BCS –I DIGITAL ELECTRONICS

UNIT 1 : NUMBER SYSTEM & BINARY CODES

NUMBER SYSTEM:

An ordered set of symbols called the number system. These symbols are called as digits. Depending upon type and number of symbols used, there can be variety of number systems. However the most standard and popular number systems used in practice are:

1) Binary Number system
2) Decimal Number system
3) Octal Number system
4) Hexadecimal Number system

Radix or Base of the Number system: the number of distinct basic symbols or Digits used in a number system is known as Radix or Base of that Number system. It is used as Subscript to the number to indicate the particular number system.

Following table shows various Number systems with their basic symbols:

Number system	Basic symbols	Base or Radix	Example
Binary	0,1	2	1101_2
Octal	0,1,2,3,4,5,6,7	8	256_8
Decimal	0,1,2,3,4,5,6,7,8,9	10	458_{10}
Hexadecimal	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	16	$2A5B_{16}$

Out of these number systems, only the binary number system is used at hardware level because, digital circuit can process only binary numbers which are formed using 0s and 1s.

Binary Number system:

The number system with base 2 is called as Binary number system. It is widely used in digital system such as computer. It has two digits e.g. 0 and 1 , and are called as bits. The base of this number system is 2.

Binary numbers are formed very similar to decimal numbers as-

0,1,10,11,100,101,110,111,- - - - - ,1000,- - - - - -,10000,- - - - -

Binary number is indicated as 1110_2 and read as 'one- one -one-zero'

Decimal weights of binary digits are as shown in the table below.

-	-	2^4	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}	2^{-4}	-	-

Binary point

-	-	16	8	4	2	1	.	0.5	0.25	0.125	0.0625	-	-

Binary number can be expressed as sum of powers of 2 as follows:

$1101_2 = 1\times2^3 + 1 \times 2^2 + 0\times2^1 + 1\times2^0$

Group of 4 bits is known as a 'Nibble' and that of 8 bits as a 'Bytes'. A group of bits simultaneously processed by a digital system such as computer is known as a 'word'

e.g. 8-bit word,16-bit word,32-bit word, etc.

Advantages of binary number system are:

- Bits can be used to designate the two voltage levels in digital Electronics.
- Binary arithmetic is simple compared to decimal arithmetic.
- It is also called as Natural or Straight binary code.

Binary Number Examples

– 11 = 1 x 20 + 1 x 21 = 310

– 101 = 1 x 22 + 0 x 21 + 1 x 20 = 4 + 1 = 510.

– 1001 = 1 x 23 + 1 x 20 = 8 + 1 = 910.

– 1100 = 1 x 23 + 1 x 22 = 8 + 4 = 1210.

– 11101 = 1 x 24 + 1 x 23 + 1 x 22 + 1 x 20 = 16 + 8 + 4 + 1 = 2910.

– 0.1 = 1 X 2–1 = ½ = 0.510

– 0.111 = 1 X 2–1 + 1 X 2–2 + 1 X 2–3 = 0.5 + 0.25 + 0.125 = 0.87510
– 0.10001 = 1 X 2–1 + 1 X 2–5 = 0.5 + 0.03125 = 0.5312510
– 1101.01 = 1 x 23 + 1 x 22 + 1 x 20 + 1 x 2–2 = 8 + 4 + 1 + 0.25 = 13.2510
– 11.001 = 1 x 21 + 1 x 20 + 1 X 2–3 = 2 + 1 + 0.125 = 3.125 1 X 210
– 10.0011 = 1 x 21 + 1 X 2–3 + 1 X 2–4 = 2 + 0.125 + 0.0625 = 2.187510

Decimal Number system

It is commonly used number system. It has ten distinct digits and they are:0,1,2,3,4,5,6,7,8,9
The base of the no. system is 10. Decimal weights of decimal digits are as shown in the table below:

Decimal number can also be expressed as sum of powers of 10 as follows:
$3548.23_{10} = 3\times10^3 + 5\times10^2 + 4\times10^1 + 8\times10^0 + 2\times10^{-1} + 3\times10^{-2}$

-	-	10^4	10^3	10^2	10^1	10^0	.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	-	-

Decimal point

Octal Number system:

The octal number system is a base -8 nu [illegible] has 8 distinct digits and they are 0,1,2,3,4,5,6,7.
Decimal weights of an Octal digits are as shown in the table below:

Octal number can also be expressed as sum of powers of 8 as follows:
$5762.23_8 = 5\times8^3 + 7\times8^2 + 6\times8^1 + 2\times8^0 + 2\times8^{-1} + 3\times8^{-2}$

-	-	8^4	8^3	8^2	8^1	8^0	.	8^{-1}	8^{-2}	8^{-3}	8^{-4}	-	-

Octal point

Hexadecimal Number system:

The hexadecimal Number system is a base-16 Number system. It has 16 distinct digits and they are:
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
The base of this number system is 16.
Hexadecimal (Hex) number is indicated either as $34BC_{16}$ or 34BCH.
Decimal weights of Hexadecimal digits are as shown in the table below:

-	-	16^4	16^3	16^2	16^1	16^0	.	16^{-1}	16^{-2}	16^{-3}	16^{-4}	-	-

Hex point

Hexadecimal number can also be expressed as sum of powers of 16 as follows:

$4762.43_8 = 4\times16^3 + 7\times16^2 + 6\times16^1 + 2\times16^0 + 4\times16^{-1} + 3\times16^{-2}$
Table below shows some decimal numbers with their equivalent numbers of other system.

Decimal	Binary	Octal	Hexa-decimal
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

Interconversion from one Number system to another Number system:

Binary to Decimal conversion: Steamline Method:-
Procedure:

1) Rewrite the given binary number with little space between the bits.
2) Write equivalent decimal weights below each bit.
3) Decimal weights corresponding to bits zero are cancelled out
4) Remaining weights are added and decimal point is appropriately placed to get decimal equivalent of a given binary number.

Ex. $1011.101_2 = (x)_{10}$

$=1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$

$= 8 + 2 + 1 + 0.5 + 0.125$

$1011.101_2 = 11.625_{10}$

Decimal to Binary conversion:(Double-Dabble Method)
Ex. $(14.25)_{10} = (x)_2$
Procedure :
(for Decimal Integer)

1) Given decimal integer is progressively divided by 2
2) Remainders after each division are recorded on one side
3) This is continued till the division is not possible.(Quotient =0)
4) Remainders taken in reverse order i.e. last remainder as MSB and first as LSB forms the binary integer of the given decimal integer.

Ex. $(14)_{10} = (x)_2$

Decimal integer	Quotient	remainder	
14 ÷ 2	7	0	LSB
7 ÷ 2	3	1	↓
3÷ 2	1	1	↓
1 ÷ 2	0	1	MSB

Thus Answer is $(14)_{10} = (1110)_2$

Converting Decimal Fractions to Binary Fractions
• Decimal-fraction-to-binary-fraction conversion rules:

– Multiply decimal fraction by 2. Record the number left of the binary point. This will always be a 1 or a 0.
– Multiply the remaining fraction by 2, repeating above action.
– Continue until the fraction is eliminated.

• The method of successive multiplication: Decimal to binary fractions.
• Example – Convert 0.25 to binary:

– 0.25 x 2 = 0.5 ; note 0 to left of binary point.
– 0.5 x 2 = 1.0 ; note 1 to left of binary point.

– There is no part of the decimal fraction left, thus we are done.
– Read the binary fraction as the numbers to left of binary point in the two results, the remainder from the first multiplication first.
• Thus, 0.2510 = 0.012.

Converting Decimal ↔ Binary Fractions (2)
Additional examples: 0.375: 0.375 x 2 = 0.75
0.75 x 2 = 1.5
0.5 x 2 = 1.0
Thus 0.37510 = 0.0112.
0.427: 0.427 x 2 = 0.854
0.854 x 2 = 1.708
0.708 x 2 = 1.416 Then, 0.42710 ≈ 0.011011012.
0.416 x 2 = 0.832 (Finite decimal fractions can
0.832 x 2 = 1.664 result in repeating binary
0.664 x 2 = 1.328 fractions!*)
0.328 x 2 = 0.656
0.656 x 2 = 1.312
* Thus we must change the last rule on the previous page to read: "Continue until the decimal fraction is eliminated or the binary fraction has at least twice as many places as
21 Lecture #2: Binary, Hexadecimal, and Decimal Numbers © N. B. Dodge 01/12
y y p
the decimal fraction." This is because each column position in a binary fraction is much more significant than each column position in a decimal fraction

Converting Mixed Decimal Numbers
• Converting mixed decimal numbers means that we must perform two operations.

• For the integer part of the number, we do the method of successive division.
• For the decimal part, we do the method of successive multiplication.
• We recognize that for decimal numbers with a fraction part, we may not be able to convert the number exactly, since we could get a repeating fraction.
• In that case, we simply do successive multiplication enough times to get the accuracy of the binary fraction that we desire, at least twice the number of decimal places. Note that if the last of the 2x
22 Lecture #2: Binary, Hexadecimal, and Decimal Numbers © N. B. Dodge 01/12
places is a 0, continue until you get a 1).

Mixed Decimal Conversions -- Examples
• 36.125: 36/2=18, 0 rem; 18/2=9, 0 rem; 9/2=4, 1 rem; 4/2=2, 0 rem; 2/2=1, 0 rem, 1/2=0, 1 rem.
.125x2=0.25; .25x2=0.5; .5x2=1.0
Then 36.12510 = 100100.0012.
Read ← for integers,
Read → for fractions.
• 19.375: 19/2=9, 1 rem; 9/2=4, 1 rem; 4/2=2, 0 rem; 2/2=1, 0 rem; 1/2=0, 1 rem.
.375x2=0.75; .75x2=1.5; .5x2=1.0.
Then 19.37510 = 10011.0112.
• 7.33: 7/2=3, 1 rem; 3/2=1, 1 rem; 1/2=0, 1 rem.
.33x2=0.66; .66x2=1.32; .32x2=0.64; .64x2=1.28
Then 7.3310 ≈ 111.01012.
• 10.17: 10/2=5, 0 rem; 5/2=2, 1 rem; 2/2=1, 0 rem; 1/2=0. 1 rem.
.17x2=0.34; .34x2=0.68; .68x2=1.36; .36x2=0.72; .72x2=1.44
Then 10.1710 ≈ 1010.001012. *
23 Lecture #2: Binary, Hexadecimal, and Decimal Numbers © N. B. Dodge 01/12
*In the last example, we had to go to five places, since the fourth place was a 0.

Exercise #3
• Convert as directed:
– 237 to binary:
– 0.648 to binary: 0 1010 01
– 48.125 to binary:
– 123.45

Hexadecimal-to Decimal Conversion
• Since hex numbers are used in computer displays, it is useful to convert decimal ↔ hex and back.
• For hex → decimal, we use the same brute-force method, as for binary-to-decimal conversion. Consider 3FB716:
Number in hex: 3 F B 7
Position as a power of 16: 3 2 1 0
Decimal value of 16n: 4096 256 16 1
"F" "B"
The decimal number is then 3(4096)+15(256)+11(16)+7(1) = 16,311.

Hexadecimal-to Decimal Conversion (2)
• For fractions, the conversion is the same, remembering that hex digits to the right of the hexadecimal point are multipliers of negative powers of 16:
0x 0.2A6 = (2/16)+(10/[16]2)+(6/[16]3)
= 0.125 + 0.03906 + 0.00146
≈ 0.1655
• Mixed numbers are treated similarly:
0x B7.CE = 11(16) + 7(1) + 12/16 + 14/256

= 183.80469

Integer Decimal-to-Hex Conversion
• For converting decimal to hexadecimal integers, we use the
method of successive division, as for decimal/binary conversions:
Convert 38210 to hex: We perform successive divisions by 16.
382□16=23, remainder 14 (= E)
23 □16=1, remainder 7 Read in reverse order as before.
1 □16=0, remainder 1
• In reverse order, the hexadecimal number is 17E, or 38210 = 17E16.
• Similarly, converting 65110:
651 □16=40, remainder 11 (=B)
40□16=2, remainder 8
2□16=0, remainder 2
Note that as in binary conversion,
the last quotient will always be 0.
However, the last remainder may
b thi f 1t F
• Thus, 65110=28B16. be anything from 1 to F.

Integer Decimal-to-Hex Conversion (2)
• Additional examples of integer conversion (read ← for the answer):
• 100 =? 100/16=6, 4 rem; 6/16=0, 6 rem; thus 100 = 0x 64.
• 4096 =? 4096/16=256, 0 rem; 256/16=16, 0 rem; 16/16=1, 0 rem;
1/16=0, 1 rem; thus 4096 = 0x 1000.
• 335 =? 335/16=20, 15 (= 0x F) rem; 20/16=1, 4 rem; 1/16=0, 1 rem;
thus 335 = 0x 14F.
• 23795 =? 23795/16=1487, 3 rem; 1487/16=92, 15 (= 0x F) rem; 92/16=5,
12 (= 0x C) rem; 5/16=0, 5 rem; thus 23795 = 0x 5CF3.
• 1024 =? 1024/16=64, 0 rem; 64/16=4, 0 rem; 4/16=0, 4 rem;
thus 1024 = 0x 400.
• Note that remainders that are > 9 must be converted to the hex digits A-F to
get correct hexadecimal number
29 Lecture #2: Binary, Hexadecimal, and Decimal Numbers © N. B. Dodge 01/12
the number.

Hexadecimal to Binary

Converting from hexadecimal to binary is as easy as converting from binary to hexadecimal. Simply look up each hexadecimal digit to obtain the equivalent group of four binary digits.

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

Hexadecimal = A 2 D E

Binary = 1010 0010 1101 1110 = 1010001011011110 binary

Hexadecimal to Octal

When converting from hexadecimal to octal, it is often easier to first convert the hexadecimal number into binary and then from binary into octal. For example, to convert A2DE hex into octal:

(from the previous example)

Hexadecimal = A 2 D E

Binary = 1010 0010 1101 1110 = 1010001011011110 binary

Add leading zeros or remove leading zeros to group into sets of three binary digits.

Binary: 1010001011011110 = 001 010 001 011 011 110

Then, look up each group in a table:

Binary:	000	001	010	011	100	101	110	111
Octal:	0	1	2	3	4	5	6	7

Binary = 001 010 001 011 011 110

Octal = 1 2 1 3 3 6 = 121336 octal

Therefore, through a two-step conversion process, hexadecimal A2DE equals binary 1010001011011110 equals octal 121336.

Hexadecimal to Decimal

Converting hexadecimal to decimal can be performed in the conventional mathematical way, by showing each digit place as an increasing power of 16. Of course, hexadecimal letter values need to be converted to decimal values before performing the math.

Hexadecimal:	0	1	2	3	4	5	6	7
Decimal:	0	1	2	3	4	5	6	7
Hexadecimal:	8	9	A	B	C	D	E	F
Decimal:	8	9	10	11	12	13	14	15

A2DE hexadecimal:
$= ((A) * 16^3) + (2 * 16^2) + ((D) * 16^1) + ((E) * 16^0)$
$= (10 * 16^3) + (2 * 16^2) + (13 * 16^1) + (14 * 16^0)$
$= (10 * 4096) + (2 * 256) + (13 * 16) + (14 * 1)$
$= 40960 + 512 + 208 + 14$
= 41694 decimal

Octal to Binary

Converting from octal to binary is as easy as converting from binary to octal. Simply look up each octal digit to obtain the equivalent group of three binary digits.

Octal:	0	1	2	3	4	5	6	7
Binary:	000	001	010	011	100	101	110	111

Octal = 3 4 5

Binary = 011 100 101 = 011100101 binary

Octal to Hexadecimal

When converting from octal to hexadecimal, it is often easier to first convert the octal number into binary and then from binary into hexadecimal. For example, to convert 345 octal into hex:

(from the previous example)

Octal = 3 4 5

Binary = 011 100 101 = 011100101 binary

Drop any leading zeros or pad with leading zeros to get groups of four binary digits (bits):
Binary 011100101 = 1110 0101

Then, look up the groups in a table to convert to hexadecimal digits.

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

Binary = 1110 0101

Hexadecimal = E 5 = E5 hex

Therefore, through a two-step conversion process, octal 345 equals binary 011100101 equals hexadecimal E5.

Octal to Decimal

Converting octal to decimal can be done with repeated division.

1. Start the decimal result at 0.
2. Remove the most significant octal digit (leftmost) and add it to the result.
3. If all octal digits have been removed, you're done. Stop.
4. Otherwise, multiply the result by 8.
5. Go to step 2.

Octal Digits	Operation	Decimal Result	Operation	Decimal Result
345	+3	3	× 8	24
45	+4	28	× 8	224
5	+5	299	done.	

The conversion can also be performed in the conventional mathematical way, by showing each digit place as an increasing power of 8.

345 octal = $(3 * 8^2) + (4 * 8^1) + (5 * 8^0) = (3 * 64) + (4 * 8) + (5 * 1) = 229$ decima

Binary to Octal

An easy way to convert from binary to octal is to group binary digits into sets of three, starting with the least significant (rightmost) digits.

Binary: 11100101 = 11 100 101

011 100 101 Pad the most significant digits with zeros if necessary to complete a group of three.

Then, look up each group in a table:

Binary:	000	001	010	011	100	101	110	111
Octal:	0	1	2	3	4	5	6	7

Binary = 011 100 101
Octal = 3 4 5 = 345 oct

Binary to Hexadecimal

An equally easy way to convert from binary to hexadecimal is to group binary digits into sets of four, starting with the least significant (rightmost) digits.

Binary: 11100101 = 1110 0101

Then, look up each group in a table:

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

Binary = 1110 0101
Hexadecimal = E 5 = E5 hex

Binary to Decimal

They say there are only 10 people in this world: those that understand binary and those that don't. Ha ha.

If you don't get that joke, you'll need a method to convert from binary to decimal. One method involves addition and multiplication.

1. Start the decimal result at 0.
2. Remove the most significant binary digit (leftmost) and add it to the result.
3. If all binary digits have been removed, you're done. Stop.
4. Otherwise, multiply the result by 2.
5. Go to step 2.

Here is an example of converting 11100000000 binary to decimal:

Binary Digits	Operation	Decimal Result	Operation	Decimal Result
11100000000	+1	1	× 2	2
1100000000	+1	3	× 2	6
100000000	+1	7	× 2	14
00000000	+0	14	× 2	28
0000000	+0	28	× 2	56
000000	+0	56	× 2	112

00000	+0	112	× 2	224
0000	+0	224	× 2	448
000	+0	448	× 2	896
00	+0	896	× 2	1792
0	+0	1792	done.	

3. Decimal Number Conversion

(article continued from previous page)

A repeated division and remainder algorithm can convert decimal to binary, octal, or hexadecimal.

1. Divide the decimal number by the desired target radix (2, 8, or 16).
2. Append the remainder as the next most significant digit.
3. Repeat until the decimal number has reached zero.

Decimal to Binary

Here is an example of using repeated division to convert 1792 decimal to binary:

Decimal Number	Operation	Quotient	Remainder	Binary Result
1792	÷ 2 =	896	0	0
896	÷ 2 =	448	0	00
448	÷ 2 =	224	0	000
224	÷ 2 =	112	0	0000
112	÷ 2 =	56	0	00000
56	÷ 2 =	28	0	000000
28	÷ 2 =	14	0	0000000
14	÷ 2 =	7	0	00000000
7	÷ 2 =	3	1	100000000
3	÷ 2 =	1	1	1100000000
1	÷ 2 =	0	1	11100000000
0	done.			

Decimal to Octal

Here is an example of using repeated division to convert 1792 decimal to octal:

Decimal Number	Operation	Quotient	Remainder	Octal Result
1792	÷ 8 =	224	0	0
224	÷ 8 =	28	0	00
28	÷ 8 =	3	4	400
3	÷ 8 =	0	3	3400
0	done.			

Decimal to Hexadecimal

Here is an example of using repeated division to convert 1792 decimal to hexadecimal:

Decimal Number	Operation	Quotient	Remainder	Hexadecimal Result
1792	÷ 16 =	112	0	0
112	÷ 16 =	7	0	00
7	÷ 16 =	0	7	700
0	done.			

The only addition to the algorithm when converting from decimal to hexadecimal is that a table must be used to obtain the hexadecimal digit if the remainder is greater than decimal 9.

Decimal:	0	1	2	3	4	5	6	7
Hexadecimal:	0	1	2	3	4	5	6	7
Decimal:	8	9	10	11	12	13	14	15
Hexadecimal:	8	9	A	B	C	D	E	F

The addition of letters can make for funny hexadecimal values. For example, 48879 decimal converted to hex is:

Decimal Number	Operation	Quotient	Remainder	Hexadecimal Result
48879	÷ 16 =	3054	15	F
3054	÷ 16 =	190	14	EF
190	÷ 16 =	11	14	EEF

11	÷ 16 =	0	11	BEEF
0	done.			

Other fun hexadecimal numbers include: AD, BE, FAD, FADE, ADD, BED, BEE, BEAD, DEAF, FEE, ODD, BOD, DEAD, DEED, BABE, CAFE, C0FFEE, FED, FEED, FACE, BAD, F00D, and my initials DAC.

Now on to octal conversions...

Addition & Subtraction

484
201
57
1073

Let's first take a look at decimal addition.

As an example we have 26 plus 36,

```
 26
+36
```

To add these two numbers, we first consider the "ones" column and calculate 6 plus 6, which results in 12. Since 12 is greater than 9 (remembering that base 10 operates with digits 0-9), we "carry" the 1 from the "ones" column to the "tens column" and leave the 2 in the "ones" column.

Considering the "tens" column, we calculate 1 + (2 + 3), which results in 6. Since 6 is less than 9, there is nothing to "carry" and we leave 6 in the "tens" column.

```
 26
+36
 62
```

Binary addition

works in the same way, except that only 0's and 1's can be used, instead of the whole spectrum of 0-9. This actually makes binary addition much simpler than decimal addition, as we only need to remember the following:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

As an example of binary addition we have,

```
 101
+101
```

a) To add these two numbers, we first consider the "ones" column and calculate 1 + 1, which (in binary) results in 10. We "carry" the 1 to the "tens" column, and the leave the 0 in the "ones" column.

b) Moving on to the "tens" column, we calculate 1 + (0 + 0), which gives 1. Nothing "carries" to the "hundreds" column, and we leave the 1 in the "tens" column.

c) Moving on to the "hundreds" column, we calculate 1 + 1, which gives 10. We "carry" the 1 to the "thousands" column, leaving the 0 in the "hundreds" column.

```
 101
+101
1010
```

Another example of binary addition:

```
 1011
+1011
10110
```

Note that in the "tens" column, we have 1 + (1 + 1), where the first 1 is "carried" from the "ones" column. Recall that in binary,

```
1 + 1 + 1 = 10 + 1
          = 11
```

Binary subtraction

is simplified as well, as long as we remember how subtraction and the base 2 number system. Let's first look at an easy example.

```
 111
- 10
 101
```

Note that the difference is the same if this was decimal subtraction. Also similar to decimal subtraction is the concept of "borrowing." Watch as "borrowing" occurs when a larger digit, say 8, is subtracted from a smaller digit, say 5, as shown below in decimal subtraction.

```
  35
-  8
  27
```

For 10 minus 1, 1 is borrowed from the "tens" column for use in the "ones" column, leaving the "tens" column with only 2. The following examples show "borrowing" in binary subtraction.

```
  10    100   1010
 - 1   - 10  - 110
   1     10    100
```

Multiplication & Division

355

160

23

818

Binary multiplication

is actually much simpler than decimal multiplication. In the case of decimal multiplication, we need to remember 3 x 9 = 27, 7 x 8 = 56, and so on. In binary multiplication, we only need to remember the following,

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Note that since binary operates in base 2, the multiplication rules we need to remember are those that involve 0 and 1 only. As an example of binary multiplication we have 101 times 11,

```
 101
 x11
```

First we multiply 101 by 1, which produces 101. Then we put a 0 as a placeholder as we would in decimal multiplication, and multiply 101 by 1, which produces 101.

```
 101
 x11
 101
1010  <-- the 0 here is the placeholder
```

The next step, as with decimal multiplication, is to add. The results from our previous step indicates that we must add 101 and 1010, the sum of which is 1111.

```
 101
 x11
 101
1010
1111
```

Binary division

is almost as easy, and involves our knowledge of binary multiplication. Take for example the division of 1011 into 11.

```
      11   R=10
11 )1011
    -11
     101
     -11
      10  <-- remainder, R
```

To check our answer, we first multiply our divisor 11 by our quotient 11. Then we add its' product to the remainder 10, and compare it to our dividend of 1011.

```
   11
 x 11
   11
  11
 1001  <-- product of 11 and 11

 1001
+  10
 1011  <-- sum of product and remainder
```

The sum is equal to our initial dividend, therefore our solution is correct.

To practice binary addition and subtraction, visit the Practice Exercises page.

Rules of Binary Addition

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

For example,

00011010 + 00001100 = 00100110 *1 1* *carries*

0 0 0 1 1 0 1 0 = $26_{(\text{base }10)}$

+ 0 0 0 0 1 1 0 0 = $12_{(\text{base }10)}$

0 0 1 0 0 1 1 0 = $38_{(\text{base }10)}$

00010011 + 00111110 = 01010001 *1 1 1 1 1* *carries*

0 0 0 1 0 0 1 1 = $19_{(\text{base }10)}$

+ 0 0 1 1 1 1 1 0 = $62_{(\text{base }10)}$

0 1 0 1 0 0 0 1 = $81_{(\text{base }10)}$

Note: The rules of binary addition (without carries) are the same as the truths of the **XOR** gate.

Rules of Binary Subtraction

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

For example,

00100101 - 00010001 = 00010100 *0* *borrows*

0 0 $\cancel{1}$ ^{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*

$$\begin{array}{r l l}
 0\ 0\ \cancel{1}\ \cancel{1}\ \cancel{0}\ {}^{1}0\ 1\ 1 & = & 51_{(\text{base } 10)} \\
 -\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0 & = & 22_{(\text{base } 10)} \\
 \hline
 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1 & = & 29_{(\text{base } 10)}
\end{array}$$

Rules of Binary Multiplication

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

For example,

00101001 × 00000110 = 11110110

$$\begin{array}{r l l}
 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1 & = & 41_{(\text{base } 10)} \\
 \times\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0 & = & 6_{(\text{base } 10)} \\
 \hline
 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & & \\
 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ & & \\
 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ & & \\
 \hline
 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0 & = & 246_{(\text{base } 10)}
\end{array}$$

00010111 × 00000011 = 01000101

$$\begin{array}{r l l}
 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1 & = & 23_{(\text{base } 10)} \\
 \times\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 & = & 3_{(\text{base } 10)} \\
 \hline
 \mathit{1\ 1\ 1\ 1\ 1}\ & & \textit{carries} \\
 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1 & & \\
 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ & & \\
 \hline
 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1 & = & 69_{(\text{base } 10)}
\end{array}$$

Note: The rules of binary multiplication are the same as the truths of the **AND** gate.

Another Method: Binary multiplication is the same as repeated binary addition; add the multicand to itself the multiplier number of times.

For example,

00001000 × 00000011 = 00011000

$$\begin{array}{rcl}
{}^{\mathit{1}}\qquad\quad & & \textit{carries}\\
0\;0\;0\;0\;1\;0\;0\;0 & = & 8_{(\text{base } 10)}\\
0\;0\;0\;0\;1\;0\;0\;0 & = & 8_{(\text{base } 10)}\\
+\,0\;0\;0\;0\;1\;0\;0\;0 & = & 8_{(\text{base } 10)}\\
\hline
0\;0\;0\;1\;1\;0\;0\;0 & = & 24_{(\text{base } 10)}
\end{array}$$

Binary Division

Binary division is the repeated process of subtraction, just as in decimal division.

For example,

00101010 ÷ 00000110 = 00000111

$$\begin{array}{rcl}
1\;1\;1 & = & 7_{(\text{base } 10)}\\
\hline
1\,1\,0\,)\;0\;0\;\cancel{1}\;{}^{1}0\;1\;0\;1\;0 & = & 42_{(\text{base } 10)}\\
-\;1\;1\;0\qquad\quad & = & 6_{(\text{base } 10)}\\
\hline
\mathit{1}\qquad\quad & & \textit{borrows}\\
\cancel{1}\;\cancel{0}\;{}^{1}0\;1\quad & & \\
-\;1\;1\;0\quad & & \\
\hline
1\;1\;0 & & \\
-\;1\;1\;0 & & \\
\hline
0 & &
\end{array}$$

10000111 ÷ 00000101 = 00011011

$$\begin{array}{rcl}
1\;1\;0\;1\;1 & = & 27_{(\text{base } 10)}\\
\hline
1\,0\,1\,)\;\cancel{1}\;\cancel{0}\;\cancel{0}\;{}^{1}0\;0\;1\;1\;1 & = & 135_{(\text{base } 10)}\\
-\;1\;0\;1\qquad\qquad\quad & = & 5_{(\text{base } 10)}\\
\hline
1\;\cancel{1}\;{}^{1}0\qquad\quad & & \\
-\;1\;0\;1\qquad\quad & &
\end{array}$$

```
        -------
              1   1
          -       0
          ---------
              1   1 1
          -   1   0 1
          -----------
                  1 0 1
              -   1 0 1
              ---------
                      0
```

Notes

Binary Number System

System Digits: 0 and 1

Bit (short for *b*inary dig*it*): A single binary digit

LSB (least significant bit): The rightmost bit

MSB (most significant bit): The leftmost bit

Upper Byte (or nybble): The right-hand byte (or nybble) of a pair

Lower Byte (or nybble): The left-hand byte (or nybble) of a pair

Binary Equivalents

1 Nybble (or nibble) = 4 bits

1 Byte = 2 nybbles = 8 bits

1 Kilobyte (KB) = 1024 bytes

1 Megabyte (MB) = 1024 kilobytes = 1,048,576 bytes

1 Gigabyte (GB) = 1024 megabytes = 1,073,741,824 bytes

Multiplying fractions

As you might expect, the multiplication of fractions can be done in the same way as the multiplication of signed numbers. The magnitudes of the two multiplicands are multiplied, and the sign of the result is determined by the signs of the two multiplicands.

There are a couple of complications involved in using fractions. Although it is almost impossible to get an overflow (since the multiplicands and results usually have magnitude less than one), it is possible to get an overflow by multiplying -1x-1 since the result of this is +1, which cannot be represented by fixed point numbers.

The other difficulty is that multiplying two Q3 numbers, obviously results in a Q6 number, but we have 8 bits in our result (since we are multiplying two 4 bit numbers). This means that we end up with two bits to the left of the decimal point. These are sign extended, so that for positive numbers they are both zero, and for negative numbers they are both one. Consider the case of multiplying -1/2 by -1/2 (using the method from the textbook):

Decimal	Fractional Binary
-0.5 x0.5 -0.25	1100 x0100 0000 0000 +111100 11110000

This obviously presents a difficulty if we wanted to store the number in a Q3 result, because if we took just the 4 leftmost bits, we would end up with two sign bits. So what we'd like to do is shift the number to the left by one and then take the 4 leftmost bit. This leaves us with 1110 which is equal to -1/4, as expected.

One's Complementing

Before we discuss this representation further, we need to introduce two operations. The first is called ***one's complementing*** or taking the one's complement of an integer. The operation can be applied to any integer, positive or negative. This operation simply reverses (flips) each bit. A 0-bit is changed to a 1-bit, a 1-bit is changed to a 0-bit.

The following shows how we take the one's complement of the integer 00110110.

Original pattern	0	0	1	1	0	1	1	0
After applying one's complement operation	1	1	0	0	1	0	0	1

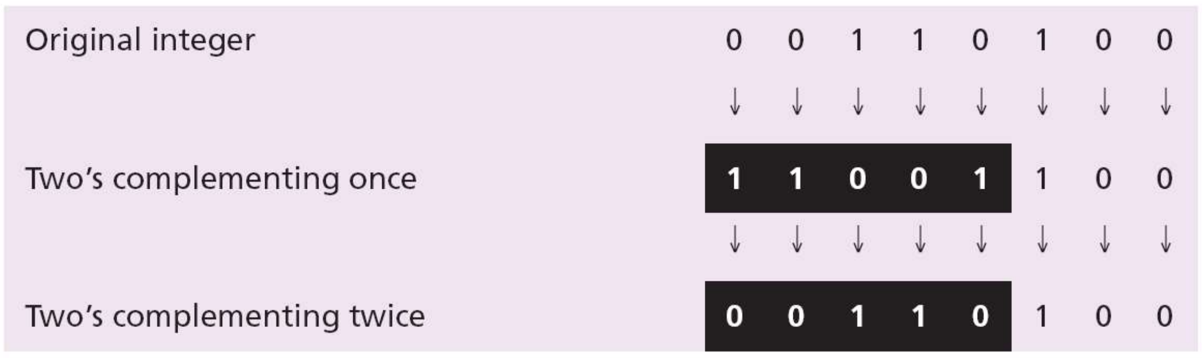
The following shows that we get the original integer if we apply the one's complement operations twice.

Original pattern	0	0	1	1	0	1	1	0
One's complementing once	1	1	0	0	1	0	0	1
One's complementing twice	0	0	1	1	0	1	1	0

Two's Complementing

The second operation is called ***two's complementing*** or taking the two's complement of an integer in binary. This operation is done in two steps. **First, we copy bits from the right until a 1 is copied; then, we flip the rest of the bits.**

The following shows how we take the two's complement of the integer 00110100.



Original integer	0	0	1	1	0	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
Two's complementing once	1	1	0	0	1	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
Two's complementing twice	0	0	1	1	0	1	0	0

An alternative way to take the two's complement of an integer is to first take the one's complement and then add 1 to the result.