The first electronic computers

- 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
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.

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.

Technology

- 18,000 vacuum tubes
- 20 registers, each around 0.5 meters long
- 24 meters long, 2.5 meters high

Technology

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Technology</th>
<th>Relative performance/cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>Vacuum tube</td>
<td>1</td>
</tr>
<tr>
<td>1965</td>
<td>Transistor</td>
<td>35</td>
</tr>
<tr>
<td>1975</td>
<td>Integrated circuit (IC)</td>
<td>900</td>
</tr>
<tr>
<td>1995</td>
<td>Very large scale IC (VLSI)</td>
<td>2,400,000</td>
</tr>
<tr>
<td>2005</td>
<td>Ultra large scale IC</td>
<td>6,200,000,000</td>
</tr>
</tbody>
</table>
Revolution?

- Human civilization:
  - Agricultural revolution
  - Industrial revolution
  - …

- Computers are the result of an incredible industry that embraces innovations at a breathtaking rate
  - A similar pace in Transportation industry would have implied that we reach New York in few seconds

Yesterday’s fiction is now reality

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

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 is 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)

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

A Safe Place for Data

- Volatile main memory
  - Loses instructions and data when power off
- Non-volatile secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)

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

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)
  - We’ll focus on response time for now…

Relative Performance

- Define Performance = 1/Execution Time
- Performance_x > Performance_y
- 1/Execution Time_x > 1/Execution Time_y
- Execution Time_y > Execution Time_x

Performance_x/Performance_y
= Execution time_y / 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
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

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
- How fast must Computer B clock be?

\[ \text{CPU Time}_B = \frac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B} \]
\[ \text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s} \]
### 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 $1.2 \times$ clock cycles
- How fast must Computer B clock be?

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

\[
\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A
= 10s \times 2GHz = 20 \times 10^9
\]

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

\[
\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s} = \frac{24 \times 10^9}{6s} = 4GHz
\]

### Instruction Count and CPI

- Clock Cycles = Instruction Count $\times$ Cycles per Instruction
- CPU Time = Instruction Count $\times$ CPI $\times$ Clock Cycle Time
  \[
  \text{CPU Time} = \frac{\text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}}{\text{Clock Rate}}
  \]

- 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?

\[
\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A
= 1 \times 2.0 \times 250ps = 1 \times 500ps
\]

\[
\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B
= 1 \times 1.2 \times 500ps = 1 \times 600ps
\]

\[
\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{1 \times 600ps}{1 \times 500ps} = 1.2
\]

A is faster... 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

Personnel

- Unmesh Bordoloi - Kursledare (examinator)
  - Kontorstider:
    - Vecka 37-43 - Tisdag 14-16
    - Epost: unmesh.bordoloi@liu.se
- Madeleine Häger Dahlqvist - Kурсsekreterare
  - Email: madeline.hager.dahlqvist@liu.se

Examination

- The course will have three obligatory exams/homework
  - Homework 1
  - Homework 2
  - Final Exam
- Homework 1
  - This is obligatory and you can receive 0-4 points. The points will be added to the final grade.
- Homework 2
  - This is obligatory and you can receive 0-8 points. The points will be added to the final grade.
- Final Exam
  - There will be a final written grade. The points will be added to the final grade
Examination

- Final score is calculated as the sum of
  - Homework 1 (maximum 4 points)
  - Homework 2 (maximum 8 points)
  - Final Exam (maximum 28 points)

Textbook

- The recommended textbook is:
  - Available as e-book in the university library

Textbook

- At the end of every lecture, there will be a slide listing the relevant portions covered from the textbook
- Today, we covered 1.1, 1.2, and 1.3

Other literature