UNIT 2

Representation of Information in Computers

Nguyen Thi Thu Huong-SoICT-HUST

UNIT 2. REPRESENTATION OF INFORMATION IN COMPUTERS

- Number Representation in Various Numeral Systems
- Data Representation in a Computer. Units of Information
- Representation of Integers
- Arithmetic Operations on Integers
- Logical Operations on Binary Numbers
- Symbol Representation
- Representation of Real Numbers

Number Representation in Various Numeral Systems

- Decimal System
- Base-b System
- Convertion from Decimal to Base-b
- Binary System
- Hexadecimal System
- Octal System

NUMBER CATEGORIES

- Natural numbers
- Integers
- Rational numbers
- Irrational numbers

- Numbers are written using positional notation

  $943 = 9 \times 10^2 + 4 \times 10 + 3$

Only in base 10 (decimal)

Nguyen Thi Thu Huong-SoICT-HUST

Decimal System

- The system has ten as its base
- Uses various symbols (called digits) for no more than ten distinct values (0, 1, 2, 3, 4, 5, 6, 7, 8 and 9) to represent any number
- Decimal separator indicates the start of a fractional part
- Sign symbols + (positive) or − (negative) in front of the numerals to indicate sign

DECIMAL SYSTEM (CONT'D)

If i is an integer written in decimal form with digits d_j:

$i = d_{n-1}d_{n-2} \ldots d_1 d_0$

then i represents the sum:

$$\sum_{j=0}^{n-1} d_j * 10^j$$

where n is the total number of digits, and d_j is the j th digit from the rightmost position in the decimal number.

BASE-B SYSTEM

- b digits is used in the representation of numbers.
- Example

  $14_{10} = 22_6 = 112_3 = 1110_2$

- If i is an integer written in base - b with digits d_j:

  $i = d_{n-1}d_{n-2} \ldots d_1d_0$

  then i represents the sum:

  $$\sum_{j=0}^{n-1} d_j * b^j$$

  Convert from base b to base 10

  where n is the total number of digits, and d_j is the j th digit from the rightmost position in the decimal number.

- When the base is higher than 10, we need symbols to represent the digits that correspondent to the decimal values of 10 and beyond.

Nguyen Thi Thu
g-SoICT-HUST

EXAMPLE

Convert 11101.11_2 to Decimal

Position of Decimal Point ←

Binary	1	1	1	0	1 .	1	1
Position	4	3	2	1	0	-1	-2
Power of 2s	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
Decimal	16	8	4	2	1	0.5	0.25

$$11101.11_2 = 1x16 + 1x8 + 1x4 + 0x2 + 1x1 + 1x0.5 + 1x0.25 = 29.75_{10}$$

Nguyen Thi Thu Huong-SoICT-HUST

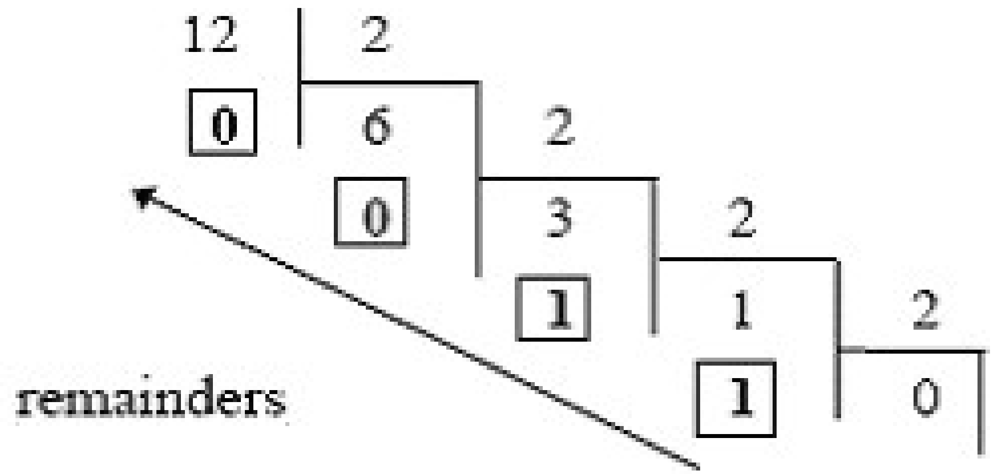
Convert integers from decimal to base b

Remainder Method:

- Let *value* = $(d_{n-1}\ d_{n-2}\ \ldots\ d_2\ d_1\ d_0)_{10}$.
- First divide *value* by b, the remainder is the least significant digit b_0.
- Divide the result by b, the remainder is b_1.
- Continue this process until the result is less than b, giving the most significant digit, b_{m-1}.
- The result is $b_{m-1}\ b_{m-2}\ \ldots\ b_2\ b_1\ b_0$

EXAMPLE 1

Use Remainder Method to convert 12 (base 10) to base 2



Result: $12_{(10)} = 1100_{(2)}$

Convert Decimal Fractions to Base-b

- Begin with the decimal fraction and multiply by b. The whole number part of the result is the first digit to the right of the point.

- Disregard the whole number part of the previous result and multiply by b once again. The whole number part of this new result is the second digit to the right of the point.

- Continue this process until you get a zero as number's decimal part or until you recognize an infinite repeating pattern.

EXAMPLE 1

Convert the decimal value .625 to a binary representation

- 0.625 x 2 = 1.25

  The first binary digit to the right of the point is a 1.
- 0.25 x 2 = 0.50,

  The second binary digit to the right of the point is a 0.
- 0.50 x 2 = 1.00

  The third binary digit to the right of the point is a 1.
- We had 0 as the fractional part of our result,

The representation of $.625_{(10)} = .101_{(2)}$

EXAMPLE 2

Binary representation of the decimal fraction 0.1

1. **.1 x 2 = 0.2**
   The first binary digit to the right of the point is a 0
2. **.2 x 2 = 0.4**
   The second binary digit to the right of the point is also a 0.
3. **.4 x 2 = 0.8**
   The third binary digit to the right of the point is also a 0.
4. **.8 x 2 = 1.6**
   The fourth binary digit to the right of the point is a 1.
5. **.6 x 2 = 1.2**
   The fifth binary digit to the right of the point is a 1.
6. **The next step to be performed (multiply 2. x 2) is exactly the same action we had in step 2. We are then bound to repeat steps 2-5, then return to Step 2 again indefinitely**

.1 (decimal) = .000110011001100110011 . . .

The repeating pattern is 0011

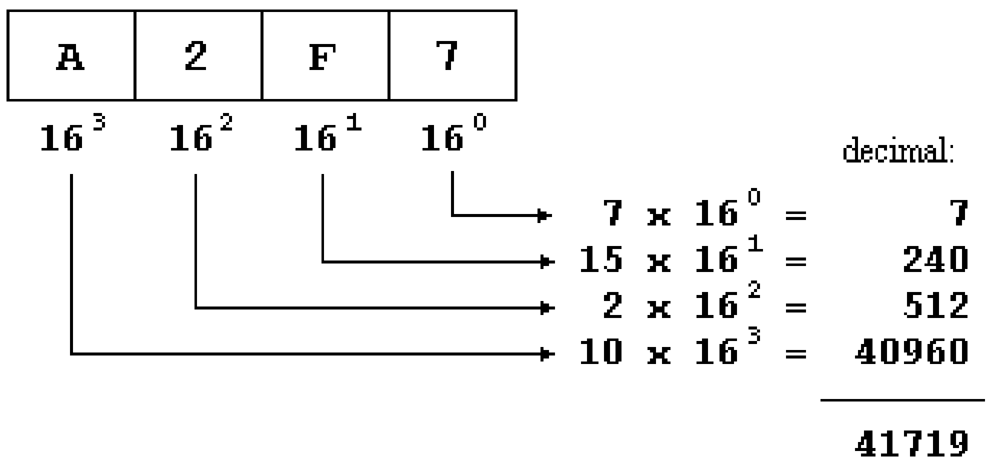
Nguyen Thi Thu Huong-SoICT-HUST

Binary System

- All data, including programs, in a computer system is represented in terms of groups of binary digits
- A single bit can represent one of two values, 0 or 1.
- If we have several symbols to represent, we can make a one-to-one correspondence between the patterns and the symbols.
  - Example : 0, 1, 2, 3 are mapped to the patterns 00, 01, 10, 11
- A group of k binary digits (bits) can be used to represent 2^k symbols

Hexadecimal System

- Hexadecimal numbers have 16 different digits, that are represented by the numbers from 0 to 9 and the letters A, B, C, D, E and F.
- Example



OCTAL SYSTEM

- The octals numbers include only the representations for the values from 0 to 7: 0, 1, 2, 3, 4, 5, 6, 7
- Example:

  $235\,.\,64_8$

  $= \mathbf{2}\text{x}8^2 + \mathbf{3}\text{x}8^1 + \mathbf{5}\text{x}8^0 + \mathbf{6}\text{x}8^{-1} + \mathbf{4}\text{x}8^{-2}$

  $= 157.\,8125_{10}$

Data Representation in a Computer
Units of Information

Basic Principles

Units of Information

Basic Principles

- Data can be numbers, symbols, images, sounds . . .
- To store in computers, it's necessary to represent data in term of bit patterns
- There are different ways to encode different types of data
  - Numbers are convert to their binary representations following some standard.
  - Symbols are assigned a bit pattern
  - Other data must be digitalized.

DATA AND COMPUTERS

- Every task, a computer manages data in some way
- In the past, computers dealt exclusively with numeric and textual data
- At present, computers are multimedia devices -> deal with different type of data:
  - Numbers
  - Text
  - Audio
  - Images and Graphics
  - Video
- All of data is stored as binary digits: strings of 1s and 0s

Analog and Digital Information

- Most part of the natural world is continuous and infinite
- Computers are finite
- Information can be represented in analog or digital

Binary Representation

- One binary digit can be either 0 or 1
- To represent more than 2 things, we need more binary digits
- N binary digits can represent 2^n things

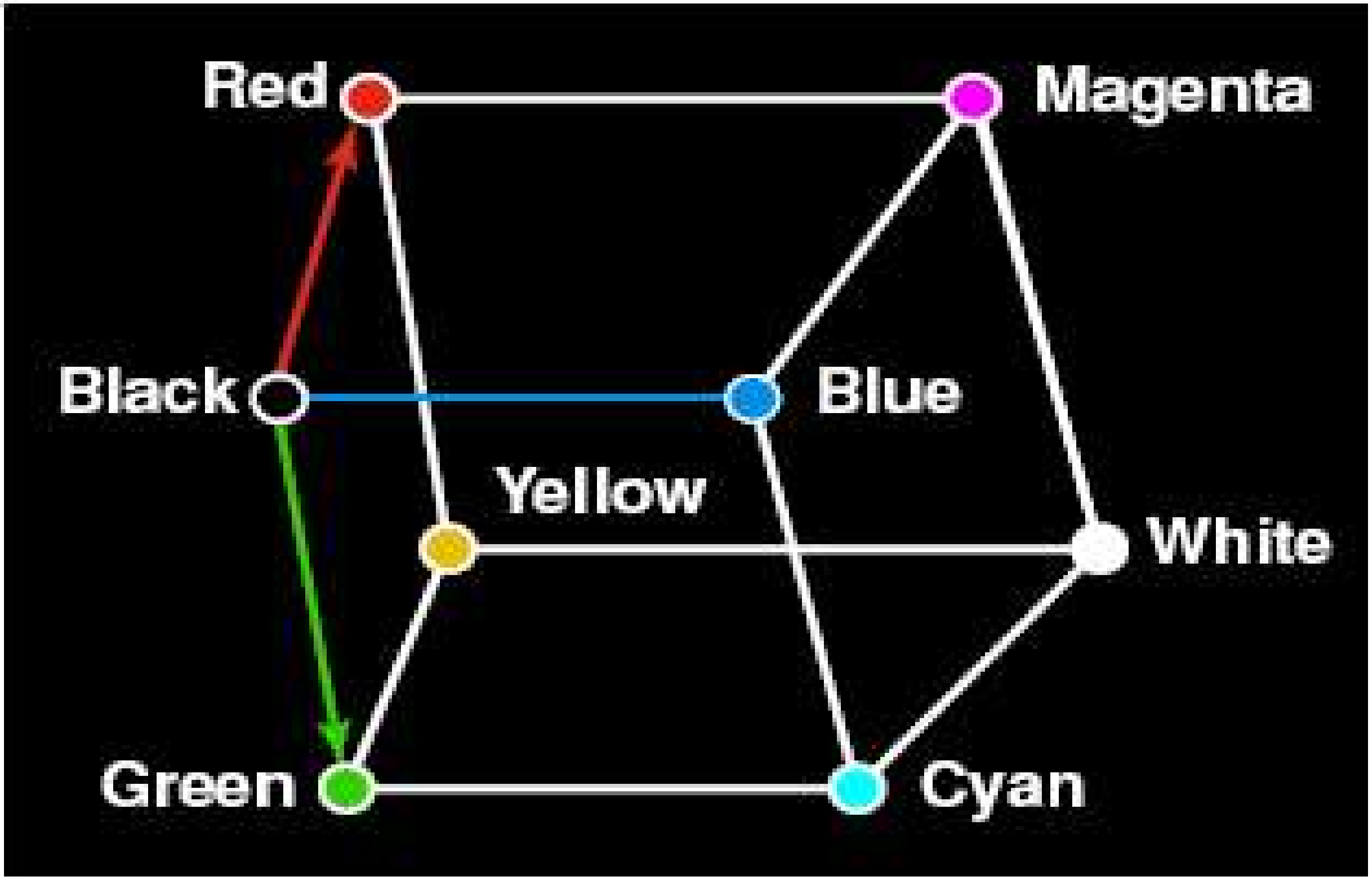
Representing Audio Information

- Digitize the sound wave
- Sampling: periodically measure the voltage of the signal and record appropriate value
- Reproduce the sound: Use stored voltage values to create a new continuous electronic signal
- Audio format : WAV,AU, AIFF, VQF, MP3

Representing Colors

- Often expressed as an RGB (red – green – blue) value
- Three numbers indicate the contribution of each of three primary colors
- Amount of data that used to represent a color is called color depth
- HiColor: 16 bit color depth
- TrueColor: 24bit color depth

Three Dimensional Color Space



Some TrueColor RGB Values

RGB Value			Actual Color
Red	Green	Blue	
0	0	0	black
255	255	255	white
255	255	0	yellow
255	130	255	pink
146	81	0	brown
157	95	82	purple
140	0	0	maroon

Digitized Images and Graphics

- Image is a collection of individual dots called pixel
- Each pixel is composed of a single color
- The number of pixel used to represent a picture is called resolution

Categorization of Data Types

- Basic types : the standard scalar predefined types that one would expect to find ready for immediate use in any imperative programming language
- Structured types: made up from such basic types or other existing higher level types.

UNITS OF INFORMATION

- The smallest unit of information a computer can use is bit (Binary Digit)
- The difference between two states (high current and low current) is represented as one of two numbers (1 or 0).
- A collection of 8 bits are put together to form a byte
- Binary prefixes can be used to quantify computer memory sizes
- Each successive prefix is multiplied by 1024 (2^{10}) rather than the 1000 (10^3)

Units of information

Name	Symbol	Value	Base 16	Base 10
kilo	k/K	$2^{10} = 1{,}024$	$= 16^{2.5}$	$> 10^3$
mega	M	$2^{20} = 1{,}048{,}576$	$= 16^5$	$> 10^6$
giga	G	$2^{30} = 1{,}073{,}741{,}824$	$= 16^{7.5}$	$> 10^9$
tera	T	$2^{40} = 1{,}099{,}511{,}627{,}776$	$= 16^{10}$	$> 10^{12}$
peta	P	$2^{50} = 1{,}125{,}899{,}906{,}842{,}624$	$= 16^{12.5}$	$> 10^{15}$
exa	E	$2^{60} = 1{,}152{,}921{,}504{,}606{,}846{,}976$	$= 16^{15}$	$> 10^{18}$
zetta	Z	$2^{70} = 1{,}180{,}591{,}620{,}717{,}411{,}303{,}424$	$= 16^{17.5}$	$> 10^{21}$
yotta	Y	$2^{80} = 1{,}208{,}925{,}819{,}614{,}629{,}174{,}706{,}176$	$= 16^{20}$	$> 10^{24}$

Representation of Integers

- Unsigned Integers
- Signed Integers

Unsigned Integers

- Unsigned integers are represented by a fixed number of bits (typically 8, 16, 32, and/or 64)
- Only a finite set of numbers that can be represented:
  - With 8 bits, 0…255 (00_{16}…FF_{16}) can be represented;
  - With 16 bits, 0…65535 (0000_{16}…$FFFF_{16}$) can be represented
- If an operation on bytes has a result outside this range, it will cause an 'overflow'

Signed Integers

- The most significant bit is set to 0 and 1 for positive and negative numbers
- Example

  $+42_{10} = \mathbf{0}0101010_2$

  and so

  $-42_{10} = \mathbf{1}1010110_2$
- All number whose leftmost bit is 1 is considered negative

SIGNED INTEGERS (CONT'D)

- If we have *4* bits in our representation then,
  - The most positive representable number using sign-and-magnitude is 0111
  - The most negative representable number using sign-and-magnitude is 1000
- How to represent negative number?

Two's Complement

- Representation for signed binary numbers
- Leading bit is a sign bit
  - Binary number with leading 0 is positive
  - Binary number with leading 1 is negative
- Magnitude of positive numbers is just the binary representation
- Magnitude of negative numbers is found by
  - Complement the bits: replace all the 1's with 0's, and all the 0's with 1's
  - Add one to the complemented number
- The carry in the most significant bit position is thrown away when performing arithmetic

EXAMPLE

Performing two's complement on the decimal 42 to get -42

- Using a eight-bit representation

42 = 00101010 Convert to binary

11010101 Complement the bits

11010101 Add 1 to the complement
+ 00000001
11010110 Result is -42 in two's complement

Two's Complement Arithmetic

- Computing 50 - 42 using a two's complement representation with eight-bit numbers

50 - 42 = 50 + (-42) = 8

00110010 50

11010110 Two's complement of 42

00110010 Add 50 and -42
\+ 11010110
100001000

Throw away the high-order carry as we are using a eight bit representation

00001000 Is the eight-bit result

Operations on Integers

- Addition and Subtraction
- Multiplication and Division

ADDITION AND SUBTRACTION

- Addition
- Subtraction

Binary Addition

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0, and carry 1 to the next more significant bit

Example

00011010 + 00001100 = 00100110

			1	1					carries	
	0	0	0	1	1	0	1	0	=	26 (base 10)
+	0	0	0	0	1	1	0	0	=	12 (base 10)
	0	0	1	0	0	1	1	0	=	38 (base 10)

00010011 + 00111110 = 01010001

		1	1	1	1	1			carries	
	0	0	0	1	0	0	1	1	=	19 (base 10)
+	0	0	1	1	1	1	1	0	=	62 (base 10)
	0	1	0	1	0	0	0	1	=	81 (base 10)

Binary Subtraction

0 - 0 = 0

0 - 1 = 1, and borrow 1 from the next more significant bit

1 - 0 = 1

1 - 1 = 0

00100101 - 00010001 = 00010100

			0						*borrows*
	0	0	~~1~~	^{1}0	0	1	0	1	= $37_{(\text{base }10)}$
−	0	0	0	1	0	0	0	1	= $17_{(\text{base }10)}$
	0	0	0	1	0	1	0	0	= $20_{(\text{base }10)}$

00110011 - 00010110 = 00011101

			0	^{1}*0*	*1*				*borrows*
	0	0	~~1~~	~~1~~	~~0~~	^{1}0	1	1	= $51_{(\text{base }10)}$
−	0	0	0	1	0	1	1	0	= $22_{(\text{base }10)}$
	0	0	0	1	1	1	0	1	= $29_{(\text{base }10)}$

Multiplication and Division

- Multiplication
- Division

Binary Multiplication

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1, and no carry or borrow bits

EXAMPLE

$00101001 \times 00000110 = 11110110$

0 0 1 0 1 0 0 1 = 41(base 10)

× 0 0 0 0 0 1 1 0 = 6 (base 10)

0 0 0 0 0 0 0 0

0 0 1 0 1 0 0 1

0 0 1 0 1 0 0 1

0 0 1 1 1 1 0 1 1 0 = 246 (base 10)

$00010111 \times 00000011 = 01000101$

0 0 0 1 0 1 1 1 = 23 (base 10)

× 0 0 0 0 0 0 1 1 = 3 (base 10)

1 1 1 1 1 carries

0 0 0 1 0 1 1 1

0 0 0 1 0 1 1 1

0 0 1 0 0 0 1 0 1 = 69 (base 10)

Binary Division

Binary division follows the same rules as in decimal division .

Logical Operations on Binary Numbers

Logical Operations with One or Two Bits

Logical Operations with One or Two Integers

Logical Operations with One or Two Bits

AND: Compares 2 bits and if they are both 1, then the result is 1, otherwise, the result is 0.

OR: Compares 2 bits and if either or both bits are 1, then the result is 1, otherwise, the result is 0.

XOR: (Exclusive OR) : Compares 2 bits and if exactly one of them is 1 then the result is 1 otherwise (if the bits are the same), the result is 0.

NOT: Changes the value of a single bit. If it is a 1, the result is 0; if it is a 0, the result is 1.

Truth Table of Logical Operations

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

NOT	0	1
	1	0

Logical Operations with One or Two Integers

- NOT
- AND
- OR
- XOR

NOT OPERATION

Unary operation which performs logical negation on each bit, forming the ones' complement of the given binary value. Digits which were 0 become 1, and vice versa.

Example:

NOT 0111

= 1000

AND OPERATION

- An AND operation takes two binary representations of equal length and performs the logical AND operation on each pair of corresponding bits. In each pair, perform AND operation on two bits
- Example:

```
      0101
AND   0011
   =  0001
```

OR OPERATION

- An OR operation takes two bit patterns of equal length, and produces another one of the same length by matching up corresponding bits (the first of each; the second of each; and so on) and performing the logical OR operation on each pair of corresponding bits.
- For example:

```
   0101
OR 0011
 = 0111
```

XOR Operation

- An exclusive or operation takes two bit patterns of equal length and performs the logical XOR operation on each pair of corresponding bits.
- For example:

```
      0101
XOR 0011
  =  0110
```

Symbol Representation

Basic Principles

ASCII Code Table

Unicode Code Table

Basic Principles

- Text documents can be decomposed into individual characters
- It's important to handle character data
- Character data isn't just alphabetic characters, but also numeric characters, punctuation, spaces, etc
- They need to be represented in binary
- There aren't mathematical properties for character data, so assigning binary codes for characters is somewhat arbitrary

Character Set

- A list of characters and the codes used to represent each one.
- By using a particular character set, computer manufacturers have made the processing of data easier.
- Two popular character sets: ASCII & Unicode

ASCII Code Table

- ASCII -- American Standard Code for Information Interchange - permitted machines from different manufacturers to exchange data.
- The ASCII standard was developed in 1963
- ASCII standard originally used 7 bits to represent characters > consists of 128 binary values (0 to 127), each associated with a character or command
- The extended ASCII character set used 8 bits to represent characters. It also consists 128 characters representing additional special, mathematical, graphic and foreign characters.

THE STANDARD ASCII CODE TABLE

Non-Printing Characters					Printing Characters								
Name	Ctrl char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
null	ctrl-@	0	00	NUL	32	20	Space	64	40	@	96	60	`
start of heading	ctrl-A	1	01	SOH	33	21	!	65	41	A	97	61	a
start of text	ctrl-B	2	02	STX	34	22	"	66	42	B	98	62	b
end of text	ctrl-C	3	03	ETX	35	23	#	67	43	C	99	63	c
end of transmit	ctrl-D	4	04	EOT	36	24	$	68	44	D	100	64	d
enquiry	ctrl-E	5	05	ENQ	37	25	%	69	45	E	101	65	e
acknowledge	ctrl-F	6	06	ACK	38	26	&	70	46	F	102	66	f
bell	ctrl-G	7	07	BEL	39	27	'	71	47	G	103	67	g
backspace	ctrl-H	8	08	BS	40	28	(	72	48	H	104	68	h
horizontal tab	ctrl-I	9	09	HT	41	29	)	73	49	I	105	69	i
line feed	ctrl-J	10	0A	LF	42	2A	*	74	4A	J	106	6A	j
vertical tab	ctrl-K	11	0B	VT	43	2B	+	75	4B	K	107	6B	k
form feed	ctrl-L	12	0C	FF	44	2C	,	76	4C	L	108	6C	l
carriage feed	ctrl-M	13	0D	CR	45	2D	-	77	4D	M	109	6D	m
shift out	ctrl-N	14	0E	SO	46	2E	.	78	4E	N	110	6E	n
shift in	ctrl-O	15	0F	SI	47	2F	/	79	4F	O	111	6F	o
data line escape	ctrl-P	16	10	DLE	48	30	0	80	50	P	112	70	p
device control 1	ctrl-Q	17	11	DC1	49	31	1	81	51	Q	113	71	q
device control 2	ctrl-R	18	12	DC2	50	32	2	82	52	R	114	72	r
device control 3	ctrl-S	19	13	DC3	51	33	3	83	53	S	115	73	s
device control 4	ctrl-T	20	14	DC4	52	34	4	84	54	T	116	74	t
negative acknowledge	ctrl-U	21	15	NAK	53	35	5	85	55	U	117	75	u
synchronous idel	ctrl-V	22	16	SYN	54	36	6	86	56	V	118	76	v
end of transmit block	ctrl-W	23	17	ETB	55	37	7	87	57	W	119	77	w
cancel	ctrl-X	24	18	CAN	56	38	8	88	58	X	120	78	x
end of medium	ctrl-Y	25	19	EM	57	39	9	89	59	Y	121	79	y
substitute	ctrl-Z	26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
escape	ctrl-[	27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
file separator	ctrl-\	28	1C	FS	60	3C	<	92	5C	\	124	7C	|
group separator	ctrl-]	29	1D	GS	61	3D	=	93	5D	]	125	7D	}
record separator	ctrl-^	30	1E	RS	62	3E	>	94	5E	^	126	7E	~
unit separator	ctrl-_	31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

THE EXTENDED ASCII CHARACTERS

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	^{n}
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	^{2}
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	□

Limitations of ASCII character set

- String data types allocated one byte per character
- The extended version of ASCII which have 256 characters is ehough for English but not enough for international use
- Logographic languages such as Chinese, Japanese, and Korean need far more than 256 characters for reasonable representation.
- Vietnamese need 61 characters for representation.
- Where can we find number for our characters?

$\Rightarrow$ 2bytes per character?

Unicode Code Table

Before Unicode was invented. . . .

- There were hundreds of different encoding systems
- No single encoding could contain enough characters
- Encoding systems conflict with one another : two encodings can use the same number for two different characters, or use different numbers for the same character.

Unicode Character Set

- Provides a unique number for every character
- Uses 16 bits per character
- Has been adopted by such industry leaders as HP, IBM, Microsoft, Oracle, Sun, and many others.
- Is supported in many operating systems, all modern browsers, and many other products.

Some Symbols in the Unicode Character Set

Code (Hex)	Character	Source
0041	A	English (Latin)
042F	Я	Russian (Cyrillic)
OE09	ฉ	Thai
13EA	Ꮺ	Cherokee
211E	℞	Letterlike Symbols
21CC	⇌	Arrows
282F	⠯	Braille
345F	㑟	Chinese/Japanese/ Korean (Common)

ADVANTAGES OF USING UNICODE

- Significant cost savings
- Enables software products to be targeted across multiple platforms, languages and countries without re-engineering.
- Allows data to be transported through many different systems without corruption.