Digital Fundamentals

Thomas L. Floyd

Number Systems, Operations, and Codes

Chapter 2

The Decimal Number System

Decimal: 10 symbols (0 through 9)

If number > 9: need more digits

Weights: powers of ten

$\ldots 10^5 \;\; 10^4 \;\; 10^3 \;\; 10^2 \;\; 10^1 \;\; 10^0 .$

For fractional: weights are negative powers of ten

$\ldots \; 10^2 \;\; 10^1 \;\; 10^0 . \; 10^{-1} \;\; 10^{-2} \;\; 10^{-3} \; \ldots$

The Decimal Number System

Decimal number: linear combination of powers

Ex) the number 9240

$$(9 \times 10^3) + (2 \times 10^2) + (4 \times 10^1) + (0 \times 10^0)$$

Ex) 480.52

$$(4 \times 10^2) + (8 \times 10^1) + (0 \times 10^0) + (5 \times 10^{-1}) + (2 \times 10^{-2})$$

The Binary Number System

Binary: 2 symbols (0 and 1)

If number > 1: need more digits

Weights: powers of two

$$\ldots\ 2^5\ \ 2^4\ \ 2^3\ \ 2^2\ \ 2^1\ \ 2^0.$$

For fractional: weights are negative powers of two

$$\ldots\ 2^2\ \ 2^1\ \ 2^0\ .\ 2^{-1}\ \ 2^{-2}\ \ 2^{-3}\ \ 2^{-4}\ \ldots$$

Ch.2 Summary

The Binary Number System

Notice the pattern of zeros and ones in each column.

Digital counters frequently have this same pattern of digits:

TABLE 2–1				
DECIMAL NUMBER	BINARY NUMBER			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Binary Weights

The positional weights for binary numbers are assigned as shown below.

TABLE 2–2 • Binary weights.

POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

Binary-to-Decimal Conversion

Add all weights with 1 (ignoring 0)

Convert the binary number 100101.01 to decimal.

2^5	2^4	2^3	2^2	2^1	2^0.	2^{-1}	2^{-2}	
32	16	8	4	2	1	0.5	0.25	
1	0	0	1	0	1.	0	1	
32			+4		+1		+0.25 =	37.25

Decimal-to-Binary Conversions - I

Iteratively find the maximum weight

Convert the decimal number 49 to binary.

49 is in [32, 64) : the maximum weight is 32

remainder is 49-32 = 17

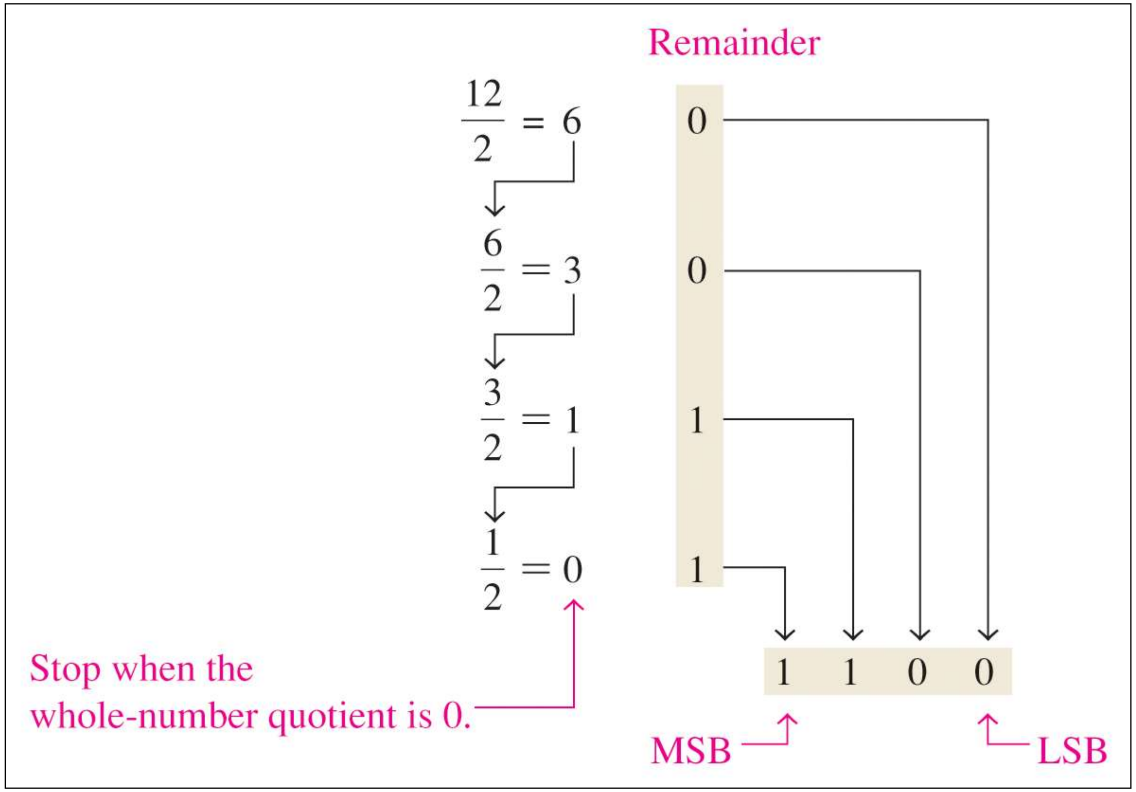
17 is in [16, 32) : the maximum weight is 16

remainder is 17-16 = 1, etc…

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1
0	1	1	0	0	0	1

Decimal-to-Binary Conversions - II

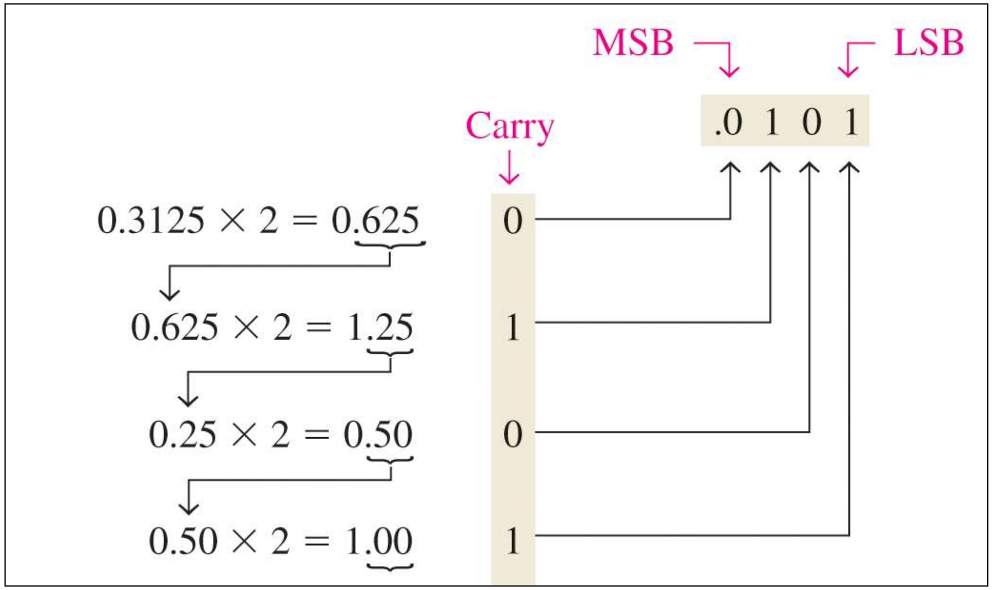
Iteratively dividing by 2 and get remainders.



Decimal-to-Binary Conversions - II

For fraction,

Iteratively multiplying by 2 and get carries.



Binary Addition

The rules for binary addition are

0 + 0 = 0	Sum = 0, carry = 0
0 + 1 = 0	Sum = 1, carry = 0
1 + 0 = 1	Sum = 1, carry = 0
1 + 1 = 10	Sum = 0, carry = 1

When an input carry = 1 due to a previous result, the rules are

1 + 0 + 0 = 01	Sum = 1, carry = 0
1 + 0 + 1 = 10	Sum = 0, carry = 1
1 + 1 + 0 = 10	Sum = 0, carry = 1
1 + 1 + 1 = 11	Sum = 1, carry = 1

Binary Subtraction

The rules for binary subtraction are

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$10 - 1 = 1 \text{ with a borrow of 1}$$

Subtract the binary number 00111 from 10101 and show the equivalent decimal subtraction.

$$\begin{array}{r} \overset{1}{\cancel{1}}\overset{1}{\cancel{0}}\overset{1}{\cancel{1}}01 \\ 00111 \\ \hline 01110 \end{array} \quad \begin{array}{c} \\ \\ \\ = \end{array} \quad \begin{array}{r} 21 \\ 7 \\ \hline 14 \end{array}$$

Binary Multiplication

The rules for binary multiplication are:

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

Perform the following binary multiplications:

(a) 11×11 **(b)** 101×111

SOLUTION

(a)

$$\begin{array}{r}
11 \\
\underline{\times\, 11} \\
\text{Partial products} \left\{ \begin{array}{r} 11 \\ \underline{+11} \end{array} \right. \\
\mathbf{1001}
\end{array}
\qquad
\begin{array}{r}
3 \\
\underline{\times\, 3} \\
9
\end{array}$$

(b)

$$\begin{array}{r}
111 \\
\underline{\times\, 101} \\
\text{Partial products} \left\{ \begin{array}{r} 111 \\ 000 \\ \underline{+111} \end{array} \right. \\
\mathbf{100011}
\end{array}
\qquad
\begin{array}{r}
7 \\
\underline{\times\, 5} \\
35
\end{array}$$

Binary Division

The rules for binary division are:

(always the same with decimal operations)

Perform the following binary divisions:

(a) 110 ÷ 11 **(b)** 110 ÷ 10

SOLUTION

(a)

$$\begin{array}{r} \mathbf{10} \\ 11\overline{)110} \\ \underline{11} \\ 000 \end{array} \qquad \begin{array}{r} 2 \\ 3\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

(b)

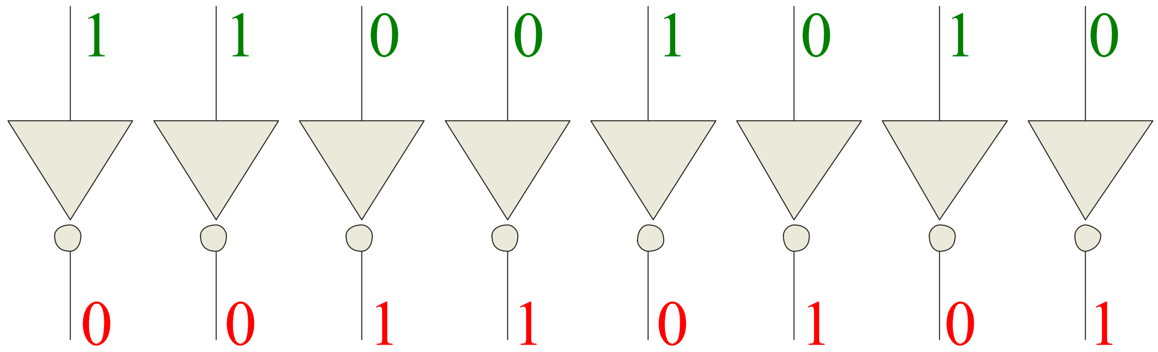
$$\begin{array}{r} \mathbf{11} \\ 10\overline{)110} \\ \underline{10} \\ 10 \\ \underline{10} \\ 00 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

1's Complement

1's complement : reversing (inverting) all the bits.

For example, the 1's complement of 11001010
is: 00110101

We can use inverter to represent this operation

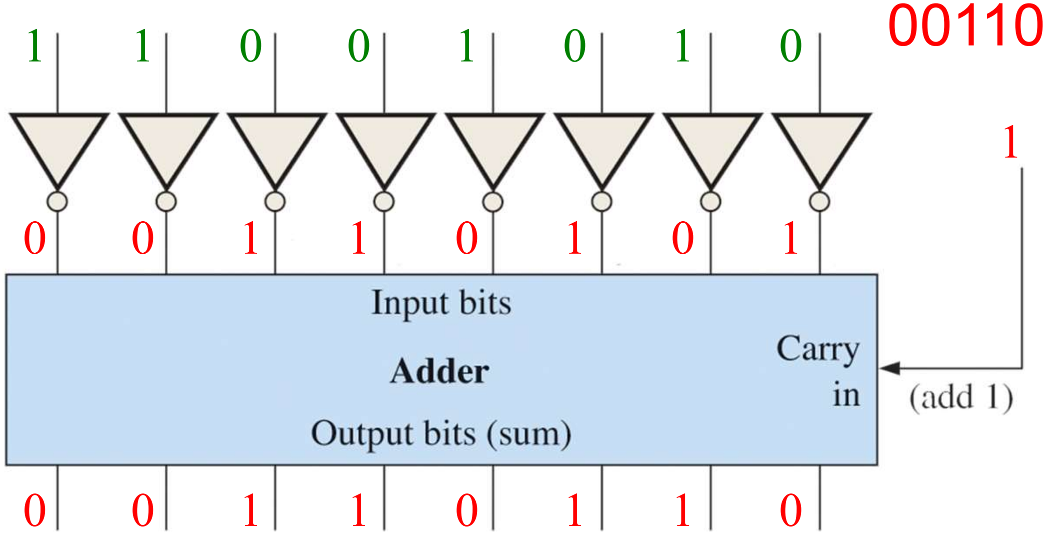


2’s Complement

The 2’s complement : 1’s complement + 1

Recall that the 1’s complement of 11001010 is

	00110101	(1’s complement)
For the 2’s complement, add 1:	+1	
	00110110	(2’s complement)



Signed Binary Numbers

MSB (most significant bit) represents the sign.

any symbol other than 0, 1 is not allowed!

Computers use a *modified 2's complement* for signed numbers.
Positive numbers: sign bit = 0

Negative numbers: sign bit = 1

For example, the positive number 58 is written using 8-bits as 00111010 (true form).

Sign bit

Magnitude bits

Signed Binary Numbers

Negative numbers = 2's complement of the positive number.

The number –58 is written as:

–58 = 11000110 (complement form)

Sign bit

Magnitude bits

MSB is a bit with weight -128.

We have 11000110 = –58, because

Column weights:	-128	64	32	16	8	4	2	1.	
	1	1	0	0	0	1	1	0	
	-128	+64				+4	+2		= –58

Floating Point Numbers

Floating point numbers: use scientific notation.

A 32-bit single precision number:

S	E (8 bits)	F (23 bits)

Sign bit   Biased exponent (+127)   Magnitude with MSB dropped

Express the speed of light, *c*, in single precision floating point notation. (c = 0.2998 x 10^9)

In binary, c = 0001 0001 1101 1110 1001 0101 1100 0000_2.

In scientific notation, c = 1.0001 1101 1110 1001 0101 1100 0000 x 2^{28}.

S = 0 because the number is positive. E = 28 + 127 = 155_{10} = 1001 1011_2. F is the next 23 bits after the first 1 is dropped.

In floating point notation, c =

0	10011011	0001 1101 1110 1001 0101 110

Arithmetic Operations With Signed Numbers

Use a *modified 2's complement* for addition and subtraction.

Rules for **addition**: Add the two signed numbers. Discard any final carries. The result is in signed form.

Examples:

00011110 = +30	00001110 = +14	11111111 = –1
00001111 = +15	11101111 = –17	11111000 = –8
00101101 = +45	11111101 = –3	~~1~~11110111 = –9

Discard carry

Arithmetic Operations With Signed Numbers

Note that if the number of bits required for the answer is exceeded, overflow will occur. This occurs only if both numbers have the same sign. The overflow is indicated by an incorrect sign bit.

Two examples are:

$$\begin{array}{r} 01000000 = +64 \\ 01000001 = +65 \\ \hline 10000001 = \cancel{-127} \end{array} \qquad \begin{array}{r} 10000001 = -127 \\ 10000001 = -127 \\ \hline \text{Discard carry} \rightarrow \cancel{1}00000010 = \cancel{+2} \end{array}$$

Wrong! The answer is incorrect and the sign bit has changed.

Arithmetic Operations With Signed Numbers

Rules for **subtraction**: use 2's complement and add the numbers. Discard any final carries. The result is in signed form.

Repeat the examples done previously, but subtract:

00011110 (+30)	00001110 (+14)	11111111 (–1)
– 00001111 –(+15)	– 11101111 –(–17)	– 11111000 –(–8)

2's complement, then add:

00011110 = +30	00001110 = +14	11111111 = –1
11110001 = –15	00010001 = +17	00001000 = +8
~~1~~00001111 = +15	00011111 = +31	~~1~~00000111 = +7

Discard carry

Discard carry

Ch.2 Summary

Hexadecimal Numbers

Hexadecimal: 16 symbols
0 through 9 and A through F.

Large binary number can easily be converted to hexadecimal by dividing it into 4-bit groups and converting each into its equivalent hexadecimal character.

Express $1001\ 0110\ 0000\ 1110_2$ in hexadecimal:

Group the binary number by 4-bits starting from the right. Thus, 1001011000001110 = 960E

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Ch.2 Summary

Hexadecimal Numbers

Weights: powers of 16

Column weights	16^3	16^2	16^1	16^0
	4096	256	16	1

Ex) express $1A2F_{16}$ in decimal.

4096	256	16	1
1	A	2	F_{16}

$1(4096) + 10(256) + 2(16) + 15(1) = 6703_{10}$

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Octal Numbers

Octal: 8 symbols
0 through 7

Binary number can easily be converted to octal by grouping bits 3 at a time and writing the equivalent octal character for each group.

Express $1\ 001\ 011\ 000\ 001\ 110_2$ in octal:

Group the binary number by 3-bits starting from the right. Thus, 1001011000001110 = 113016_8

Decimal	Octal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

Octal Numbers

Octal is also a weighted number system. The column weights are powers of 8, which increase from right to left.

Column weights	8^3	8^2	8^1	8^0
	512	64	8	1

Ex) express 3702_8 in decimal.

512	64	8	1
3	7	0	2

$3(512) + 7(64) + 0(8) + 2(1) = 1986_{10}$

Decimal	Octal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

Binary Coded Decimal(BCD)

Convert each decimal digits to binary number

BCD represents each decimal digit with a 4-bit code.

Codes 1010 through 1111 are not used in BCD.

Decimal	Binary	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	00010000
11	1011	00010001
12	1100	00010010
13	1101	00010011
14	1110	00010100
15	1111	00010101

BCD

Weights of BCD: 80 40 20 10 8 4 2 1.

Example: What are the weights for the BCD number 1000 0011 0101 1001?

8000 4000 2000 1000 800 400 200 100 80 40 20 10 8 4 2 1

Get the decimal number by adding weights:

$8000 + 200 + 100 + 40 + 10 + 8 + 1 = 8359_{10}$

Gray Code

Gray code is an unweighted code

A single bit change between one code word and the next in a sequence.

To avoid problems in systems where an error can occur if more than one bit changes at a time.

Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

ASCII

ASCII is a code for characters.

Originally, ASCII encoded 128 characters and symbols using 7-bits.

Ex) 35(0100011) -> #, 97(1100001) -> a

In 1981, IBM introduced extended ASCII, which is an 8-bit code and increased the character set to 256.
Other extended sets (such as Unicode) have been introduced to handle characters in languages other than English.

Parity Method for Error Detection

Method of error detection

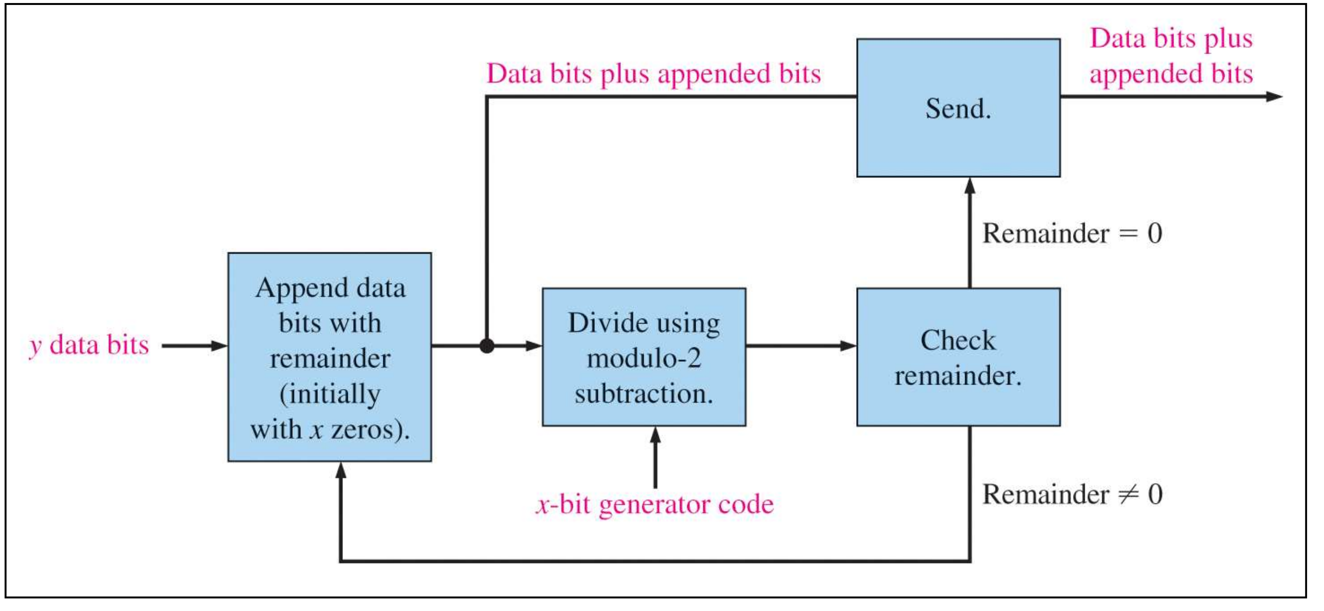
parity bit : “extra” bit to force the number of 1’s to be either even (*even parity*) or odd (*odd parity*).

The ASCII character for “a” is 1100001 and for “A” is 1000001. What is the correct bit to append to make both have odd parity?

The ASCII “a” has an odd number of bits that are equal to 1; therefore the parity bit is 0. The ASCII “A” has an even number of bits that are equal to 1; therefore the parity bit is 1.

Cyclic Redundancy Check

The cyclic redundancy check (CRC) is an error detection method that can detect multiple errors in larger blocks of data. At the sending end, a **checksum** is appended to a block of data. At the receiving end, the checksum is generated and compared to the sent checksum. If the check sums are the same, no error is detected.



Cyclic Redundancy Check

Example

11010011101100 with 1011