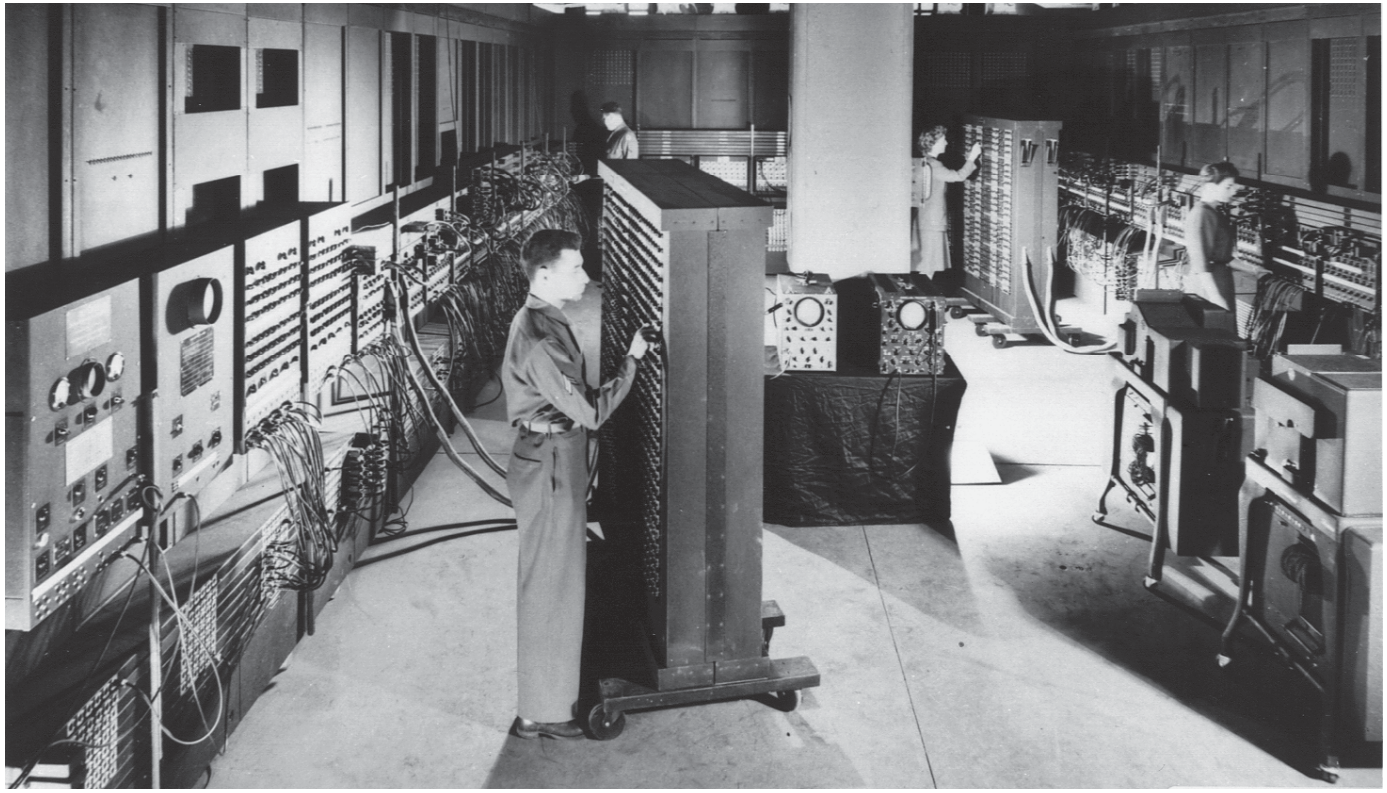
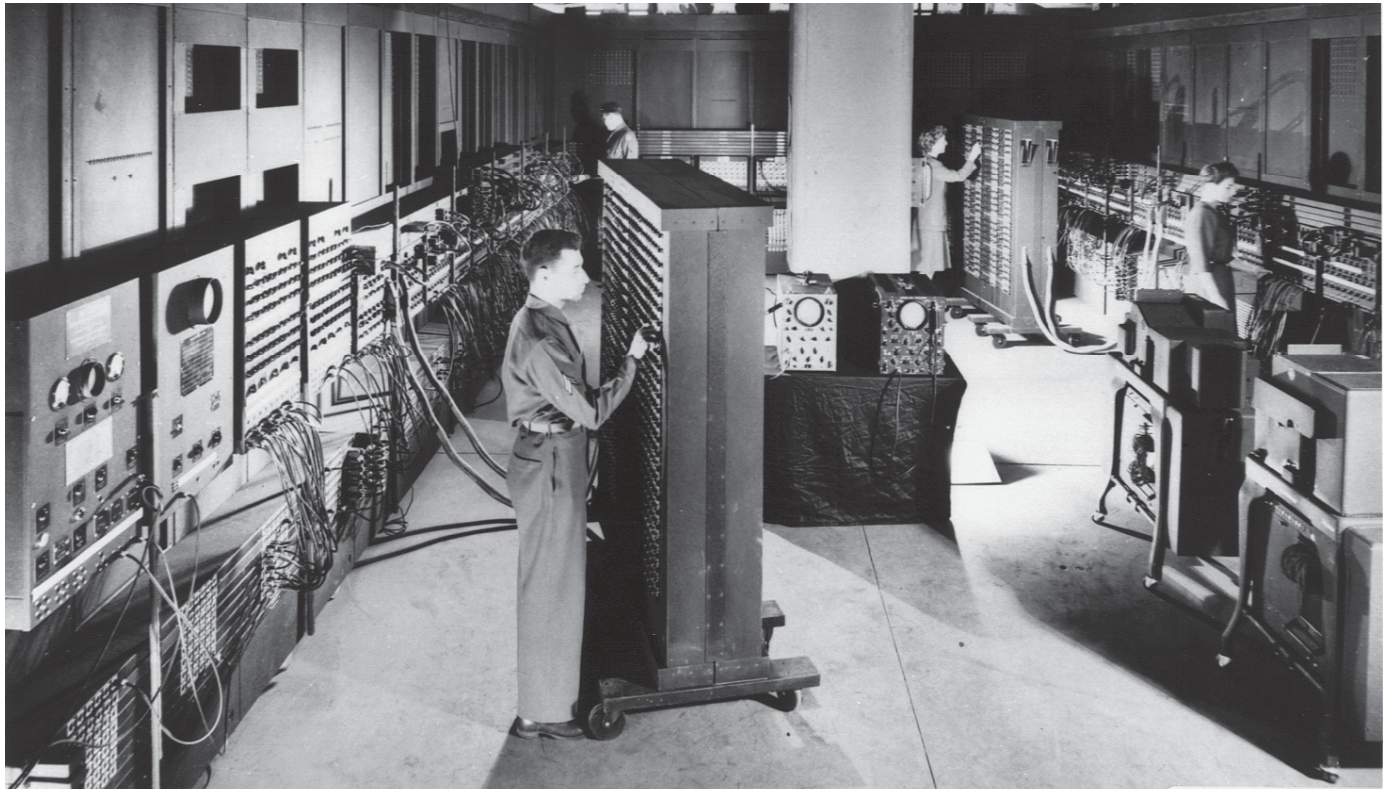
Computer Architecture

Unmesh D. Bordoloi



The first electronic computers



ENIAC (Electronic Numerical Integrator and Calculator) –
The world's first electronic computer

ENIAC

- ENIAC provided conditional jumps and was programmable, clearly distinguishing it from earlier calculators

- Programming was done manually by plugging cables and setting switches, and data was entered on punched cards. Programming for typical calculations required from half an hour to a whole day

- ENIAC was a general-purpose machine, limited primarily by a small amount of storage and tedious programming

ENIAC

- J. Presper Eckert and John Mauchly at the Moore School of the University of Pennsylvania
- Funded by the United States Army
- Became operational during World War II but was not publicly disclosed until 1946

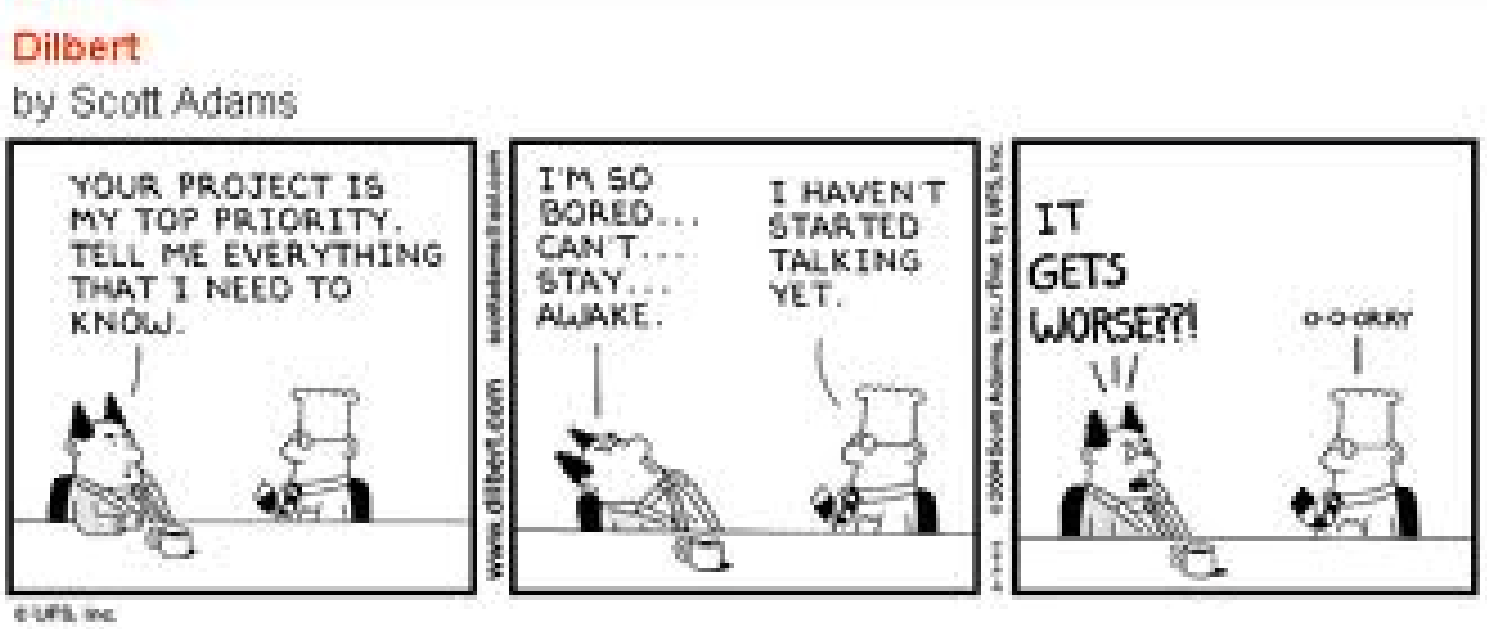
Von Neumann

- In 1944, John von Neumann was attracted to the ENIAC project. The group wanted to improve the way programs were entered and discussed storing programs as numbers; von Neumann helped crystallize the ideas and wrote a memo proposing a stored-program computer called EDVAC (Electronic Discrete Variable Automatic Computer).

- Herman Goldstine distributed the memo and put von Neumann's name on it, much to the dismay of Eckert and Mauchly, whose names were omitted. This memo has served as the basis for the commonly used term *von Neumann computer*. Several early pioneers in the computer field believe that this term gives too much credit to von Neumann, who wrote up the ideas, and too little to the engineers, Eckert and Mauchly, who worked on the machines.

- For this reason, the term does not appear elsewhere in your textbook book.

Why study Computer Architecture ?



Main goals of this course

- Understand and explain the main components of a computer
- Make connections between programs and the way they are executed on hardware infrastructure
- Evaluate how the different design choices influence the performance of computer
- Understand the interaction between the different components and their impact on performance

Will be evaluated by …

- Labs
- Written exam: Ability to solve problems
- Written exam: Ability to explain the fundamental concepts

In the classroom

- Sometimes, you will solve problems
  - I will interact with you, but this will not be graded

- For this, please be seated in a small group of 2 or 3
  - You may use your lab group

An experiment with Twitter

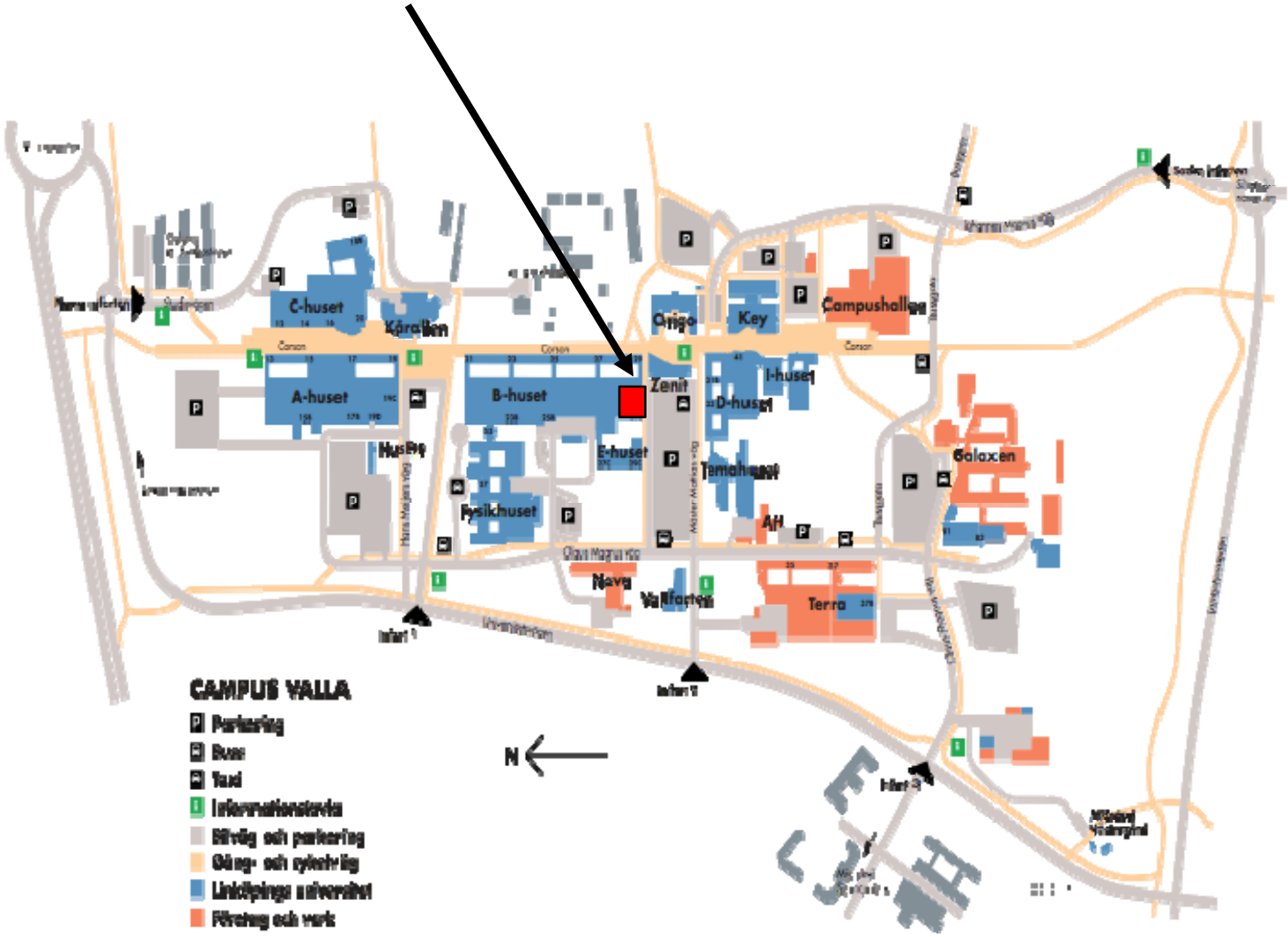
- Optional
- Use #tdts10
- Use to ask doubts, or share interesting results, when you do not feel like spamming the entire group
- Use also in classroom – only when solving problems

Examination

- The course will have three obligatory exams/homework
  - Exam
  - Lab
- Lab
  - This is obligatory and you can get a pass/fail grade
- Final Exam
  - There will be a final written exam. More details will be shared later

Personnel

- Unmesh Bordoloi- Kursledare (examinator)
  - Epost: unmesh.bordoloi@liu.se
- Dimitar Nikolov (Teaching Assistant)
  - Epost: dimitar.nikolov@liu.se
- Madeleine Dahlqvist (Kurssekreterare)
  - Epost: madeline.hager.dahlqvist@liu.se
- Patrick Lambrix (Director of Studies)
  - Epost: patrick.lambrix@liu.se


C-huset
Kårallen
A-huset
B-huset
Zenit
Origo
Key
Campushallen
I-huset
D-huset
E-huset
Galaxen
Temahuset
Fysikhuset
Navet
Terra
Infart 1
Infart 2
Infart 3
CAMPUS VALLA
Parkering
Buss
Taxi
N

14

Textbook

- The recommended textbook is:
  - David A. Patterson and John L. Hennessy: Computer Organization and Design - The Hardware / Software Interface, 4th Edition, Morgan Kaufmann
  - Available as e-book in the university library

Other literature

- William Stallings: Computer Organization and Architecture, Prentice Hall International, Inc..
- Sven Eklund, Avancerad datorarkitektur, Studentlitteratur, 1994.
- Andrew S. Tanenbaum, Structured Computer Organization, 4th edition, Prentice Hall International, Inc., 1999.
- V. C. Hamacher, et al.: Computer Organization, 4th edition, McGraw-Hill, 1996.

Today

- 4 hours!
- 3 + 1 hour!

So …

Technology

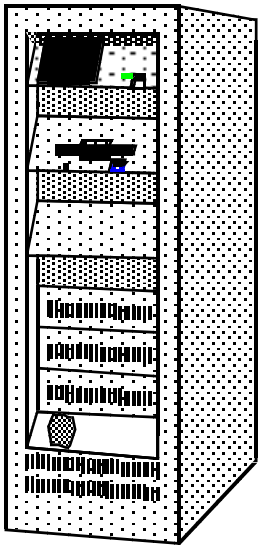
- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost

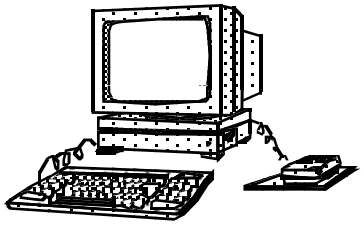
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

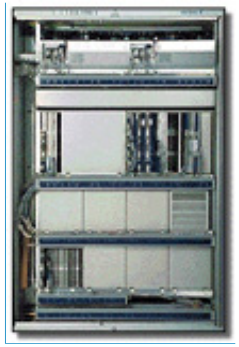
Yesterday's fiction is now reality

- Applications empowered by computers:
  - Human genome project
  - World Wide Web
  - Search Engines
  - Social Networks

Classes of Computers







Classes of Computers

- Desktop computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Below your program

- We constantly interact with these computers
  - E.g., via apps on iPhone or via a Word Processor

- How does an application interact with the hardware?

Language of the hardware

- The hardware understands *on* or *off*. Hence, the language of the hardware needs two symbols 0 and 1.

- Commonly referred to as binary digits or bits.

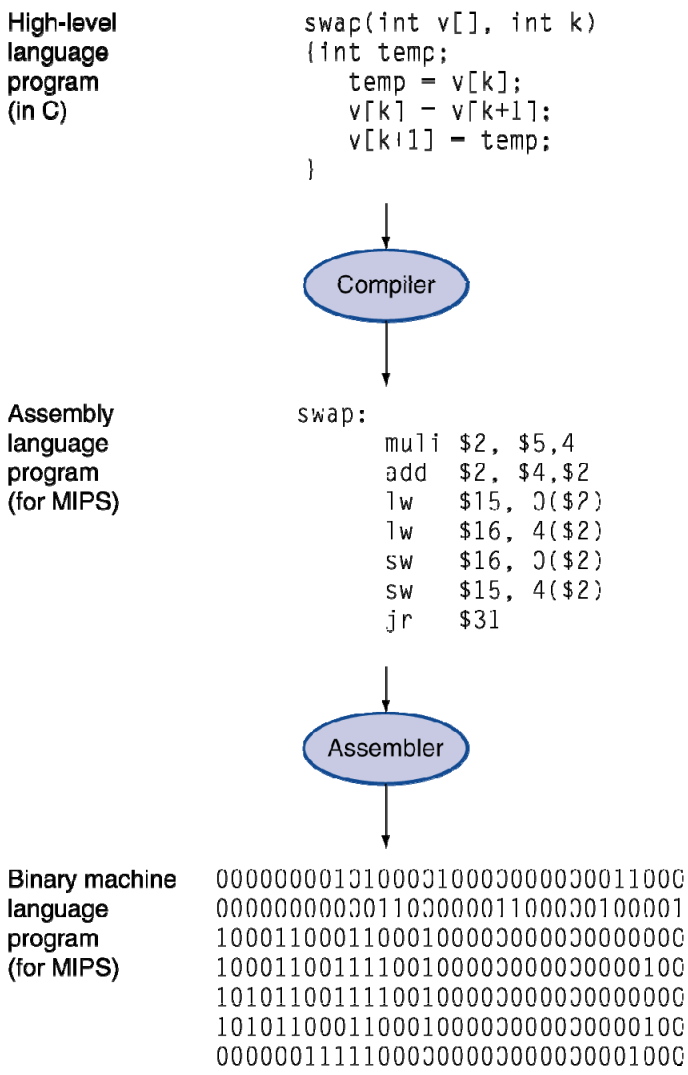
- 2 letters do not limit what computers can do (just like the finite letters in our languages)

Language of the hardware

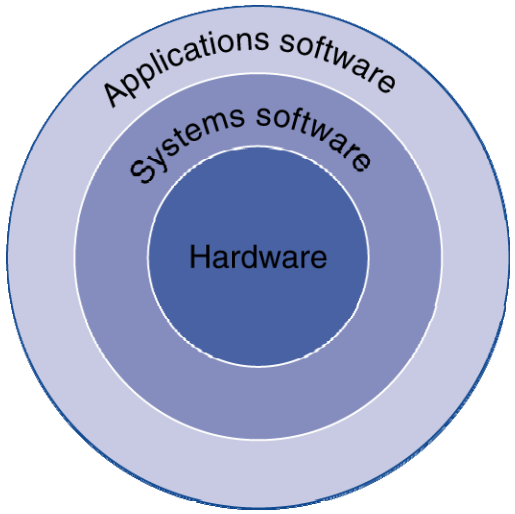
- Instructions are collections of bits that the computer understands
- What our programs instruct the computer to do are the *instructions*:
- Hundreds/thousands/millions of lines of code (instructions) are hidden beneath our apps and programs

Hierarchical Layers of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



Below your program



- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

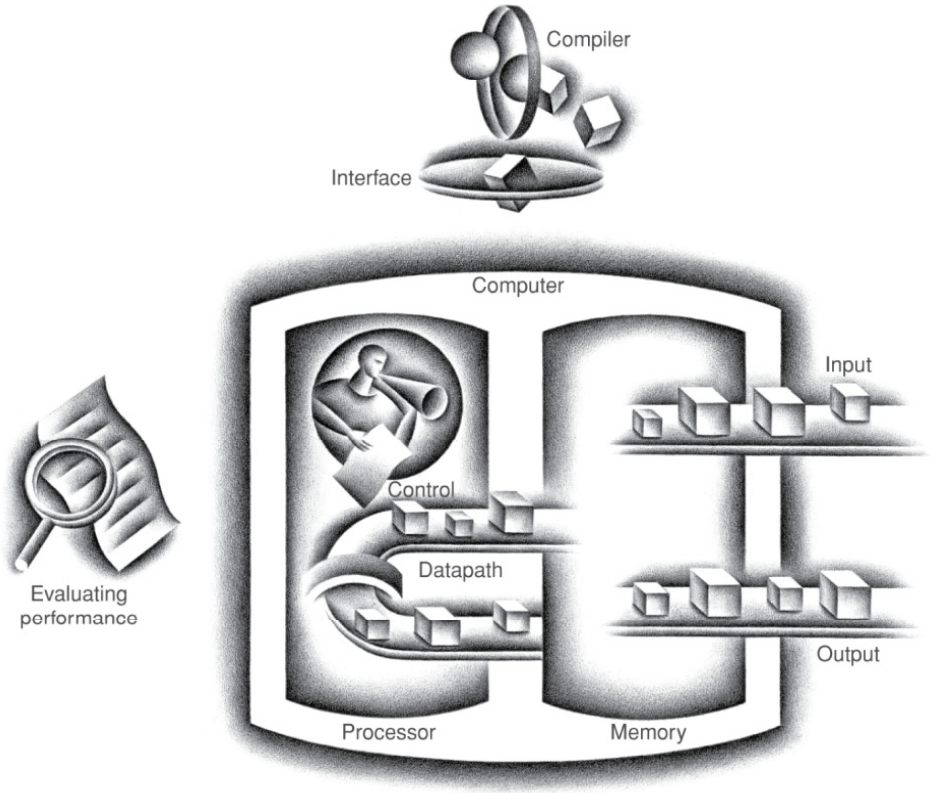
Under the Covers

- Understanding the underlying hardware (the computer!) is the main focus of this course

- So, what are the main components of the computer?

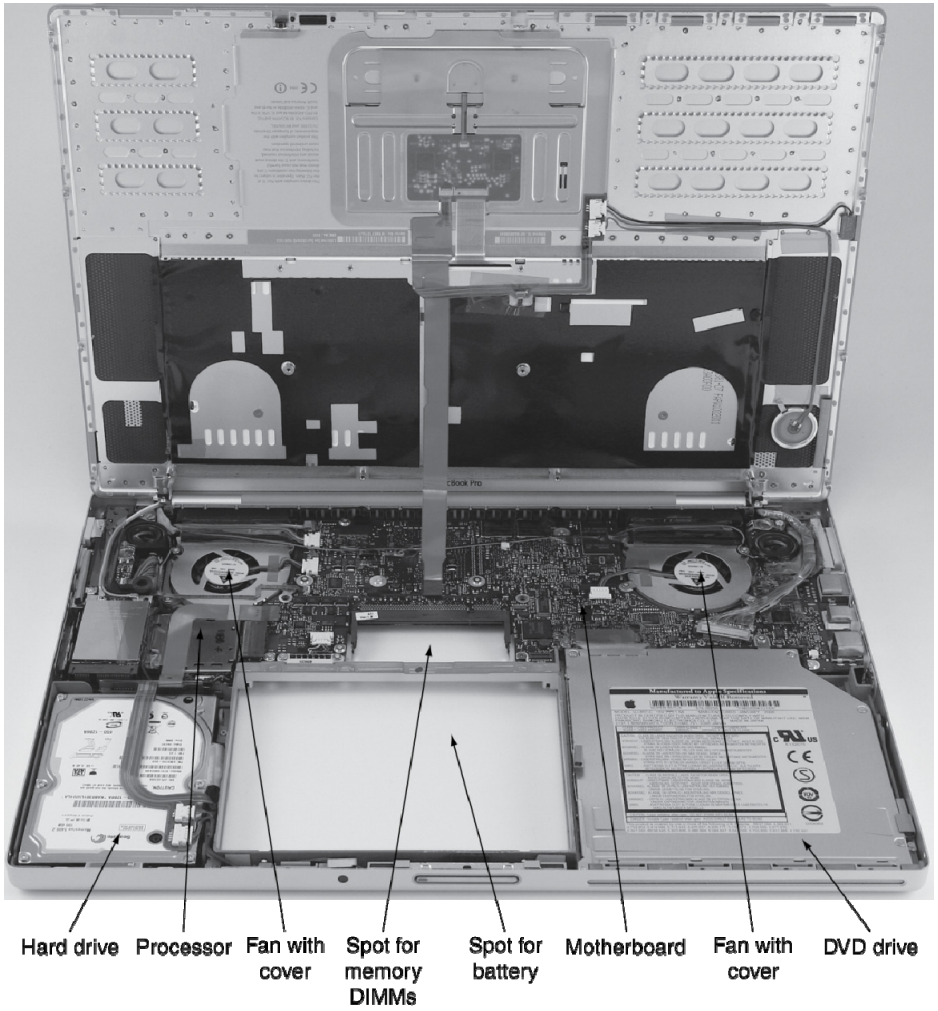
Components of a Computer

- Same components for all kinds of computer
  - Desktop, server, embedded



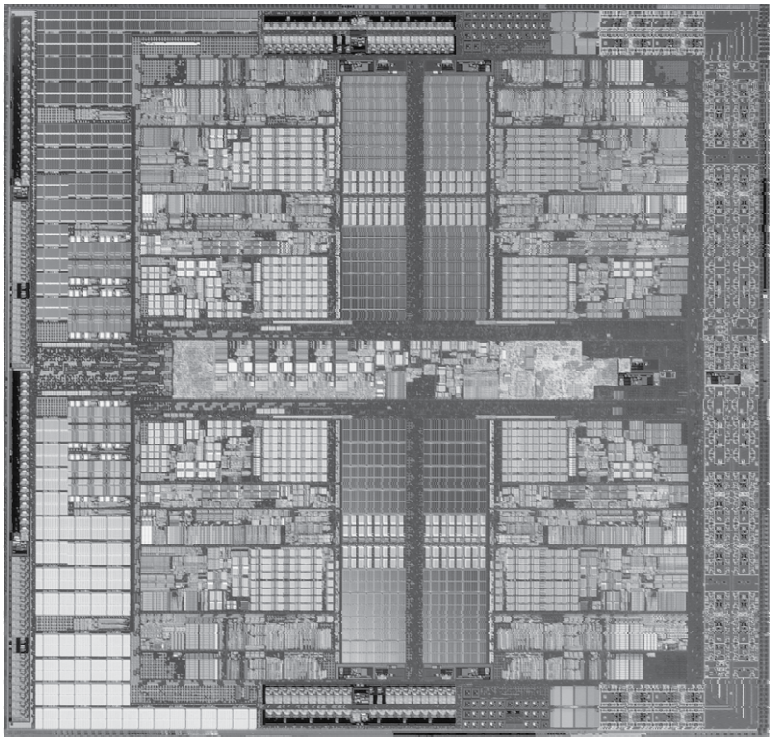
Anatomy of a Computer

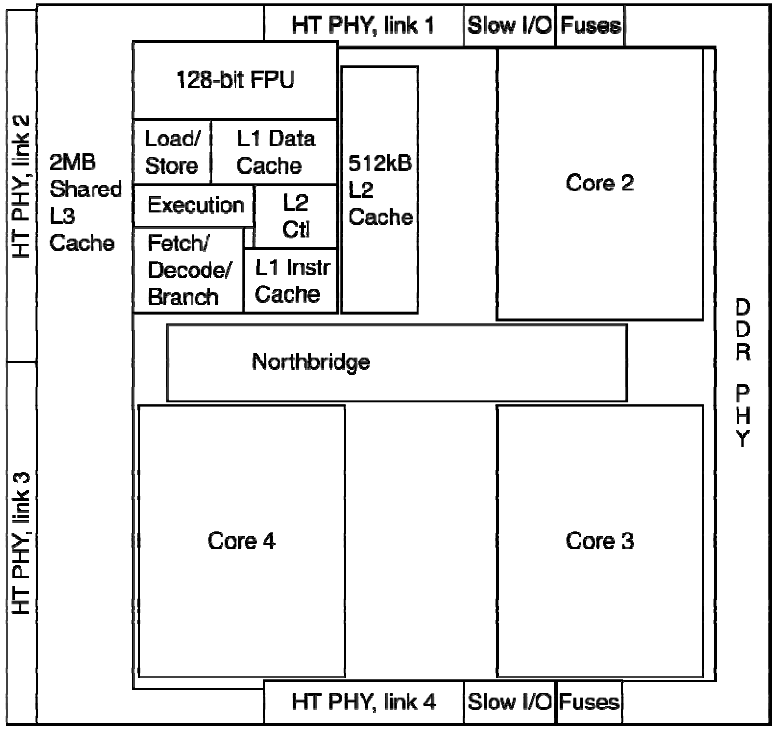
Opening the Box



- Motherboard
  - I/O connections
  - Memory (DRAM)
  - CPU or central processing unit

AMD Barcelona: 4 processor cores





Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

Abstractions

- Abstraction helps us deal with complexity
  - Note abstraction in both hardware and software
  - Hide lower-level detail
- Instruction set architecture (ISA)
  - The hardware/software interface
- Implementation
  - The details underlying and interface

Our Topics in this Course

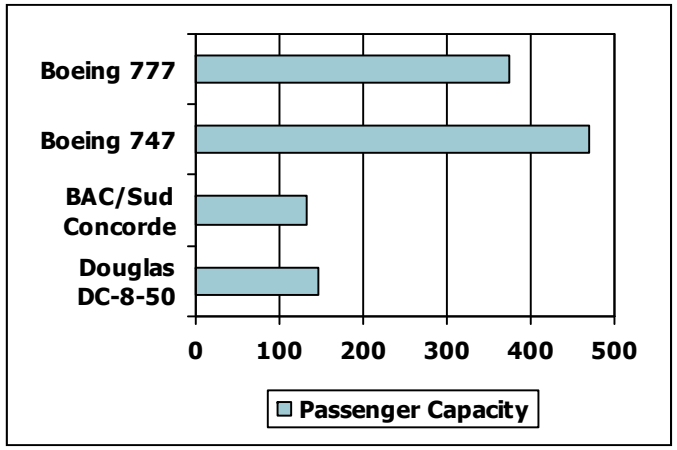
- Instructions
- Arithmetic
- The Processor
- Memory
- Input/Output
- Multi-cores and GPUs

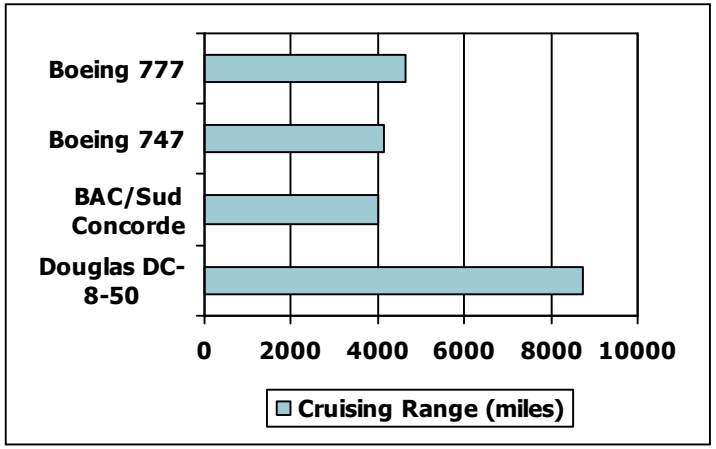
Performance

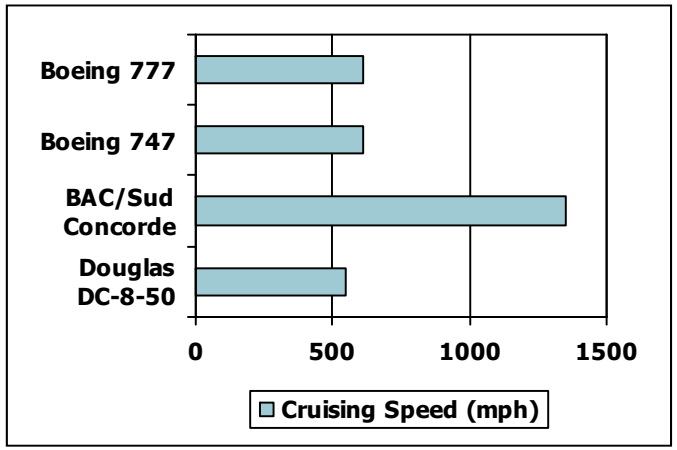
- Why is performance important?
  - For purchasers: to choose between computers
  - For designers: to make the sales pitch
- Defining performance is not straightforward!
  - An analogy with airplanes shows the difficulty

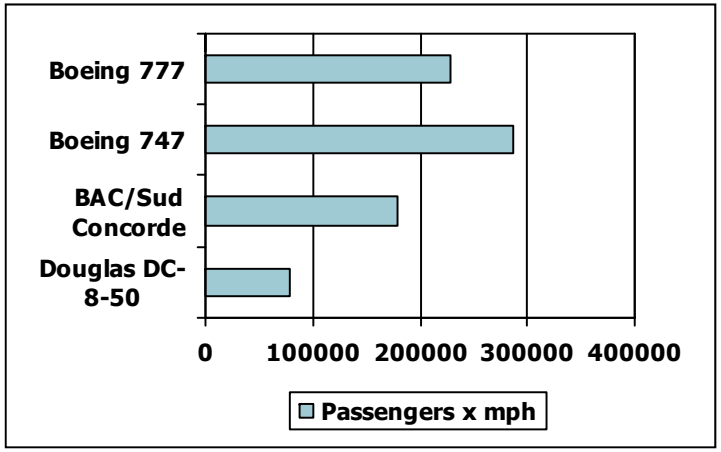
Defining Performance

- Which airplane has the best performance?









Response Time and Throughput

- Response time
  - As a user of a smart phone (embedded computer), or laptop, the one that responds faster is the better!
  - Response time = How long it takes to do a task?
  - Response time = the total time required for the computer to complete a task, including disk accesses, memory accesses, I/O activities, operating system overheads, CPU execution time …

Response Time and Throughput

- Throughput
  - If I am running a data center with several servers, faster computer is the one that completes several tasks in one day!
  - Total work done per unit 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?
  - Look in the textbook for a discussion (Page 28)

Relative Performance

- Define Performance = $1/\text{Execution Time}$
- $\text{Performance}_x > \text{Performance}_y$
- $1/\text{Execution Time}_x > 1/\text{Execution Time}_y$
- $\text{Execution Time}_y > \text{Execution Time}_x$

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock

  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0 GHz = 4000 MHz = 4.0×10^{9}Hz

CPU Time

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

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles compared to A
- How fast must Computer B clock be i.e., what is the clock rate for Computer B?

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

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles compared to A
- How fast must Computer B clock be?

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

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - We aim for 6s CPU time
  - We can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

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

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10\text{s} \times 2\text{GHz} = 20 \times 10^9\end{aligned}$$

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

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

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

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
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

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

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

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

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

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

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance