

Lecture 1 - Outline

Binary Values and Number Systems, Data Representation.

- **Know the different types of numbers**
- **Describe positional notation**
- **Convert numbers in other bases to base 10**
- **Convert base 10 numbers into numbers of other bases**
- **Describe the relationship between bases 2, 8, and 16**
- **Explain computing and bases that are powers of 2**
- **Decimal Codes [BCD (binary coded decimal)]**
- **Gray Codes**
- **ASCII Codes**

Electronic Data and Instructions

- **Data and instructions are represented electronically**
- **Two-state system or Binary System**
 - **Off/On electrical states**
 - **Characters represented by 0's (off) and 1's (on)**
 - **Bits**
 - **Bytes**

Character Coding Schemes

Three types of binary coding schemes

- ASCII - American Standard Code for Information Exchange
- EBCDIC - Extended Binary Coded Decimal Interchange Code
- Unicode - handles languages with large numbers of characters

Decimal	Binary	Hex
00	00000000	00
01	00000001	01
02	00000010	02
03	00000011	03
04	00000100	04
05	00000101	05
06	00000110	06
07	00000111	07
08	00001000	08
09	00001001	09
10	00001010	0A
11	00001011	0B
12	00001100	0C
13	00001101	0D
14	00001110	0E
15	00001111	0F

Numbers

Natural Numbers

Zero and any number obtained by repeatedly adding one to it.

Examples: 100, 0, 45645, 32

Negative Numbers

A value less than 0, with a – sign

Examples: -24, -1, -45645, -32

Integers

A natural number, a negative number, zero

Examples: 249, 0, - 45645, - 32

Rational Numbers

Rational Number is an integer number that can be written as a simple fraction (i.e. as a **ratio** of two integers).

Examples: -249, -1, 0, 3/7, -2/5

Irrational Numbers

Numbers that can not be represented by the quotient of two integers

Examples: $\pi = 3.14159 \dots$, e , $\sqrt{2}$, etc.

Natural Numbers

How many ones are there in 642?

Is it $600 + 40 + 2$?

Or is it $384 + 32 + 2$?

Or maybe... $1536 + 64 + 2$?

We need a base for each number .

The base of a number determines the number of digits and the value of digit positions

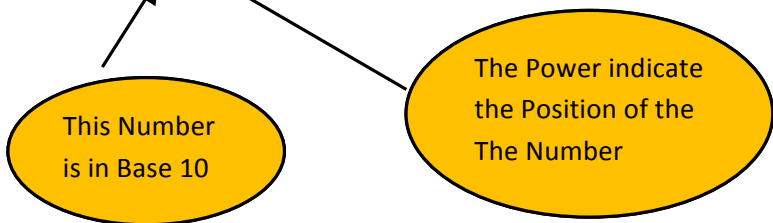
Thus 642 is 600 + 40 + 2 in BASE 10

Positional Notation

Continuing with our example...

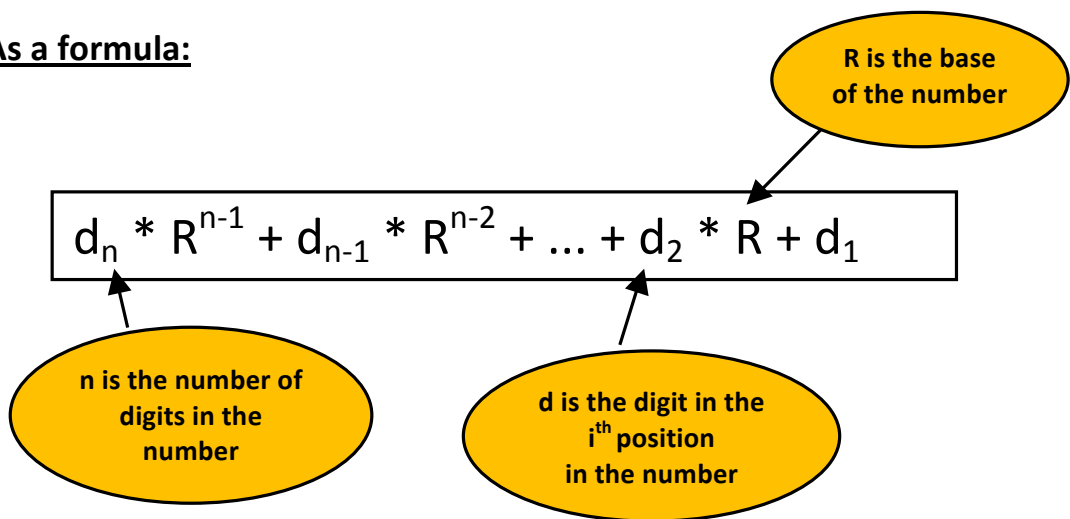
642 in base 10 positional notation is:

$$\begin{aligned} 6 \times 10^2 &= 6 \times 100 = 600 \\ + 4 \times 10^1 &= 4 \times 10 = 40 \\ + 2 \times 10^0 &= 2 \times 1 = 2 = 642 \text{ in base 10} \end{aligned}$$



$$642 \text{ is } 6_3 * 10^2 + 4_2 * 10^1 + 2_1$$

As a formula:



What if 642 has the base of 13?

$$\begin{aligned} \text{Then} \quad & 6 \times 13^2 = 6 \times 169 = 1014 \\ & + 4 \times 13^1 = 4 \times 13 = 52 \\ & + 2 \times 13^0 = 2 \times 1 = 2 \\ & \qquad \qquad \qquad = 1068 \text{ in base 10} \end{aligned}$$

642 in base 13 is equivalent to 1068 in base 10

Decimal Numeral System - Base-10

Decimal numbers uses digits from 0..9. These are the regular numbers that we use. Each digit has an associated value of an integer raised to the power of 10

Example: $2538_{10} = 2 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 8 \times 10^0$

Example: $724.5_{10} = 7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$

Octal Numeral System - Base-8

Octal numbers uses digits from 0..7.

Converting Octal to Decimal

What is the decimal equivalent of the **octal** number **642**?

$$\begin{aligned}6 \times 8^2 &= 6 \times 64 = 384 \\+ 4 \times 8^1 &= 4 \times 8 = 32 \\+ 2 \times 8^0 &= 2 \times 1 = 2\end{aligned}$$

= 418 in base 10

What is the decimal equivalent of the **octal** number **(127.4)₈** ?

Sol : $(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$

Bases Higher than 10

How are digits in bases higher than 10 represented?

With distinct symbols for 10 and above.

Base 16 has 16 digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F

Hexadecimal Numeral System - Base-16

Hex numbers uses digits from 0..9 and A..F.

H denotes hex prefix.

Converting Hexadecimal to Decimal

What is the decimal equivalent of the hexadecimal number DEF?

$$\begin{aligned} D \times 16^2 &= 13 \times 256 = 3328 \\ + E \times 16^1 &= 14 \times 16 = 224 \\ + F \times 16^0 &= 15 \times 1 = 15 \end{aligned}$$

= 3567 in base 10

What is the decimal equivalent of the hexadecimal number

(B65F)₁₆ ?

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46687)_{10}$$

Remember, the digits in base 16 are 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

So far , we have the following :

Decimal is base 10 and has 10 digits:	0,1,2,3,4,5,6,7,8,9
Octal is base 8 and uses digits from	0..7.
Hexadecimal is Base 16 has 16 digits: and F	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,

Note :

For a number to exist in a given number system, the number system must include those digits. For example, the number 284 only exists in base 9 and higher.

Binary

Binary is base 2 and has 2 digits: 0,1

Each digit has an associated value of 0 or 1 raised to the power of 2 .

B denotes binary prefix

Converting Binary to Decimal

What is the decimal equivalent of the **binary** number **1101110**?

$$\begin{aligned} 1 \times 2^6 &= 1 \times 64 = 64 \\ + 1 \times 2^5 &= 1 \times 32 = 32 \\ + 0 \times 2^4 &= 0 \times 16 = 0 \\ + 1 \times 2^3 &= 1 \times 8 = 8 \\ + 1 \times 2^2 &= 1 \times 4 = 4 \\ + 1 \times 2^1 &= 1 \times 2 = 2 \\ + 0 \times 2^0 &= 0 \times 1 = 0 \\ &= \mathbf{110} \text{ in base 10} \end{aligned}$$

What is the decimal equivalent of the **binary number** $(11010.11)_2$

$$\begin{aligned}\text{Sol : } (11010.11)_2 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \\ &= \quad \quad \quad \mathbf{(26.75)_{10}}\end{aligned}$$

Conversion Decimal to Base-R

Step:

- 1) Convert the Integer Part
- 2) Convert the Fraction Part
- 3) Join the two results with a radix point

Conversion Details

To Convert the Integral Part:

- Repeatedly divide the number by the new radix and save the remainders.
- The digits for the new radix are the remainders in *reverse order* of their computation.
- If the new radix is > 10 , then convert all remainders > 10 to digits A, B, ...

To Convert the Fractional Part:

- Repeatedly multiply the fraction by the new radix and save the integer digits that result.
- The digits for the new radix are the integer digits in *order* of their computation.
- If the new radix is > 10 , then convert all integers > 10 to digits A, B, ...

Algorithm: To convert the fractional part:

Repeatedly multiply the fraction by the radix and save the integer digits that result. The new radix fraction digits are the integer numbers in *computed order*.

For example, to convert 0.6875_{10} to base 2, we repeatedly multiply the fractional part by 2 until we can 0 for fractional part or desired accuracy is obtained.

For example, to convert 0.6875_{10} to base 2, we repeatedly multiply the fractional part by 2 until we can get 0 for fractional part or desired accuracy is obtained.

$$0.6875 * 2 = 1.3750, \text{ integer digit} = 1$$

$$0.3750 * 2 = 0.7500, \text{ integer digit} = 0$$

$$0.7500 * 2 = 1.5000, \text{ integer digit} = 1$$

$$0.5000 * 2 = 1.0000, \text{ integer digit} = 1$$

$$0.0000 \quad (\text{nothing more to do})$$

Thus : 0.6875 in base 10 is equal to 0.1011 in base 2

Converting integer and fraction parts :

You now should be able to convert any number *with both integer and fractional part* to any other bases by simply decomposing it to an integer part and fractional part, find the representation separately then merge them using a radix point (“.”).


For example, 46.6875_{10} in base 2 is $101110_2 + 0.1011_2$ or 101110.1011_2 .

Arithmetic in Binary

Remember that there are only 2 digits in binary, 0 and 1

Position is key, carry values are used:

$$\begin{array}{r}
 111111 \\
 101011 \\
 +100101 \\
 \hline
 10100010
 \end{array}$$



Subtracting Binary Numbers

Remember borrowing? Apply that concept here:

$$\begin{array}{r}
 12 \\
 202 \\
 101011 \\
 - 111011 \\
 \hline
 0011100
 \end{array}$$

Converting Binary to Octal

- Groups of Three (from right)
- Convert each group

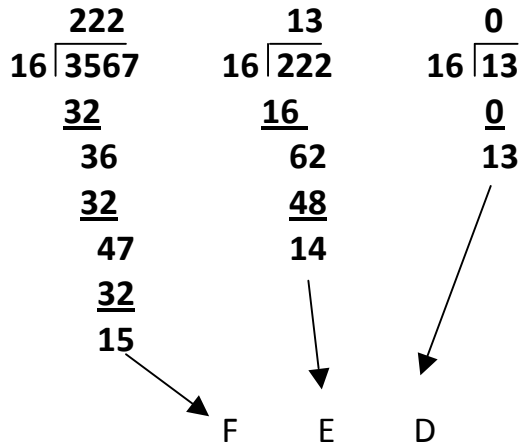
$$\begin{array}{r}
 10101011 \quad \underline{10} \quad \underline{101} \quad \underline{011} \\
 \quad \quad \quad 2 \quad 5 \quad 3
 \end{array}$$

10101011 is 253 in base 8

Converting Decimal to Hexadecimal

Try a Conversion

The base 10 number 3567 is what number in base 16?



Binary and Computers

Binary computers have storage units called binary digits or bits

Low Voltage = 0

High Voltage = 1 all bits have 0 or 1

Problem: How are negative numbers stored ?

Solutions

Sign-magnitude :

- First (high-order) bit represents the sign bit
 - 0 = positive
 - 1 = negative
- Remaining bits represent the magnitude of the number

2's Complement Notation :

- fixed length code
 - 1st (high-order) bit represents the sign bit
 - 0 = positive
 - 1 = negative

Converting from decimal to 2's complement

1. Complement each bit
2. Add 1 to the low order bit
3. Retain the original sign of the number

These are examples of converting an **eight-bit** two's complement number to decimal.

Interpret 11011011 as a two's complement binary number, and give its decimal equivalent. ?

Solution :

1. First, note that the number is negative, since it starts with a 1.
2. Change the sign to get the magnitude of the number.

$$\begin{array}{r}
 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\
 -\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0 \\
 +\ \underline{\hspace{1.5cm}1} \\
 \hline
 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1
 \end{array}$$

← Complement of 11011011

3. Convert the magnitude to decimal: $00100101_2 = 37_{10}$.
4. Since the original number was negative, the final result is -37.

Interpret 01101001 as a two's complement binary number, and give its decimal equivalent.

Solution :

The number is positive, so simply convert it to decimal: $01101001_2 = ???$

Interpret 11110010 as a two's complement binary number, and give its decimal equivalent.

Solution :

$$\begin{array}{r} 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ -\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1 \\ +\ \underline{\hspace{1.5cm}}\ 1 \\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0 \end{array}$$

$00001110_2 = ???$

2's complement arithmetic :

- addition
 - use binary addition
 - discard any final carry

For Example : what is the value of $9 + (-5)$??

Take the 2's complement of the negative number and use regular binary addition

9 \longrightarrow 00001001

(-5) \longrightarrow 00000101

↓↓↓↓↓↓↓↓

11111010 (00000101's Complement Process)

$$\begin{array}{r}
 \\
 + \\
 \hline
 11111011
 \end{array}$$

Thus

$9 + (-5) = 00001001$

$$\begin{array}{r}
 11111011 \\
 \hline
 1]00000100
 \end{array}$$

8th Bit = 0: Answer is Positive - Disregard 9th Bit

Relationship between Bits and Byte → 8 bits = 1 Byte.

Name	Abbreviation	# of Bytes	Size
Byte	B	1 = 8 bits	Could hold 1 Character of data
Kilobyte	KB	1,024 Bytes = 1024 * 8 = 8192 bits	Could hold 1024 Characters of data (half of double spaced typewriter paper = 2^{10})
Megabyte	MB	KB * KB = 1,048,576 Bytes	A floppy disk holds 1.4 MB of data – around 768 pages of typed text $\sim 2^{20}$
Gigabyte	GB	MB * KB = 1,073,741,824 Bytes	Around 786,432 pages of text - Stack of papers that is 262 feet high $\sim 2^{30}$
Terabyte	TB	GB * KB = 1,099,511,627,776 Bytes	Stack of typewritten pages that is almost 51 miles high $\sim 2^{40}$
Petabyte	PB	TB * KB = 1,125,899,906,842,624 Bytes	Stack of typewritten pages that is almost 52,000 miles high – about one-fourth distance from the earth to the moon.

The number of bits in a word determines the word length of the computer, but it is usually a multiple of 8

- 32-bit machines
- 64-bit machines etc.

Numeral Systems Conversion Table

Decimal - Base-10	Binary -Base-2	Octal - Base-8	Hexadecimal - Base-16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20

Number Systems – Examples

	General	Decimal	Binary
Radix (Base)	r	10	2
Digits	$0 \Rightarrow r - 1$	$0 \Rightarrow 9$	$0 \Rightarrow 1$
	r^0	1	1
0	r^1	10	2
1	r^2	100	4
2	r^3	1000	8
3	r^4	10,000	16
4	r^5	100,000	32
5	r^{-1}	0.1	0.5
-1	r^{-2}	0.01	0.25
-2	r^{-3}	0.001	0.125
-3	r^{-4}	0.0001	0.0625
-4	r^{-5}	0.00001	0.03125
-5			

Number Systems – Examples

Powers of Two

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

Binary Numbers and Binary Coding

Flexibility of representation

- Within constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded.

Information Types

- Numeric
 - Must represent range of data needed
 - Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
 - Tight relation to binary numbers
- Non-numeric
 - Greater flexibility since arithmetic operations not applied.
 - Not tied to binary numbers

Non-numeric Binary Codes

Given n binary digits (called bits), a binary code is a mapping from a set of elements ... things ... to a subset of the 2^n binary numbers.

Example: A binary code for the seven colors of the rainbow

Note : Code 100 is not used

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

Binary Coded Decimal (BCD)

Binary coded decimal (BCD) is a system of writing numerals that assigns a four-digit [binary](#) code to each digit 0 through 9 in a [decimal](#) (base-10) numeral. The four-bit BCD code for any particular single base-10 digit is its representation in binary notation, as follows:

0 = 0000
 1 = 0001
 2 = 0010
 3 = 0011
 4 = 0100
 5 = 0101
 6 = 0110
 7 = 0111
 8 = 1000
 9 = 1001

Numbers larger than 9, having two or more digits in the decimal system, are expressed digit by digit. For example, the BCD rendition of the base-10 number 1895 is

0001 1000 1001 0101

The binary equivalents of 1, 8, 9, and 5, always in a four-digit format, go from left to right.

Example : $(791)_{10} = (0111\ 1001\ 0001)_{BCD}$

Note : BCD was used in some of the early decimal computers, as well as the IBM System/360 series systems.

Warning: Conversion or Coding?

Do NOT mix up:

- Conversion of a decimal number to a binary number
- With coding a decimal number with a binary code.

$13_{10} = 1101_2$ **(This is conversion)**

$13 \Leftrightarrow 0001|0011$ **(This is coding)**

GRAY CODE– Decimal

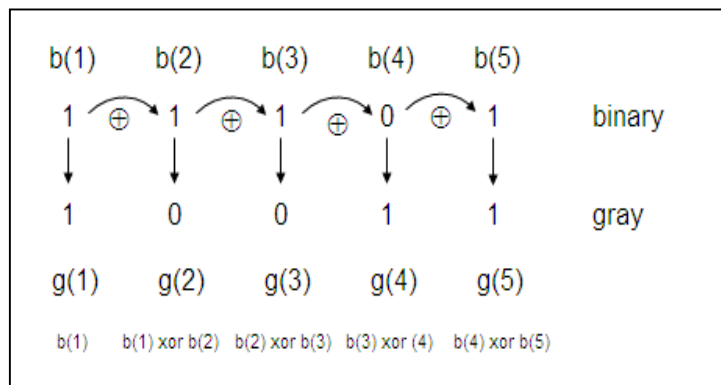
The code word for the Gray code changes in only one bit position as we go from decimal digit to digit including from 9 to 0. The reflected binary code was originally designed to prevent spurious output from electromechanical switches. Today, Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television

Converting Binary to Gray

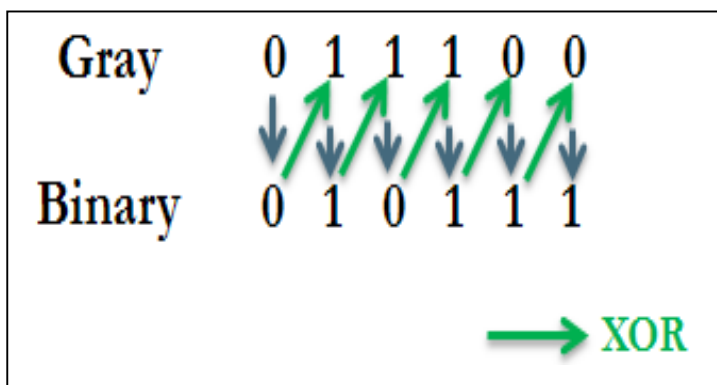
1. Write down the number in binary code .
2. The most significant bit of the gray number is the most significant bit of the binary code .
3. **XOR** the significant bit of the binary number to the next significant bit of the binary number to obtain the next gray coded bit

Example : Convert the following decimal number 29 code to its equivalent Gray Code.

Convert 29 to binary $\sim\sim$ 11101



Convert from Gray to Binary



ASCII Character Code

American Standard Code for Information Interchange

This code is a popular code used to represent information sent as character-based data (like this presentation).

It uses 7-bits to represent:

- 94 Graphic printing characters.
- 34 Non-printing characters

Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return)

Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).

ASCII Table :

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Escape	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	File Separator	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	Group Separator	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	Record Separator	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	Unit Separator	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		Del

asciicharstable.com

ASCII Character Codes

American Standard Code for Information Interchange (ASCII)								
B ₇ B ₆ B ₅								
B ₂ B ₁ B ₀	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	.	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SOH	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII Character Codes (cont): Non-printing characters

Control Characters (34 non-printing characters)			
NULL	NULL	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Navigative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End of transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SOH	Shift: out	RS	Record separator
SI	Shift: in	US	Unit separator
SP	Space	DEL	delete

PARITY BIT Error-Detection Codes

Extra bits, can be incorporated into binary code words to detect and correct errors. These bits are sometimes called “redundant” bits.

A simple form of redundancy is **parity**, an extra bit prepended or appended onto the code word to make the number of 1’s odd or even. Parity can detect all single-bit errors and some multiple-bit errors.

A code word has **even** parity if the number of 1’s in the code word is even.

A code word has **odd** parity if the number of 1’s in the code word is odd.

4-Bit Parity Code Example

Fill in the even and odd parity bits:

The codeword "1111" has **even** parity and the codeword "0111" has **odd** parity. Both can be used to represent 3-bit data.

Even Parity		Odd Parity	
Parity Bit	Data	Parity Bit	Data
0	000	1	000
1	001	0	001
1	010	0	010
0	011	1	011
1	100	0	100
0	101	1	101
0	110	1	110
1	111	0	111

Reading Assignment

Appendix A: Sections A3, A4
 Chapter 2: Sections 1 - 3
 Be prepared to discuss

- Signed Numbers and operations on them
- Binary Logic and Logic Gates