Data Representation

BIT211: Computer Systems Organization

Learning outcomes

- By the end of this Chapter you will be able to:
  - Explain how integers are represented in computers using:
    - Unsigned, signed magnitude, excess, and two’s complement notations
  - Calculate the decimal value represented by a binary sequence in:
    - Unsigned, signed notation, excess, and two’s complement.
  - Explain how characters are represented in computers
    - E.g. using ASCII and Unicode
  - Explain how colours, images, sound and movies are represented

Positional Number Systems

Different Representations of Natural Numbers

XXVII	Roman numerals (not positional)
27	Radix-10 or decimal number (positional)
11011_2	Radix-2 or binary number (also positional)

Fixed-radix positional representation with *k* digits

Number N in radix $r = (d_{k-1}d_{k-2} \; . \; . \; . \; d_1d_0)_r$

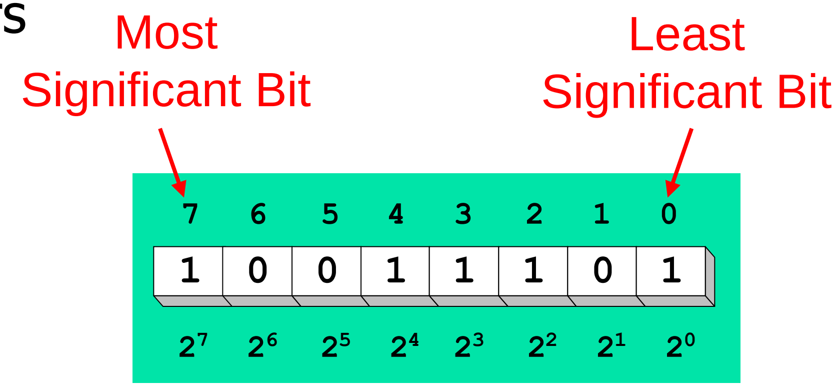
Value $= \mathrm{d}_{k-1}\times r^{\,k-1} + \mathrm{d}_{k-2}\times r^{\,k-2} + \ldots + \mathrm{d}_1\times r + \mathrm{d}_0$

Examples: $(11011)_2 = 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2 + 1 = 27$

$(2103)_4 = 2\times 4^3 + 1\times 4^2 + 0\times 4 + 3 = 147$

Binary Numbers

- Each binary digit (called bit) is either 1 or 0
- Bits have no inherent meaning, can represent
  - Unsigned and signed integers
  - Characters
  - Floating-point numbers
  - Images, sound, etc.



- Bit Numbering
  - Least significant bit (LSB) is rightmost (bit 0)
  - Most significant bit (MSB) is leftmost (bit 7 in an 8-bit number)

Binary numbers

- Binary number is simply a number comprised of only 0's and 1's.
- Computers use binary numbers because it's easy for them to communicate using electrical current -- 0 is off, 1 is on.
- You can express any base 10 (our number system -- where each digit is between 0 and 9) with binary numbers.
- The need to translate decimal number to binary and binary to decimal.
- There are many ways in representing decimal number in binary numbers.

How does the binary system work?

- It is just like any other system
- In decimal system the base is 10
  - We have 10 possible values 0-9

- In binary system the base is 2
  - We have only two possible values 0 or 1.
  - The same as in any base, 0 digit has non contribution, while 1 has contribution depending with their position.

Example

- $30405_{10} = 30000 + 400 + 5$
  $= 3*10^4+4*10^2+5*10^0$

- $10101_2 = 10000+100+1$
  $=1*2^4+1*2^2+1*2^0$

- "There are 10 kinds of people in the world - those who understand binary and those who don't."

Convert Unsigned Decimal to Binary

- Repeatedly divide the decimal integer by 2
- Each remainder is a binary digit in the translated value

Division	Quotient	Remainder
37 / 2	18	1
18 / 2	9	0
9 / 2	4	1
4 / 2	2	0
2 / 2	1	0
1 / 2	0	1

least significant bit

$37 = (100101)_2$

most significant bit

stop when quotient is zero

Examples: decimal -- binary

- Find the binary representation of 129_{10}.
- Find the decimal value represented by the following binary representations:
  - 10000011
  - 10101010

Decimal fractional to binary

- Find the binary representations of:
  - $0.5_{10} = 0.1$
  - $0.75_{10} = 0.11$

- Using only 8 binary digits find the binary representations of:
  - 0.2_{10}
  - 0.8_{10}

Decimal fraction to binary

0.8125

0.8125 × 2	1.6250	1	0.625
0.625 × 2	1.250	1	0.250
0.25 × 2	0.50	0	0.5
0.5 × 2	1.00	1	0

Multiply by 2
Keep the integer/whole part of the result

Number representation

- Representing whole numbers
- Representing fractional numbers

Integer Representations

- Unsigned notation
- Signed magnitude notation
- Excess notation
- Two’s complement notation.

Unsigned Representation

- Represents positive integers.
- Unsigned representation of 157:

position	7	6	5	4	3	2	1	0
Bit pattern	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
contribution	2^7			2^4	2^3	2^2		2^0

- Addition is simple:

  1 0 0 1 + 0 1 0 1 = 1 1 1 0.

Advantages and disadvantages of unsigned notation

- Advantages:
  - One representation of zero
  - Simple addition

- Disadvantages
  - Negative numbers can not be represented.
  - The need of different notation to represent negative numbers.

Representation of negative numbers

- Is a representation of negative numbers possible?
- Unfortunately:
  - you can not just stick a negative sign in front of a binary number. (it does not work like that)
- There are three methods used to represent negative numbers.
  - Signed magnitude notation
  - Excess notation
  - Two’s complement notation

1) Signed Magnitude Representation

- Unsigned: **-** and **+** are the same.
- In signed magnitude
  - the left-most bit represents the sign of the integer.
    - 0 for positive numbers.
    - 1 for negative numbers.
- The remaining bits represent magnitude of the numbers.

Example

- Suppose 10011101 is a signed magnitude representation.
- The sign bit is 1, then the number represented is negative

position	7	6	5	4	3	2	1	0
Bit pattern	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
contribution	-			2^4	2^3	2^2		2^0

- The magnitude is 0011101 with a value $2^4+2^3+2^2+2^0= 29$
- Then the number represented by 10011101 is –29.

Exercise 1

1. 37_{10} has 0010 0101 in signed magnitude notation. Find the signed magnitude of -37_{10} ?

2. Using the signed magnitude notation find the 8-bit binary representation of the decimal value 24_{10} and -24_{10}.

3. Find the signed magnitude of –63 using 8-bit binary sequence?

Disadvantage of Signed Magnitude

- Additions and subtractions are difficult.
- Signs and magnitude, both have to carry out the required operation.
- There are two representations of 0
  - 00000000 = $+ 0_{10}$
  - 10000000 = $- 0_{10}$
  - To test if a number is 0 or not, the CPU will need to see whether it is 00000000 or 10000000.
  - 0 is always performed in programs.
    - Therefore, having two representations of 0 is inconvenient.

Signed-Summary

- In signed magnitude notation,
  - The most significant bit is used to represent the sign.
  - 1 represents negative numbers
  - 0 represents positive numbers.
  - The unsigned value of the remaining bits represent The magnitude.
- Advantages:
  - Represents positive and negative numbers
- Disadvantages:
  - two representations of zero,
  - Arithmetic operations are difficult.

2) Excess Notation

- In excess notation:
  - The value represented is the unsigned value with a fixed value subtracted from it.
    - For n-bit binary sequences the subtracted fixed value is $2^{(n-1)}$.
  - Most significant bit:
    - 0 for negative numbers
    - 1 for positive numbers

Excess Notation with n bits

- 1000…0 represent 2^{n-1} is the decimal value in unsigned notation.

Decimal value In unsigned notation	$- 2^{n-1}$	=	Decimal value In excess notation

- Therefore, in excess notation:
  - 1000…0 will represent 0 .

Binary (in excess) to decimal

- Find the decimal number represented by 10011001 in excess notation.
  - Unsigned value
    - $10011001_2 = 2^7 + 2^4 + 2^3 + 2^0 = 128 + 16 + 8 + 1 = 153_{10}$
  - Excess value:
    - excess value = $153 – 2^7 = 152 – 128 = 25$.

Binary to decimal - example 2 (10101)

- Unsigned
  - $10101_2 = 16+4+1 = 21_{10}$
  - The value represented in unsigned notation is 21
- Sign Magnitude
  - The sign bit is 1, so the sign is negative
  - The magnitude is the unsigned value $0101_2 = 5_{10}$
  - So the value represented in signed magnitude is -5_{10}
- Excess notation
  - As an unsigned binary integer $10101_2 = 21_{10}$
  - subtracting $2^{5-1} = 2^4 = 16$, we get $21-16 = 5_{10}$.
  - So the value represented in excess notation is 5_{10}.

Decimal to binary (excess) - example 3

- Represent the decimal value 24 in 8-bit excess notation.
- We first add, 2^{8-1}, the fixed value
  - $24 + 2^{8-1} = 24 + 128 = 152$
- then, find the unsigned value of 152
  - $152_{10} = 10011000$ (unsigned notation).
  - $24_{10} = 10011000$ (excess notation)

Decimal to binary (excess) – example 4

- Represent the decimal value -24 in 8-bit excess notation.
- We first add, 2^{8-1}, the fixed value
  - $-24 + 2^{8-1} = -24 + 128 = 104$
- then, find the unsigned value of 104
  - $104_{10} = 01101000$ (unsigned notation).
  - $-24_{10} = 01101000$ (excess notation)

Advantages of Excess Notation

- It can represent positive and negative integers.
- There is only one representation for 0.
- It is easy to compare two numbers.
- When comparing the bits can be treated as unsigned integers.
- Excess notation is not normally used to represent integers.
- It is mainly used in floating point representation for representing fractions.

Exercise

1. Consider the 8-bit binary sequence 10011001
   - Find the decimal value it represents if it was in unsigned and signed magnitude.
   - Suppose this representation is excess notation, find the decimal value it represents?

2. Using 8-bit binary sequence notation, find the unsigned, signed magnitude and excess notation of the decimal value 11_{10}

Excess notation - Summary

- In excess notation, the value represented is the unsigned value with a fixed value subtracted from it.
  - i.e. for n-bit binary sequences the value subtracted is $2^{(n-1)}$.

- Most significant bit:
  - 0 for negative numbers .
  - 1 positive numbers.

- Advantages:
  - Only one representation of zero.
  - Easy for comparison.

3) Two's Complement Notation

- The most used representation for integers.
  - All positive numbers begin with 0.
  - All negative numbers begin with 1.
  - One representation of zero
    - i.e. 0 is represented as 0000 using 4-bit binary sequence.

Two's Complement Representation

- Positive numbers
  - Signed value = Unsigned value
- Negative numbers
  - Signed value = Unsigned value – 2^n
    n = number of bits
- Negative weight for MSB
  - Another way to obtain the signed value is to assign a negative weight to most-significant bit

1	0	1	1	0	1	0	0
-128	64	32	16	8	4	2	1

= -128 + 32 + 16 + 4 = -76

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	+1
00000010	2	+2
. . .	. . .	. . .
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
. . .	. . .	. . .
11111110	254	-2
11111111	255	-1

Forming the Two's Complement

starting value	`00100100 = +36`
step1: reverse the bits (1's complement)	`11011011`
step 2: add 1 to the value from step 1	`+        1`
sum = 2's complement representation	`11011100 = -36`

Sum of an integer and its 2's complement must be zero:

00100100 + 11011100 = 00000000 (8-bit sum) $\Rightarrow$ Ignore Carry

Another way to obtain the 2's complement:

Start at the least significant 1

Leave all the 0s to its right unchanged

Complement all the bits to its left

Binary Value

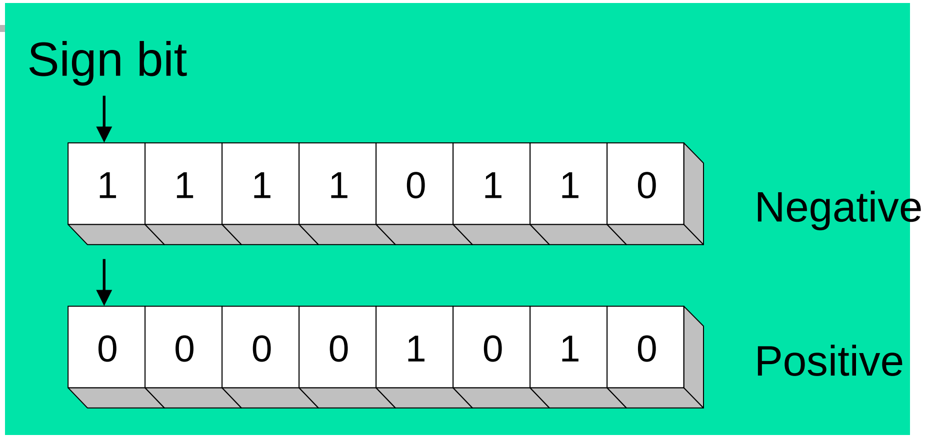
= 00100 $\boxed{1}$ 00 least significant 1

2's Complement

= 11011 $\boxed{1}$ 00

Sign Bit

- Highest bit indicates the sign
- 1 = negative
- 0 = positive



For Hexadecimal Numbers, check most significant digit

If highest digit is > 7, then value is negative

Examples: 8A and C5 are negative bytes

B1C42A00 is a negative word (32-bit signed integer)

Two's Complement of a Hexadecimal

- To form the two's complement of a hexadecimal
  - Subtract each hexadecimal digit from 15
  - Add 1
- <u>Examples:</u>

  2's complement of **`6A3D = 95C2 + 1 = 95C3`**

  2's complement of **`92F15AC0 = 6D0EA53F + 1 = 6D0EA540`**

  2's complement of **`FFFFFFFF = 00000000 + 1 = 00000001`**
- No need to convert hexadecimal to binary

Sign Extension

Step 1: Move the number into the lower-significant bits

Step 2: Fill all the remaining higher bits with the sign bit

- This will ensure that both magnitude and sign are correct
- Examples
  - Sign-Extend 10110011 to 16 bits

    `10110011 = -77` ⇨ `11111111 10110011 = -77`
  - Sign-Extend 01100010 to 16 bits

    `01100010 = +98` ⇨ `00000000 01100010 = +98`
- Infinite 0s can be added to the left of a positive number
- Infinite 1s can be added to the left of a negative number

Two’s Complement Notation with 4-bits

Binary pattern	Value in 2’s complement.
0 1 1 1	7
0 1 1 0	6
0 1 0 1	5
0 1 0 0	4
0 0 1 1	3
0 0 1 0	2
0 0 0 1	1
0 0 0 0	0
1 1 1 1	-1
1 1 1 0	-2
1 1 0 1	-3
1 1 0 0	-4
1 0 1 1	-5
1 0 1 0	-6
1 0 0 1	-7
1 0 0 0	-8

Properties of Two's Complement Notation

- Positive numbers begin with 0
- Negative numbers begin with 1
- Only one representation of 0, i.e. 0000
- Relationship between +n and –n.

- 0 1 0 0 +4 0 0 0 1 0 0 1 0 +18
- 1 1 0 0 -4 1 1 1 0 1 1 1 0 -18

Advantages of Two's Complement Notation

- It is easy to add two numbers.

	0 0 0 1	+1		1 0 0 0	-8
+	0 1 0 1	+5	+	0 1 0 1	+5
	0 1 1 0	**+6**		**1 1 0 1**	**-3**

- Subtraction can be easily performed.
- Multiplication is just a repeated addition.
- Division is just a repeated subtraction
- Two's complement is widely used in ***ALU***

Evaluating numbers in two's complement notation

- Sign bit = 0, the number is positive. The value is determined in the usual way.
- Sign bit = 1, the number is negative. three methods can be used:

Method 1	decimal value of (n-1) bits, then subtract 2^{n-1}
Method 2	$-\ 2^{n-1}$ is the contribution of the sign bit.
Method 3	● Binary rep. of the corresponding positive number. ● Let V be its decimal value. ● - V is the required value.

Example: 10101 in two's complement

- The most significant bit is 1, hence it is a negative number.
- Method 1
  - 0101 = +5 $(+5 - 2^{5-1} = 5 - 2^4 = 5\text{-}16 = -11)$
- Method 2

4	3	2	1	0		
1	0	1	0	1		
-2^4		2^2		2^0	=	-11

- Method 3
  - Corresponding + number is 01011 = 8 + 2+1 = 11 the result is then –11.

Two's complement-summary

- In two's complement the most significant bit for an n-bit number has a contribution of **$-2^{(n-1)}$**.
- One representation of zero
- All arithmetic operations can be performed by using addition and inversion.
- The most significant bit: **0** for positive and 1 for negative.
- Three methods can the decimal value of a negative number:

Method 1	decimal value of (n-1) bits, then subtract **2^{n-1}**
Method 2	-2^{n-1} is the contribution of the sign bit.
Method 3	• Binary rep. of the corresponding positive number. • Let V be its decimal value. • - V is the required value.

Exercise - 10001011

- Determine the decimal value represented by 10001011 in each of the following four systems.
  1. Unsigned notation?
  2. Signed magnitude notation?
  3. Excess notation?
  4. Two's complement?

Character representation

1) ASCII
   - A character encoding standard for electronic communication
2) Unicode
   - Unicode is the universal character encoding used to process, store and facilitate the interchange of text data in any language while ASCII is used for the representation of text such as symbols, letters, digits, etc. in computers.

a) American Standard Code for Information Interchange (ASCII)

- It is the scheme used to represent characters.
- Digits (1,2,3, etc.), letters (a, b, c, etc.) and symbols (!) are called characters.
- Each character is represented using 7-bit binary code.
- If 8-bits are used, the first bit is always set to 0
- ASCII is a standard used to represent characters on electronic devices.
- Represents 128 English characters, each assigned to a specific number in the range 0 to 127.

ASCII – example

Symbol	decimal	Binary
7	55	00110111
8	56	00111000
9	57	00111001
:	58	00111010
;	59	00111011
<	60	00111100
=	61	00111101
>	62	00111110
?	63	00111111
@	64	01000000
A	65	01000001
B	66	01000010
C	67	01000011

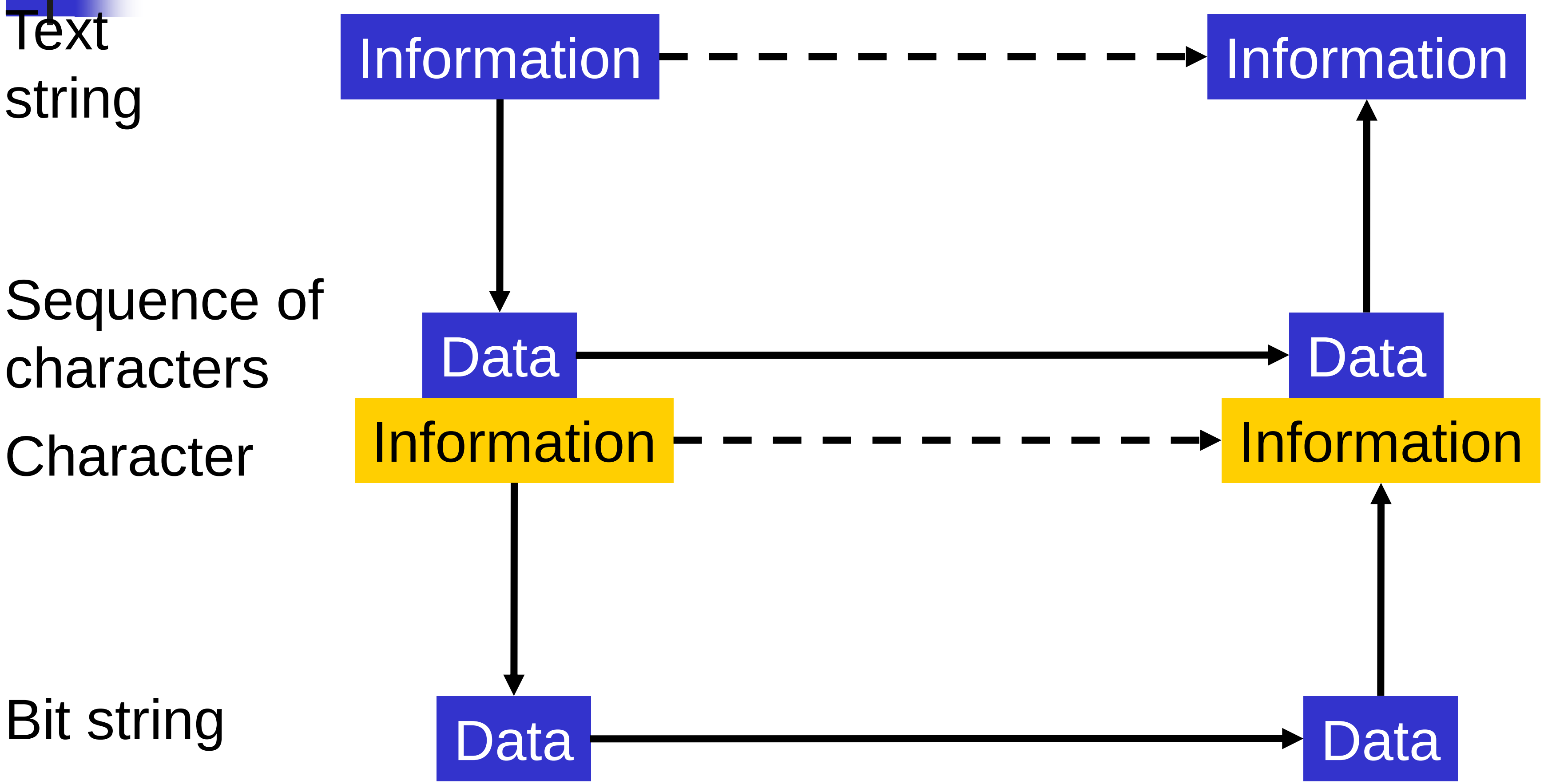
Character strings

- How to represent character strings?
- A collection of adjacent “words” (bit-string units) can store a sequence of letters

'H'	'e'	'l'	'l'	o'	' '	'W'	'o'	'r'	'l'	'd'	'\0'

- Notation: enclose strings in double quotes
  - "Hello world"
- Representation convention: null character defines end of string
  - Null is sometimes written as '\0'
  - Its binary representation is the number 0

Layered View of Representation



Working With A Layered View of Representation

- Represent “SI” at the two layers shown on the previous slide.
- Representation schemes:
  - **Top layer - Character string to character sequence**:
    Write each letter separately, enclosed in quotes. End string with ‘\0’.
  - **Bottom layer - Character to bit-string**:
    Represent a character using the binary equivalent according to the ASCII table provided.

Solution

- SI
- ‘S’ ‘I’ ‘\0’
- **010100110100100000000000**
    - **The colors are intended to help you read it; computers don’t care that all the bits run together.**

ASCII

The ASCII table contains letters, numbers, control characters, and other symbols. Each character is assigned a unique 7-bit code.

Refer to the complete ASCII table

Decimal	Octal	Hex	Binary	Value
073	111	49	0100 1001	I
074	112	4A	0100 1010	J
075	113	4B	0100 1011	K
076	114	4C	0100 1100	L
077	115	4D	0100 1101	M
078	116	4E	0100 1110	N
079	117	4F	0100 1111	O
080	120	50	0101 0000	P
081	121	51	0101 0001	Q
082	122	52	0101 0010	R
083	123	53	0101 0011	S

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Most modern character-encoding schemes are based on ASCII, although they support many additional characters.

Exercise

- Use the ASCII table to write the ASCII code for the following:
  - CIS110
  - 6=2*3

b) Unicode representation

- **ASCII** code can represent only $128 = 2^7$ characters.
- It only represents the English Alphabet plus some control characters.
- **Unicode** is designed to represent the worldwide interchange.
- It uses 16 bits and can represents 32,768 characters.
- For compatibility, the first 128 Unicode are the same as the one of the **ASCII**.

Unicode

- It assigns a code to every character and symbol in every language in the world.
- Since no other encoding standard supports all languages, Unicode is the only encoding standard that ensures that you can retrieve or combine data using any combination of languages

Colour representation

- Colours can be represented using a sequence of bits.

- 256 colours – how many bits?
  - Hint for calculating
    - To figure out how many bits are needed to represent a range of values, figure out the smallest power of 2 that is equal to or bigger than the size of the range.
    - That is, find *x* for $2^{x} => 256$
- 24-bit colour – how many possible colors can be represented?
  - Hints
    - 16 million possible colours (why 16 millions?)

24-bits -- the True colour

- 24-bit color is often referred to as the true colour.
- Any real-life shade, detected by the naked eye, will be among the 16 million possible colours.

Example: 2-bit per pixel

- $4=2^2$ choices
  - 00 (off, off)=white
  - 01 (off, on)=light grey
  - 10 (on, off)=dark grey
  - 11 (on, on)=black

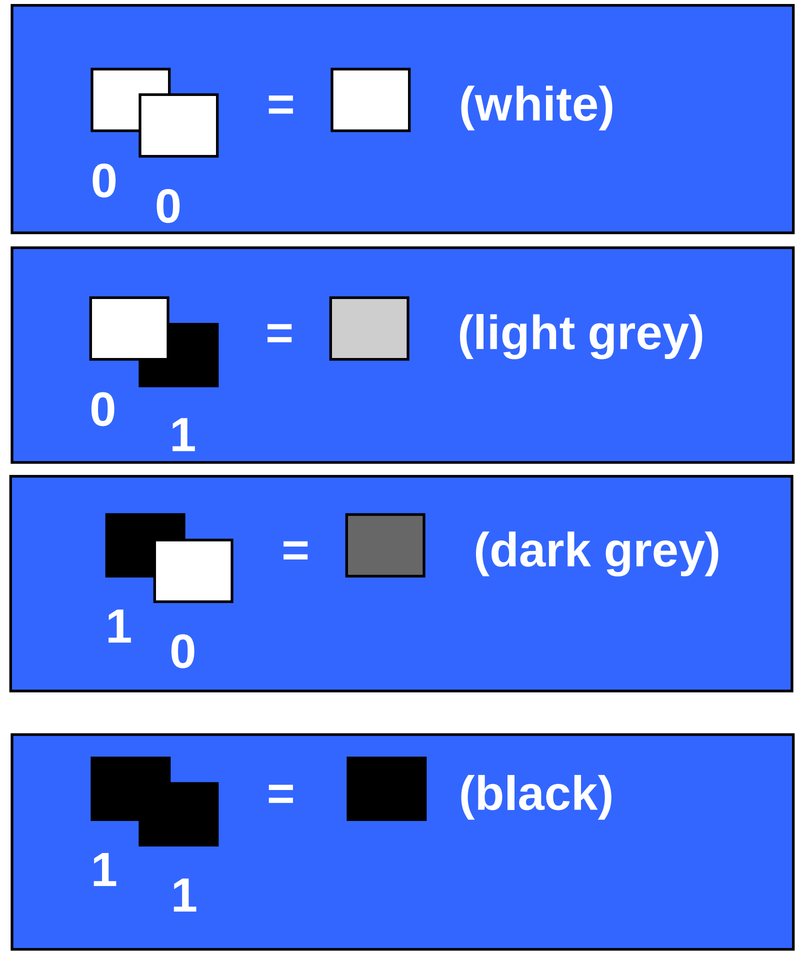
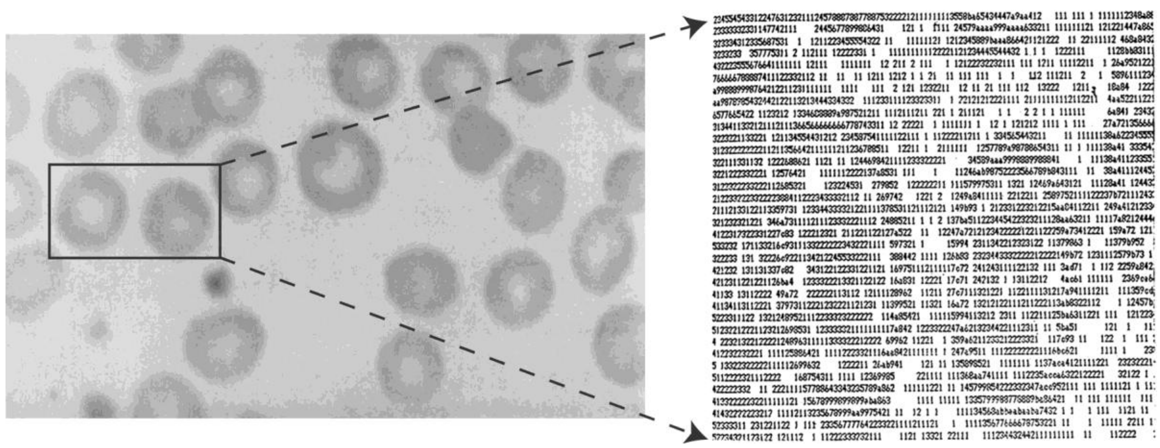


Image representation

- An image can be divided into many tiny squares, called pixels.
- Each pixel has a particular colour.
- The quality of the picture depends on two factors:
  - the density of pixels.
  - The length of the word representing colours.
- The resolution of an image is the density of pixels.
- The higher the resolution the more information information the image contains.

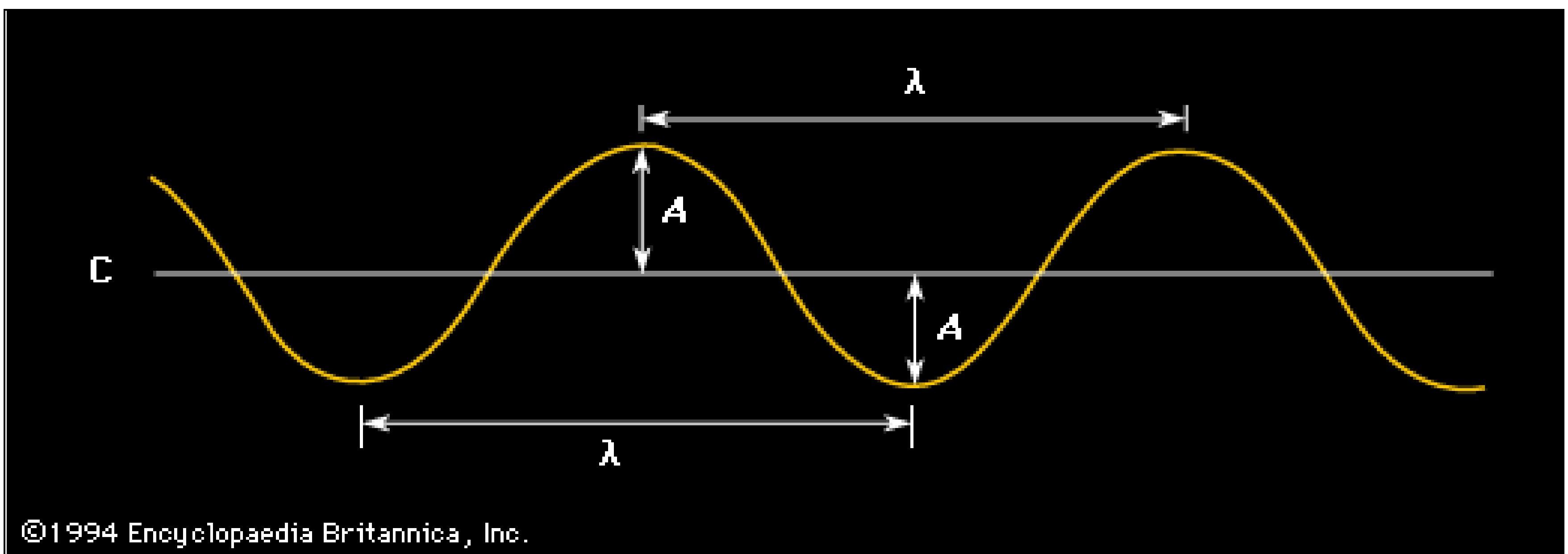
Bitmap Images

- Each individual *pixel* (*pi(x)*cture *el*ement) in a graphic stored as a binary number
  - Pixel: A small area with associated coordinate location
  - Example: each point below is represented by a 4-bit code corresponding to 1 of 16 shades of gray



Representing Sound Graphically

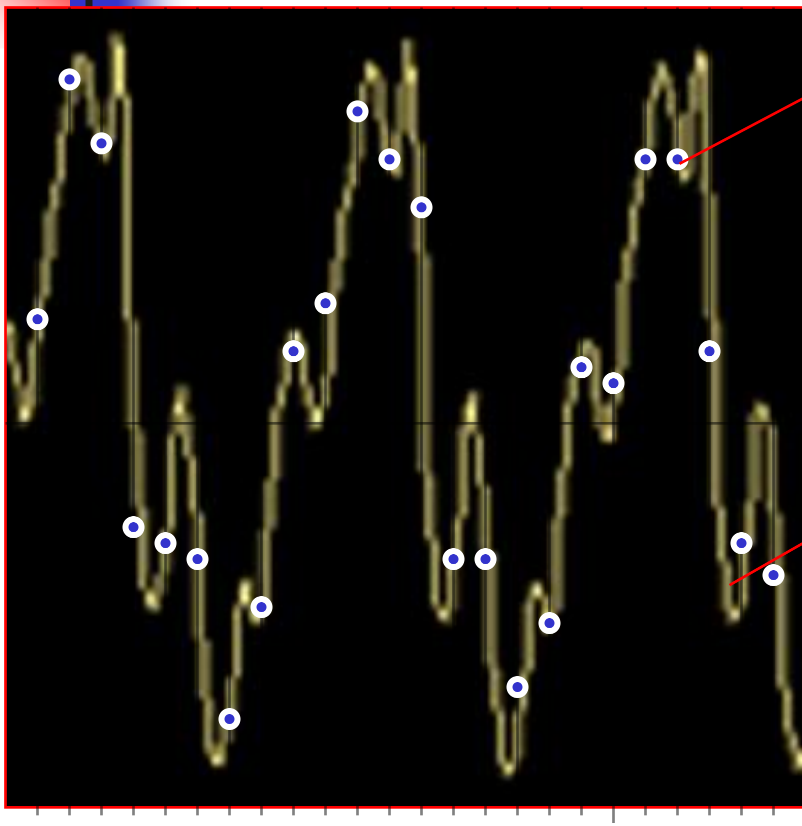
- X axis: time
- Y axis: pressure
- A: amplitude (volume)
- ●: wavelength (inverse of frequency = $1/\lambda$)



Sampling

- Sampling is a method used to digitise sound waves.
- A sample is the measurement of the amplitude at a point in time.
- The quality of the sound depends on:
  - The sampling rate, the faster the better
  - The size of the word used to represent a sample.

Digitizing Sound



Capture amplitude at these points

Lose all variation between data points

Zoomed Low Frequency Signal

Summary

- Integer representation
  - Unsigned,
  - Signed,
  - Excess notation, and
  - Two’s complement.
- Fraction representation
- Character representation
- Colour representation
- Sound representation