 1x2 + 2x3 = 10
- So, **Avg. CPI = 10/5 = 2.0**

- Sequence 2: IC = 6, so Clock Cycles = 4x1 + 1x2 + 1x3 = 9
- So, **Avg. CPI = 9/6 = 1.5**

Other Factors to CPU Performance: **Power Consumption**

Market trends DEMAND that power consumption of CPUs keep decreasing.

Power and Performance DON'T always go together…

- **Power = Capacitive Load x Voltage^2 x Clock Frequency**
- So:
  - Decreasing Voltage helps to get lower power, but it can make individual logic go slower!
  - Increasing clock frequency helps performance, but increases power!

- It's a dilemma that has contributed to Moore's Law "plateau"

YOUR TO-DOs for the Week

- **BRING YOUR MIPS REF CARDS TO CLASS!!!**

- Do your reading for next class (see syllabus)

- Finish up Assignment #1 for lab (***lab01***)

  - You have to submit it as a **PDF** using ***Gradescope***
  - Due on **Wednesday, 1/15, by 11:59:59 PM**

</LECTURE>