

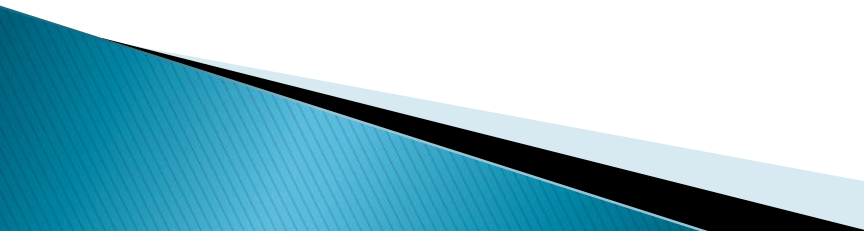
Switching Theory

Chapter 1

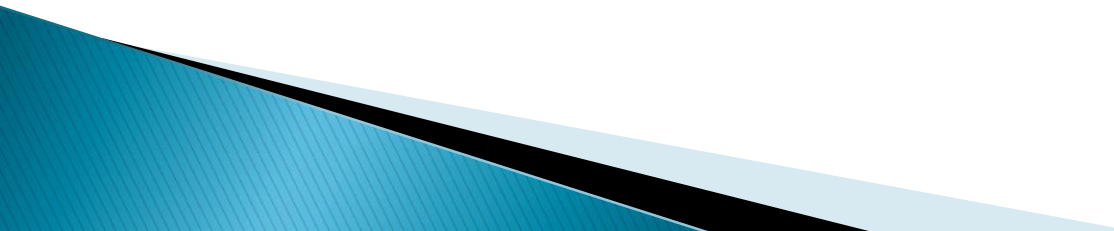
Emad Felemban



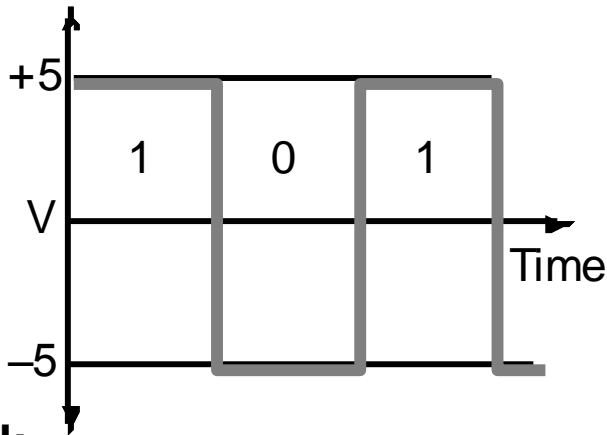
Contents

- ▶ Digital Computers and Digital Systems
 - ▶ Binary Numbers
 - ▶ Number Base Conversion
 - ▶ Octal and Hexadecimal Numbers
 - ▶ Complements
 - ▶ Signed Binary Numbers
 - ▶ Binary Codes
 - ▶ Binary Storage and Registers
 - ▶ Binary Logic
- 

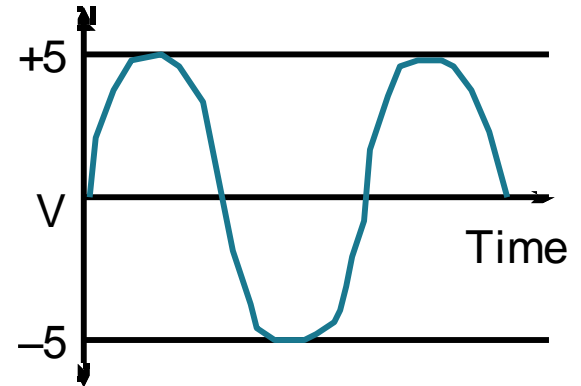
Digital Computers and Digital Systems



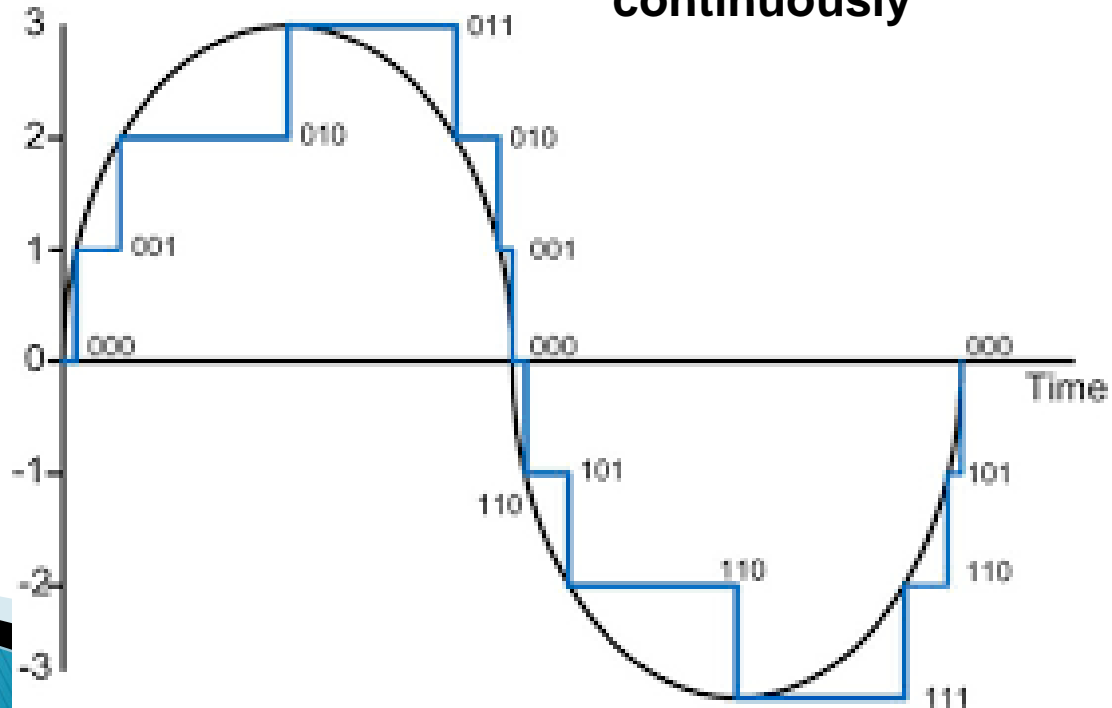
Digital vs. Analog Waveforms



Digital:
only assumes discrete values

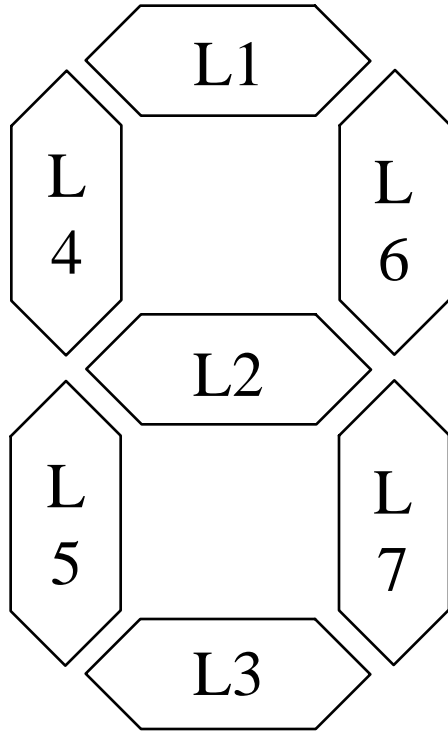


Analog:
values vary over a broad range continuously



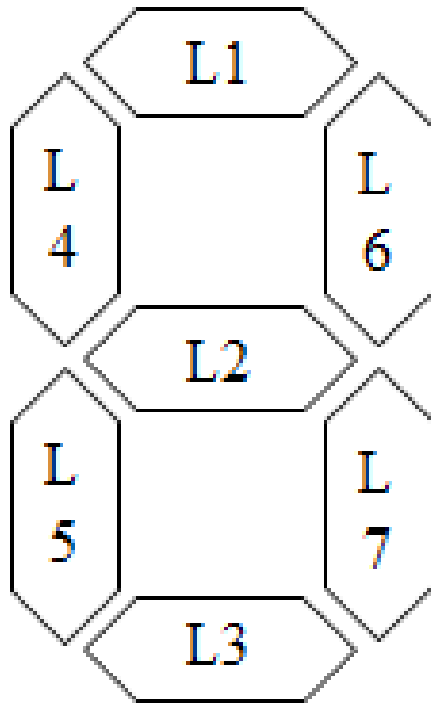
Case Study of a Simple Logic Design: Seven Segment Display

Chip to drive digital display ▶

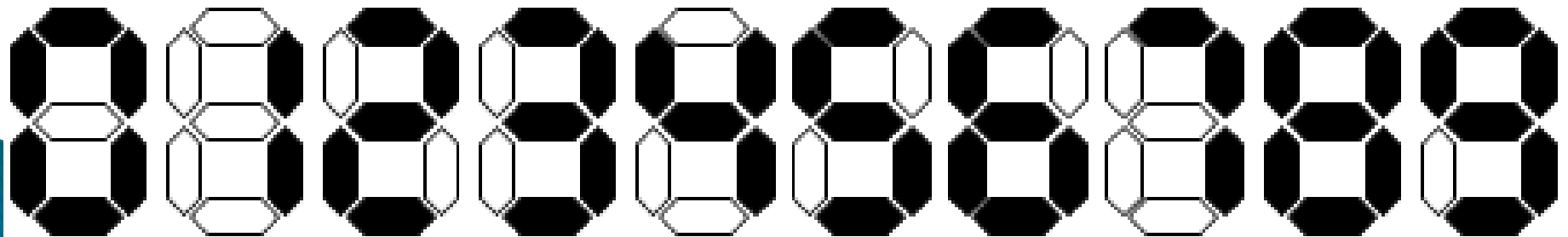


B3	B2	B1	B0	Val
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

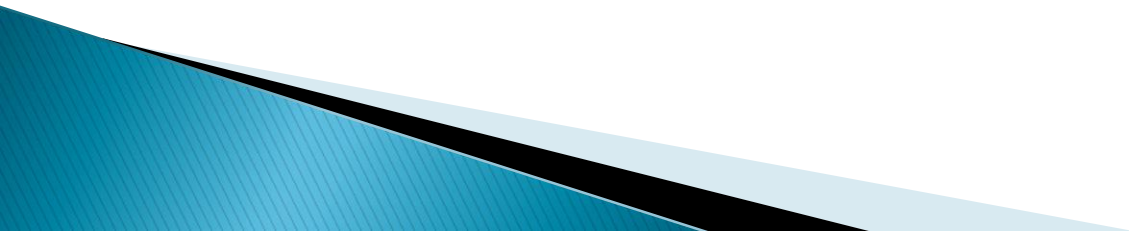
Case Study (cont.)



B3	B2	B1	B0	Val	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1



Binary Numbers



Why Binary?

- ▶ The basic unit of storage in a computer is the *bit* (binary digit), which can have one of just two values: 0 or 1.
- ▶ This is easier to implement in hardware than a unit that can take on 10 different values.
 - For instance, it can be represented by a transistor being off (0) or on (1).
 - Alternatively, it can be a magnetic stripe that is magnetized with North in one direction (0) or the opposite (1).
- ▶ Binary also has a convenient and natural association with logical values of False (0) and True (1).

Building on Binary

- ▶ Binary bits are grouped together to allow them to represent more information:
 - A *nybble* is a group of 4 bits, e.g. 1011
 - A *byte* is a group of 8 bits, e.g. 11010010
 - A *word* is a larger grouping: usually 32 bits. A *halfword* is half as many bits as a word, thus usually 16 bits. A *doubleword* is twice as many bits, usually 64 bits. However, computers have been designed with various word sizes, such as 36, 48, or 60 bits.

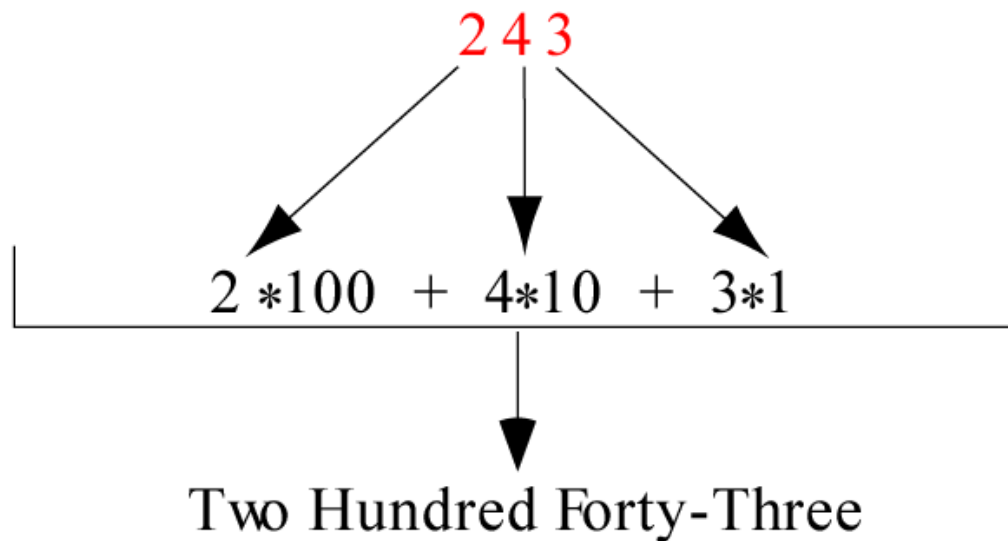
Building on Binary

- ▶ Clearly, the number of possible combinations of a group of N bits is $2^N = 2 \times 2 \times 2 \dots \times 2$ (N 2s). Thus:
 - A nybble can form $2^4 = 16$ combinations
 - A byte can form $2^8 = 256$ combinations
 - A 32-bit word can form $2^{32} = 4,294,967,296$ combinations

Decimal System

10^4	10^3	10^2	10^1	10^0
10,000	1000	100	10	1

Decimal Positions



Binary System

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Binary Positions

1 1 1 1 0 0 1 1

$$1*128 + 1*64 + 1*32 + 1*16 + 0*8 + 0*4 + 1*2 + 1*1$$

Two Hundred Forty-Three

Binary Numbers

Here are the first 16 binary numbers (using 4 bits), and their decimal equivalents.

$$0000 = 0$$

$$0001 = 1$$

$$0010 = 2$$

$$0011 = 3$$

$$0100 = 4$$

$$0101 = 5$$

$$0110 = 6$$

$$0111 = 7$$

$$1000 = 8$$

$$1001 = 9$$

$$1010 = 10$$

$$1011 = 11$$

$$1100 = 12$$

$$1101 = 13$$

$$1110 = 14$$

$$1111 = 15$$

Binary and Hexadecimal

- ▶ Because binary numbers are rather unwieldy, programmers prefer to use a more compact way to represent them. Historically, octal (base 8) and hexadecimal (base 16, or hex) have been used.
- Octal has the advantage that it uses only familiar digits, but it groups digits by threes, which is inconvenient for word sizes that are multiples of 4. So it is seldom used nowadays.
- Hexadecimal works nicely for bytes and for word sizes that are multiples of 4, but requires the introduction of 6 new digits.
- ▶ Conversion between binary and decimal is slow and is preferably avoided if there is no need. Octal and hex allow immediate conversion to and from binary.

Binary and Hexadecimal

Hexadecimal works by grouping binary bits into groups of 4 (starting from the right). Each group (a nybble) is assigned a hex digit value. The digits are the same as for decimal up to 9, and then letters A through F are used for 10 through 15.

0000 = 0

1000 = 8

0001 = 1

1001 = 9

0010 = 2

1010 = A

0011 = 3

1011 = B

0100 = 4

1100 = C

0101 = 5

1101 = D

0110 = 6

1110 = E

0111 = 7

1111 = F

Thus the 16-bit binary number

1011 0010 1010 1001

converted to hex is

B2A9

Number Base Conversion



Binary to decimal conversion

0 1 0 1 1 0 1

binary number

64 32 16 8 4 2 1

position values

0 + 32 + 0 + 8 + 4 + 0 + 1

results

45

decimal number

Example 2

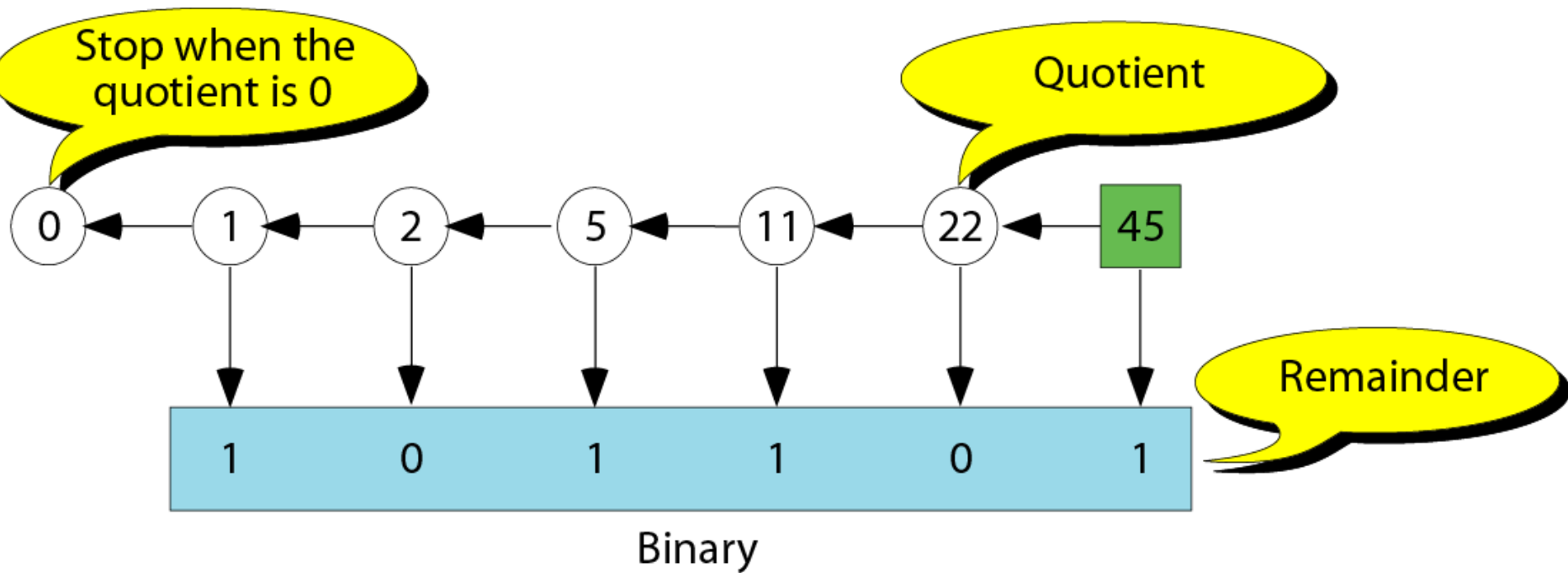
Convert the decimal number 35 to binary.

Solution

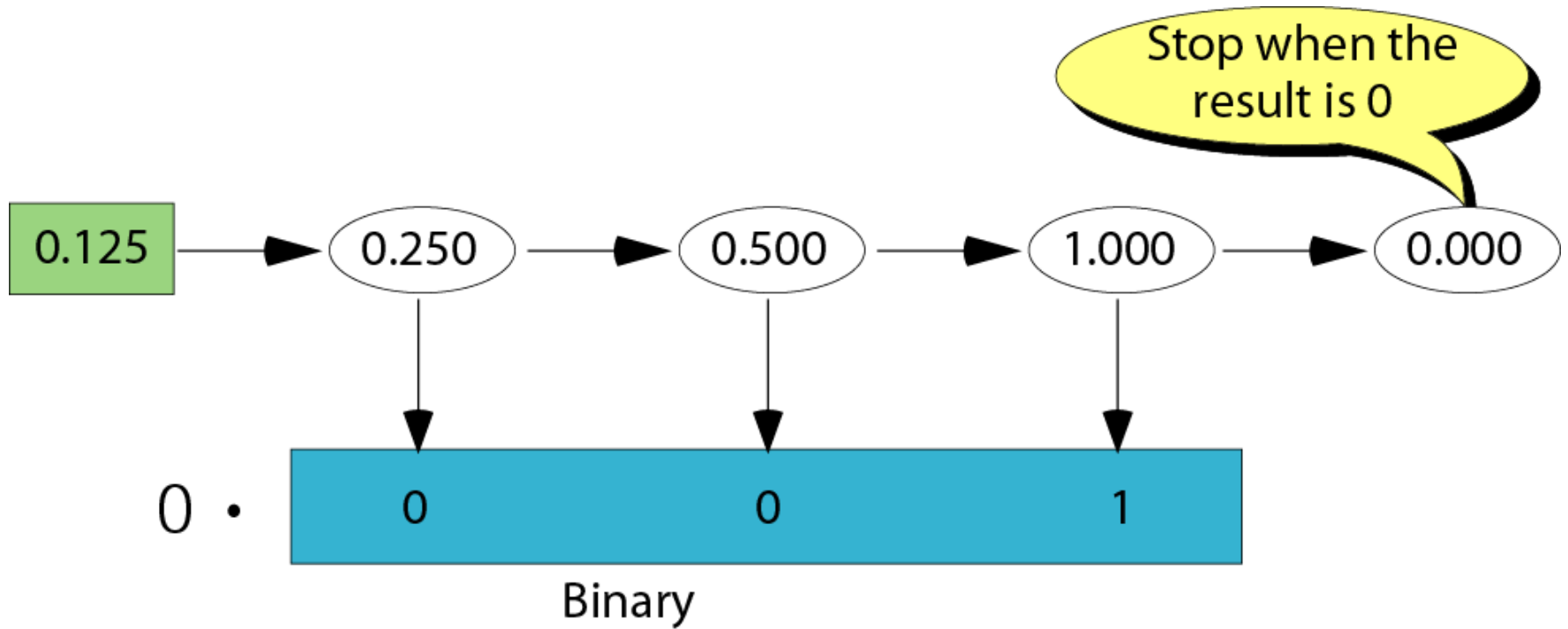
Write out the number at the right corner. Divide the number continuously by 2 and write the quotient and the remainder. The quotients move to the left, and the remainder is recorded under each quotient. Stop when the quotient is zero.

0	←	1	←	2	←	4	←	8	←	17	←	35	Dec.
Binary	1		0		0		0		1		1		

Decimal to binary conversion



Changing fractions to binary



Example 17

Transform the fraction 0.875 to binary

Solution

Write the fraction at the left corner. Multiply the number continuously by 2 and extract the integer part as the binary digit. Stop when the number is 0.0.

$$0.875 \rightarrow 1.750 \rightarrow 1.5 \rightarrow 1.0 \rightarrow 0.0$$

0 . 1 1 1

Example 18

Transform the fraction 0.4 to a binary of 6 bits.

Solution

Write the fraction at the left corner. Multiply the number continuously by 2 and extract the integer part as the binary digit. You can never get the exact binary representation. Stop when you have 6 bits.

0.4 → 0.8 → 1.6 → 1.2 → 0.4 → 0.8 → 1.6
0 . 0 1 1 0 0 1

Octal and Hexadecimal

Binary and Hexadecimal

- ▶ Because binary numbers are rather unwieldy, programmers prefer to use a more compact way to represent them. Historically, octal (base 8) and hexadecimal (base 16, or hex) have been used.
 - Octal has the advantage that it uses only familiar digits, but it groups digits by threes, which is inconvenient for word sizes that are multiples of 4. So it is seldom used nowadays.
 - Hexadecimal works nicely for bytes and for word sizes that are multiples of 4, but requires the introduction of 6 new digits.
- ▶ Conversion between binary and decimal is slow and is preferably avoided if there is no need. Octal and hex allow immediate conversion to and from binary.

Binary and Hexadecimal

Hexadecimal works by grouping binary bits into groups of 4 (starting from the right). Each group (a nybble) is assigned a hex digit value. The digits are the same as for decimal up to 9, and then letters A through F are used for 10 through 15.

0000 = 0

1000 = 8

0001 = 1

1001 = 9

0010 = 2

1010 = A

0011 = 3

1011 = B

0100 = 4

1100 = C

0101 = 5

1101 = D

0110 = 6

1110 = E

0111 = 7

1111 = F

Thus the 16-bit binary number

1011 0010 1010 1001

converted to hex is

B2A9

Complements

Complements

▶ Diminished Radix Complement

- 9's Complement of 546700 is $999999 - 546700 = 453299$
- 9's Complement of 012398 is $999999 - 012398 = 987601$
- 1's Complement of 1011000 is 0100111
- 1's Complement of 0101101 is 1010010

▶ Radix Complement

- 10's Complement of 546700 is 453300
- 10's Complement of 012398 is 987602
- 2's Complement of 1011000 is 0101000
- 2's Complement of 0101101 is 1010011

Subtraction with Complements

**Example
1-7**

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction (a) $X - Y$ and (b) $Y - X$ using 2's complements.

(a)

$$\begin{array}{r} X = \quad \quad \quad 1010100 \\ 2\text{'s complement of } Y = \quad + \underline{0111101} \\ \text{Sum} = \quad \quad \quad 10010001 \\ \text{Discard end carry } 2^7 = \quad - \underline{10000000} \\ \text{Answer: } X - Y = \quad \quad \quad 0010001 \end{array}$$

(b)

$$\begin{array}{r} Y = \quad \quad \quad 1000011 \\ 2\text{'s complement of } X = \quad + \underline{0101100} \\ \text{Sum} = \quad \quad \quad 1101111 \end{array}$$

There is no end carry.

$$\text{Answer: } Y - X = -(2\text{'s complement of } 1101111) = -0010001$$



Subtraction with Complements

**Example
1-8**

Repeat Example 1-7 using 1's complement.

(a) $X - Y = 1010100 - 1000011$

$$\begin{array}{r} X = \quad \quad \quad 1010100 \\ 1\text{'s complement of } Y = \quad + \underline{0111100} \\ \text{Sum} = \quad \quad \quad 10010000 \\ \text{End-around carry} \quad \quad \quad \rightarrow + \underline{1} \\ \text{Answer: } X - Y = \quad \quad \quad 0010001 \end{array}$$

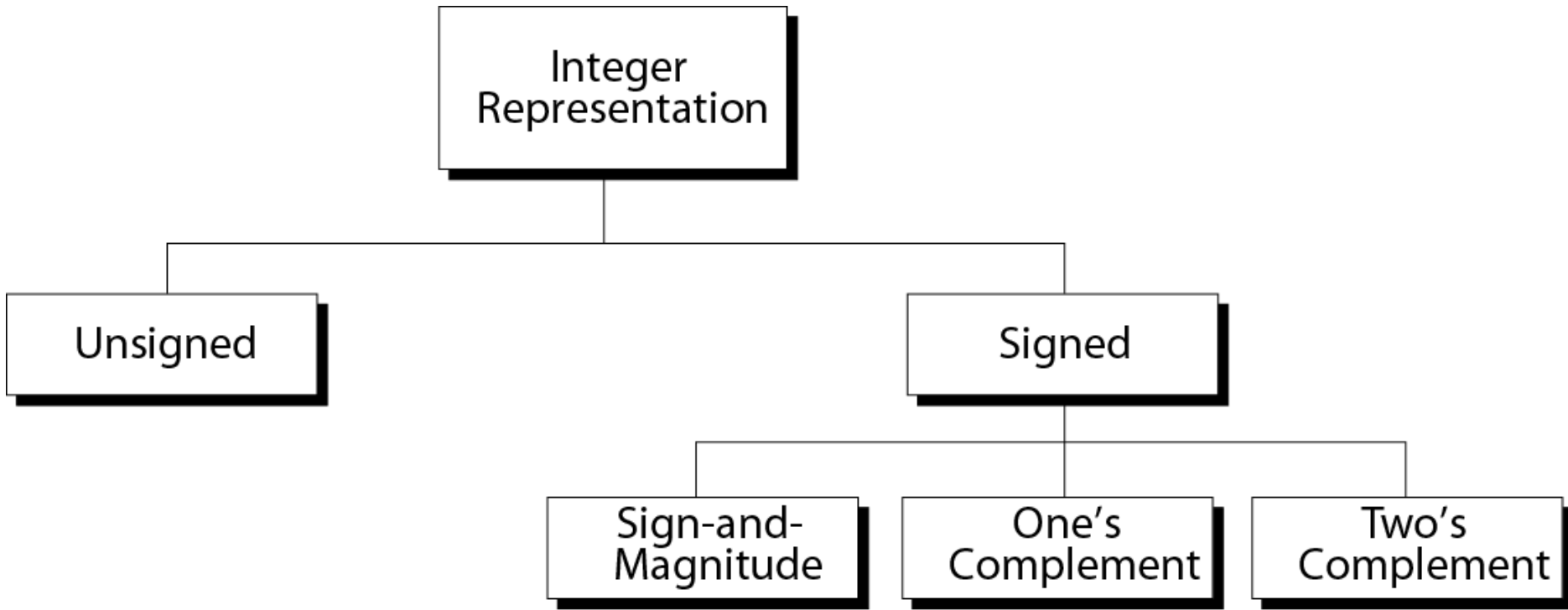
(b) $Y - X = 1000011 - 1010100$

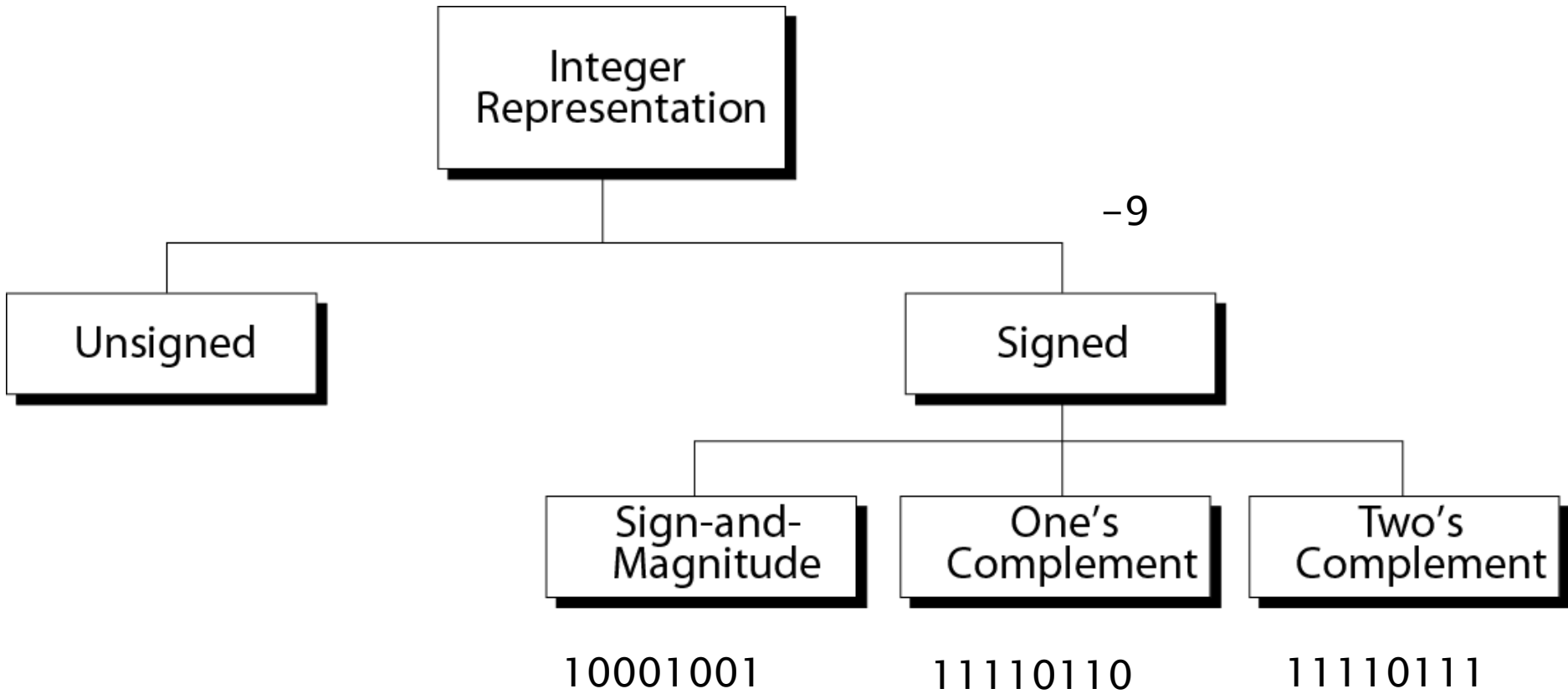
$$\begin{array}{r} Y = \quad \quad \quad 1000011 \\ 1\text{'s complement of } X = \quad + \underline{0101011} \\ \text{Sum} = \quad \quad \quad 1101110 \end{array}$$

There is no end carry.

Answer: $Y - X = -(1\text{'s complement of } 1101110) = -0010001$

Signed Binary Numbers





Arithmetic Addition and Subtraction

$$\begin{array}{r} + 6 \quad 00000110 \\ + 13 \quad 00001101 \\ \hline + 19 \quad 00010011 \end{array}$$

$$\begin{array}{r} + 6 \quad 00000110 \\ - 13 \quad 11110011 \\ \hline - 7 \quad 11111001 \end{array}$$

$$\begin{array}{r} - 6 \quad 11111010 \\ + 13 \quad 00001101 \\ \hline + 7 \quad 00000111 \end{array}$$

$$\begin{array}{r} - 6 \quad 11111010 \\ - 13 \quad 11110011 \\ \hline - 19 \quad 11101101 \end{array}$$

Binary Codes

TABLE 1-2
Binary codes for the decimal digits

Decimal digit	(BCD) 8421	Excess-3	84-2-1	2421	(Biquinary) 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Decimal	Binary	BCD	84-2-1	2421
23				
245				
11				
99				
100				
512				
1				

TABLE 1-5
American Standard Code for Information Interchange (ASCII)

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	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	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Binary Logic

TABLE 1-6
Truth Tables of Logical Operations

AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		



(a) Two-input AND gate



(b) Two-input OR gate



(c) NOT gate or inverter

Quiz

- 1-7
- 1-8
- 1-15
- 1-18
- 1-19
- 1-23