COM1006 Devices and Networks (Autumn)
COM1090 Computer Architectures

Lecture #1

▶ Digital systems

Dr Dirk Sudholt
Department of Computer Science
University of Sheffield

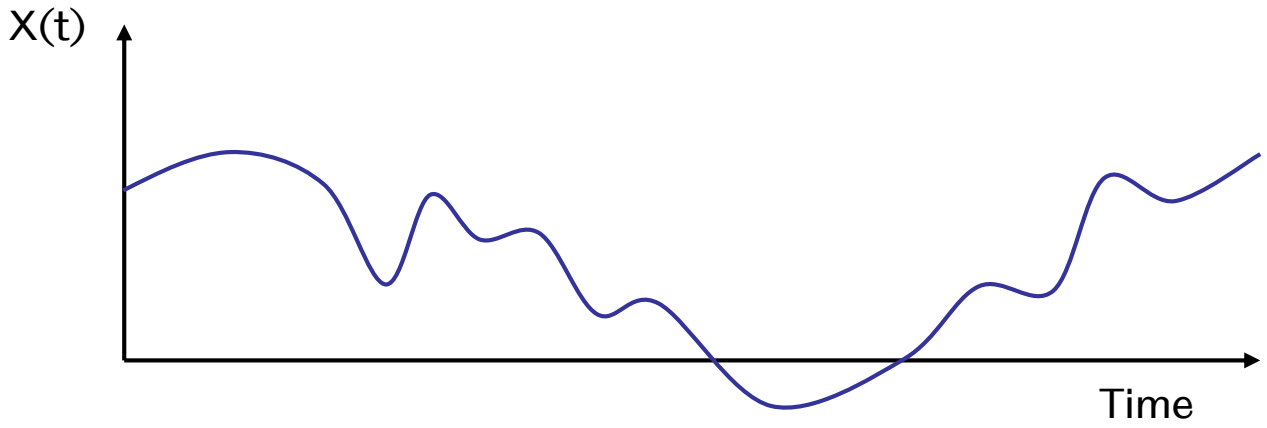
d.sudholt@sheffield.ac.uk
http://staffwww.dcs.shef.ac.uk/~dirk/campus_only/com1006/

Aims of this lecture

- To explain the distinction between analog and digital systems
- To show how data is stored in a computer
- To introduce binary and hexadecimal numbers
- To show how numbers can be converted between number systems

Analog systems

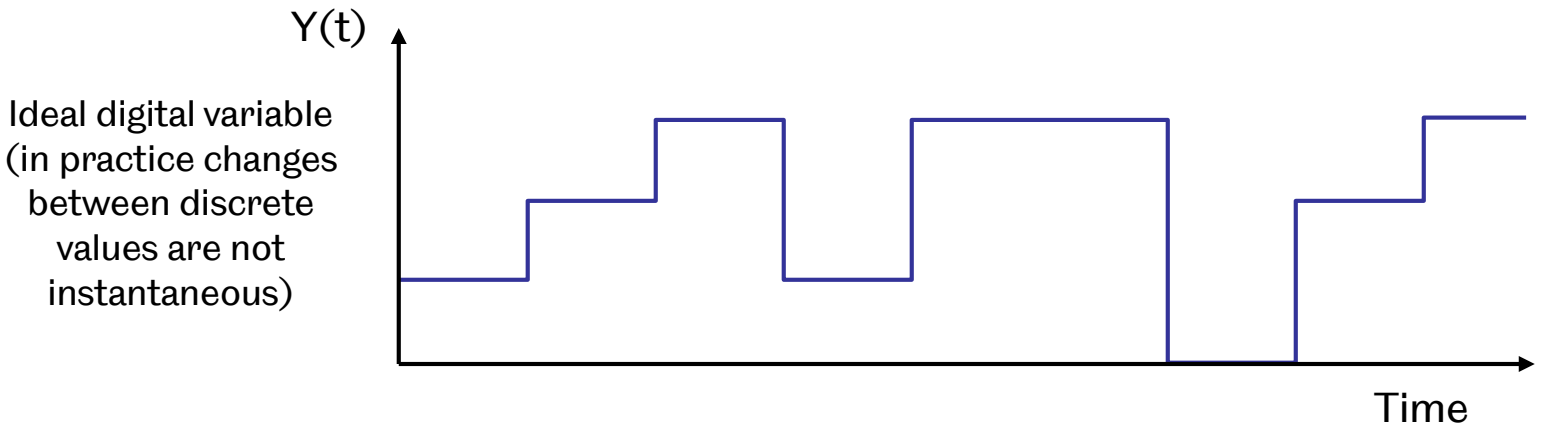
- Analog electronic circuits represent physical quantities in terms of voltages and currents:
  - Analog variables are **continuous**, vary between maximum and minimum values by arbitrarily small amounts



  - Can you think of examples of analog systems?

Digital systems

- Information in a computer is represented in digital form:
  - Digital variables are **discrete** in value and time.
  - Variables in a digital system take a value from a set of values called an **alphabet**.
  - Digital computers use a **binary** system in which the alphabet is composed of two symbols, 0 and 1 (logical false and true, low and high, off and on).



Logic values and signal levels

- Logic variables assume one of two discrete states, 0 or 1.
- Represented by low/high voltage.
- In a system using a 5V power supply you might expect a variable to be represented by exactly 0V or 5V:
  - Such precise electronic devices cannot be made cheaply
  - In practice, two **ranges** of voltage are used to represent logic values, e.g. 0–0.4V for 0 state and 2.8–5V for 1 state.

What is the advantage of representing data in digital form?

Why do computers use only two values, 0 and 1?

Bits, words and bytes

- A **bit** is the smallest unit of information in a computer, and takes the value 0 or 1.
- Computers store information as groups of bits called **words**.
- The trend is towards increasing word length (1st generation microprocessors were 8 bit; 4th generation are 64 bit).

What is the advantage of a 64-bit word length over an 8-bit word length?

- A group of 8 bits is called a **byte**.

Representing data in binary

- An n-bit word can take the value of 2^n unique bit patterns.
- These bit patterns have no intrinsic meaning; the meaning of the bits is determined by the programmer.

What entities could a word represent?

- Natural numbers
- Text
- Computer programs
- Pictures
- Music, sound, voice, movies, …

Representing text

How to represent text in a computer?

- 26 letters A...Z can be encoded by 5 bits.

THISALPHABETMIGHTBEABITSMALL

- What's missing: lower case letters, punctuation, digits, ...
- ASCII Code (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)
  - established in 1963 by the American Standards Organization
  - uses **7 bits** to represent **128 characters**

ASCII Code

	000	001	010	011	100	101	110	111
0000...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
0001...	BS	HT	LF	VT	FF	CR	SO	SI
0010...	DLE	DC1	XON	DC3	XOF	NAK	SYN	ETB
0011...	CAN	EM	SUB	ESC	FS	GS	RS	US
0100...		!	"	#	$	%	&	'
0101...	(	)	*	+	,	-	.	/
0110...	0	1	2	3	4	5	6	7
0111...	8	9	:	;	<	=	>	?
1000...	@	A	B	C	D	E	F	G
1001...	H	I	J	K	L	M	N	O
1010...	P	Q	R	S	T	U	V	W
1011...	X	Y	Z	[	\	]	^	_
1100...	`	a	b	c	d	e	f	g
1101...	h	i	j	k	l	m	n	o
1110...	p	q	r	s	t	u	v	w
1111...	x	y	z	{	|	}	~	DEL

ASCII Code: Extensions

- 7 bits leaves one more bit of a byte
- Further 128 characters can be encoded
- Different extensions are available:
- ISO-8859-1 or ISO Latin1 includes
  - £
  - umlauts: äöüÄÖÜß
  - accented characters

Unicode

- International standard with 1,112,064 codepoints
- Unicode characters can have 1-6 bytes.
- UTF-8 is a standard in Java, E-Mail, WWW, etc.
- First 128 characters in UTF-8 are ASCII codes.

Number bases

- Numbers can be represented in different number systems.
- Decimal system uses digits 0,1,...,9.
- The decimal number 904531_{10} (base 10) can be written:

$$9x10^5+0x10^4+4x10^3+5x10^2+3x10^1+1x10^0$$

- The **position** of a digit and the **base** of the representation determine the magnitude.
- Computers use base 2 (**binary**), e.g. the binary number 1011010_2 has a value:

$$1x2^6+0x2^5+1x2^4+1x2^3+0x2^2+1x2^1+0x2^0$$

- Manual conversion between number bases is a useful skill, e.g. for assembly programming.

Counting in binary

decimal	binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

Conversion: decimal to binary

- How to convert decimal numbers to binary numbers?
- What is 3_{10} in binary?

$$3_{10} = 11_2$$

- What is 18_{10} in binary?

$$18_{10} = 10010_2$$

- What is 245_{10} in binary?

Conversion: decimal to binary

Algorithm:

- Successively divide the number by 2 and record the remainder (which is either 0 or 1).
- Stop when the result of the division is 0.
- Remainders read "backwards" give binary number.

Example:

- **Convert 245_{10} to binary**

245 / 2 = 122	R = 1
122 / 2 = 61	R = 0
61 / 2 = 30	R = 1
30 / 2 = 15	R = 0
15 / 2 = 7	R = 1
7 / 2 = 3	R = 1
3 / 2 = 1	R = 1
1 / 2 = 0	R = 1

The result is read upwards:

$245_{10} = 11110101_2$

Why does this work?

- **Convert 245_{10} to binary**

245 / 2 = 122	R = 1	$245_{10} = 11110101_2$
122 / 2 = 61	R = 0	$122_{10} = 1111010_2$
61 / 2 = 30	R = 1	$61_{10} = 111101_2$
30 / 2 = 15	R = 0	$30_{10} = 11110_2$
15 / 2 = 7	R = 1	$15_{10} = 1111_2$
7 / 2 = 3	R = 1	$7_{10} = 111_2$
3 / 2 = 1	R = 1	$3_{10} = 11_2$
1 / 2 = 0	R = 1	$1_{10} = 1_2$

- Remainder of division by 2 identifies last bit
- Integer division by 2 removes last bit
- Algorithm successively identifies bits from right to left

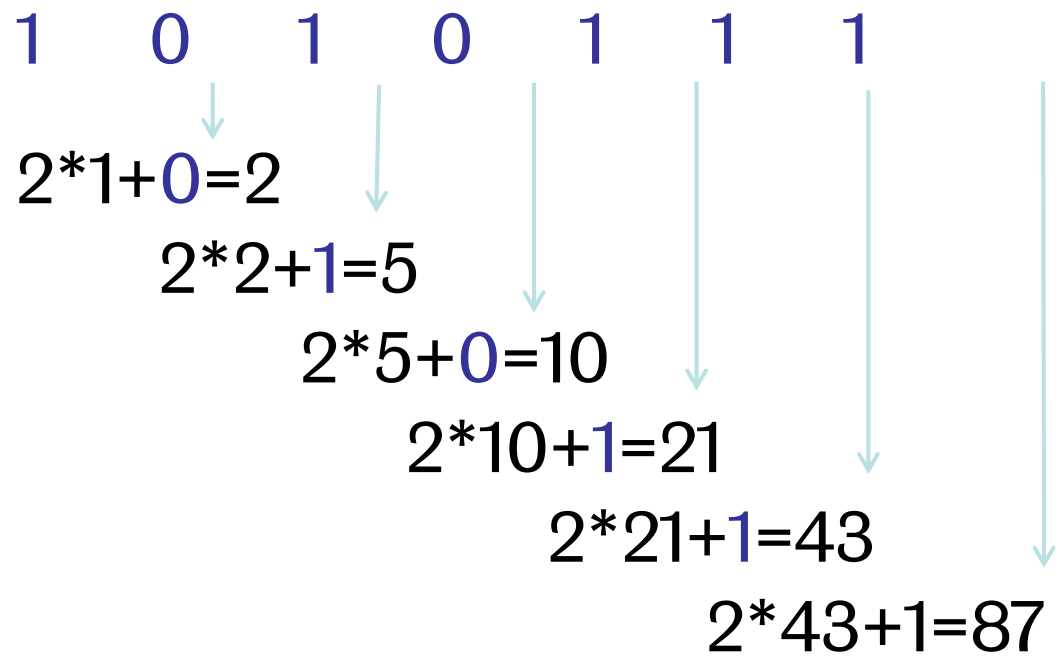
Conversion: binary to decimal

Conversion: binary to decimal

Algorithm:

- Multiply the first non-zero bit by 2 and add the bit on its right.
- Continue with the remaining bits and stop after adding the smallest bit.

- **Convert 1010111_2 to decimal:**



- Therefore $1010111_2 = 87_{10}$

Why does this work?

Binary	Decimal
10_2	2*1+0=2
101_2	2*2+1=5
1010_2	2*5+0=10
10101_2	2*10+1=21
101011_2	2*21+1=43
1010111_2	2*43+1=87

- Multiplication by 2 shifts all bits to the left.
- Add the next bit in decimal also adds it in binary.
- Subsequently converting bits from left to right.

Hexadecimal numbers

- **Exercise**: remember 0011101000001111_2
- Binary numbers can be long and cumbersome!
- **Hexadecimal system**: base 16
- Digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- One hexadecimal digit represents **4 bits**.
- One byte can be expressed by 2 hexadecimal digits.
- Common notations: $3B_{16}$ = 0x3B = $3B

Hexadecimal vs. binary

Binary	Hexadecimal
0000_2	0_{16}
0001_2	1_{16}
0010_2	2_{16}
0011_2	3_{16}
0100_2	4_{16}
0101_2	5_{16}
0110_2	6_{16}
0111_2	7_{16}

Binary	Hexadecimal
1000_2	8_{16}
1001_2	9_{16}
1010_2	A_{16}
1011_2	B_{16}
1100_2	C_{16}
1101_2	D_{16}
1110_2	E_{16}
1111_2	F_{16}

- Conversion: look up blocks of 4 bits in the above table.
- What is $0011\ 1010\ 0000\ 1111_2$ in hexadecimal?

Summary

- Analog variables are continuous, digital variables are discrete
- Boolean variables (0 and 1) can be used to encode numbers, text, programs, and much more.
- Learned how to convert decimal numbers into binary and vice versa.
- Hexadecimal numbers are a more compact representation for binary numbers.