

COMPUTER SYSTEMS ARCHITECTURE - COURSE 0x01

**THE EVOLUTION OF COMPUTING SYSTEMS
AND THE BINARY NUMBER SYSTEM**

Cristian Rusu, Bogdan Macovei

CONTENTS

- **brief history of computing systems**
- **the binary system**
 - binary representation
 - two's complement representation
 - logic functions
- **bibliographic references**

A BRIEF HISTORY OF COMPUTING SYSTEMS

- **major contributions by:**
 - Blaise Pascal
 - Gottfried Wilhelm von Leibniz
 - George Boole
 - Charles Babbage
 - Ada Lovelace
 - Konrad Zuse
 - Alan Turing
 - John von Neumann
 - Claude Shannon
 - after 1945, interest in computing systems grew significantly, and progress is difficult to attribute only to individuals

BLAISE PASCAL (1623 - 1662)

- in 1642, before he was 19, he created the **Pascaline**
 - a mechanical calculator
 - capable of addition/subtraction (used for tax calculations)
 - it was not a practical machine
 - fewer than 50 were made
 - it was used as a “toy” by the aristocracy
- the **Pascal language** is named in his honor



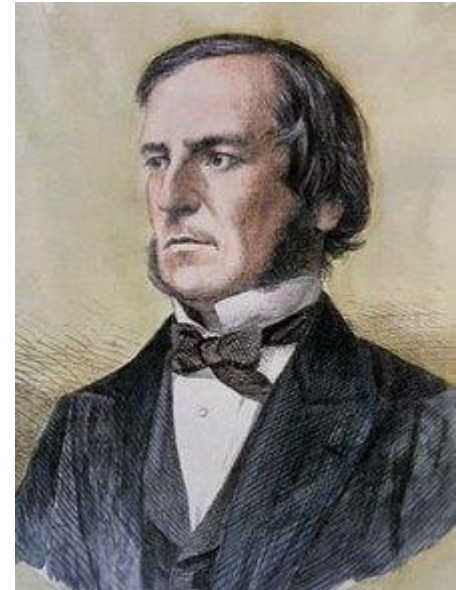
GOTTFRIED WILHELM VON LEIBNIZ (1646 – 1716)

- it is impossible to list all of his contributions
- two major contributions:
 - he studied the binary system
 - he extended Pascal's machine by adding multiplication and division operations — still a mechanical machine, created in 1673



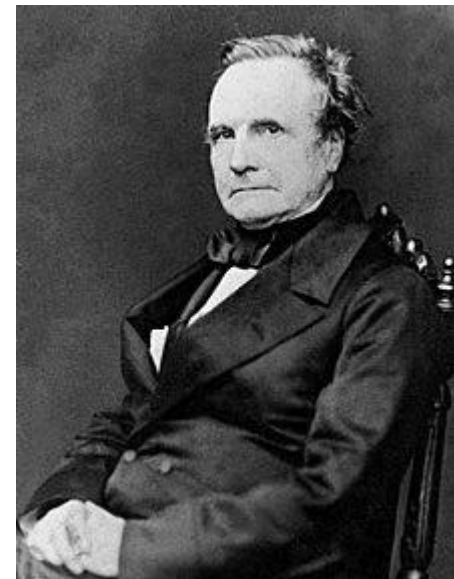
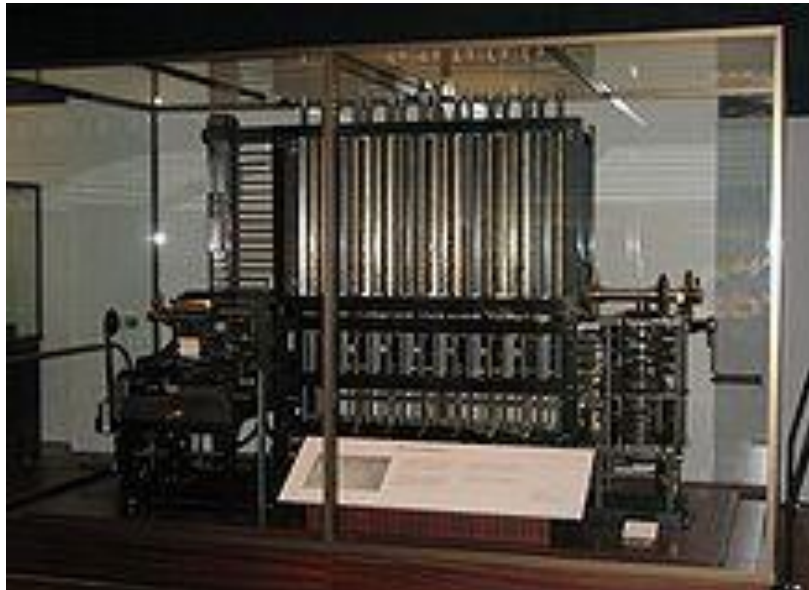
GEORGE BOOLE (1815 – 1864)

- wrote “The Laws of Thought” (1854)
- introduced Boolean logic and analyzed the basic operations
 - negation
 - conjunction
 - disjunction
 - exclusive disjunction
- all of these underlie information theory



CHARLES BABBAGE (1791 – 1871)

- designed Difference Engine No. 2
- only theoretically; the design was built only in 1991
- nevertheless, it is the first programmable (mechanical) computing machine
- its prototypes already weighed more than 13 tons
- he is considered the “father of modern computers”



ADA LOVELACE (1815 – 1852)

- Babbage's collaborator
- wrote the first program, computing Bernoulli numbers
- there were no programming languages, but she described a sequence of steps to be executed by a machine
- she is considered the first “programmer”

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.												Working Variables.												Result Variables.			
						v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}	v_{25}	v_{26}	v_{27}	
						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	$\times$	$v_2 \times v_3$	v_6, v_8, v_9	$v_6 = v_2 \cdot v_3$ $v_8 = v_2 \cdot v_3$ $v_9 = v_2 \cdot v_3$	$= 2n$	2	n	2n	2n	2n																							
2	$-$	$v_6 - v_7$	v_4	$v_4 = v_6 - v_7$	$= 2n - 1$	1																											
3	$+$	$v_4 + v_5$	v_3	$v_3 = v_4 + v_5$	$= 2n + 1$	1																											
4	$+$	$v_3 + v_4$	v_{11}	$v_{11} = v_3 + v_4$	$= 2n - 1$																												
5	$+$	$v_{11} + v_{12}$	v_{11}	$v_{11} = v_{11} + v_{12}$	$= \frac{1}{2} \cdot \frac{2n-1}{2}$	2																											
6	$-$	$v_{11} - v_{10}$	v_{13}	$v_{13} = v_{11} - v_{10}$	$= \frac{1}{2} \cdot \frac{2n-1}{2} = A_n$																												
7	$-$	$v_6 - v_5$	v_{10}	$v_{10} = v_6 - v_5$	$= n - 1 (= 3)$	1																											
8	$+$	$v_5 + v_7$	v_7	$v_7 = v_5 + v_7$	$= 2 + 0 = 2$	2																											
9	$+$	$v_7 + v_{11}$	v_{11}	$v_{11} = v_7 + v_{11}$	$= 2n = A_1$																												
10	$\times$	$v_{11} \times v_{12}$	v_{12}	$v_{12} = v_{11} \times v_{12}$	$= B_1 \cdot \frac{2n-1}{2} = B_1 A_1$																												
11	$+$	$v_{12} + v_{13}$	v_{13}	$v_{13} = v_{12} + v_{13}$	$= \frac{1}{2} \cdot \frac{2n-1}{2} + B_1 \cdot \frac{2n-1}{2}$																												
12	$-$	$v_{10} - v_5$	v_{10}	$v_{10} = v_{10} - v_5$	$= n - 2 (= 2)$	1																											
13	$-$	$v_6 - v_5$	v_8	$v_8 = v_6 - v_5$	$= 2n - 1$	1																											
14	$+$	$v_8 + v_7$	v_7	$v_7 = v_8 + v_7$	$= 2 + 1 = 3$																												
15	$+$	$v_7 + v_8$	v_8	$v_8 = v_7 + v_8$	$= 2n - 1$																												
16	$\times$	$v_8 \times v_{11}$	v_{11}	$v_{11} = v_8 \times v_{11}$	$= \frac{2n-1}{2} \cdot \frac{2n-1}{2}$																												
17	$-$	$v_6 - v_5$	v_8	$v_8 = v_6 - v_5$	$= 2n - 2$	1																											
18	$+$	$v_7 + v_8$	v_7	$v_7 = v_7 + v_8$	$= 3 + 1 = 4$																												
19	$+$	$v_7 + v_8$	v_{10}	$v_{10} = v_7 + v_8$	$= 2n - 2$																												
20	$\times$	$v_{10} \times v_{11}$	v_{11}	$v_{11} = v_{10} \times v_{11}$	$= \frac{2n-1}{2} \cdot \frac{2n-2}{2} = A_2$																												
21	$\times$	$v_{10} \times v_{11}$	v_{10}	$v_{10} = v_{10} \times v_{11}$	$= \frac{2n-1}{2} \cdot \frac{2n-2}{2} = B_2 A_2$																												
22	$+$	$v_{10} + v_{11}$	v_{11}	$v_{11} = v_{10} + v_{11}$	$= A_2 + B_1 A_1 + B_2 A_2$																												
23	$-$	$v_{10} - v_5$	v_{10}	$v_{10} = v_{10} - v_5$	$= n - 3 (= 1)$	1																											
Here follows a repetition of Operations thirteen to twenty-three.																																	
24	$+$	$v_{10} + v_{11}$	v_{24}	$v_{24} = v_{10} + v_{11}$	$= B_2$																												
25	$+$	$v_{11} + v_8$	v_8	$v_8 = v_{11} + v_8$	$= n + 1 = 4 + 1 = 5$	1																											



KONRAD ZUSE (1910 – 1995)

- introduced a series of computers: Z1, Z2, Z3, and Z4
- the first prototypes appeared in 1940–1941
- uses the binary system
- instructions are stored on a tape
- introduced floating-point representation and computation
- did almost everything in isolation (1936–1945)



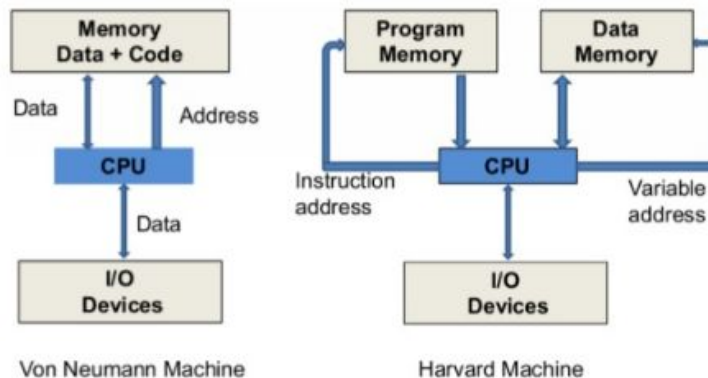
ALAN TURING (1912 – 1954)

- **widely known for his contribution to the rapid breaking of Enigma messages using the “Bombe” machine**
 - in practice, the machine performed a brute-force search to reduce the number of possibilities when decrypting messages
- **introduced the Turing Machine**
 - a theoretical model for implementing any algorithm
 - the concept of Turing-completeness
 - intuition: a system that can recognize and analyze sets of rules for manipulating data (an infinite amount, theoretically)
- **introduced the Turing Test**
 - The imitation game: “The original question, ‘Can machines think!’ I believe to be too meaningless to deserve discussion” A. Turing



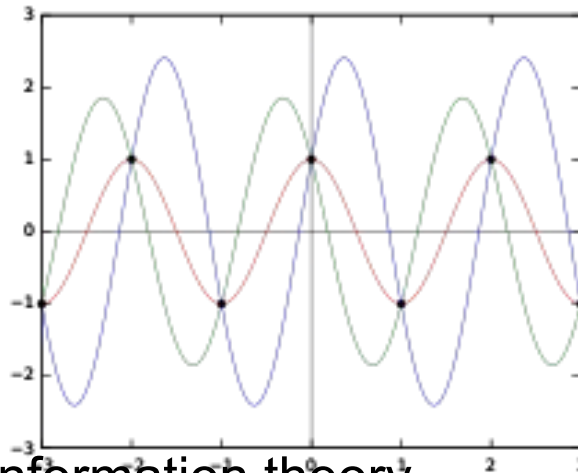
JOHN VON NEUMANN (1903 – 1957)

- considered one of the greatest mathematicians of the last century, with contributions in many fields
- helped create the first electronic computer, ENIAC (Electronic Numerical Integrator And Computer), 1939–1944
- improved ENIAC by helping create EDVAC (Electronic Discrete Variable Automatic Computer); the system is binary and has stored programs
- introduced the von Neumann architecture



CLAUDE SHANNON (1916 – 2001)

- considered the “father of information theory”
- three exceptional contributions:
 - showed that Boolean logic problems can be solved using electronic circuits
 - the Shannon–Nyquist sampling theorem (from analog to digital and back, without losing information)



- invented information theory
 - we will discuss this in detail next lecture



THE BIG IDEAS

- from mechanical to electrical
- from a machine that automatically does one single thing to a programmable machine
- modular design
- theory about what is possible on these machines
- the desire to do things optimally, at the limit, and without waste

AFTER SHANNON ...

- **after World War II, research in computing began to grow exponentially**
- **there were many small technological inventions and discoveries along the way**
- **we cannot cover all of them**
- **the important actors in the field became professional groups (e.g., IEEE, ACM, Bell Labs, etc.) and states (e.g., the United States, DARPA research programs, etc.)**
- **this progress rests on fundamental mathematical concepts; we begin with the binary system**

GUESS WHO ...



THE BINARY SYSTEM

- it is the basis of modern computing systems
- any number (integer or, more generally, real) can be represented by a (potentially infinite) number of bits
- bit = binary digit
- we are in the base- $B = 2$ numeral system
- we have only two available digits: 0 and 1

THE BINARY SYSTEM

- a natural number represented in base $B = 2$

bit b_i :	...	0	1	1	1	1	0	0	0	1
2^i :	...	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $$x = \sum_{i=0}^{N-1} b_i 2^i$$

- *N is the number of bits used in the representation*
- in the example above:
 - what number is represented?
 - how many bits is this number represented on?

THE BINARY SYSTEM

- a natural number represented in base $B = 2$

bit b_i :	...	0	1	1	1	1	0	0	0	1
2^i :	...	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $$x = \sum_{i=0}^{N-1} b_i 2^i$$

- *N is the number of bits used in the representation*
- in the example above:
 - $0 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 241$
 - above we have $N = 9$, but in fact we need $N = 8$

THE BINARY SYSTEM

- our intuition is in base $B = 10$
- it is useful to abstract and consider a general base B
- in base B we have:
 - digits from 0 to $B-1$ (the rest are called numbers)
 - the representation is $x = \sum_{i=0}^{N-1} b_i B^i$
 - bit b_0 is called the Least Significant Bit (LSB), and bit b_{N-1} is called the Most Significant Bit (MSB)
 - how do we represent a number from base 10 in base B ?
 - repeated divisions by B , keeping the remainders

THE BINARY SYSTEM

- an explicit example: $(4215)_{10} = (1000001110111)_2$

2	4215		
2	2107	— 1	← LSB
2	1053	— 1	
2	526	— 1	
2	263	— 0	
2	131	— 1	
2	65	— 1	
2	32	— 1	
2	16	— 0	
2	8	— 0	
2	4	— 0	
2	2	— 0	
2	1	— 0	
	0	— 1	← MSB

THE BINARY SYSTEM

- **conversion between numeral systems is very useful**
- **what happens in base $B = 10$?**
 - what happens if we want to move from base $B_{\text{old}} = 10$ to the new base $B_{\text{new}} = 100$?
 - how many digits are there in base $B_{\text{new}} = 100$?

THE BINARY SYSTEM

- **conversion between numeral systems is very useful**
- **what happens in base $B = 10$?**
 - what happens if we want to move from base $B_{\text{old}} = 10$ to the new base $B_{\text{new}} = 100$?
 - how many digits are there in base $B_{\text{new}} = 100$? 100
 - what are the digits in base $B_{\text{new}} = 100$?

THE BINARY SYSTEM

- conversion between numeral systems is very useful
- what happens in base $B = 10$?
 - what happens if we want to move from base $B_{\text{old}} = 10$ to the new base $B_{\text{new}} = 100$?
 - how many digits are there in base $B_{\text{new}} = 100$? 100
 - what are the digits in base $B_{\text{new}} = 100$? from digit “0” to digit “99”
 - I receive a number in base 10; how do I convert it to base 100?
 - ex: $(4837103)_{10} = (?????)_{100}$

THE BINARY SYSTEM

- conversion between numeral systems is very useful
- what happens in base $B = 10$?
 - what happens if we want to move from base $B_{\text{old}} = 10$ to the new base $B_{\text{new}} = 100$?
 - how many digits are there in base $B_{\text{new}} = 100$? 100
 - what are the digits in base $B_{\text{new}} = 100$? from digit “0” to digit “99”
 - I receive a number in base 10; how do I convert it to base 100?
 - ex: $(4837103)_{10} = (“4” “83” “71” “3”)_{100}$
 - how did we obtain the result? we only grouped consecutive digits two by two — why two?

THE BINARY SYSTEM

- conversion between numeral systems is very useful
- what happens in base $B = 10$?
 - what happens if we want to move from base $B_{\text{old}} = 10$ to the new base $B_{\text{new}} = 100$?
 - how many digits are there in base $B_{\text{new}} = 100$? 100
 - what are the digits in base $B_{\text{new}} = 100$? from digit “0” to digit “99”
 - I receive a number in base 10; how do I convert it to base 100?
 - ex: $(4837103)_{10} = (“4” “83” “71” “3”)_{100}$
 - how did we obtain the result? we only grouped consecutive digits two by two — why two?

General rule: when we move from base B to base B^p , we only need to group the new number into blocks of p digits

THE BINARY SYSTEM

- someone says: “I spent 1,000,000 euros. An enormous digit!”
is this correct?

21:57 56%

m.hotnews.ro/stire/2433680

€ 4,1438 miercuri 07 oct 2020

HotNews.ro
STIRI NON-STOP

Versiune Mobile

Home Ultimele Economie Sport Actual

Coronavirus Franța: Aproape 19.000 noi cazuri în ultimele 24 de ore, o cifră record

de R.M. Miercuri, 7 octombrie 2020, 21:08

- a A+ Intra in cont

f t w e +



Autoritățile sanitare franceze au anunțat miercuri 18.746 noi cazuri de COVID-19 confirmate în ultimele 24 de ore, un nou record și aproape dublu față de bilanțul de marți, de 10.489 cazuri, potrivit agenției Reuters.

THE BINARY SYSTEM

- **someone says: “I spent 1,000,000 euros. An enormous digit!”**
- **1,000,000 is a number, not a digit**
- **when can 1,000,000 be a digit?**
 - only if the speaker is referring to numbers in a numeral base $B \geq 1,000,001$
 - and then they should say “1,000,000”
- **one last time**
 - in base $B = 10$, the digits are from “0” to “9”
 - the rest are numbers
 - the concept generalizes to any B

THE BINARY SYSTEM

- an explicit example: $(1110111000001)_2$
 - in base 4:
 - in base 8:
 - in base 16 (hexadecimal):

0 _{hex}	=	0 _{dec}	=	0 _{oct}	0	0	0	0
1 _{hex}	=	1 _{dec}	=	1 _{oct}	0	0	0	1
2 _{hex}	=	2 _{dec}	=	2 _{oct}	0	0	1	0
3 _{hex}	=	3 _{dec}	=	3 _{oct}	0	0	1	1
4 _{hex}	=	4 _{dec}	=	4 _{oct}	0	1	0	0
5 _{hex}	=	5 _{dec}	=	5 _{oct}	0	1	0	1
6 _{hex}	=	6 _{dec}	=	6 _{oct}	0	1	1	0
7 _{hex}	=	7 _{dec}	=	7 _{oct}	0	1	1	1
8 _{hex}	=	8 _{dec}	=	10 _{oct}	1	0	0	0
9 _{hex}	=	9 _{dec}	=	11 _{oct}	1	0	0	1
A _{hex}	=	10 _{dec}	=	12 _{oct}	1	0	1	0
B _{hex}	=	11 _{dec}	=	13 _{oct}	1	0	1	1
C _{hex}	=	12 _{dec}	=	14 _{oct}	1	1	0	0
D _{hex}	=	13 _{dec}	=	15 _{oct}	1	1	0	1
E _{hex}	=	14 _{dec}	=	16 _{oct}	1	1	1	0
F _{hex}	=	15 _{dec}	=	17 _{oct}	1	1	1	1

THE BINARY SYSTEM

- an explicit example: $(1110111000001)_2$
 - in base 4: $(\text{"1" "11" "01" "11" "00" "00" "01"})_4 = (1313001)_4$
 - in base 8:
 - in base 16 (hexadecimal):

0 _{hex}	=	0 _{dec}	=	0 _{oct}	0	0	0	0
1 _{hex}	=	1 _{dec}	=	1 _{oct}	0	0	0	1
2 _{hex}	=	2 _{dec}	=	2 _{oct}	0	0	1	0
3 _{hex}	=	3 _{dec}	=	3 _{oct}	0	0	1	1
4 _{hex}	=	4 _{dec}	=	4 _{oct}	0	1	0	0
5 _{hex}	=	5 _{dec}	=	5 _{oct}	0	1	0	1
6 _{hex}	=	6 _{dec}	=	6 _{oct}	0	1	1	0
7 _{hex}	=	7 _{dec}	=	7 _{oct}	0	1	1	1
8 _{hex}	=	8 _{dec}	=	10 _{oct}	1	0	0	0
9 _{hex}	=	9 _{dec}	=	11 _{oct}	1	0	0	1
A _{hex}	=	10 _{dec}	=	12 _{oct}	1	0	1	0
B _{hex}	=	11 _{dec}	=	13 _{oct}	1	0	1	1
C _{hex}	=	12 _{dec}	=	14 _{oct}	1	1	0	0
D _{hex}	=	13 _{dec}	=	15 _{oct}	1	1	0	1
E _{hex}	=	14 _{dec}	=	16 _{oct}	1	1	1	0
F _{hex}	=	15 _{dec}	=	17 _{oct}	1	1	1	1

THE BINARY SYSTEM

- an explicit example: $(1110111000001)_2$
 - in base 4: $(“1” “11” “01” “11” “00” “00” “01”)_4 = (1313001)_4$
 - in base 8: $(“1” “110” “111” “000” “001”)_8 = (16701)_8$
 - in base 16 (hexadecimal):

0 _{hex}	=	0 _{dec}	=	0 _{oct}	0	0	0	0
1 _{hex}	=	1 _{dec}	=	1 _{oct}	0	0	0	1
2 _{hex}	=	2 _{dec}	=	2 _{oct}	0	0	1	0
3 _{hex}	=	3 _{dec}	=	3 _{oct}	0	0	1	1
4 _{hex}	=	4 _{dec}	=	4 _{oct}	0	1	0	0
5 _{hex}	=	5 _{dec}	=	5 _{oct}	0	1	0	1
6 _{hex}	=	6 _{dec}	=	6 _{oct}	0	1	1	0
7 _{hex}	=	7 _{dec}	=	7 _{oct}	0	1	1	1
8 _{hex}	=	8 _{dec}	=	10 _{oct}	1	0	0	0
9 _{hex}	=	9 _{dec}	=	11 _{oct}	1	0	0	1
A _{hex}	=	10 _{dec}	=	12 _{oct}	1	0	1	0
B _{hex}	=	11 _{dec}	=	13 _{oct}	1	0	1	1
C _{hex}	=	12 _{dec}	=	14 _{oct}	1	1	0	0
D _{hex}	=	13 _{dec}	=	15 _{oct}	1	1	0	1
E _{hex}	=	14 _{dec}	=	16 _{oct}	1	1	1	0
F _{hex}	=	15 _{dec}	=	17 _{oct}	1	1	1	1

THE BINARY SYSTEM

- an explicit example: $(1110111000001)_2$
 - in base 4: $(“1” “11” “01” “11” “00” “00” “01”)_4 = (1313001)_4$
 - in base 8: $(“1” “110” “111” “000” “001”)_8 = (16701)_8$
 - in base 16 (hexadecimal): $(“1” “1101” “1100” “0001”)_{16} = (1DC1)_{16}$

0 _{hex}	=	0 _{dec}	=	0 _{oct}	0	0	0	0
1 _{hex}	=	1 _{dec}	=	1 _{oct}	0	0	0	1
2 _{hex}	=	2 _{dec}	=	2 _{oct}	0	0	1	0
3 _{hex}	=	3 _{dec}	=	3 _{oct}	0	0	1	1
4 _{hex}	=	4 _{dec}	=	4 _{oct}	0	1	0	0
5 _{hex}	=	5 _{dec}	=	5 _{oct}	0	1	0	1
6 _{hex}	=	6 _{dec}	=	6 _{oct}	0	1	1	0
7 _{hex}	=	7 _{dec}	=	7 _{oct}	0	1	1	1
8 _{hex}	=	8 _{dec}	=	10 _{oct}	1	0	0	0
9 _{hex}	=	9 _{dec}	=	11 _{oct}	1	0	0	1
A _{hex}	=	10 _{dec}	=	12 _{oct}	1	0	1	0
B _{hex}	=	11 _{dec}	=	13 _{oct}	1	0	1	1
C _{hex}	=	12 _{dec}	=	14 _{oct}	1	1	0	0
D _{hex}	=	13 _{dec}	=	15 _{oct}	1	1	0	1
E _{hex}	=	14 _{dec}	=	16 _{oct}	1	1	1	0
F _{hex}	=	15 _{dec}	=	17 _{oct}	1	1	1	1

on an earlier slide I said that “99” is a digit in base 100, because we do not have letters up to 99

for “99” we can continue through the extended ASCII list (starting from F, which is “15”) until we reach “99”: š

THE BINARY SYSTEM

- **negative integers**
 - so far we have only seen natural numbers
 - what would you do to represent negative numbers?
 - what is the first (simplest) idea?

THE BINARY SYSTEM

- **negative integers**
 - so far we have only seen natural numbers
 - what would you do to represent negative numbers?
 - what is the first (simplest) idea?
 - we must store the sign of the number
 - how much space does this take? 1 bit
 - so 1 bit for the sign, and the rest for the absolute value
 - 1 101 would be -5
 - 0 101 would be 5
 - what does “zero” look like when represented this way?
 - 1 000 would be -0
 - 0 000 would be 0
 - it is redundant
 - and there is another problem: we need special circuits to perform operations with these numbers (the first bit must be checked, and the operations must be chosen based on that)

THE BINARY SYSTEM

- negative integers

bit b_i :	1	1	1	1	0	0	0	1
2^i :	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- what happens? The MSB is negative

- $$x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$$

- in the example above:
 - what number is represented?
 - how many bits is this number represented on?

THE BINARY SYSTEM

- negative integers

bit b_i :	1	1	1	1	0	0	0	1
2^i :	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- what happens? The MSB is negative

- $$x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$$

- in the example above:

- $-1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = -15$
- 8 bits

- this is the two's complement representation system

THE BINARY SYSTEM

- negative integers

bit b_i :	1	1	1	1	0	0	0	1
2^i :	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $$x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$$

- the representation is called two's complement

- numbers in the interval from -2^{N-1} to $2^{N-1} - 1$
- one bit is “lost” for the sign; it is optimal
- the MSB is the sign; the remaining bits are the value
- in order to use signed numbers in

arithmetic operations, we need to perform some transformation

see the table on the right; positive numbers are the same as before (but on only 3 bits); negative numbers look the same except that the first bit is “1” (the upper half of the table has MSB “0”; the lower half is identical, but with MSB “1”)

two's complement	decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

what happens if we add 1 to 7?

THE BINARY SYSTEM

- negative integers

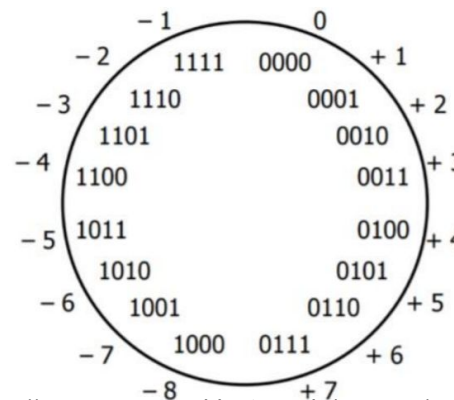
bit b_i :	1	1	1	1	0	0	0	1
2^i :	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $$x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$$

- the representation is called two's complement

- numbers in the interval from -2^{N-1} to $2^{N-1} - 1$
- one bit is "lost" for the sign; it is optimal
- the MSB is the sign; the remaining bits are the value
- in order to use signed numbers in

arithmetic operations, we need to perform some transformation



two's complement	decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

THE BINARY SYSTEM

- negative integers

bit b_i :	1	1	1	1	0	0	0	1
2^i :	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $$x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$$

- how do we represent a negative decimal number x ? (ex: $x = -30$)

- write $|x|$ in binary

- set the MSB and

invert the remaining bits

- add one

	0	0	1	1	1	1	0
1	1	1	0	0	0	0	1
1	1	1	0	0	0	1	0

- deci $(-30)_{10} = (11100010)_2$

THE BINARY SYSTEM

- negative integers

bit b_i :	1	1	1	1	0	0	0	1
2^i :	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $$x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$$

- how do we interpret a binary number x ? (ex: $x = 1011\ 1010$)

- write x in binary

1	0	1	1	1	0	1	0
---	---	---	---	---	---	---	---

- MSB = 1, so x is negative (if MSB = 0, then x is positive)

- invert the bits

0	1	0	0	0	1	0	1
0	1	0	0	0	1	1	0

- add one

- therefore $(10111010)_2 = (-70)_{10}$ (negative, $64 + 4 + 2 = 70$)

THE BINARY SYSTEM

- arithmetic operations, natural-number example

	1	0	1	1	0	0	1	0	178
	0	0	1	1	0	1	1	0	54
+	1	1	1	0	1	0	0	0	232

THE BINARY SYSTEM

- **why do we use this two's complement system?**
- **because the addition algorithm is the same as for natural numbers; we do not need to change anything**
- **the previous circuits that implement natural-number addition can still be used**

THE BINARY SYSTEM

- arithmetic operations, integer examples
 - the negative number has smaller magnitude than the positive one

	1	1	0	1	1	0	0	1	-39
	0	1	0	1	1	1	0	0	92
+	0	0	1	1	0	1	0	1	53

- how did we obtain the representation for -39?

	0	1	0	0	1	1	1	39
1	1	0	1	1	0	0	0	bit inversion
1	1	0	1	1	0	0	1	plus one

THE BINARY SYSTEM

- arithmetic operations, integer examples
 - the negative number has greater magnitude than the positive one

	1	1	0	1	1	0	0	1	-39
	0	0	0	1	0	1	0	1	21
+	1	1	1	0	1	1	1	0	-18

- where did -18 come from?

1	1	1	0	1	1	1	0	the result: the first bit is
0	0	0	1	0	0	0	1	1 bit inversion
0	0	0	1	0	0	1	0	plus one

THE BINARY SYSTEM

- arithmetic operations, integer examples
 - both numbers are negative

	1	1	0	1	1	0	0	1	-39
	1	1	1	0	1	1	1	0	-18
+	1	1	0	0	0	1	1	1	-57

- where did -57 come from?

1	1	0	0	0	1	1	1	the result: the first bit is 1 bit inversion
0	0	1	1	1	0	0	0	
0	0	1	1	1	0	0	1	plus one

THE BINARY SYSTEM

- arithmetic operations, integer examples
 - overflow at the boundary

	0	1	1	1	1	1	1	127
	0	0	0	0	0	0	1	1
+	1	0	0	0	0	0	0	-128

- where did -128 come from?

1	0	0	0	0	0	0	0	the result: the first bit is 1 bit inversion
0	1	1	1	1	1	1	1	
1	0	0	0	0	0	0	0	plus one: 128

-128 has the same
representation

THE BINARY SYSTEM

- arithmetic operations, integer examples
 - both numbers are positive and large

	0	1	1	1	0	1	1	1	119
	0	1	0	0	1	0	1	1	75
+	1	1	0	0	0	0	1	0	-62

- where did -62 come from?

1	1	0	0	0	0	1	0	the result: the first bit is 1 bit inversion
0	0	1	1	1	1	0	1	
0	0	1	1	1	1	1	0	plus one

- $119 + 75 = 194$ (does not fit on 7 bits)

THE BINARY SYSTEM

- **arithmetic operations, integer examples**
 - both numbers are positive and large

	0	1	1	1	0	1	1	1	119
	0	1	0	0	1	0	1	1	75
+	1	1	0	0	0	0	1	0	-62

- **intuition: where did -62 come from?**
 - $119 + 75 = 194$ (does not fit on 7 bits)
 - the maximum is 127, so we have $194 - 127 = 67$ “extra”
 - overflow starts after 127; after 127 comes -128 (we use one extra)
 - therefore the remaining 66 extra start from -128
 - therefore we have $-128 + 66 = -62$

THE BINARY SYSTEM

- arithmetic operations, integer examples
 - overflow at the boundary

	0	1	1	1	1	1	1	127
	0	0	0	0	0	0	1	1
+	1	0	0	0	0	0	0	-128

- intuition: where did -128 come from?
 - $127 + 1 = 128$ (does not fit on 7 bits)
 - the maximum is 127, so we have $128 - 127 = 1$ “extra”
 - overflow starts after 127; after 127 comes -128 (we use one extra)
 - therefore the remaining 0 extra start from -128
 - therefore we have $-128 + 0 = -128$

THE BINARY SYSTEM

- **arithmetic operations, integer examples**
 - both numbers are negative and large

	1	0	0	0	1	0	0	1	-119
	1	0	1	1	0	1	0	1	-75
+	0	0	1	1	1	1	1	0	62

- **intuition: where did 62 come from?**
 - $-119 - 75 = -194$ (does not fit on 7 bits)
 - the minimum is -128 , so we have $+194 - 128 = 66$ “extra”
 - underflow starts before -128 ; before -128 comes 127 (we use one extra)
 - therefore the remaining 65 extra start from 127
 - therefore we have $127 - 65 = 62$

THE BINARY SYSTEM

- **extending the number of bits used for representation**
 - suppose we have the number 01 11 11 represented on 6 bits
 - we are told that the number is natural
 - we are told that we may use 2 more bits for the representation
 - the new representation is:

THE BINARY SYSTEM

- **extending the number of bits used for representation**
 - suppose we have the number 01 11 11 represented on 6 bits
 - we are told that the number is natural
 - we are told that we may use 2 more bits for the representation
 - the new representation is: 0001 1111
 - what happens now if we have the number 10 00 11 represented on 6 bits (but we know that we are using two's complement)
 - we are told again that we may use 2 more bits for the representation
 - the new representation is: 0010 0011 ?

THE BINARY SYSTEM

- **extending the number of bits used for representation**
 - suppose we have the number 01 11 11 represented on 6 bits
 - we are told that the number is natural
 - we are told that we may use 2 more bits for the representation
 - the new representation is: 0001 1111
- what happens now if we have the number 10 00 11 represented on 6 bits (but we know that we are using two's complement)
- we are told again that we may use 2 more bits for the representation
 - the new representation is: 0010 0011 ?
 - this number is not even negative (the MSB is 0)
 - so we cannot extend with “zero”
 - **what do we extend with?**

THE BINARY SYSTEM

- **extending the number of bits used for representation**
 - suppose we have the number 01 11 11 represented on 6 bits
 - we are told that the number is natural
 - we are told that we may use 2 more bits for the representation
 - the new representation is: 0001 1111
- what happens now if we have the number 10 00 11 represented on 6 bits (but we know that we are using two's complement)
- we are told again that we may use 2 more bits for the representation
 - the new representation is: 0010 0011 ?
 - this number is not even negative (the MSB is 0)
 - so we cannot extend with “zero”
 - what do we extend with? with “one”
 - the new representation is: 1110 0011

THE BINARY SYSTEM

- **one more example:**
- **extending the number of bits used for representation**
 - suppose we have the number 10111 represented on 5 bits
 - we move to base $B = 8$
 - the number in base $B = 8$ is $10\ 111 = 27$
 - but if we see $(27)_8$, then we might think that in binary we have 010111
 - but 010111 is on 6 bits and is positive
 - **idea: when we move from binary to base $B = 8$ (and we know that here we implicitly have 6 representation bits), we extend the representation**
 - therefore, we start with 110 111
 - and in base $B = 8$ we then have $110\ 111 = 67$
- the problem is caused by the fact that 3 does not divide 5 exactly
- both variants are correct: $(27)_8$ if you know there are 5 bits, or $(67)_8$ if you assume there are 6 bits (implicitly, when you see two digits in base $B = 8$, you assume there are 6 bits)

THE BINARY SYSTEM

- **binary logic** (0 = False, 1 = True)
- **logic operations:**
 - NOT (negation)
 - AND (conjunction)
 - OR (disjunction)
 - XOR (exclusive disjunction)

X	NOT X
0	1
1	0

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

- **for numbers represented in binary, the logic operation is performed bit by bit (for N-bit numbers, there are N operations)**

THE BINARY SYSTEM

- **binary logic, example**

	1	0	1	1	0	0	1	0	178
	0	0	1	1	0	1	1	0	54
AND:	0	0	1	1	0	0	1	0	50
OR:	1	0	1	1	0	1	1	0	182
XOR:	1	0	0	0	0	1	0	0	132

- **apparently, the decimal values do not have a clear interpretation**
- **nevertheless, we can say something: OR encourages the occurrence of “1” bits, AND discourages it, and for XOR ... it depends**
- **nevertheless, we will see that binary logic (in interesting combinations) can also tell us a lot about decimal numbers**

WHAT WE DID TODAY

- **we discussed the history of computing systems**
- **we covered the basics of the binary system**
 - binary representation (natural numbers)
 - calculations with different bases B
 - two's complement representation (integers)
 - basic binary operations
- **in the seminar, we will see how to perform operations in the binary system and how to use two's complement**

NEXT TIME ...

- **introduction to information theory**
- **how we measure the amount of information**
- **Shannon entropy**
- **we continue studying the binary system**
- **... later: multiplication and division**
- **... and later: floating point**

FURTHER READING

- **PH book**
 - 2.4 Signed and Unsigned Numbers
 - 2.17 Real Stuff: x86 Instructions
 - 3.2 Addition and Subtraction
- **Thomas Finley course notes,**
<https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html>
- **the Turing Machine and the concept of Turing-completeness:**
 - Turing Machines Explained – Computerphile,
<https://youtube.com/watch?v=dNRDvLACg5Q>
 - Turing Complete – Computerphile,
<https://www.youtube.com/watch?v=RPQD7-AOjMI>

FURTHER READING

(NOT INCLUDED IN THE EXAM)

- if you are interested in the history of computing systems, I suggest the following presentations:
 - Computer Pioneers: Pioneer Computers Part 1 (1935 – 1945),
<https://www.youtube.com/watch?v=qundvme1Tik>
 - Computer Pioneers: Pioneer Computers Part 2 (1946 – 1950),
<https://www.youtube.com/watch?v=wsirYCAocZk>
 - BBC History of Computers,
<https://www.youtube.com/playlist?list=PL1331A4548513EA81>
 - <https://www.gresham.ac.uk/lectures-and-events/a-very-brief-history-of-computing-1948-2015>
 - The Grand Narrative of the History of Computing,
<https://www.youtube.com/watch?v=njwQgz63rls>

