Fundamentals of Computer Architecture

Course Overview

Teacher: **Lobar Asretdinova**

Department of Control and Computer Engineering

email: l.asretdinova@polito.uz

Course Overview (part 1)

Course Structure:

Lobar Asretdinova*:*

- Lectures: 38 hours
- Laboratory work - 38 hours (divided into groups)

Prof. Giovanni Squillero

- Lectures: ~44 hours

Total 8 credits

Goal:

- This course aims to provide a strong foundation for students to understand modern computer system architecture and to apply these insights and principles to future computer designs.

Course Overview (part 2)

Learning Outcomes:

- important advances in the history of modern computing, and latest trends in the computing industry;
- how programs written in high-level programming languages, such as C or Java, can be translated into the language of the hardware;
- digital logic design (both combinational and sequential);
- memory hierarchy in computer design, and memory design impacts on overall hardware performance;
- storage and I/O devices, their performance measurement;

Course Overview (part 3)

Assessment and Examination:

- Laboratory and Practice work : **4 points** (max)
  - In total, 19 Classes (2 hours / each)
- Final exam:
  - **30L points** (max)
- Attendance is checked:
  - Students with 60% and higher attendance are allowed to get extra points during the lab classes.
  - Examination of the lab classes for extra point will be announced during the lessons. It is going to be after we cover at least 50% of our material.

Fundamentals of Computer Architecture

Unit 1: Introduction to Computer Systems

Teacher: **Lobar Asretdinova**

Department of Control and Computer Engineering

email: l.asretdinova@polito.uz

Introduction to Computer Systems

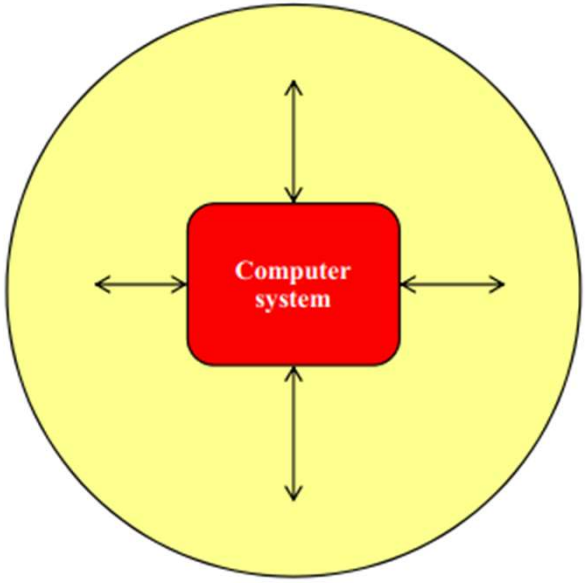
Goals:

- To summarize how we reached the current state of the art in the area of computer systems
- To introduce in an informal manner some basic concepts
- To describe the current situation.

Computer systems

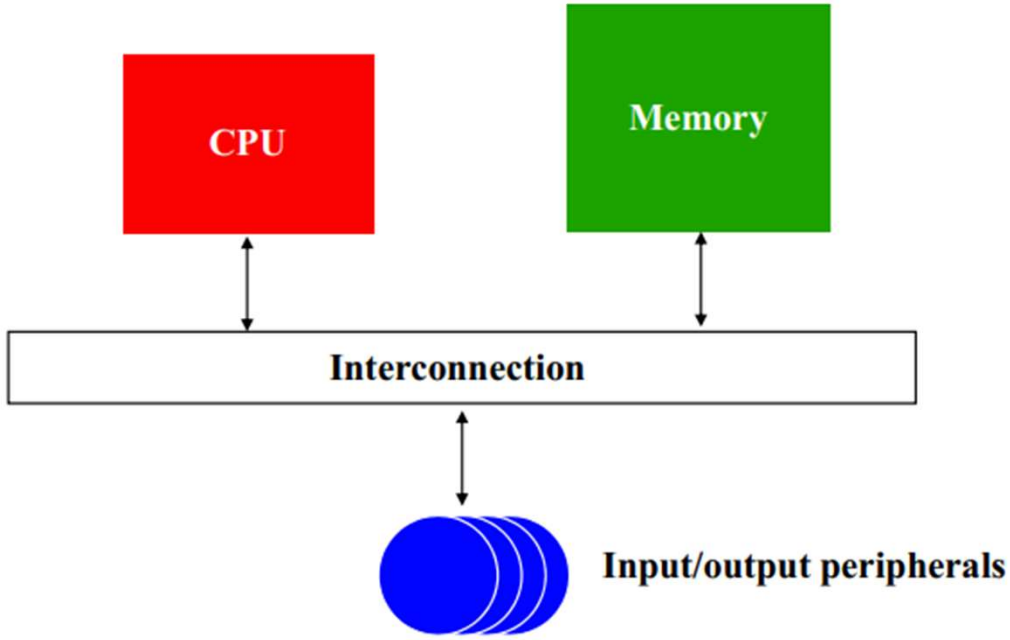
A *computer system* is able to

- Receive inputs from the outside
- Perform some processing
- Send commands and data outside.



Processor-based systems

A computer system often corresponds to a processor-based system.



Classes of Computers

- Personal computers
- Server computers
- Supercomputers
- Embedded computers

Classes of Computers (2)

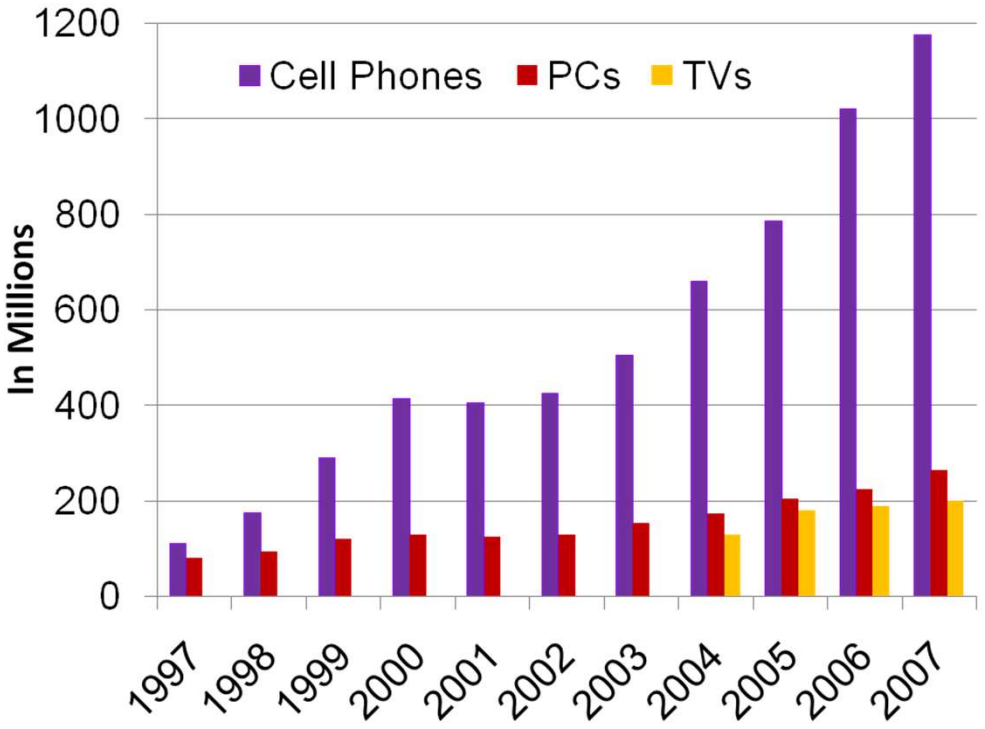
- Personal computers
  - Computers designed for use by an individual
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server computers
  - Computers used for running larger programs for multiple users, often simultaneously
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

Classes of Computers (3)

- Supercomputers
  - Consist thousands of processors and many terabytes of memory
  - High-level scientific and engineering calculations
  - Highest capability and cost but represent a small fraction of the overall computer market
- Embedded computers
  - Hidden as components of systems
  - Largest class of computers and span the widest range of applications
  - Strict power/performance/cost limitations

Growth in Embedded Processor Sales

(embedded growth >> desktop growth !!!)



- Where else are embedded processors found?

Key ideas

Computer systems are based on

- Techniques to *represent information* (numbers, text, multimedia)
- Techniques to define the *sequence of operations* to be performed to solve a problem (i.e., algorithms)
- Techniques to *store data* and *automate* operation execution.

Ages (Historical Evolution)

- Mechanic age (until 1945)
- Electronic age (from 1945 to 1975) organized in 3 generations
- Very Large Scale Integration (VLSI) age (since 1975).

Mechanic age

- B. Pascal (1623-1662)
- G. Leibniz (1646-1716)
- J. Jacquard (1752-1834)
- C. Babbage (1792-1871)
- Z1 (1938), Z3 (1941) by K. Zuse
- Mark I (1944) and Mark II by H. Aiken

B. Pascal

In 1642 he built a computing machine

- First working computer
- Able to perform additions and subtractions
- Included 2 sets of 6 toothed wheels each. Each wheel had 10 positions, corresponding to a decimal character
- Each set corresponded to a register, able to store a decimal value

How it works – video
https://www.youtube.com/watch?v=3h71HAJWnVU

Leibniz

Around 1671 he built a new, more powerful computing machine:

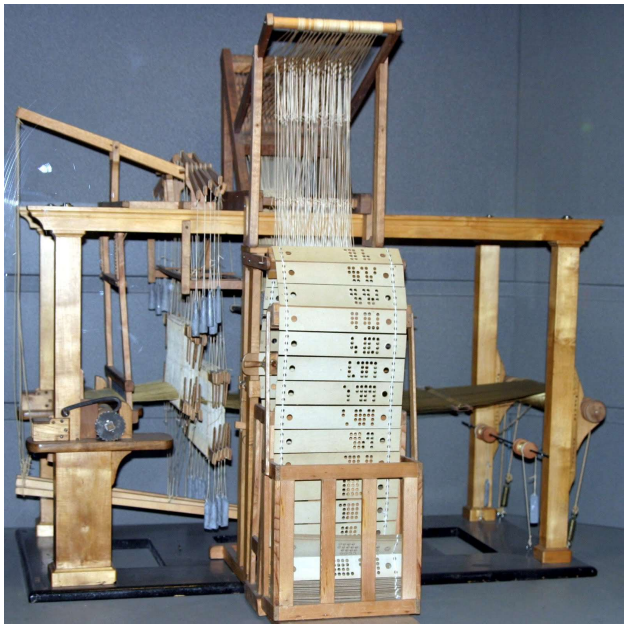
- Able to perform sum, subtraction, multiplication and division operations
- Composed of two Pascal machines (4 sets of wheels)
- Multiplication and division operations were performed by sequences of additions and subtractions.

Jacquard loom

It was built around 1801 by J. Jacquard. – video

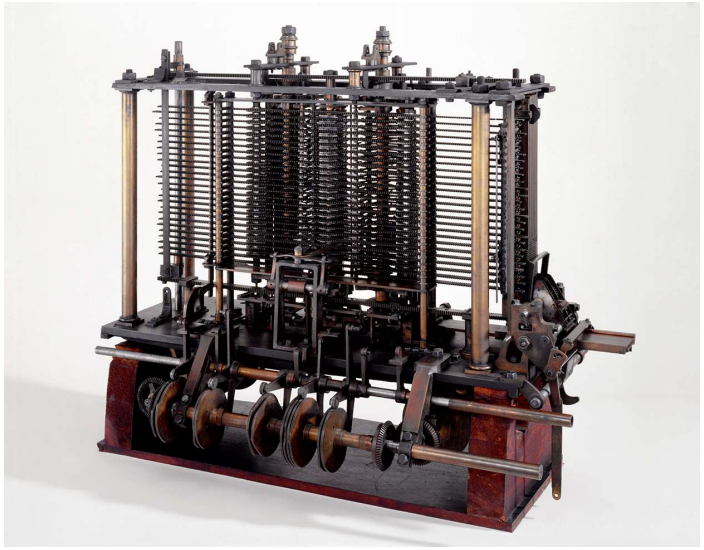
https://www.youtube.com/watch?v=OlJns3fPItE

- It is the first example of programmable machine, aimed at controlling the weaving process.
- The sequence of operations performed by the loom was decided by a set of pasteboard cards with punched holes.
- Jacquard wrote a program composed of 24,000 cards to weave a tapestry with his image.



Differential machine

- It was designed by C. Babbage (the inventor of the tachymeter) around 1823. - video
- Its purpose was to compute tables of numbers to support sailors.
- It was able to execute a single algorithm (finite differences with polynomials).
- It included an output device based on engraving a copper plate with a steel tip.



Analytical machine

It is the evolution of the differential machine and was designed around 1834.

It was composed of 4 parts:

- The storage (i.e., the memory, composed of 1,000 words, each composed of 50 decimal characters)
- The mill (i.e., the processing unit)
- The input device (reading punched cards); cards defined the sequence of operations to be performed and the origin of the operands
- The output device (copper plate or punched cards).

Zuse

Between 1939 and 1944 Konrad Zuse designed and manufactured in Germany 4 computer products (Z1, Z2, Z3, Z4).

They are the first examples of programmable computers and are based on electromagnetic relays.

For the first time a binary representation for numbers was used.

All the Zuse prototypes were destroyed in Berlin during WWII.

Harvard Mark I and II

They were manufactured (the former) and designed (the latter) by H. Aiken in Harvard, MA.

The design was based on Babbage ideas, using relays.

Storing was implemented through wheels

Mark I

Mark I was completed in 1944 in the IBM labs in Endicott.

It owned 72 words of 23 decimal characters each, and a cycle time of 6 seconds.

Input and output were based on tapes of punched paper.

Problems:

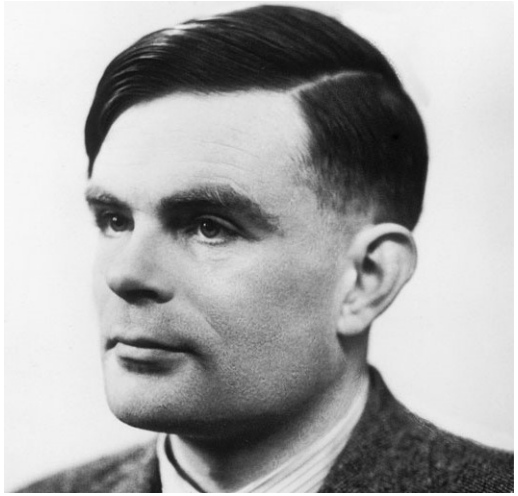
- Speed was limited by the movements of the composing parts
- Information were transferred through levers and pulleys in a very unreliable manner.

Turing

When the first computing systems were being designed, in 1937 Alan Turing proposed a computer model able to execute a generic algorithm (Universal Machine).

This model allows to

- Distinguish what is computable and what is not
- Identify the complexity of an algorithm.



Colossus

During WWII, Turing was part of the group of scientists who worked at Bletchley Park to develop techniques aimed at decoding the German messages encoded via the Enigma devices).

In Bletchley Park the Colossus was built.

At the end of the war everything was destroyed and kept secret.

How Enigma Machine work – video

https://www.youtube.com/watch?v=ybkkiGtJmkM

Electronic age

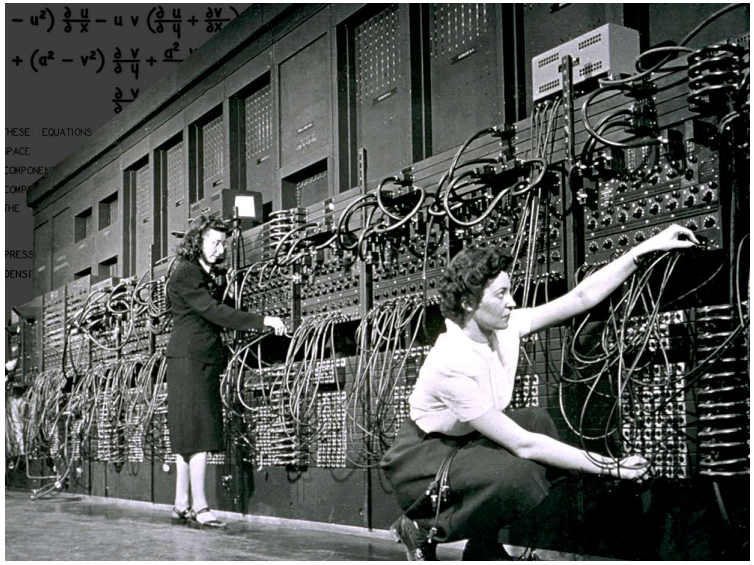
- Data flow correspond to movements of electrons
- Switching is fast thanks to vacuum tubes (also called valves).

ENIAC

ENIAC (Electronic Numerical Integrator and Calculator) was built between 1943 and 1946 at the University of Pennsylvania by J.W. Mauchly and J.P. Eckert.

It was composed of 18,000 valves and its weight was 30 tons.

It was used to compute ballistic tables for the U.S. Army.



Characteristics - ENIAC

Decimal representation: each character was stored in a ring of 10 valves, where only one valve was on

- 20 accumulators, each storing 10 characters
- Special units for multiplication, division and square root
- Manual programming (via switches and cables)
- *master programmer* unit for multiple or iterative operations
- Special memories for constants (Function Tables)
- Data and code were separated
- Data input via card readers
- Data output via teletypewriters.

EDVAC

EDVAC (Electronic Discrete Variable Automatic Computer) was designed in 1951 by the same team of ENIAC, including also *J. von Neumann* and *John Atanasof*.

Characteristics:

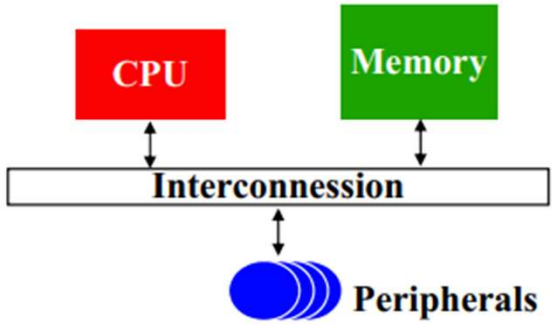
- Binary representation
- Wider memory, divided in two parts:
  - Main memory: 1K word, lines of mercury technology
  - Secondary memory: 20Kword, coil technology
- Code was also stored in memory
- Data input/output via teletypewriters.

EDVAC was never manufactured, because Eckert and Mauchley left the Università of Pennsylvania to fond *Unisys*.

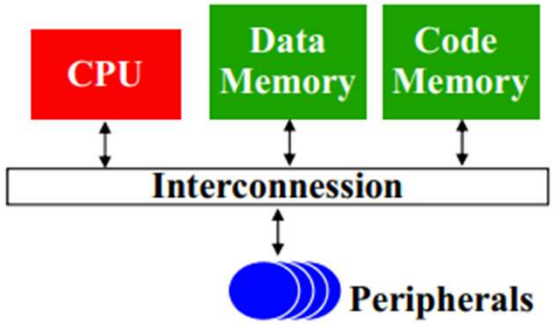
Von Neumann architecture

We defined as *von Neumann architecture* the one based on a single memory storing both data and code, which is accessed by the CPU via a bus.

The *von Neumann* architecture contrasts with the *Harvard architecture*, where the data memory is separated from the code memory.



Von Neumann architecture



Harvard architecture

First generation

It is composed of the computers built around 1950.

Characteristics:

- Technology based on valves
- Centralized architecture: every operation is performed by the CPU
- Machine language
- No operating system
- One user at a time can access the machine.

Second generation

It is composed of the computers built around 1960.

Characteristics:

- transistors substitute valves
- Cathodic ray and delay lines memory are substituted by those based on ferrite cores and magnetic drums
- CPUs include index registers and hardware for floatingpoint operations
- The first hardware-independent languages, such as ALGOL, COBOL and FORTRAN, are introduced
- I/O processors free the CPU from the burden of controlling all the peripherals
- First system software products (operating systems, compilers, libraries) are released.

Transistor was invented at the Bell Labs in 1948 by J. Bardeen, W. Brattain and W. Shokley (Nobel prize winners in 1956).
The first transistor machine was the TX-0 (*Transistorized eXperimental computer 0*), built at MIT in the first 50s and then followed by TX-2.

Third generation

Computers of the third generation appeared around 1965.

Characteristics:

- Integrated Circuits (ICs) substituted transistors
- Semiconductor memories substitute the ferrite core ones
- Microprogramming is introduced
- Pipelining is introduced
- Operating systems allow resource sharing in a system.

The VLSI age

Integrating tens of thousands of transistors in a single circuit (Very Large Scale of Integration) makes it possible that entire systems (CPU, peripherals, memories) are manufactured in a single device.

This provides significant advantages in terms of:

- cost
- size
- speed

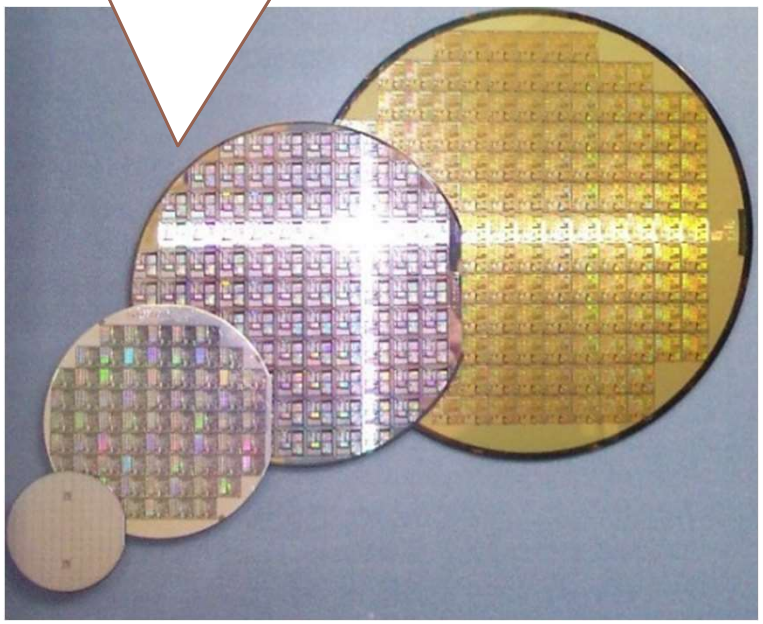
Integrated circuits

An Integrated Circuit (or IC) is composed of:

- silicon
- package
- pins.

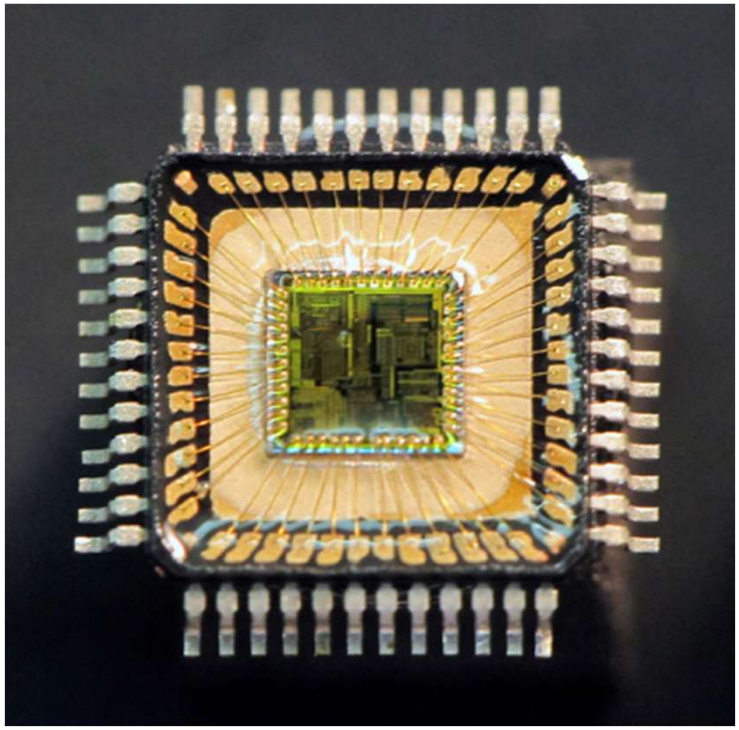
Today, a single IC can include several millions (in some cases, billions) of transistors.

Silicon Wafer where several dies are manufactured



Bonding

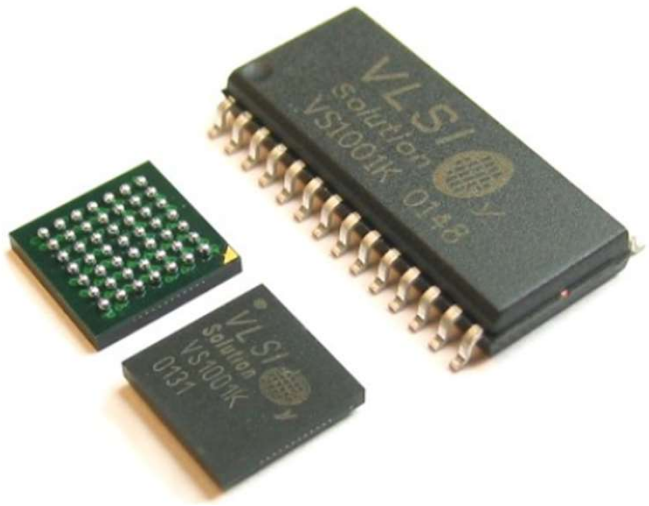
Once the different dies are cut from the wafer (slicing), each of them is inserted in its package and the input / output pads are connected to the pins (bonding).



Package types

The main ones include:

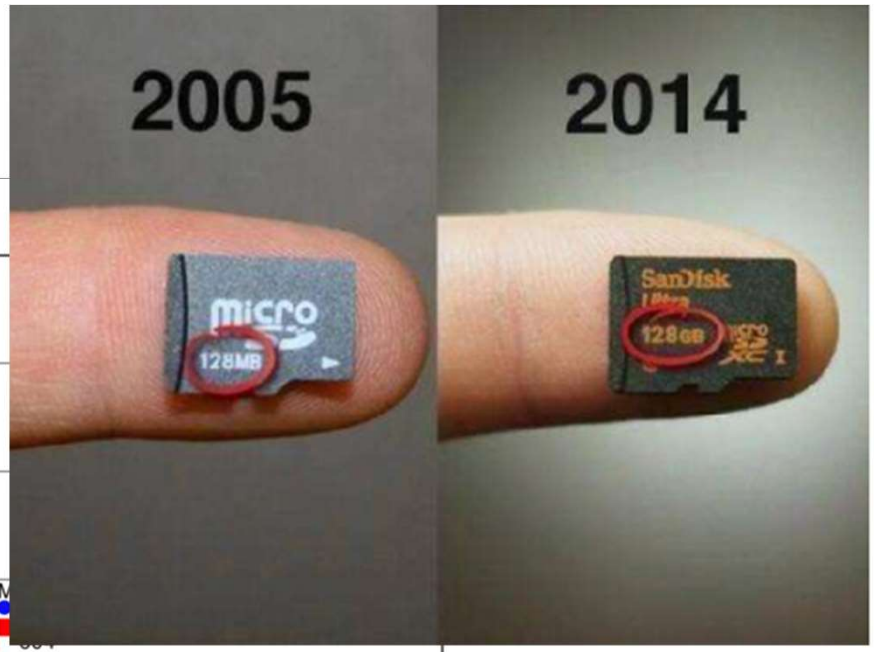
- DIP (Dual In-Line Package): from 14 to 68 pins on 2 rows
- PGA (Pin-Grid Array): 144 pins
- BGA (Ball-Grid Array).

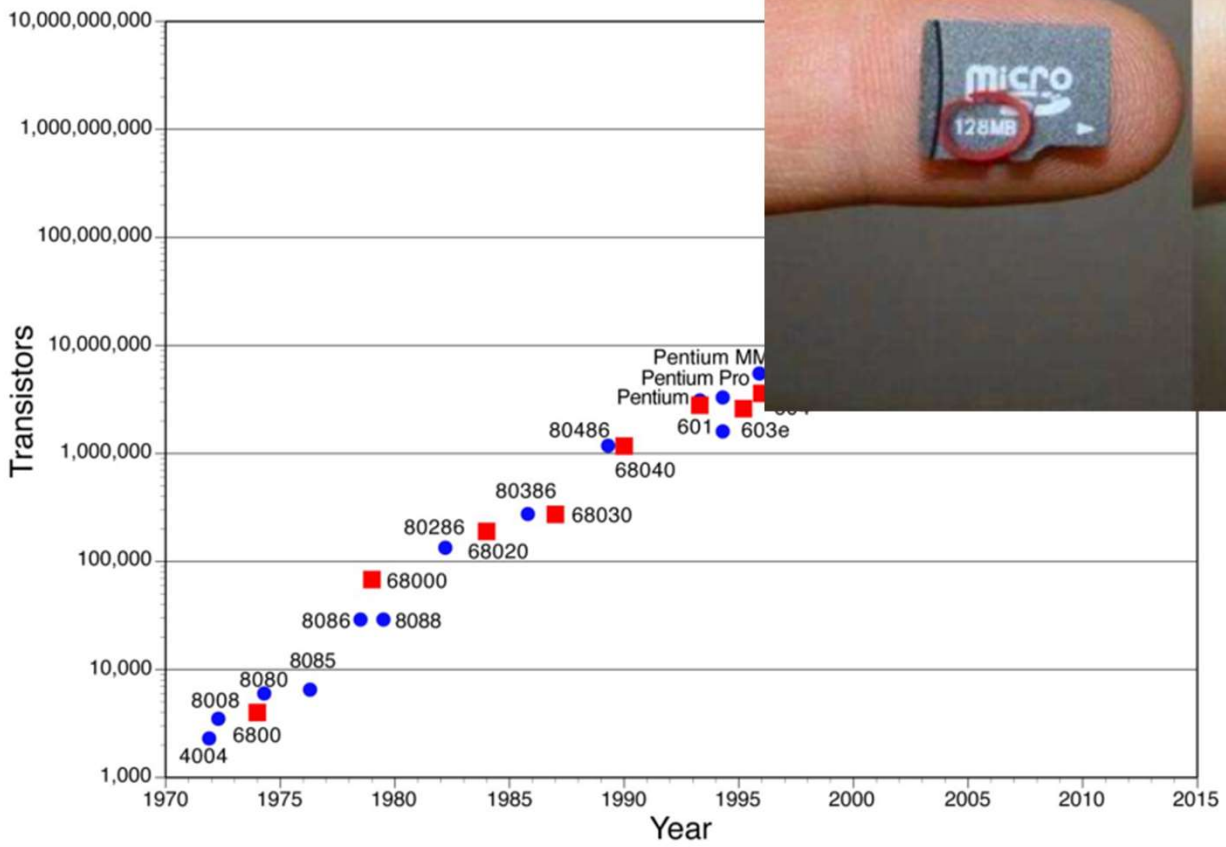


Moore law

It says tl
18/24 m

The obser

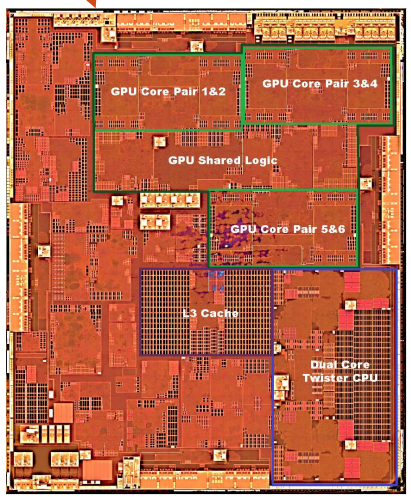


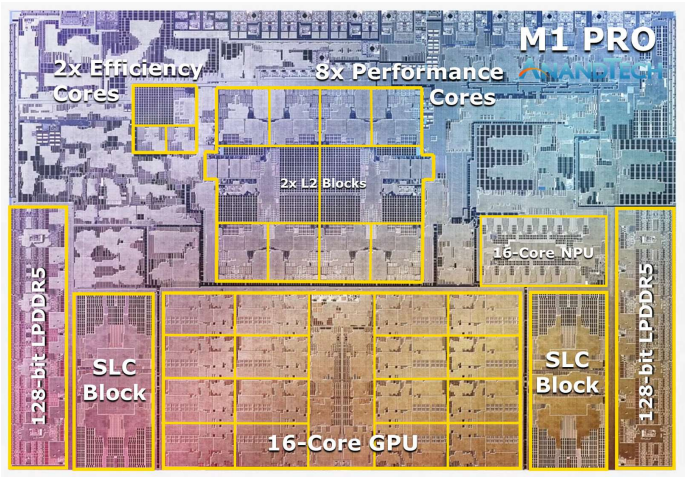


Integration density

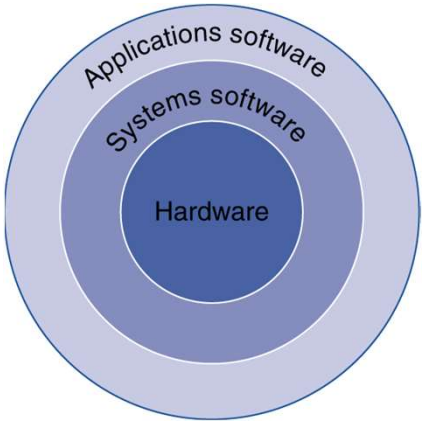
Thanks to the very high integration capabilities, today we can manufacture System on a Chip (SoC) devices, integrating in a single device one or more processors, memories, peripherals, special-purpose logic blocks.

Example: the **A9** chip by Apple OR recent **M1** chip inside Macbooks by Apple





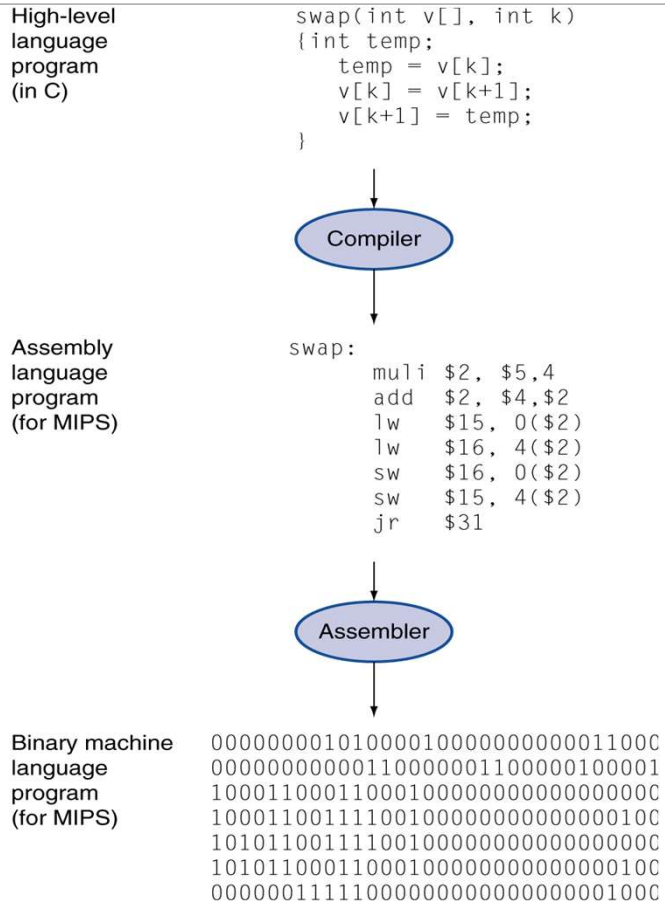
Below Your Program



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

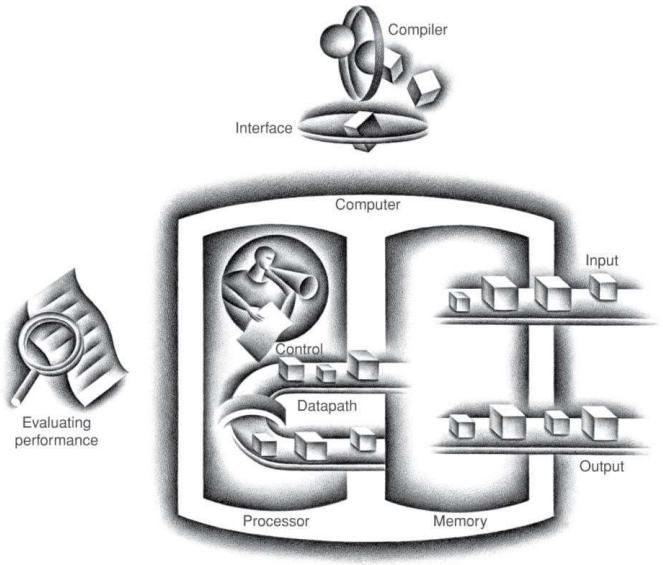
Levels of Program Code

- 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



Components of a Computer

The BIG Picture



- Same components for all kinds of computer
  - Desktop, Server, Embedded
- Input/Output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Microprocessors

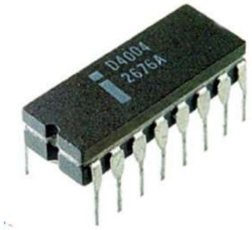
Microprocessor = processor manufactured on a single IC

Microcomputer = computer based on a microprocessor

The first commercial microprocessor was the Intel 4004 (1971).

All processors until the late 80s belonged to the CISC (Complex Instruction Set Computer) category, and were characterized by a wide set of instructions (>100 instructions).

CISC processors require a certain number of clock periods to execute each instruction.



RISC processors

RISC (Reduced Instruction Set Computer) processors are characterized by:

- Reduced set of instructions (e.g., 32 instructions)
- Pipelined architecture
- High number of registers (e.g., 128).

A RISC processor is typically able to complete the execution of an instruction at each clock period.

The first RISC processors were released in the late 80s.

Superscalar processors

Superscalar processors can complete the execution of more than one instruction per clock period

They use a set of RISC instructions

Their architecture is based on a pipeline structure including multiple functional units.

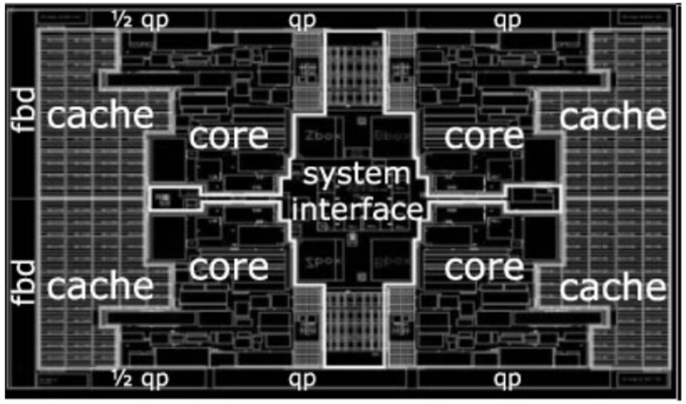
Multicore processors

Increase indefinitely the performance of a single processor is difficult

Hence, over the last few years, multicore devices have been introduced, integrating multiple processors into a single device.

Examples

- Cell (IBM, Sony, Toshiba)
- UltraSPARC T1 and T2 (Sun)
- Phenom (AMD)
- Core i7 (Intel)
- Tegra 3 (Nvidia).



	FET count	Voltage	Power
Core logic	430M	0.9-1.15V	100W
Sys Int	157M	0.9-1.15V	30W
L3 cache	1,420M	1.10V	20W
IO logic	39M	1.10V	20W
Chip Total	**2.046B**		**170W**

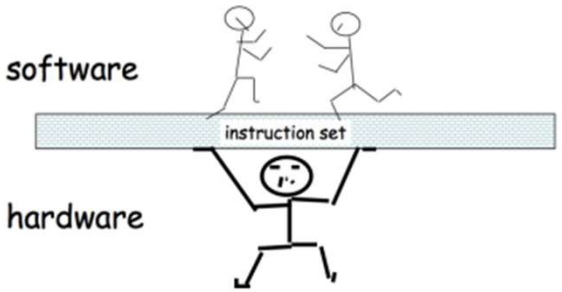
Instruction Set Architecture

Processors are organized in families

Processors of the same family are compatible at the software level, that is

The code of any application developed for a processor can be run on all subsequent processors in the same family

Compatibility is obtained by adopting the same *Instruction Set Architecture (ISA),* corresponding to the processor information set used by the programmer.



- Instruction set architecture (ISA) is a part of the abstract model of a computer, which generally defines how software controls the CPU.
- A device that executes instructions described by that ISA, such as a central processing unit (CPU), is called an implementation.

- In general, an ISA defines the supported instructions, data types, registers, the hardware support for managing main memory, fundamental features (such as the memory consistency, addressing modes, virtual memory), and the input/output model of a family of implementations of the ISA.

Microcontrollers (or MCUs)

These devices are built on a single IC and each targets some specific special-purpose application.

They are composed of:

- processor (with very variable features depending on the application)
- memory ROM, RAM, Flash modules
- ports and interfaces for I / O
- counters.

In the field of microcontrollers performance is as important as other parameters, such as cost, consumption, size, reliability, and ease of use

MCU market

	2012	2013	2014	2015	2016	2017	CAGR
Total Semiconductor	325,367	339,666	361,612	385,052	395,974	413,602	4.9%
Microcontroller (MCU)	16,008	16,202	17,211	18,799	19,307	20,480	5.1%
4 bit MCU	154	159	161	157	145	133	-2.8%
8 bit MCU	6,057	6,565	6,936	7,532	7,768	8,259	6.4%
16 bit MCU	4,021	3,611	3,765	4,060	4,053	4,019	0.0%
32 bit MCU	5,776	5,868	6,349	7,050	7,341	8,069	6.9%

All figures are in millions of USD
CAGR = Compound Annual Growth Rate

STM32 (example of MCU)

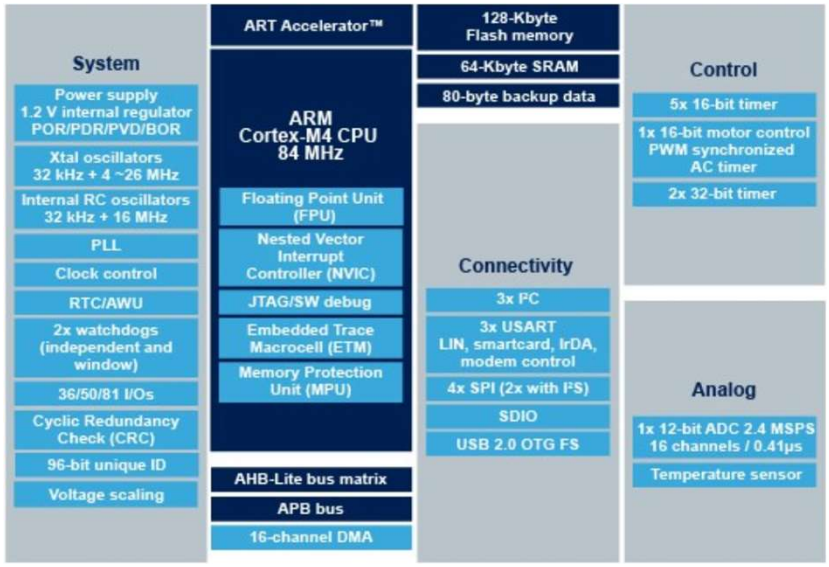
It is a family of MCUs with 32-bit CPUs designed and manufactured by STMicroelectronics

The first family products were introduced in 2006

Family devices are used in many applications, such as

- Automotive
- Industrial
- Medical
- Railway ...

More than one billion devices of this family have been sold already.



Special-Purpose systems

These systems are designed and built to run a single application

They are often inserted into more complex systems, which may include mechanical parts (mechatronic systems)

Special-Purpose system design

There are several alternatives when designing a special purpose system:

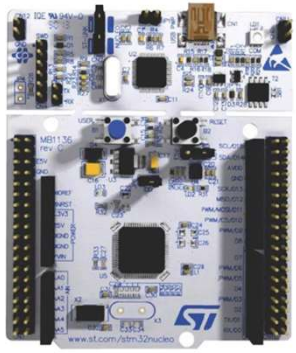
- SW Solution:
  - Use an existing board (equipped with a microprocessor or microcontroller that runs a program, usually contained in a ROM or Flash)
- HW Solution:
  - Based on an ASIC (*Application Specific IC*) designed and implemented specifically for that application.
  - If an ASIC contains one or more processors, it is called System on Chip (SoC).

Software solution

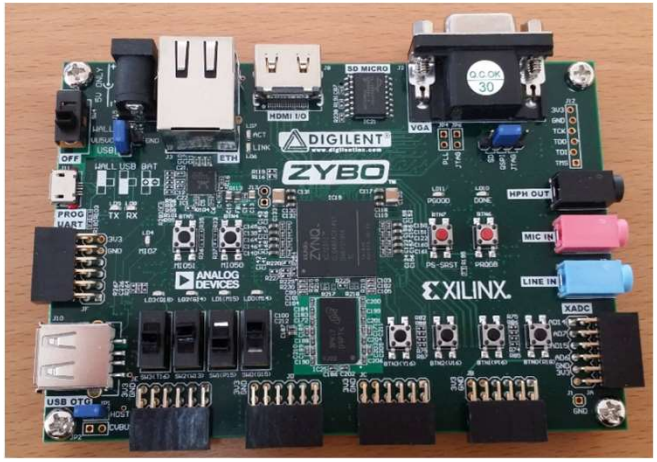
It is so defined because it often uses commercial hardware, based on a microprocessor (or microcontroller) board

In this case, the implementation work is focused on writing the software executed by the processor

It is typically the least efficient solution in terms of execution speed, but also the least expensive and more flexible.



STM32 Nucleo board



Xilinx ZINQ FPGA evaluation board

Hardware solution

It consists in realizing an ASIC that implements the desired functionality

The cost is much higher in terms of design and production, but the unit cost of the finished product is much lower

It makes sense when the volumes are high (> 1 million)

Moreover (in general)

- the performance is higher
- the footprint is smaller
- consumption is minimized
- flexibility is much smaller: once designed and built, the ASIC is no longer modifiable.

Bitcoin Miner ASICs

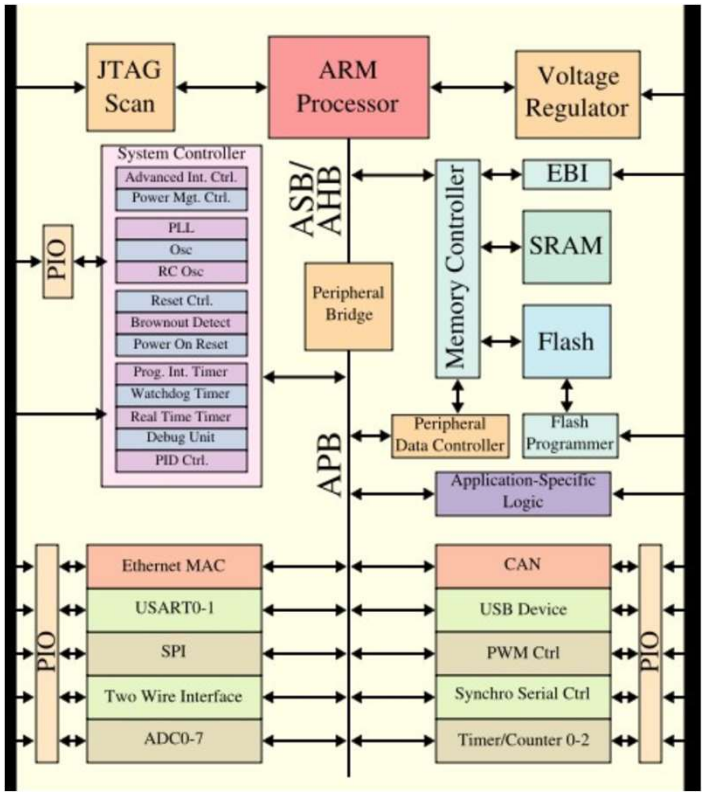
System on Chip

Many applications (such as mobile phones) are based on complex ICs (called Systems on Chip or SoCs)

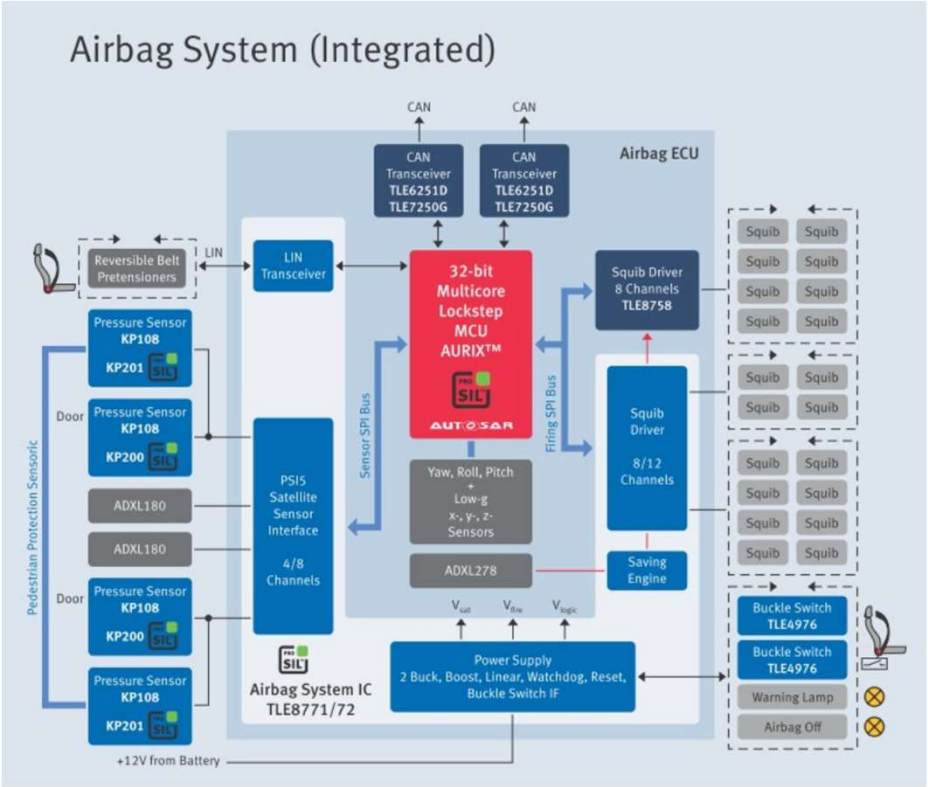
They integrate

- one or more processors
- various memory modules
- the interface modules
- any specific modules

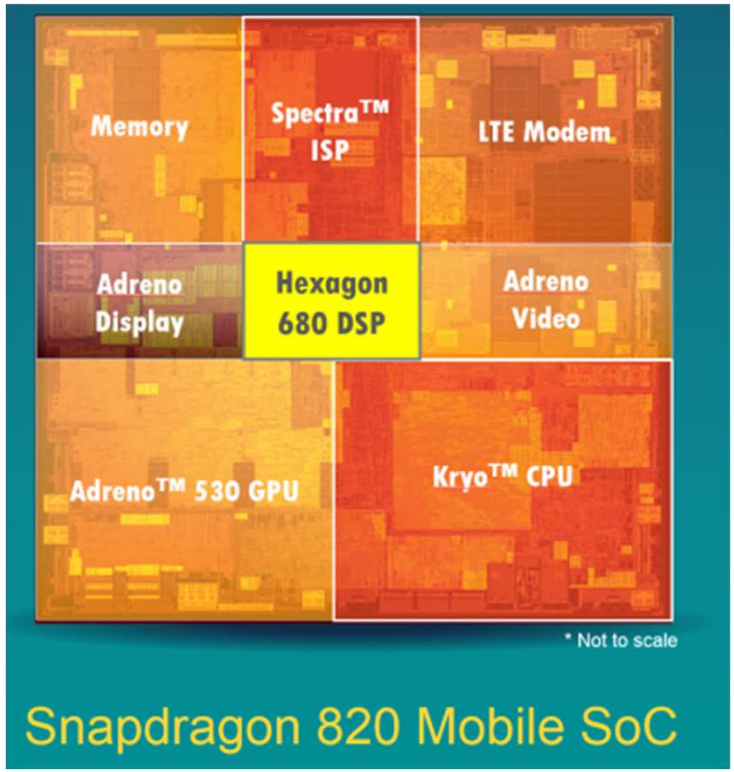
Each SoC is aimed at a specific application and integrates the necessary IP (*Intellectual Property)* modules.



Infineon Airbag SoC



Quallcom Snapdragon SoC



Additional Materials (video/article)

1) How do Smartphone CPUs Work? || Inside the System on a Chip – video link

2) History of Computing Hardware – wikipage

3) The Most Powerful Computers You've Never Heard Of – video link

Question #1

What is a *microcontroller*?

A	**A processor integrated on board a single integrated circuit**
B	**A processor with a reduced set of instructions**
C	**A processor made with VLSI technology**
D	**A device designed for special purpose applications that integrates a processor, some memory modules, and some peripherals on the same device**

Question #2

What does the *Moore law* say?

A	**The maximum number of integrated transistors on a single integrated circuit doubles every 18/24 months**
B	**That the maximum operating frequency of the integrated circuits doubles on average every 18/24 months**
C	**The processing power of processors doubles every 18/24 months**
D	**The cost of a single integrated circuit halves on average every 18/24 months**

Question #3

Which of the following statements about special-purpose systems is true?

A	**A special purpose system is designed and built to have a very low cost**
B	**A special purpose system is designed and built to run a single application**
C	**A special purpose system is designed and built to have very low consumption**
D	**A special-purpose system is designed and built to last much longer than a general-purpose system**

Question #4

What differentiates a system with a Harvard architecture compared to a von Neumann architecture system?

A	**A Harvard architecture system uses a decimal representation, while a von Neumann architecture system uses a binary representation.**
B	**A Harvard architecture system uses a processor, while a von Neumann architecture system uses a microcontroller.**
C	**A Harvard architecture system uses two separate data and code memories, while a von Neumann architecture system uses only one memory for both.**
D	**A system with Harvard architecture is designed and built to run a single application, while a von Neumann architecture system can run different applications.**

Question #5

Many systems can be implemented through a hardware solution or a software solution.

Which of the following advantages is generally proper of the hardware solution?

A	**Flexibility**
B	**Low design cost**
C	**Short design time**
D	**High performance**

Question #6

When were the first microprocessors on the market?

A	**In the 40s**
B	**In the 50s**
C	**In the 60s**
D	**In the 70s**

Question #7

What is a SoC (*System on Chip*)?

A	An integrated device containing over one billion transistors
B	A device for special-purpose applications
C	A device designed for a specific product that integrates a whole system internally
D	A standard device that integrates a processor, one or more memory modules, a number of interfaces to the outside

Question #8

What is the ISA (*Instruction Set Architecture*) of a processor?

A	The set of detailed information on the architecture and implementation of a processor
B	The set of information about a processor needed for its use by the SW
C	The SW that can be run by a processor
D	The set of processor specifications that can be deduced from its manual

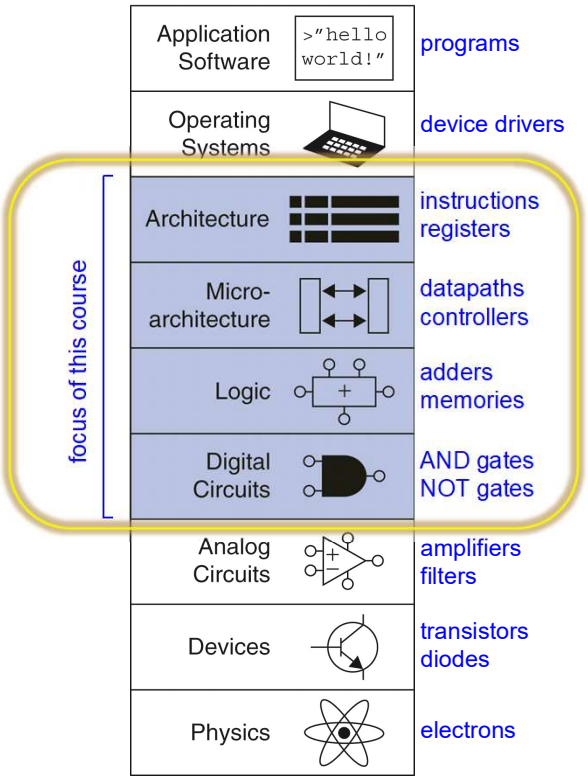
Question #9

What is the typical characteristic of a *Reduced Instruction Set Computer* (RISC) processor?

A	The ability to run more than 1 million instructions per second
B	The ability to perform one memory access for each clock period
C	The ability to complete (at best) one instruction for each clock period
D	The ability to complete (at best) two instructions for each clock period

Abstraction

Hiding details when they aren't important



The Digital Abstraction

Most physical variables are **continuous**

- Voltage on a wire
- Frequency of an oscillation
- Position of a mass

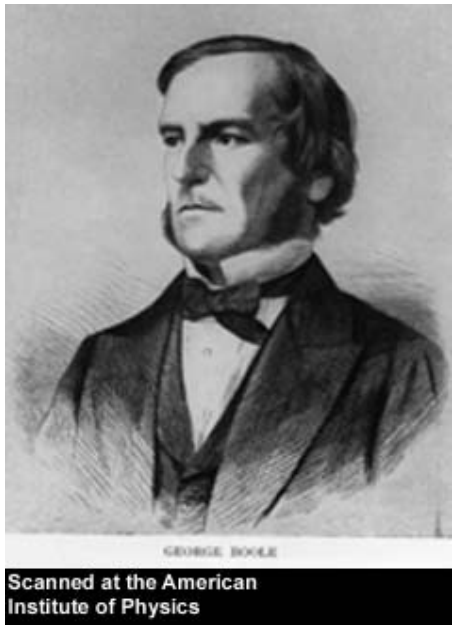
Digital abstraction considers **discrete** subset of values

Digital Discipline: Binary Values

- **Two discrete values**:
  - 1's and 0's
  - 1, TRUE, HIGH
  - 0, FALSE, LOW
- 1 and 0: voltage levels, rotating gears, fluid levels, etc.
- Digital circuits use voltage levels to represent 1 and 0
- **Bit**: Binary digit

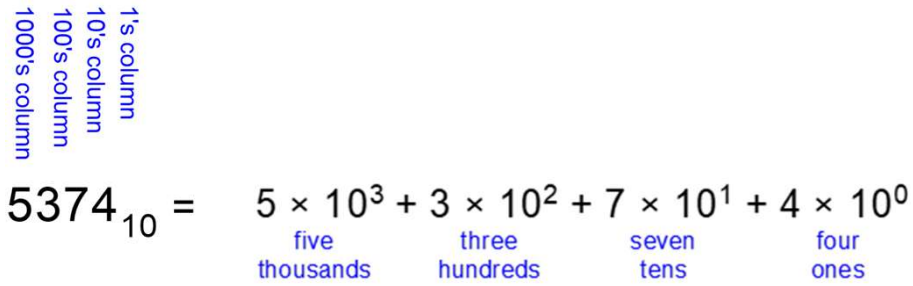
George Boole, 1815-1864

- Born to working class parents
- Taught himself mathematics and joined the faculty of Queen's College in Ireland
- Wrote *An Investigation of the Laws of Thought* (1854)
- Introduced binary variables
- Introduced the three fundamental logic operations: AND, OR, and NOT



Number Systems

- Decimal numbers



- Well, who came up with this brilliant idea? Who discovered this way of representing values and quantities ?
  - In the 12th century, **al-Khwārizmī** introduced *Hindu-Arabic numerals* and their *arithmetic* to the West. It is preserved only in a Latin translation, *Algoritmi de numero Indorum* ("Al-Khwārizmī Concerning the Hindu Art of Reckoning"). From the name of the author, rendered in Latin as Algoritmi, originated the term algorithm. *

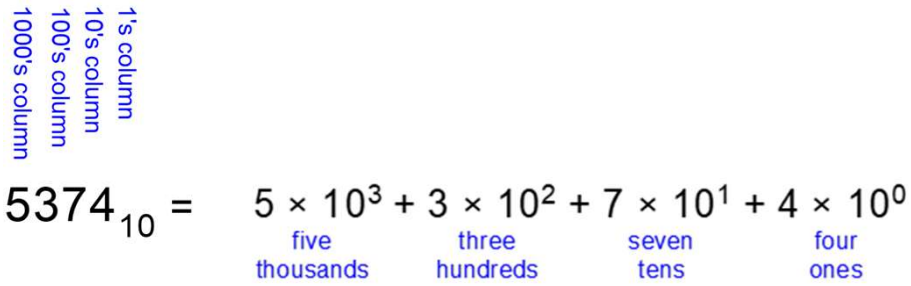
* Brittanica Encyclopedia

Number Systems

Clip from BBC documentary “Science and Islam” – about Al-Khwarizmi

Number Systems

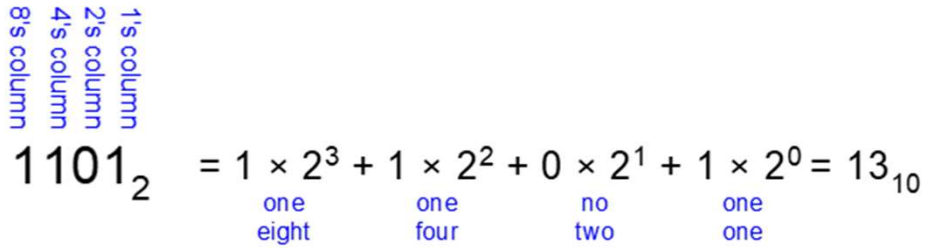
- Decimal numbers



An N-digit decimal number represents one of 10^N possibilities: 0, 1, 2, 3, …, $10^N - 1$. This is called the *range* of the number. For example, a three-digit decimal number represents one of 1000 possibilities in the range of 0 to 999.

Number Systems

- Binary numbers:
  - uses only 1 or 0



Powers of Two

- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$
- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$

Powers of Two

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$
- Handy to memorize up to 2^9

Number Conversion

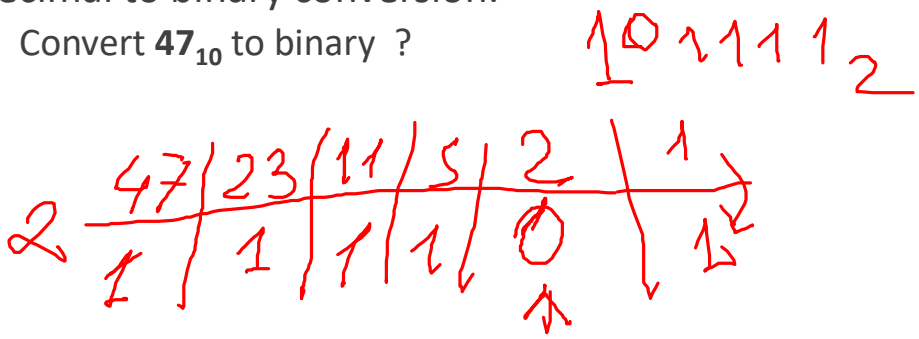
Table 1.1 Binary numbers and their decimal equivalent

1-Bit Binary Numbers	2-Bit Binary Numbers	3-Bit Binary Numbers	4-Bit Binary Numbers	Decimal Equivalents
0	00	000	0000	0
1	01	001	0001	1
	10	010	0010	2
	11	011	0011	3
		100	0100	4
		101	0101	5
		110	0110	6
		111	0111	7
			1000	8
			1001	9
			1010	10
			1011	11
			1100	12
			1101	13
			1110	14
			1111	15

Number Conversion

- Binary to Decimal conversion:
  - Convert **10011_2** to decimal

- Decimal to binary conversion:
  - Convert **47_{10}** to binary ?



Number Systems: Binary Values and Range

- *N*-digit decimal number
  - How many values?
  - Range?
  - Example: 3-digit decimal number:

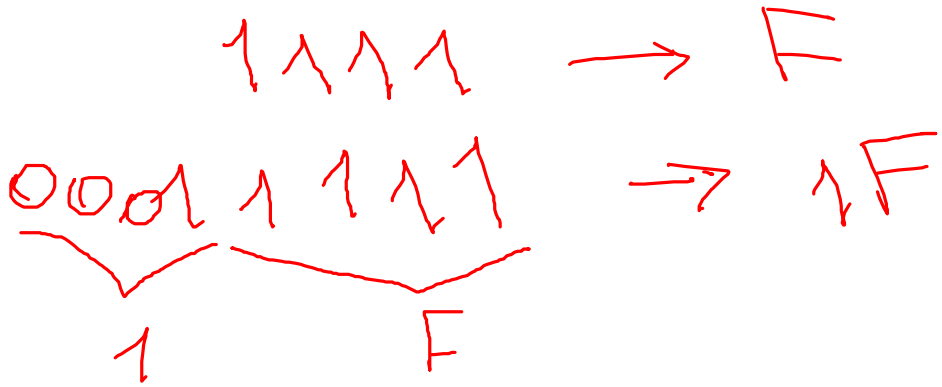
- *N*-bit binary number
  - How many values?
  - Range:
  - Example: 3-digit binary number:

Number Systems: Binary Values and Range

- *N*-digit decimal number
  - How many values? 10^N
  - Range? $[0, 10^N - 1]$
  - Example: 3-digit decimal number:
    - $10^3 = 1000$ possible values
    - Range: [0, 999]
- *N*-bit binary number
  - How many values? 2^N
  - Range: $[0, 2^N - 1]$
  - Example: 3-digit binary number:
    - $2^3 = 8$ possible values
    - Range: $[0, 7] = [000_2$ to $111_2]$

Number Systems

- Writing long binary numbers becomes tedious and prone to error. A group of four bits represents one of 24 = 16 possibilities.
- Hence, it is sometimes more convenient to work in base 16, called **hexadecimal**.
- Hexadecimal numbers use the digits **0** to **9** along with the letters **A** to **F**



Number Systems

Table 1.2 Hexadecimal number system

Hexadecimal Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Number Systems: Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary

0100 1010 1111

12 bit

- Hexadecimal to decimal conversion:
  - Convert 0x4AF to decimal

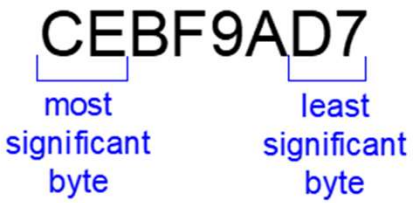
$4 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 = 1199$

Bits, Bytes, Nibbles…

- Bits

- Bytes & Nibbles

- Bytes



32 bit

Large Powers of Two

- 2^{10} = 1 kilo ≈ 1000 (1024)
- 2^{20} = 1 mega ≈ 1 million (1,048,576)
- 2^{30} = 1 giga ≈ 1 billion (1,073,741,824)

Estimating Powers of Two

- What is the value of 2^{24}?

- How many values can a 32-bit variable represent?

Addition

- Decimal

```
   3734
+  5168
------
```

- Binary

```
   1011
+  0011
------
```

Binary Addition Examples

- Add the following 4-bit binary numbers

```
  1001
+ 0101
------
  1110
```

1 + 1 = 10

- Add the following 4-bit binary numbers

```
  1011
+ 0110
------
 10001
```

1

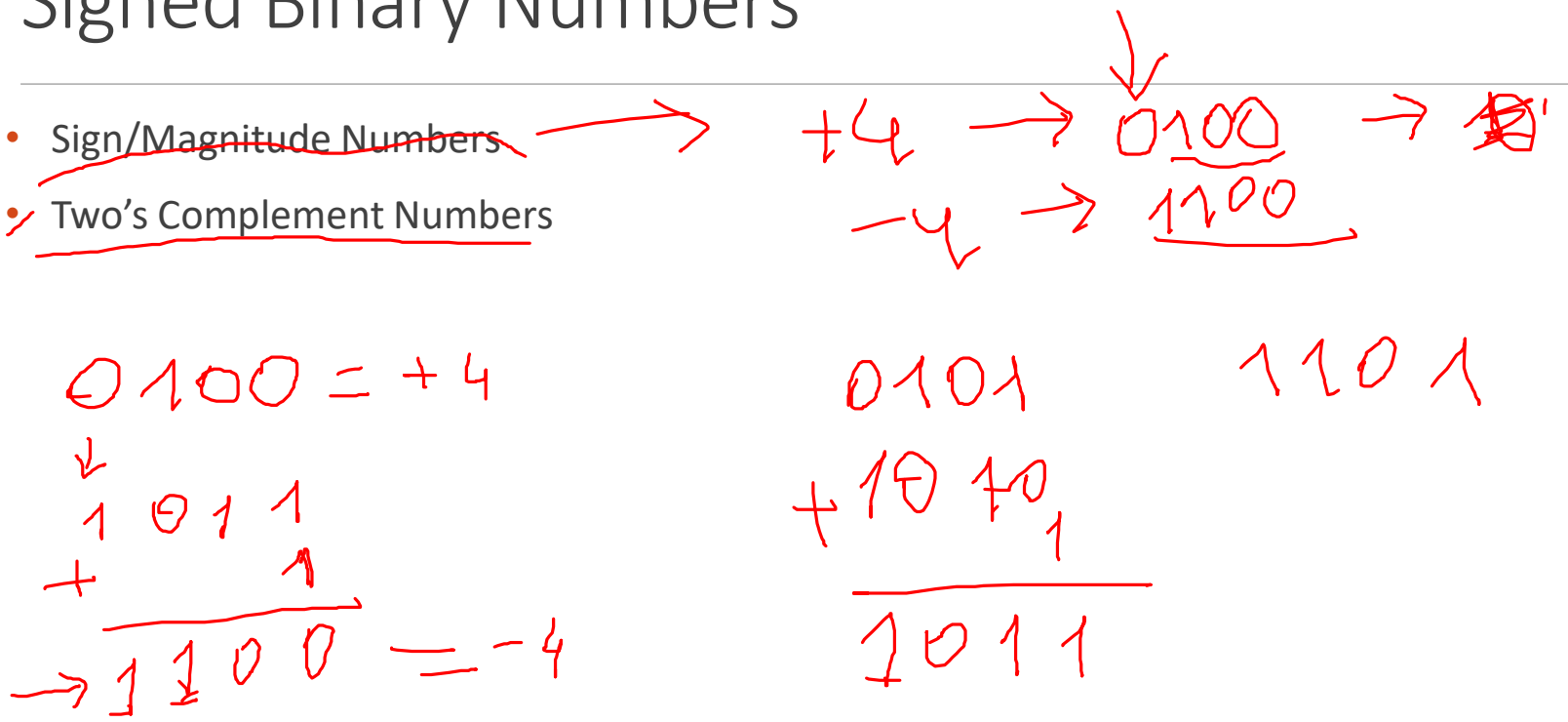
Overflow!

Overflow

- Digital systems operate on a fixed number of bits
- Overflow: when result is too big to fit in the available number of bits
- See previous example of 11 + 6

Signed Binary Numbers

- Sign/Magnitude Numbers
- Two's Complement Numbers



Sign/Magnitude Numbers

- 1 sign bit, *N*-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

- Example, 4-bit sign/mag representations of ± 6:

  +6 =

  - 6 =
- Range of an *N*-bit sign/magnitude number:

Sign/Magnitude Numbers

- 1 sign bit, *N*-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A : \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- Example, 4-bit sign/mag representations of ± 6:

  +6 = **0110**

  - 6 = **1110**

- Range of an *N*-bit sign/magnitude number:

  $[-(2^{N-1}-1), 2^{N-1}-1]$

Sign/Magnitude Numbers

- Problems:
  - Addition doesn't work, for example -6 + 6:

```
  1110      -6
+ 0110      +6
 10100 (wrong!)
```

  - Two representations of 0 (± 0):

```
 1000
 0000
```

Two's Complement Numbers

- Don't have same problems as sign/magnitude numbers:
  - Addition works
  - Single representation for 0

Two's Complement Numbers

- Msb has value of -2^{N-1}

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

- Most positive 4-bit number: **0111**
- Most negative 4-bit number: **1000**
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two's comp number: **$[-(2^{N-1}), 2^{N-1}-1]$**

"Taking the Two's Complement"

- Flip the sign of a two's complement number
- Method:
  1. Invert the bits
  2. Add 1
- Example: Flip the sign of $3_{10} = 0011_2$
  1. **1100**
  2. **+ 1**

  $1101 = -3_{10}$

Two's Complement Examples

- Take the two's complement of $6_{10} = 0110_2$

- What is the decimal value of 1001_2?

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

$$\begin{array}{r} 0110 \\ +\ 1010 \\ \hline \end{array}$$

- Add -2 + 3 using two's complement numbers

$$\begin{array}{r} 1110 \\ +\ 0011 \\ \hline \end{array}$$

Increasing Bit Width

- **Extend number from *N* to *M* bits (*M* > *N*) :**
  - Sign-extension
  - Zero-extension

Sign-Extension

- Sign bit copied to msb's
- Number value is same

- **Example 1:**
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value: 00000011
- **Example 2:**
  - 4-bit representation of -5 = 1011
  - 8-bit sign-extended value: 11111011

Zero-Extension

- Zeros copied to msb's
- Value changes for negative numbers

- **Example 1:**
  - 4-bit value = $0011_2 = 3_{10}$
  - 8-bit zero-extended value: $00000011 = 3_{10}$
- **Example 2:**
  - 4-bit value = $1011 = -5_{10}$
  - 8-bit zero-extended value: $00001011 = 11_{10}$

Number System Comparison

Number System	Range
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation: