DM548

Computer Architecture & System Programming

Fall 2017

Lecture 1 - Introduction

Who is Talking?

- Richard Röttger
- from Munich

- Studied Computer Science at TUM
- PhD. from Max Planck Institute for Informatics
- Joined IMADA 2014

- Main Interests:
  - Bioinformatics
  - Regulatory networks
  - Unsupervised learning
  - Clustering …

Course Organization

- **Evaluation**
  - Written Exam at the end of the semester
  - 7 scale grading
- **Mandatory Projects**
  - Project in Assembly
  - Project in C
  - Evaluated pass/fail
- **You need to pass both projects in order to take the exam!**
- **We will have lab and exercises**

Introduction to the Course

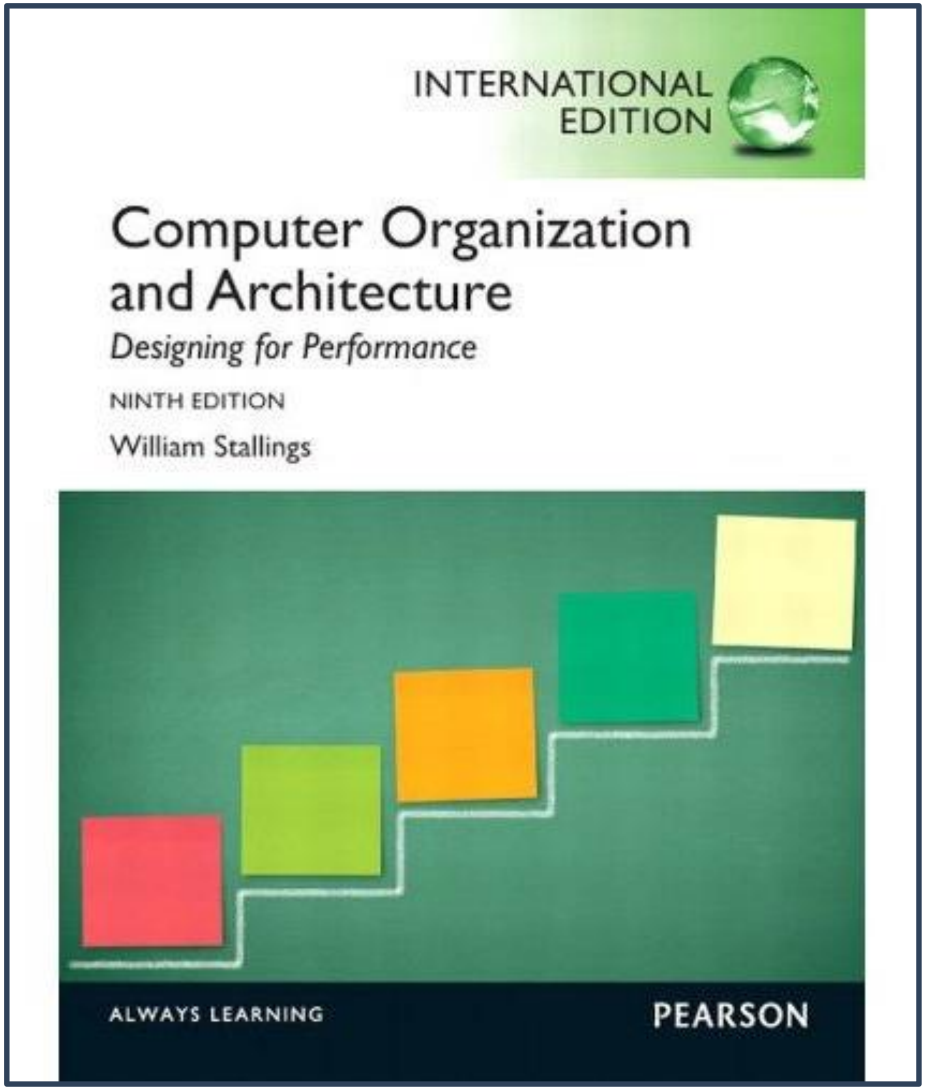
- Course web-page (there is a link on blackboard): http://imada.sdu.dk/~roettger/teaching/2017_fall_dm548.php

- We will have about 20 Lectures
- Slides will be available after each lecture
- All relevant course material will be available on the website

- TA: Caroline
- Contact: just email us
  - cknud14@student.sdu.dk

Book and Slides

- Lecture and slides are based on:
  - William Stallings: *Computer Organization and Architecture*, 9th edition, Pearson, 2013.
- Some slides are based on different courses and are indicated accordingly
- Slides will be available on the website after the lecture
  - **User: DM548**
  - **Pass: architecture**



Structure of the Course

- Fundamental Basics
- Assembly
- Classical Computer Architecture
- System Programming

Structure of the Course

- **Fundamental Basics**
  - Computer arithmetic
  - Floating point arithmetic
  - Boolean logic
- **Assembly**
- **Classical Computer Architecture**
- **System Programming**

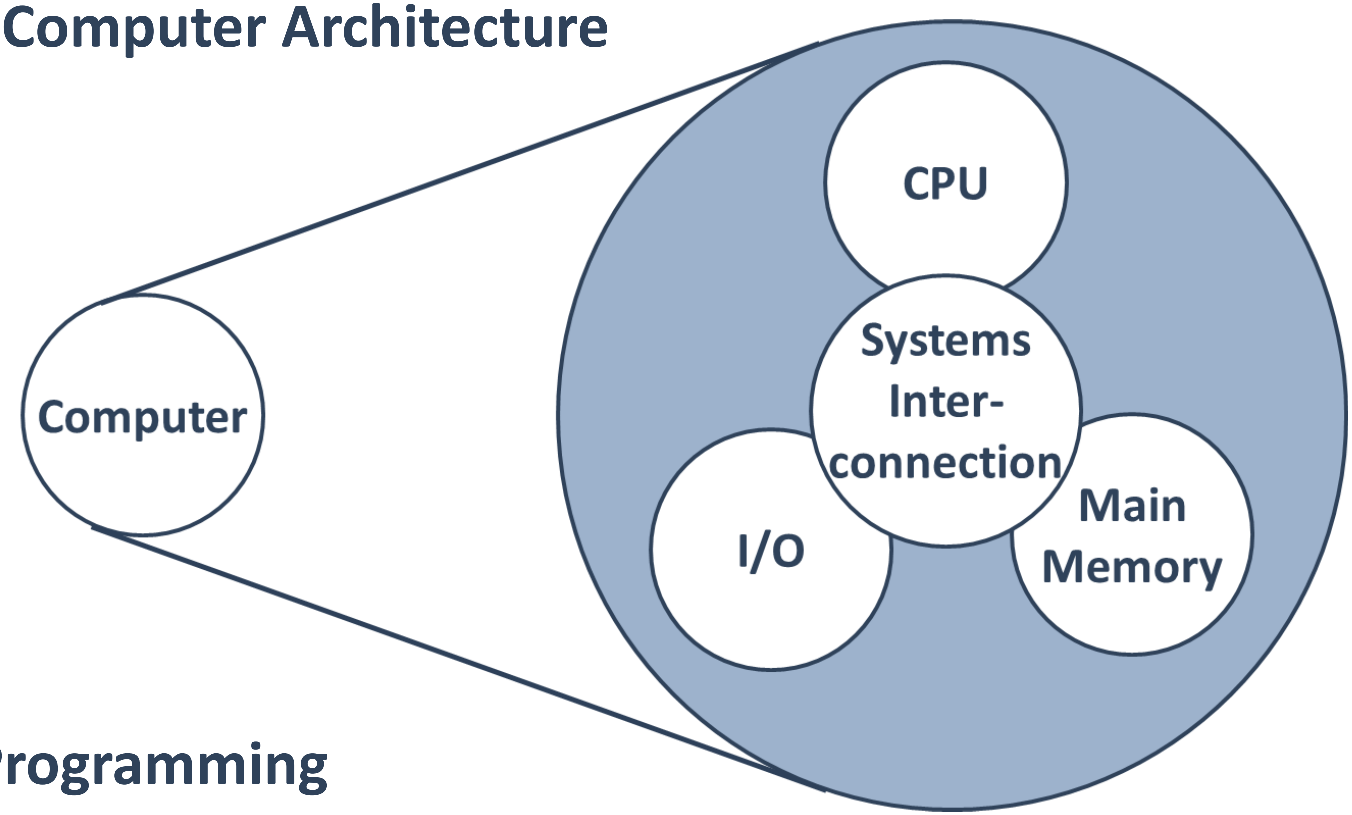
Structure of the Course

- **Fundamental Basics**
- **Assembly**
  - Very small and rather rough introduction to assembly
  - This is the first practical part of the course
  - Hands-on in the labs
  - We will write real x86_64 assembly code running on your linux machine, no toy language with a toy machine!

- **Classical Computer Architecture**
- **System Programming**

Structure of the Course

- Fundamental Basics
- Assembly
- Classical Computer Architecture

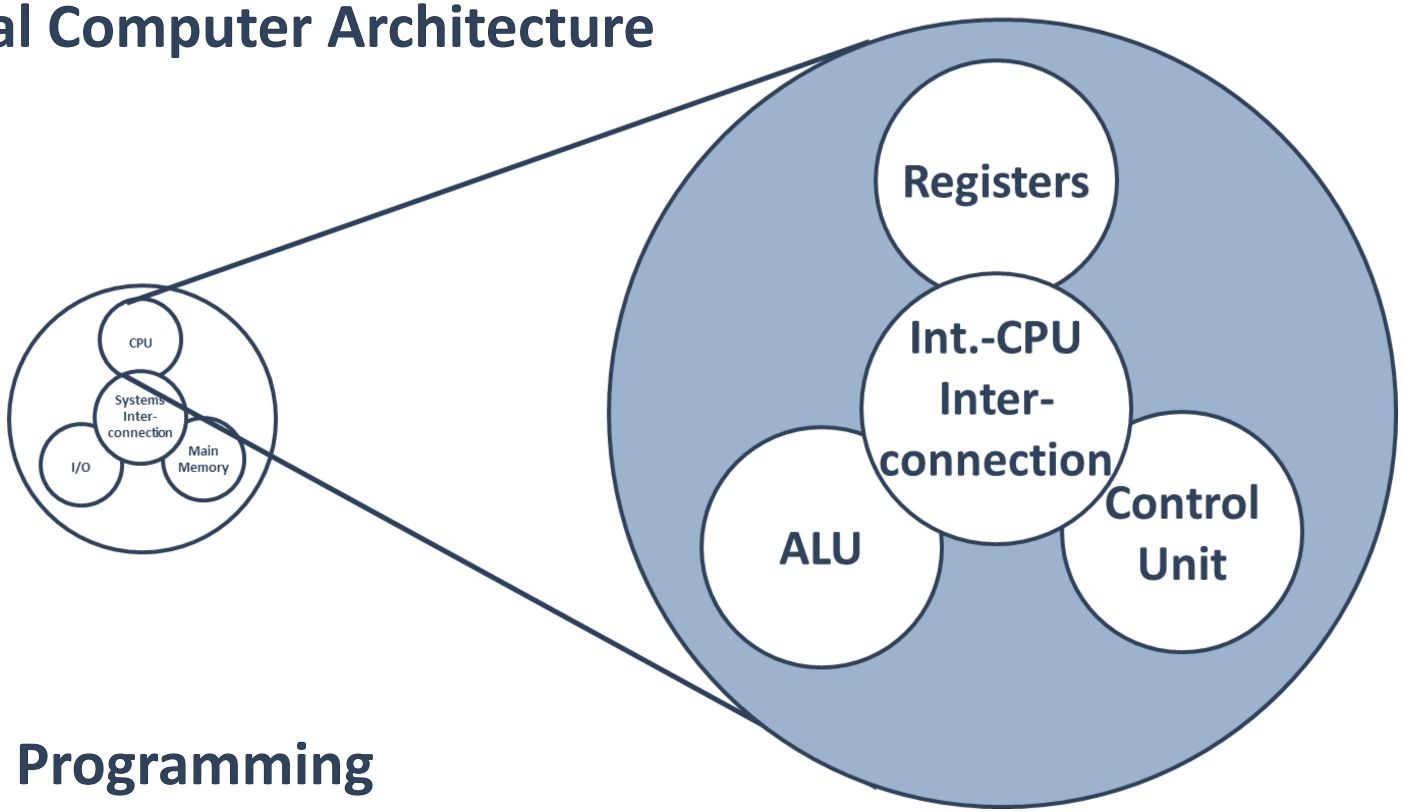


- System Programming

UNIVERSITY OF SOUTHERN DENMARK.DK

Structure of the Course

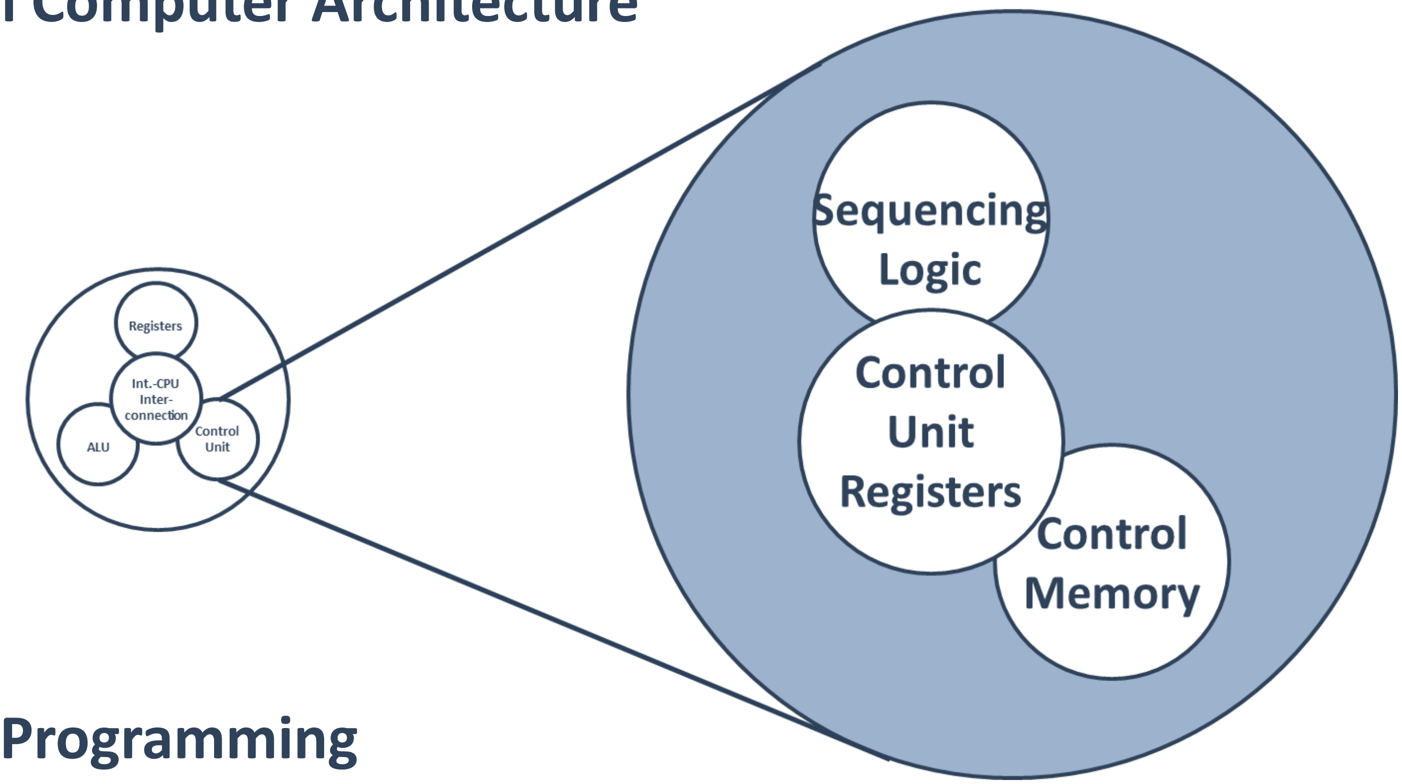
- Fundamental Basics
- Assembly
- Classical Computer Architecture



- System Programming

Structure of the Course

- Fundamental Basics
- Assembly
- Classical Computer Architecture



- System Programming

Structure of the Course

- **Fundamental Basics**
- **Assembly**
- **Classical Computer Architecture**
- **System Programming**
  - Last part of the course
  - We will have an introduction to C
  - Understanding the abstraction from assembly to higher programming languages

Some Remarks

- I will explain the theory in the lectures
- I am willing and happy to repeat parts if necessary
- You can always contact me or the TA with questions
- The Lab will help you tremendously to get through the assignments
- The Exercises will help you to be prepared for the exam

Introduction & Motivation

- **Why are we doing this?**
- **Computer History**
- **How to assess the performance of a computer?**

Why Computer Architecture?

- Obviously: We are computer scientists and should know what we are working with

- Most CS courses emphasize abstraction
  - Abstract data types
  - Asymptotic analysis
  - Why then bother with architecture details?

- Chances are, you'll never write assembly
  - Programs are too complex
  - Compilers are more efficient and can do nasty optimizations
  - Why then bother with assembly?

Facing the Reality: Arithmetic

- Addition is addition is addition, isn't?
- Common data types: `int`, `float`
- Is $x^2 \geq 0$?
  - Float: yepp
  - Ints: Not necessarily
    - $40{,}000 \cdot 40{,}000 = 1{,}600{,}000{,}000$
    - $50{,}000 \cdot 50{,}000 = ?$

- Is $(x + y) + z = x + (y + z)$?
  - Ints: yepp
  - Floats:
    - $(10^{20} - 10^{20}) + 3.14 = 3.14$
    - $10^{20} + (-10^{20} + 3.14) = ?$

➢ Slide is based on a slide from the lecture of Randal E. Bryant, David O'Hallaron and Andrew Tanenbaum used in last year's course.

Facing the Reality: Arithmetic

- **Good News:** It does not generate “random” values
  - They have provable mathematical properties

- **Bad News:** They are different from what we know from math
  - Due to finiteness of representations
  - Integers satisfy “Ring” properties
    - commutativity, associativity and distributivity
  - Floats only “ordering” properties
    - Monotonicity, values of signs

- **Impact:**
  - We need to understand which concepts apply in which context
  - Can lead to serious difficulties

➢ Slide is based on a slide from the lecture of Randal E. Bryant, David O’Hallaron and Andrew Tanenbaum used in last year’s course.

Facing the Reality: Patriot Missile Air Defense System

- Patriot missile system operating in Dhahran (Saudi Arabia) during the Gulf War in 1990’s
- 25th of February, 1991 the system failed to track and intercept an incoming Scud
- Scud hit army barracks, killing 28 U.S. soldiers and injuring another 98
- At the end, a software bug caused this failure

> http://sydney.edu.au/engineering/it/~alum/patriot_bug.html
> http://fas.org/spp/starwars/gao/im92026.htm

Facing the Reality: Patriot Background

- The Patriot is a surface-to-air defense missile system
- Designed to protect against Soviet cruise missiles and medium to high altitude aircraft
- These cruise missiles travel at speeds up to about MACH 2 (1500 mph / 2400 kmh)
- To avoid detection, it was designed to operate only a few hours, when needed

- http://sydney.edu.au/engineering/it/~alum/patriot_bug.html
- http://fas.org/spp/starwars/gao/im92026.htm

Facing the Reality: New Objective for Patriot

- Scud missiles fly at Mach 5 (3750 mph / 6000 kmh)
- Scuds fly short range
- No time to start-up the Patriot system only when needed
- Some Patriot batteries where running over 100 hours straight

- http://sydney.edu.au/engineering/it/~alum/patriot_bug.html
- http://fas.org/spp/starwars/gao/im92026.htm

Facing the Reality: The Patriot Software Bug

- Time is kept continuously by the system's internal clock in tenths of seconds
- Is expressed as an integer or whole number (e.g., 32, 33, 34...)
- To predict where the Scud will next appear, both time and velocity must be expressed as real numbers.
- But 1/10 has a non-terminating binary expansion, was chopped at 24 bits after the radix point
- The longer the system was running, the more significant the error became

- http://sydney.edu.au/engineering/it/~alum/patriot_bug.html
- http://fas.org/spp/starwars/gao/im92026.htm

Facing the Reality: The Patriot Software Bug

Figure 3: Correctly Calculated Range Gate

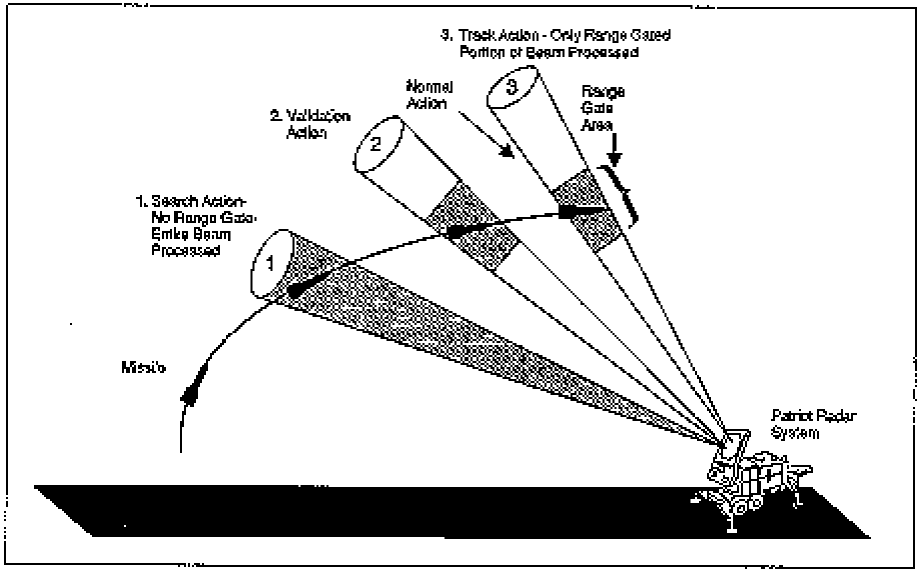
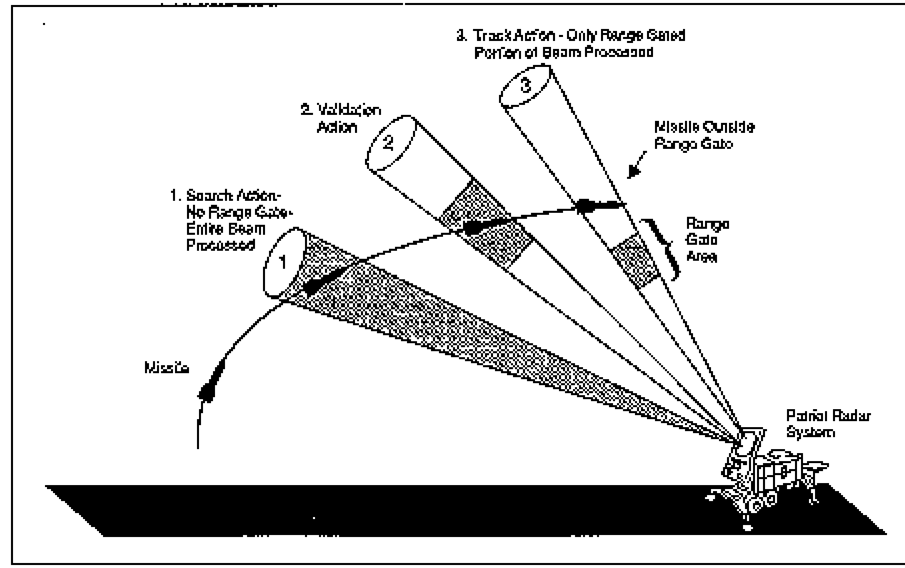


Figure 5: Incorrectly Calculated Range Gate



Hours	Seconds	Calculated Time (s)	Inaccuracy (s)	Shift in Range Gate (m)
0	0	0	0	0
1	3600	3599.9966	.0034	7
8	28800	8799.9725	.0250	55
20	72000	71999.9313	.0687	137
48	172800	172799.8352	.1648	330
72	259200	259199.7528	.2472	494
100	360000	359999.6667	.3433	687

- http://sydney.edu.au/engineering/it/~alum/patriot_bug.html
- http://fas.org/spp/starwars/gao/im92026.htm

Facing the Reality: Random Access Memory

- Memory is not unbounded
  - It must be allocated and managed
  - Many applications are memory dominated

- Memory referencing bugs are especially pernicious
  - Effects are distant in both time and space
  - Common security leak

- Memory performance is not uniform
  - Cache and virtual memory effects can greatly affect program performance
  - Adapting program to characteristics of memory system can lead to major speed improvements

Facing the Reality: System Performance Example

```c
void copyij(int src[2048][2048],
    int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

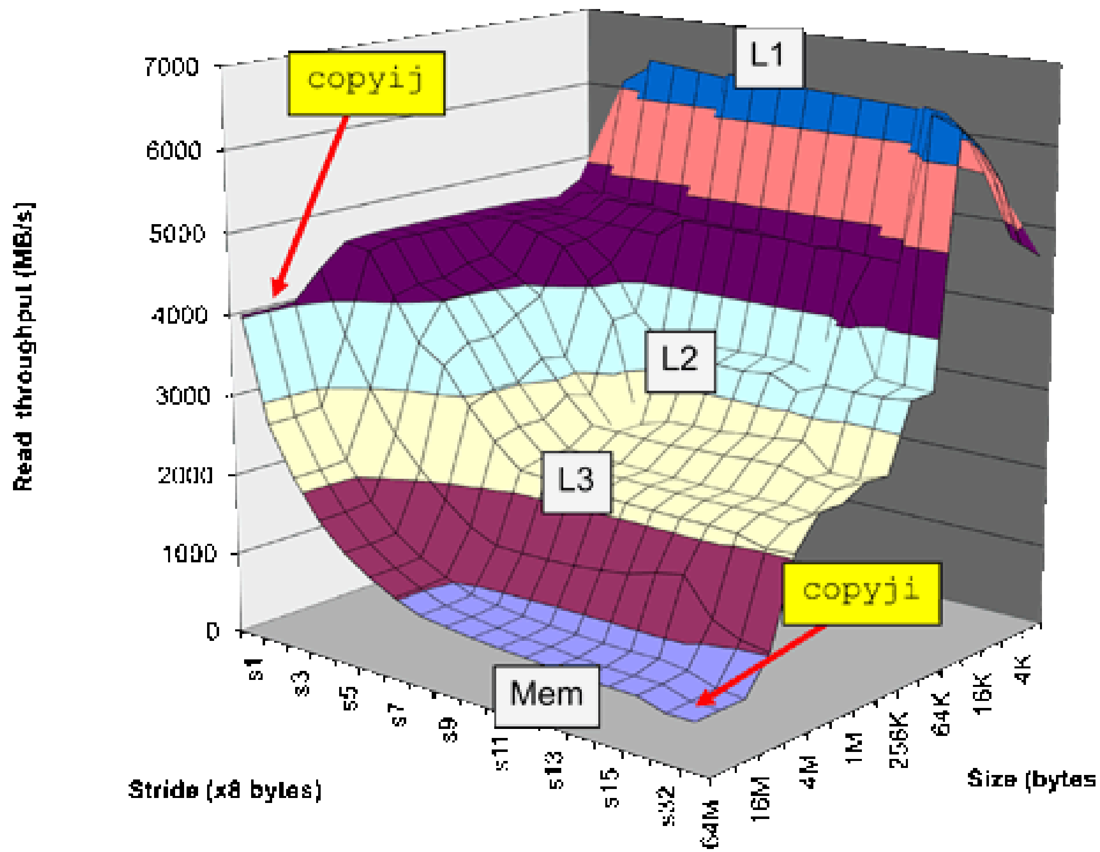
```c
void copyij(int src[2048][2048],
    int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

- The sequence of how the memory is accessed matters!
- The right example is **21 times slower** (Pentium 4)

- Generally, the efficient use of the Cache may speed-up programs significantly!

> Randal E. Bryant and David R. O'Hallaron : *Computer Systems: A Programmer's Perspective*

Facing the Reality: The Memory Mountain



➢ Randal E. Bryant and David R. O'Hallaron : *Computer Systems: A Programmer's Perspective*

Facing the Reality: Asymptotic Performance

- $O(n) \neq O(n)$
  - Constant factors matter

- Even exact Op-count does not predict performance
  - You can observe dramatic speed-ups depending how code is written
  - Multi-Level-Problem: Optimization of the algorithm, data structure, loop structure, …

- In order to write highly efficient code, you must know …
  - ... how programs are compiled
  - … how to measure and monitor program’s performance
  - … how to actually improve the code without raising other issues

Facing the Reality: Assembly matters

- Behavior of programs in presence of bugs
  - High-level language models may break down

- Tuning program performance

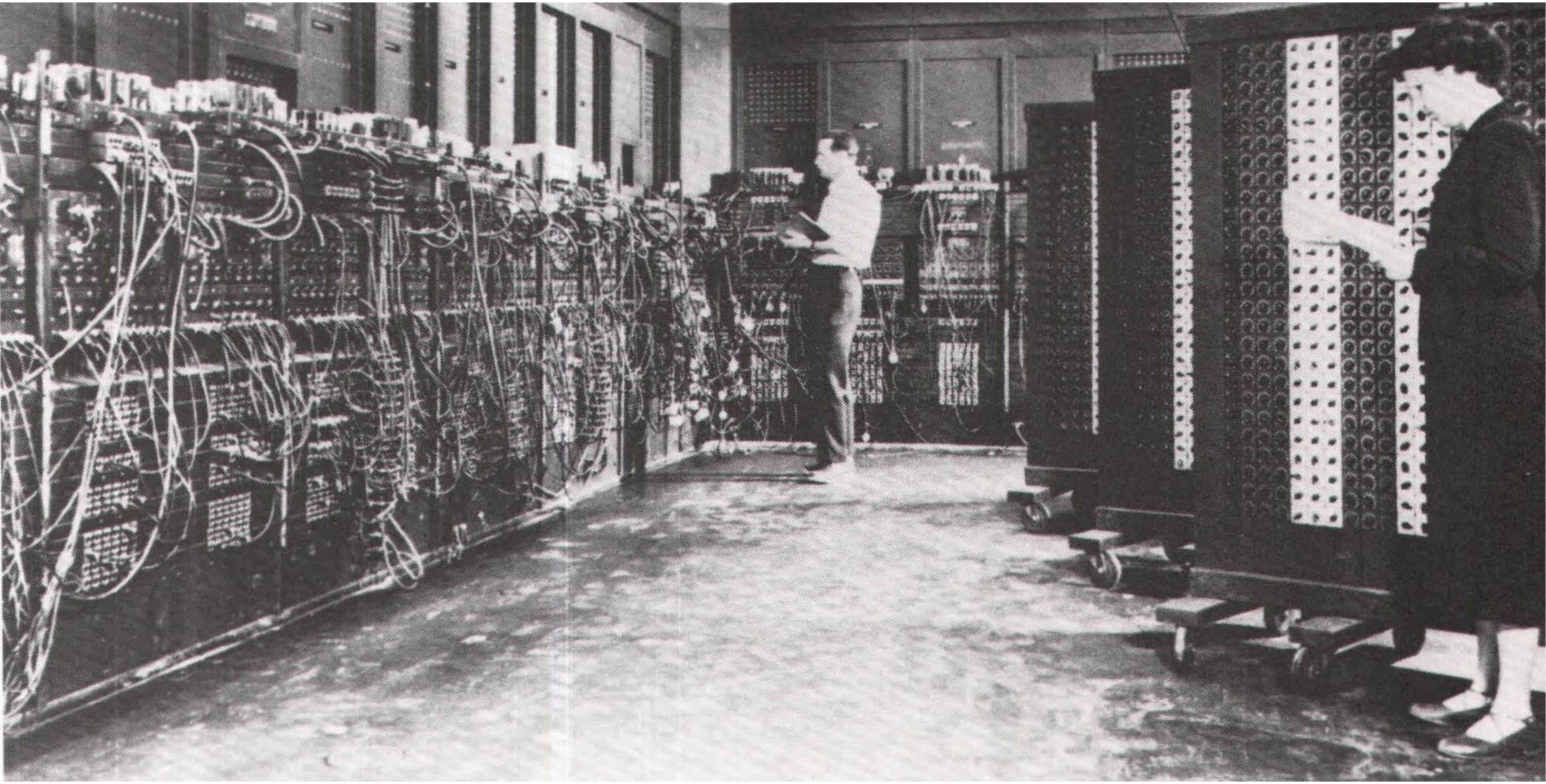
- Fighting malware, detecting security issues
  - x86 assembly is language of choice
  - E.g., nop sliding, breaking cryptography by counting cache misses

- Helps a lot in mastering other courses:
  - Operating Systems
  - Compiler

Introduction & Motivation

- Why are we doing this?
- Computer History
- How to assess the performance of a computer?

The ENIAC



➢ http://ds.haverford.edu/bitbybit/bit-by-bit-contents/chapter-four/4-8-project-px-and-the-eniac/

The ENIAC

- ENIAC – Electronic Numerical Integrator and Computer
- Design to calculate trajectory tables for artillery weapons

- Decimal System
- 17,468 vacuum tubes
- 70,000 resistors
- 10,000 capacitors
- 1,500 relays
- 6,000
- Eight feet high, eighty feet long, weighed thirty tons
- consumed 174,000 watts of power

➢ http://ds.haverford.edu/bitbybit/bit-by-bit-contents/chapter-four/4-8-project-px-and-the-eniac/

Operating the ENIAC

- “One-way ticket to the madhouse”
- The machine had 40 control panels
  - 9 basic units
  - 3 controlled the operations:
    - Initiating Unit started and stopped the machine
    - A master programmer orchestrated its overall activity
    - A cycling unit generated an internal drumbeat of 100,000 pulses a second.
  - 3 performed the arithmetic: a multiplier; a divider/ square-rooter
  - 20 accumulators
  - some more for I/O
- Programming: you set thousands of switches and plugged in hundreds of cables by hand, one at a time.
- It took about two days to set up ENIAC to carry out a program.

➢ http://ds.haverford.edu/bitbybit/bit-by-bit-contents/chapter-four/4-8-project-px-and-the-eniac/

Operating the ENIAC

- Again, it was designed for calculating trajectory tables
- Once, the calculation was set for one table, calculating another one was quite “fast” and “simple”

- No need for a convenient easy programming
- As the machine came too late for war, the usage for other tasks (actually, doing math for the hydrogen bomb) revealed those shortcomings

=> Need for a more convenient programming

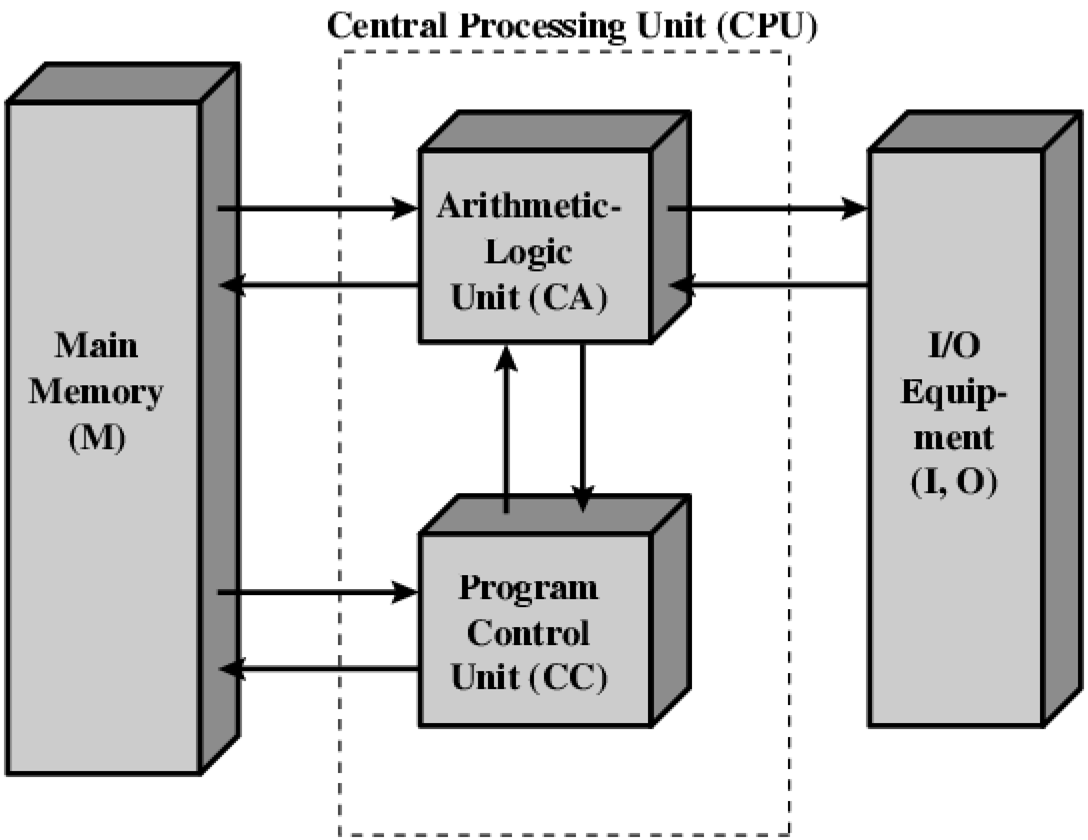
➢ http://ds.haverford.edu/bitbybit/bit-by-bit-contents/chapter-four/4-8-project-px-and-the-eniac/

von Neumann / IAS

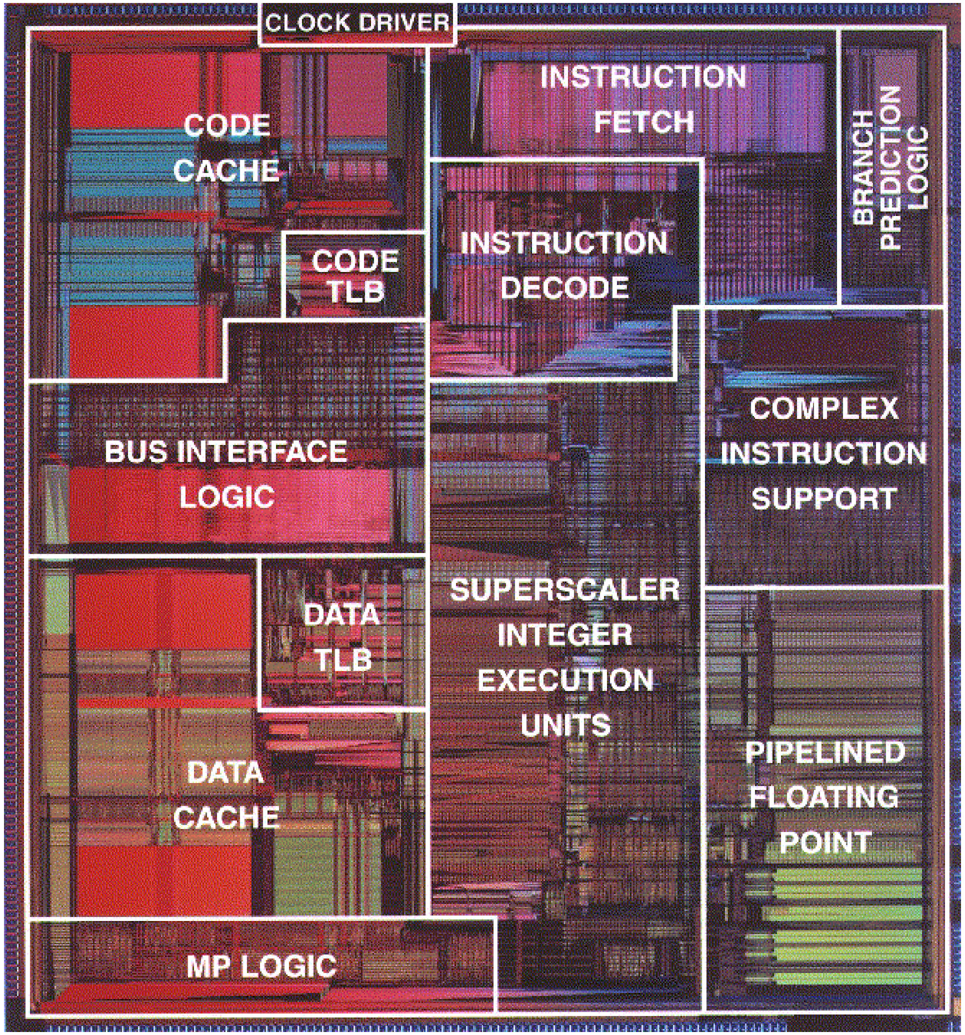
- 1945 von Neumann proposed the design for a new computer, the EDVAC
- Princeton Institute for Advanced Studies (IAS)
- "**stored-program concept**"
- Basically all computers follow this concept now

- Consisted of four parts:
  - **Main Memory** stores both data and instructions
  - **Arithmetic and Logical Unit (ALU)** capable of operating on binary data
  - **Control Unit** interprets the instructions and causes them to be executed
  - **I/O** operated by the control unit

von Neumann / IAS

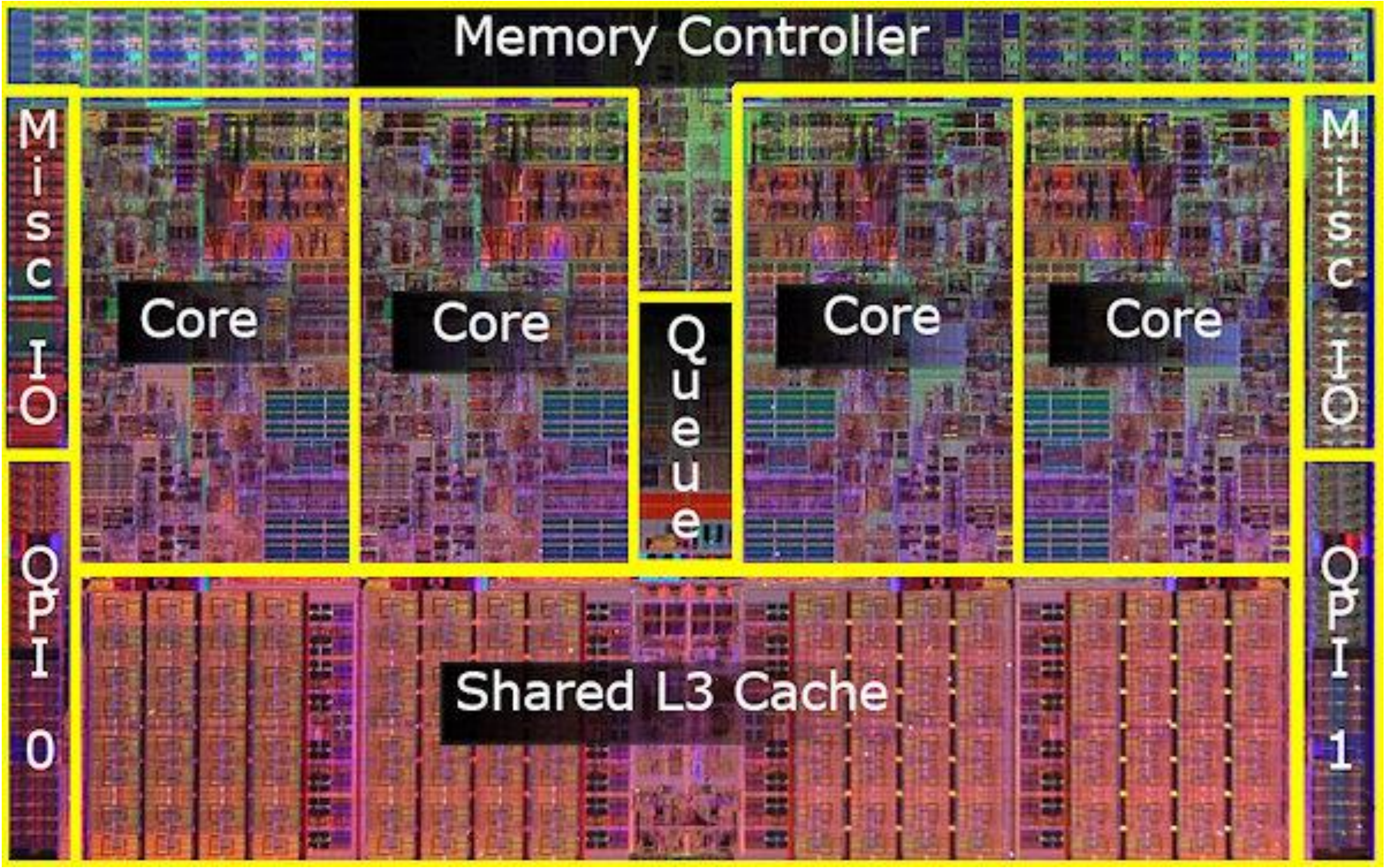


Pentium Pro vs. von Neumann



- http://web.eecs.umich.edu/~bartlett/w99si-cpu.html

Intel Core i7



Milestones in Computer History Overview

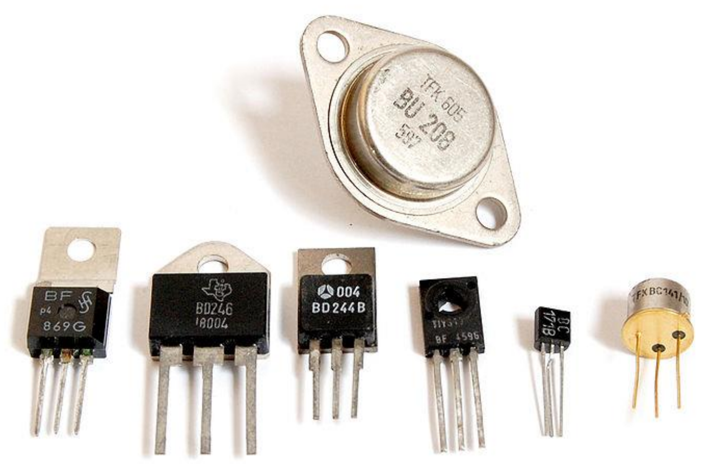
- A computer generation emerges when the fundamental hardware technology changes

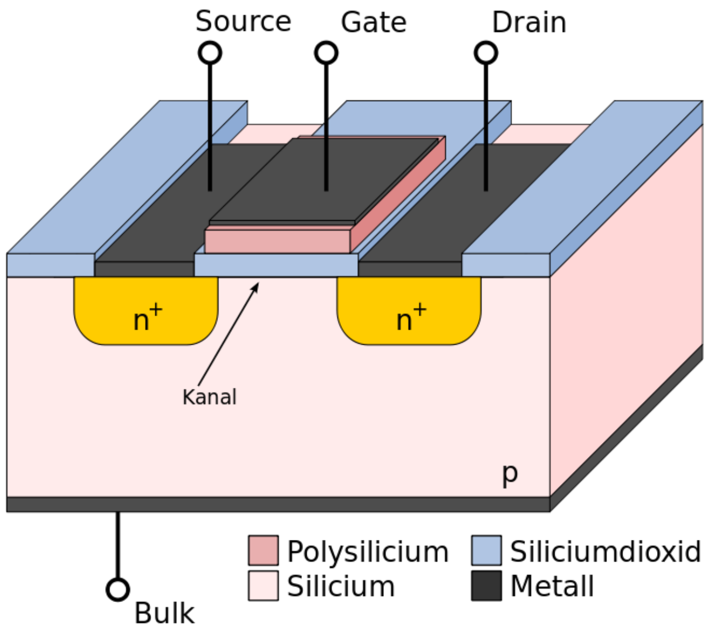
Generation	~Dates	Key Technology	Typical Speed (in Ops)
1	1946-1957	Vacuum Tubes	40,000
2	1958-1964	Transistors	200,000
3	1965-1971	Small- and medium-scale integration	1,000,000
4	1972-1977	Large-scale integration	10,000,000
5	1978-1991	Very-large-scale integration	100,000,000
6	1991-	Ultra-large-scale integration	1,000,000,000

- Integration means that the transistors are integrated in a silicon chip and not on a circuit board
- You can probably imagine, that after the 3rd generation there is room for interpretation

Milestones in Computer History
1947

- The transistor was invented at Bell Labs.
- It is smaller, cheaper, and dissipates less heat than vacuum tubes
- Took a couple of years until commercialized

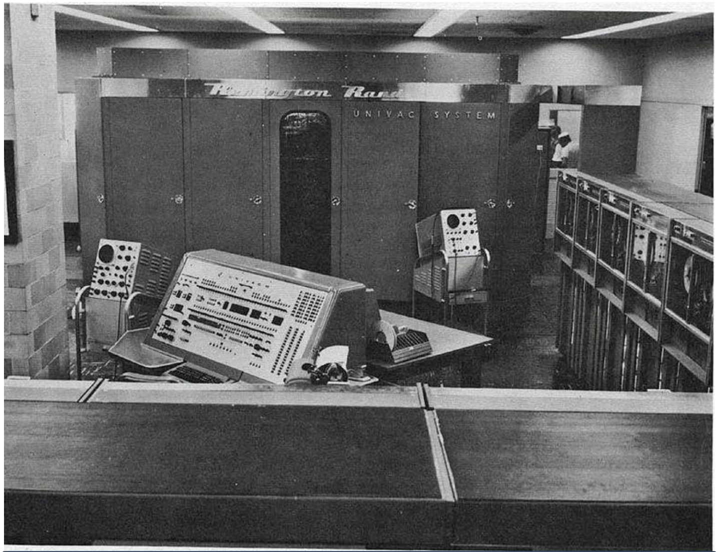




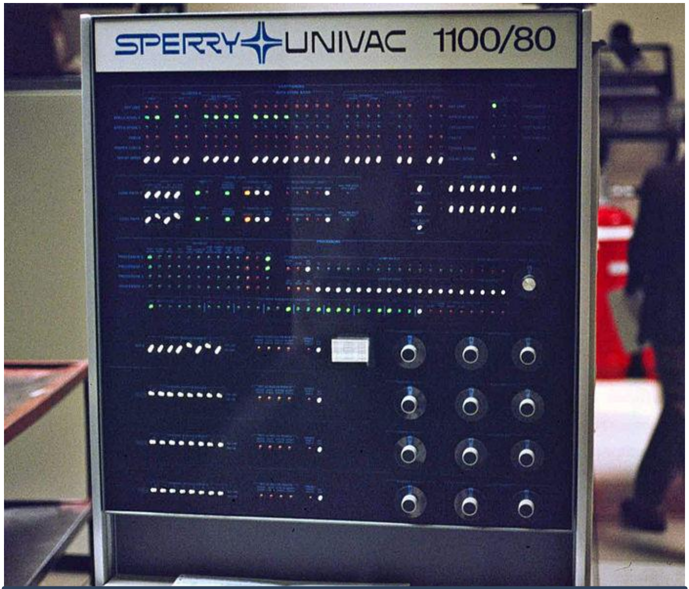
➢ Images: http://www.wikipedia.de

Milestones in Computer History
1951

- First commercially successful computer: UNIVAC I (UNIVersal Automatic Computer)
- Introduced by Remington Rand.
- Publicity Stunt: UNIVAC correctly predicted that Dwight D. Eisenhower will win the presidential election.



UNIVAC II



UNIVAC 1100/80

➢ Images: http://www.wikipedia.org

Milestones in Computer History 1952-1964

- IBM built the 700/7000 series mainframes
- 700 series still used vacuum tubes, the 7000 series was transistorized
- These machines made IBM to the market leader for business computers

IBM 702 System

IBM 7090 System

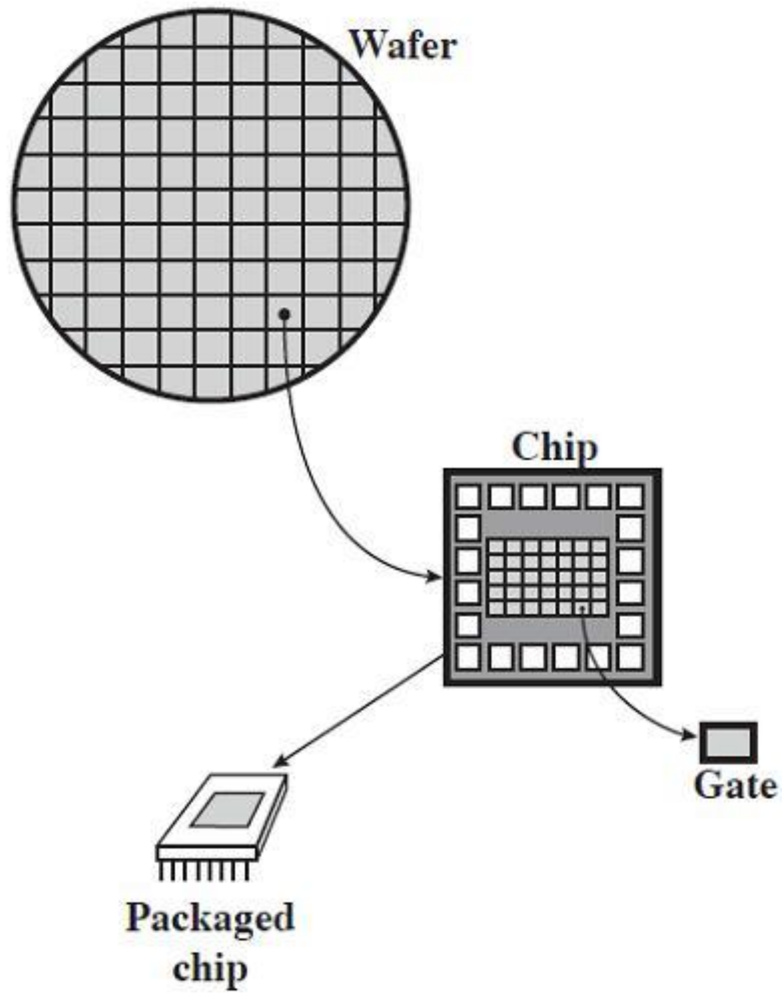
➢ Images: http://www.wikipedia.de

Milestones in Computer History
The Third Generation

- Computers contained up to 10,000 transistors
- Back then, the transistor was a discrete component in its own package
- Integrated Circuits (IC) packed a couple of those into one chip
- Those were connected on circuit boards

- Important Computers of this generation:
  - IBM System/360
  - DEC PDP-8

Milestones in Computer History
From Wafer to Packed Chip



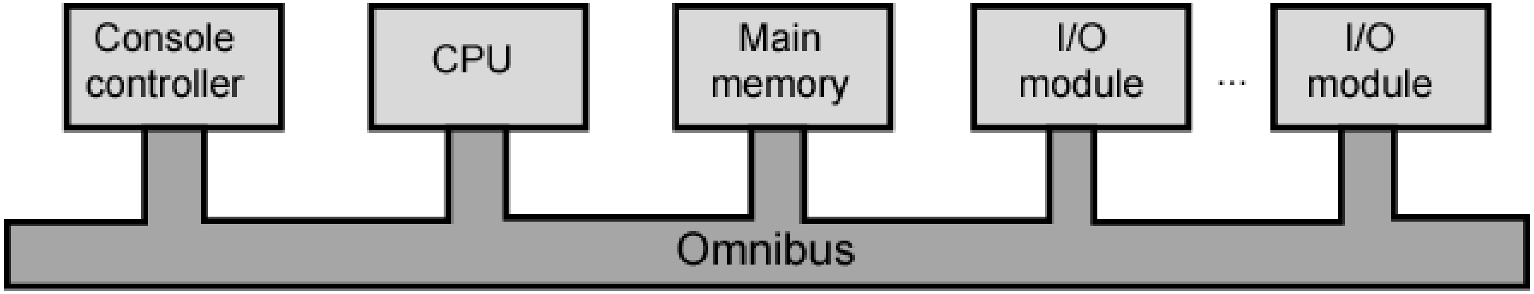
Milestones in Computer History
1964 IBM System/360

- First Computer Family, i.e., different variants
- All variants shared (more or less)
  - Identical Instruction Set
  - Operating System
- Enabled an “upgrade” of hardware without changing the software.
- Cemented IBMs market leadership (~70%)
- Todays IBM Mainframes still follow the 360 architecture

Milestones in Computer History
1964 DEC’s PDP-8

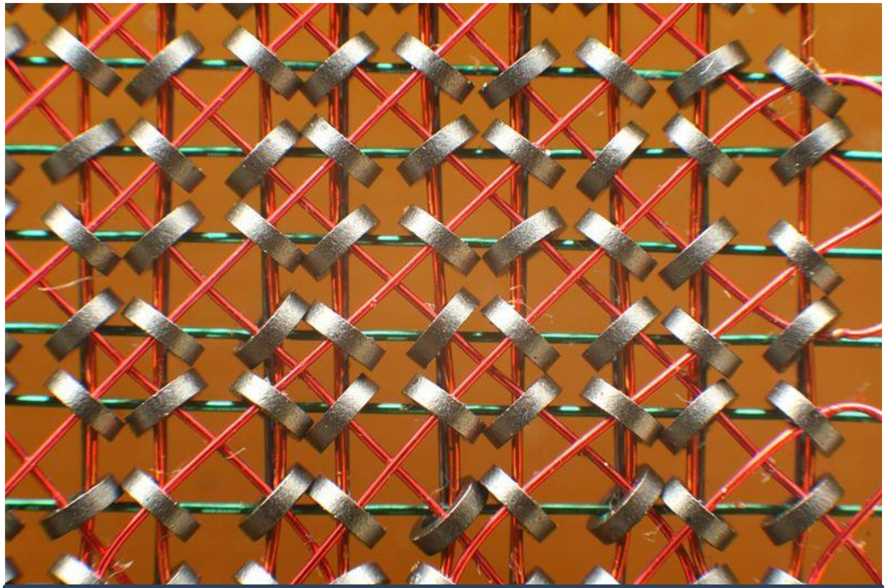
- First “small” computer, only $16,000
- Didn’t require air condition, could be placed on a desk
- Other manufacturers bought the PDP-8 and integrated it into a total system for resale
  - Original Equipment Manufacturers (OEMs)
- Important deviation from the von Neumann model:
  - The bus structure
  - Called Omnibus by DEC

Milestones in Computer History
The Omnibus

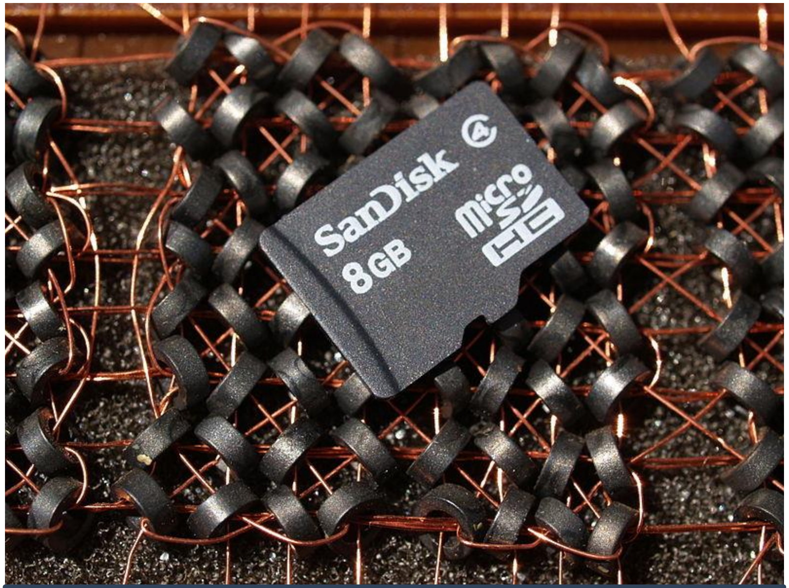


- 96 separate signal paths carry address, data and control signals
- Easy to plug new modules into bus
- Bus is controlled by CPU
- Now virtually universal in microcomputers

Milestones in Computer History
Core Memory



Close-up of a core plane. The distance between the rings is roughly 1 mm (0.04 in).



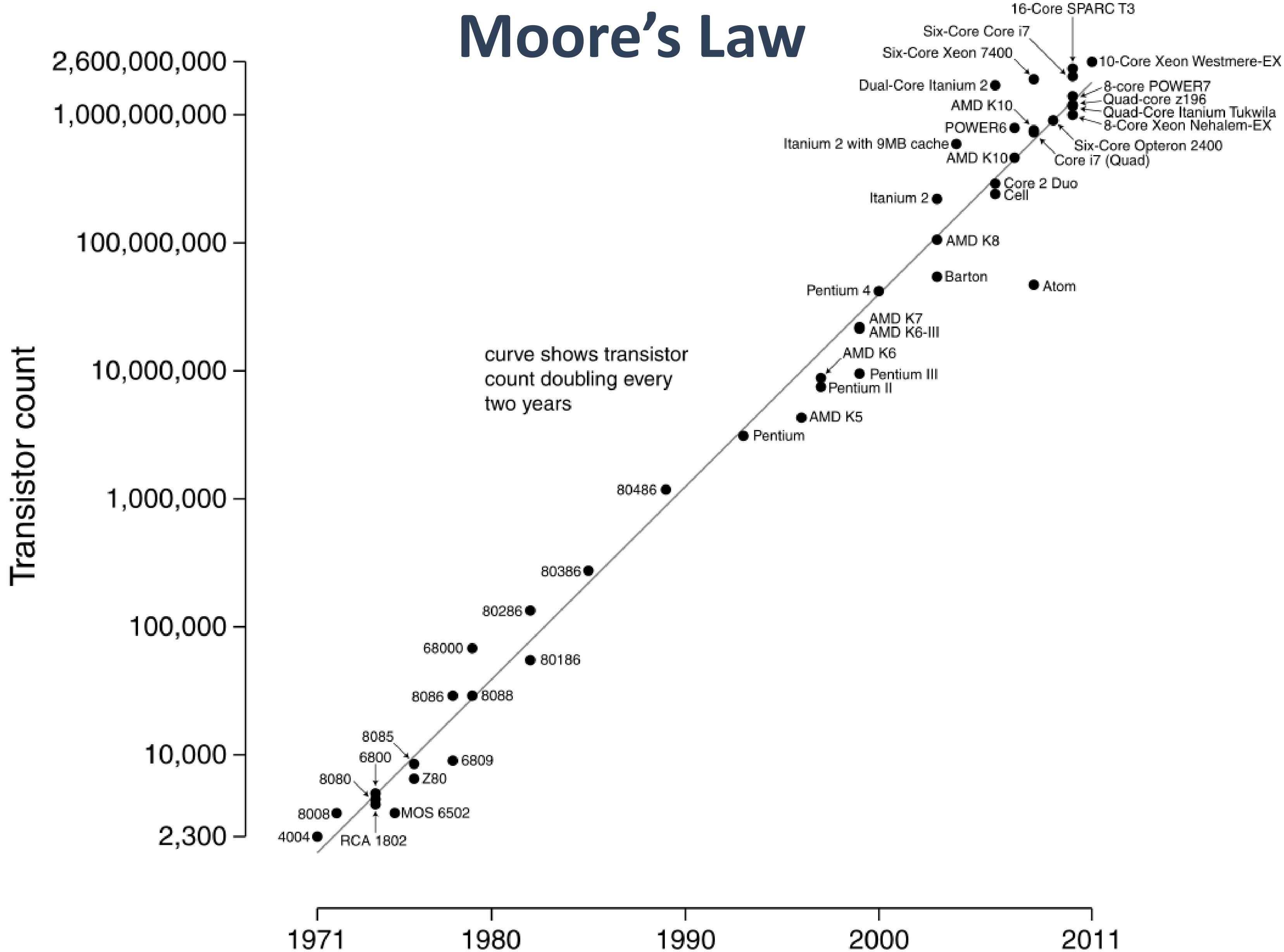
8 Byte vs. 8GB

- www.wikipedia.org

Milestones in Computer History
Moore's Law

- In 1965, the rapid development of ICs lead Intel's Co-founder Gordon Moore to the prediction that the number of transistors doubles every year.
- To the surprise of many, Moore was right (it's every 18 month, though)
- Impacts:
  - The prize for a chip remained unchanged
  - The closer packaging resulted in shorter path length which in turn means higher operating speed
  - The computer becomes smaller
  - Reduction in power and cooling costs
  - Chip connections are more reliable than solder connections. More components on a chip, less solder connections.

Milestones in Computer History
Moore's Law



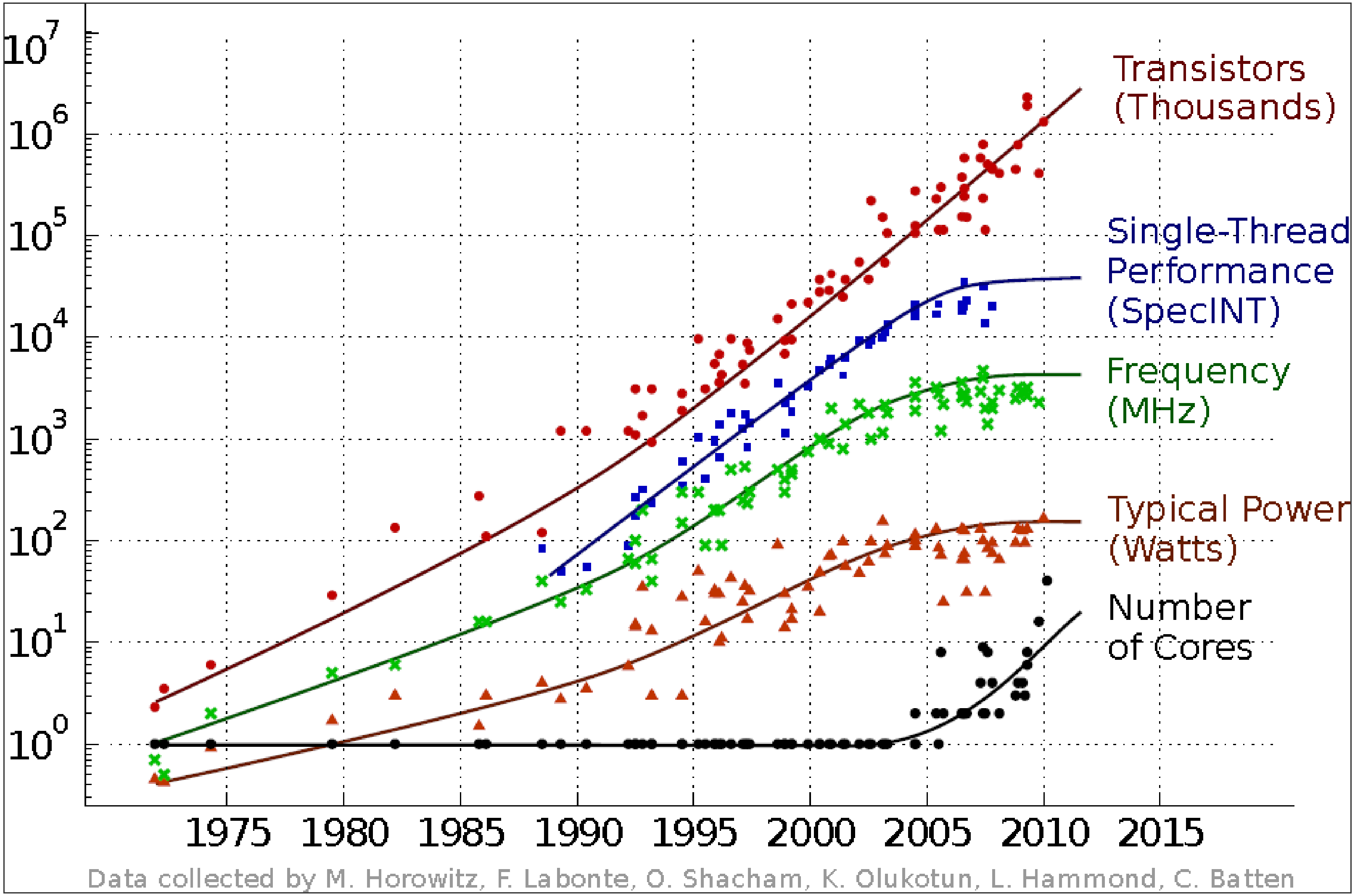
Milestones in Computer History
Intel Processors

Architectures	Processors
X86-16	8086
	286
X86-32/IA32	386
	486
	Pentium
MMX	Pentium MMX
SSE	Pentium III
SSE2	Pentium 4
SSE3	Pentium 4E
X86-64 / EM64t	Pentium 4F
	Core 2 Duo
SSE4	Core i7

time

- Slide is based on a slide from the lecture of Randal E. Bryant, David O'Hallaron and Andrew Tanenbaum used in last year's course.

Moore's Law Revisited



- http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/

Introduction & Motivation

- Why are we doing this?
- Computer History
- How to assess the performance of a computer?

How to Assess a CPU?

- We have seen, there is lots more to a processor than clock speed.
- What do we have to consider when assessing a CPU
  - Performance
  - Instruction Set
  - Power consumption / Heat emission
  - Package Size
  - Security options
  - Reliability
  - …

How to Assess Performance of a CPU

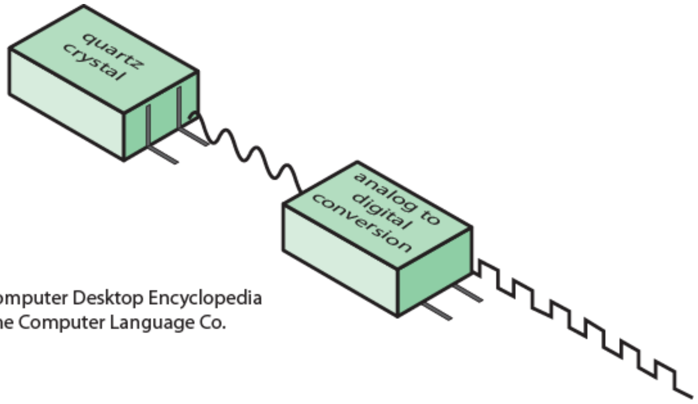
- There exists several characteristic numbers:
  - Clock Speed
  - Millions of instructions per second (MIPS)
  - Millions of floating point instructions per second (MFLOPS)

- These numbers normally give the theoretical maximum, not achieved in practice

- Benchmarks:
  - More realistic
  - Depends on the quality of the compiler

Clock Speed

- Every step in a CPU is governed by a system clock
- Generated by a quartz crystal, conversed into a digital signal
- Typically, operations start with the begin of the pulse
- With some exceptions, the speed of each elementary operation cannot be faster than one tick
- The physical layout limits the clock speed:
  - It takes a finite amount of time for voltage levels to settle
  - Depending on the inner-CPU line lengths only a certain maximal clock speed can be reached



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

What Can be Done Within One Tick? Let's calculate

- Most operations require more than one cycle some up to couple of dozen cycles
- Depends on the CPU (e.g., pipelining) and the program and the compiler, how many instructions per second can be executed

- Let f be the frequency, $\tau = {}^{1}/_{f}$ the cycle time
- Let I_c be the instruction count for a program and I_i the instruction count of type i
- An important measure is the cycles per instruction (CPI)
- CPI is dependent of the type i thus we have CPI_i

What Can be Done Within One Tick?
Let's calculate

- We now can define the cycles per instruction (CPI) as

$$CPI = \frac{\sum_{i=0}^{n}(CPI_i \cdot I_i)}{I_c}$$

- Time to execute a given program is

$$T = I_c \cdot CPI \cdot \tau$$

- It's not that easy:
  - We have m memory accesses
  - k is the ratio between the memory cycle time and the CPU cycle time
  - Let p the number of cycles the processor need to decode an instruction

- Then we already end up at

$$T = I_c \cdot [p + (m \cdot k)] \cdot \tau$$

Are we done yet?

	I_c	p	m	k	τ
Instruction Set Architecture	x	x			
Compiler	x	x	x		
Processor Implementation		x			x
Cache and Memory hierarchy				x	x

- Still not all performance factors accounted for
- Should demonstrate, assessing a processor is not that trivial
- A more common measure is the MIPS and MFLOPS rate for a given program:

$$\text{MIPS} = \frac{I_c}{T \cdot 10^6} = \frac{f}{CPI \cdot 10^6}$$

$$\text{MFLOPS} = \frac{\text{Number of float-ops}}{T \cdot 10^6}$$

Still Depends on the ISA

- Example: $A = B + C$
- All machines require the same time to execute

- CISC machine (1 MIPS):

```
add     mem(b),mem(C),mem(A)
```

- RISC machine (4 MIPS):

```
load    R1, B
load    R2, C
add     R3, R2, R1
store   A, R3
```

Who should be the winner?

Benchmarks

- Introduction of Benchmarks
  - Written in high-level language
  - Representative for a certain kind of application
  - Can be measured easily
  - Should be widely distributed
- Example SPEC Benchmarks (Systems Performance Evaluation Corporation)
  - SPEC CPU2006
    - processor intensive applications
    - 17 FP programs (written in C, C++ and Fortran) and 12 integer programs (C, C++)
  - SPECjbb2000
    - Benchmark for evaluating the performance of the machine for JAVA business servers
  - …

➢ Further reading: http://www.spec.org/benchmarks.html

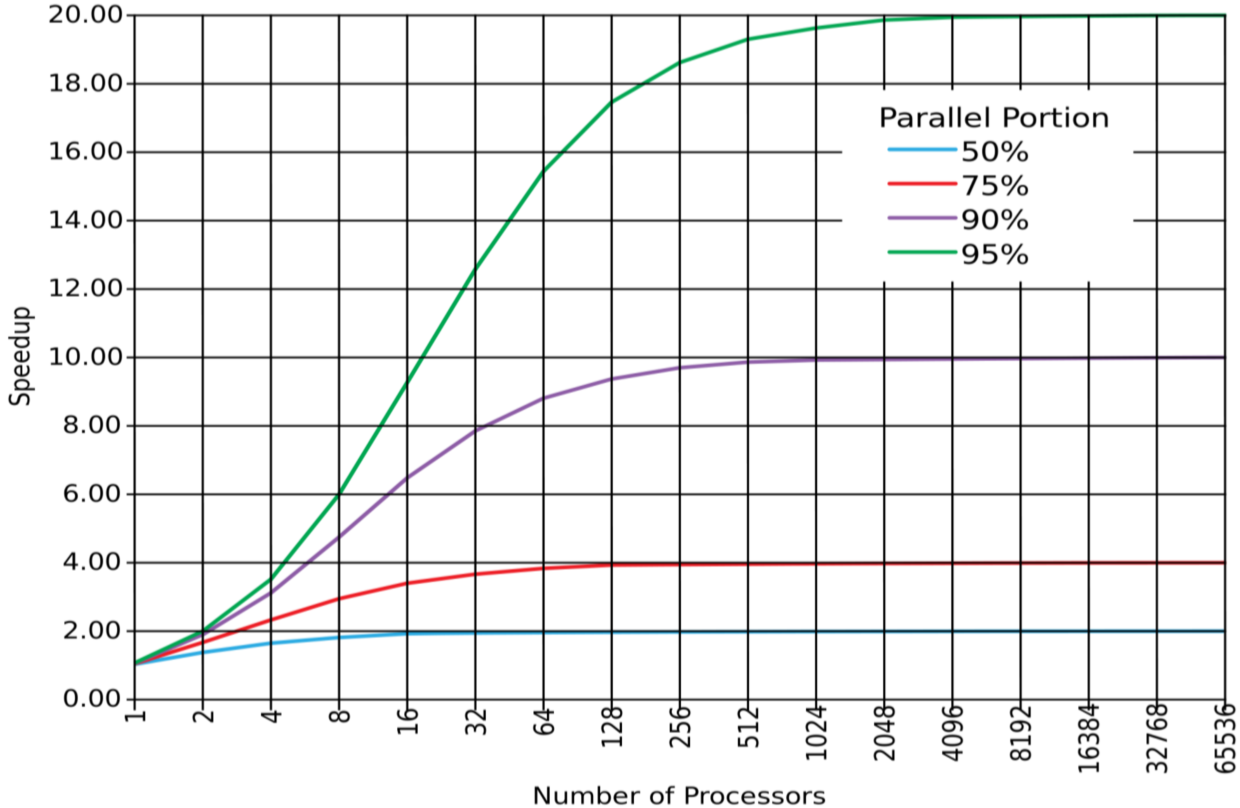
Amdahl's Law

- Gene Amdahl's law deals with the potential speed-up of a program when using multiple processors
- Let f be the part of the program which is infinitely parallelizable, T is the execution time on one processor

$$\text{Speed–Up} = \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

- In other words: Divide the single-CPU time by the time with multiple CPUs.

Amdahl's Law



- When f is small, little sped-up
- For $N \rightarrow \infty$ the speed-up is bound by $\frac{1}{(1-f)}$

➢ http://thedailyomnivore.net/2012/03/13/amdahls-law/

To End the History Part

- *"Computers in the future may weigh no more than one-and-a-half tons."*
  - Popular Mechanics, 1949
- *"I think there is a world market for maybe five computers."*
  - Thomas Watson, Chairman of IBM, 1943
- *"I can assure you that data processing is a fad that won't last the year."*
  - Chief Business Editor, Prentice Hall, 1957
- *"Yeah, microchips, but what... is it good for?"*
  - an IBM senior engineer, 1968
- *"There is no reason anyone in the right state of mind will want a computer in their home."*

  - Ken Olson, President of Digital Equipment Corp, 1977.
- *"640k is enough for anyone, and by the way, what's a network?"*
  - William Gates III, President of Microsoft Corporation, 1984.
- *"Linux is not portable."*
  - Linus Torvalds.