

Section 1

Digital Logic Design Notes

Number Systems and Binary Codes

Revision 1
Ray Mitchell

Table Of Contents - Section 1
Number Systems and Binary Codes

I. Positional Number Systems	3
A. Characteristics	3
B. Converting Any Positional Number to Decimal	3
C. Converting a Decimal Number to Another Base	4
D. Converting Between Hex/Octal/Binary -- Regrouping Technique	6
E. Math In Any Base	7
II. Representation of Negative Numbers	8
A. Signed Magnitude Representation	8
B. Complement Representations	8
C. Diminished Radix Complement Representation (r - 1)'s	8
D. Radix Complement Representation r's	8
E. Subtraction Using Complements	9
F. Logical Complement	10
III. Binary Codes	11
A. Definitions	11
B. Decimal Codes - Represent numeric values	11
C. Gray (Reflected) Codes	12
D. Alphanumeric Codes	14
E. Control Codes	15
F. Code Distance	15
G. Error Detection Codes	16
IV. Binary Storage and Registers	17
A. Computer Storage	17
Section 1 Exercises	19
Section 1 Exercise Answers	21

C. Converting a Decimal Number to Another Base

1. Example: Convert 61.61_{10} to binary -- integral and fractional parts must be found separately:

a) The integral part (61_{10}) - series of divisions by the new base must be performed. All math is performed in decimal.

(1) Place the integral part of the number to be converted in the upper left position in the table calling it the dividend. Then place the desired base in the upper right, calling it the divisor.

Dividend	61	2	Divisor

(2) Divide the dividend by the divisor. Place the quotient on the left side of the table below the dividend and place the remainder on the right side of the table beside the quotient.

Dividend	61	2	Divisor
Quotient	30	1	Remainder

(3) The quotient then becomes the dividend for the next division, (however, the divisor remains the same). Repeat steps 1 and 2 until the quotient becomes zero.

	61	2	Divisor
	30	1	
	15	0	
	7	1	
	3	1	
Dividend	1	1	
Quotient	0	1	Remainder

(4) First remainder found goes next to the radix point in the new number (it is the LSD - least significant digit) and the rest of the remainders follow in order. Integral part of the new number is:

111101_2

- b) The fractional part ($.61_{10}$) - series of multiplications by the new base must be performed. All math is performed in decimal.

- (1) Place the fractional part of the number to be converted in the upper right position in the table, calling it the multiplicand. Then place the desired base in the upper left, calling it the multiplier.

Multiplier	2	.61	Multiplicand

- (2) Multiply the multiplicand by the multiplier. Place the fractional part of the product on the right side of the table below the multiplicand and place the integral part on the left side of the table beside the product.

Multiplier	2	.61	Multiplicand
Int Prod	1	.22	Fract Prod

- (3) The fractional part becomes the multiplicand for the next multiplication, (however, the multiplier remains the same). Repeat steps 1 and 2 until the fractional part becomes zero or until the desired fractional precision is obtained.

- (a) This second condition is necessary because most decimal fractions cannot be exactly represented in a power of 2 base like binary, octal, and hex, and as a result, the fractional product will never become zero.

Multiplier	2	.61	
	1	.22	Multiplicand
Int Prod	0	.44	Fract Prod

Multiplier	2	.61	
	1	.22	
	0	.44	Multiplicand
Int Prod	0	.88	Fract Prod

STOP

- (4) The integral product found as a result of the very first multiplication goes next to the radix point in the new number as was the case when the integral conversion of the original number was found. In this case, however, it is not the LSD but rather the MSD of the fractional part. So, the fractional part of the new number is:

$$\cong .100_2$$

2. The entire converted number may now be found by concatenating the results of the previous two procedures:

$$61.61_{10} \cong 111101.100_2$$

D. Converting Between Hex/Octal/Binary -- Regrouping Technique

1. The hex, octal, and binary bases are all powers of two, that is, 2^4 , 2^3 , and 2^1 respectively. This results in easy conversions between them using regrouping techniques.

2. Octal or Hex to Binary

a) Any octal or hex digit can be represented by a unique combination of 3 or 4 binary bits, respectively. Substitute this combination in place of each digit in the number to create its binary equivalent.

b) Always pad with 0s when necessary. Some examples are:

$$abcd.ef_{16} = 1010\ 1011\ 1100\ 1101 . 1110\ 1111_2$$

$$7510.16_{16} = 0111\ 0101\ 0001\ 0000 . 0001\ 0110_2$$

$$7510.16_8 = 111\ 101\ 001\ 000 . 001\ 110_2$$

3. Binary to Octal or Hex

a) Group the binary number into groups of 3 bits for octal and 4 bits for hex, starting at the radix point. Substitute the equivalent octal or hex digit in place of each binary group, as in:

$$11011001100001.0011001100_2$$

$$= 0011\ 0110\ 0110\ 0001 . 0011\ 0011\ 0000_2 = 3661.330_{16}$$

$$= 011\ 011\ 001\ 100\ 001 . 001\ 100\ 110\ 000_2 = 33141.1460_8$$

4. Hex to Octal

a) Convert the hex number to binary (in 4 bit groups). Regroup the binary into 3 bit groups and convert those groups to their octal equivalents. For example:

$$abcd.ef_{16} = 1010\ 1011\ 1100\ 1101 . 1110\ 1111_2$$

$$= 001\ 010\ 101\ 111\ 001\ 101 . 111\ 011\ 110_2$$

$$= 125715.736_8$$

5. Octal to Hex

a) Convert the octal number to binary (in 3 bit groups). Regroup the binary into 4 bit groups and convert those groups to their hex equivalents. For example:

$$12345670.012_8 = 001\ 010\ 011\ 100\ 101\ 110\ 111\ 000 . 000\ 001\ 010_2$$

$$= 0010\ 1001\ 1100\ 1011\ 1011\ 1000 . 0000\ 0101_2$$

$$= 29cbb8.05_{16}$$

E. Math In Any Base

1. Addition

- a) Add column at a time using base 10 math
- b) Divide sum of each column by the actual base of the numbers
 - (1) Keep remainder as sum for that column
 - (2) Use quotient as carry into next column

$$\begin{array}{r}
 c_{11} \\
 73_{10} \\
 +47_{10} \\
 \hline
 120_{10}
 \end{array}
 \qquad
 \begin{array}{r}
 c_{11} \\
 73_8 \\
 +47_8 \\
 \hline
 142_8
 \end{array}
 \qquad
 \begin{array}{r}
 c_0 \\
 73_{16} \\
 +47_{16} \\
 \hline
 (11)(10) \\
 BA_{16}
 \end{array}
 \qquad
 \begin{array}{r}
 c_{1111} \\
 1101_2 \\
 +1011_2 \\
 \hline
 11000_2
 \end{array}$$

2. Subtraction

- a) Subtract column at a time using base 10 math
- b) If a borrow is needed, borrow the value of the actual base of the numbers

$$\begin{array}{r}
 b_{10} \\
 73_{10} \\
 -47_{10} \\
 \hline
 26_{10}
 \end{array}
 \qquad
 \begin{array}{r}
 b_8 \\
 73_8 \\
 -47_8 \\
 \hline
 24_8
 \end{array}
 \qquad
 \begin{array}{r}
 b_{16} \\
 73_{16} \\
 -47_{16} \\
 \hline
 (2)(12) \\
 2C_{16}
 \end{array}
 \qquad
 \begin{array}{r}
 b_2 \\
 1101_2 \\
 -1011_2 \\
 \hline
 0010_2
 \end{array}$$

3. Multiplication

- a) Multiply digit at a time using base 10 math
- b) Divide each digit product by the actual base of the numbers
 - (1) Keep remainder as the digit product
 - (2) Use quotient as carry into next column
- c) Be sure to use previous rules for addition when adding partial products

$$\begin{array}{r}
 c_{22} \\
 23_4 \\
 \times 13_4 \\
 \hline
 201_4 \\
 +23_4 \\
 \hline
 1031_4
 \end{array}$$

II. Representation of Negative Numbers

A. Signed Magnitude Representation

1. Consists of a sign symbol followed by a magnitude, for example

$$23_{10}, +23_{10}, -23_{10}, 101_2, -101_2$$

2. For binary, an extra bit is typically used to represent the sign: 0 => positive, 1 => negative

$$101_2 \text{ or } +101_2 = 0101_2 \qquad -101_2 = 1101_2$$

B. Complement Representations

1. Negates a number by taking its complement
2. Complementing twice gives back the original number
3. Has a fixed number of digits

C. Diminished Radix Complement Representation $(r - 1)$'s

1. Subtract each digit from the largest possible digit (base - 1)

$$79.2_{10} \quad (r - 1)\text{'s (9's) complement} = 99.9_{10} - 79.2_{10} = 20.7_{10}$$

$$10.01_2 \quad (r - 1)\text{'s (1's) complement} = 11.11_2 - 10.01_2 = 01.10_2$$

$$0000_2 \quad (r - 1)\text{'s (1's) complement} = 1111_2 - 0000_2 = 1111_2 \quad (\text{negative } 0)$$

D. Radix Complement Representation r 's

1. Method 1: Add 1 to the LSD of the $(r - 1)$'s complement

$$79.2_{10} \quad r\text{'s (10's) complement} = 99.9_{10} - 79.2_{10} + .1 = 20.8_{10}$$

$$10.01_2 \quad r\text{'s (2's) complement} = 11.11_2 - 10.01_2 + .01 = 01.11_2$$

$$0000_2 \quad r\text{'s (2's) complement} = 1111_2 - 0000_2 + 1 = 0000_2 \quad (\text{discard carry})$$

2. Method 2: Subtract the number from r^n using base r math

$$79.2_{10} \quad r\text{'s (10's) complement} = 100.0_{10} - 79.2_{10} = 20.8_{10}$$

$$10.01_2 \quad r\text{'s (2's) complement} = 100.00_2 - 10.01_2 = 01.11_2$$

3. Method 3 for binary only:

a) Moving right-to-left, leave all leading 0s and the first 1 unaltered

b) Swap all remaining 1s and 0s

$$01101.11000_2 \quad \Rightarrow \quad 01101.\underline{11000}_2 \quad (\text{swap all but the underlined bits})$$

$$\Rightarrow \quad 10010.01000_2$$

E. Subtraction Using Complements

1. Computers normally subtract indirectly by adding complements (negations)
2. Procedures are independent of whether the numbers are signed or unsigned and may be generalized to numbers of any radix.
3. Subtrahend and the minuend must be adjusted to contain the same number of digits.
 - a) Right pad fractional part with 0s and left pad with 0s (unsigned) or sign extend (signed)
4. $(r-1)$'s Complement Subtraction
 - a) Take the $(r-1)$'s complement of the subtrahend
 - b) Obtain a sum by adding this complement to the unaltered minuend
 - c) If the addition causes a carry out of the most significant digit position
 - (1) The final answer will be a positive number and is obtained by discarding that carry and adding the digit 1 to the least significant position of the sum, even if it is a fractional position.
 - d) Else (no carry)
 - (1) The final answer will be a negative number and is obtained by taking the $(r-1)$'s complement of the sum and placing a minus sign in front of it.
 - e) Examples of $(r-1)$'s complement subtraction

$$\begin{array}{r} 920.8_{10} \\ -275.6_{10} \\ \hline \end{array} \Rightarrow \begin{array}{r} 920.8_{10} \\ +724.3_{10} \\ \hline 1645.1_{10} \end{array} \Rightarrow 645.1_{10} + 000.1_{10} \Rightarrow \mathbf{645.2_{10}}$$

$$\begin{array}{r} 275.6_{10} \\ -920.8_{10} \\ \hline \end{array} \Rightarrow \begin{array}{r} 275.6_{10} \\ +079.1_{10} \\ \hline 354.7_{10} \end{array} \Rightarrow \text{recomp (9's) \& chg sign} \Rightarrow \mathbf{-645.2_{10}}$$

$$\begin{array}{r} 101.0_2 \\ -001.1_2 \\ \hline \end{array} \Rightarrow \begin{array}{r} 101.0_2 \\ +110.0_2 \\ \hline 1011.0_2 \end{array} \Rightarrow 011.0_2 + 000.1_2 \Rightarrow \mathbf{011.1_2}$$

$$\begin{array}{r} 001.1_2 \\ -101.0_2 \\ \hline \end{array} \Rightarrow \begin{array}{r} 001.1_2 \\ +010.1_2 \\ \hline 100.0_2 \end{array} \Rightarrow \text{recomp (1's) \& chg sign} \Rightarrow \mathbf{-011.1_2}$$

5. r 's Complement Subtraction

- a) Take the $(r-1)$'s complement of the subtrahend (as before)
- b) Obtain a sum by adding together this complement, the unaltered minuend, and, in the least significant position, the digit 1.
- c) If the addition causes a carry out of the most significant digit position
 - (1) The final answer will be a positive number and is obtained by discarding that carry.
- d) Else (no carry)
 - (1) The final answer will be a negative number and is obtained by taking the r 's complement of the sum and placing a minus sign in front of it.
- e) Examples of r 's complement subtraction

$$\begin{array}{r}
 920.8_{10} \\
 -275.6_{10} \\
 \hline
 \end{array}
 \Rightarrow
 \begin{array}{r}
 000.1_{10} \\
 920.8_{10} \\
 +724.3_{10} \\
 \hline
 1645.2_{10}
 \end{array}
 \Rightarrow \text{throw away carry} \Rightarrow \mathbf{645.2_{10}}$$

$$\begin{array}{r}
 275.6_{10} \\
 -920.8_{10} \\
 \hline
 \end{array}
 \Rightarrow
 \begin{array}{r}
 000.1_{10} \\
 275.6_{10} \\
 +079.1_{10} \\
 \hline
 354.8_{10}
 \end{array}
 \Rightarrow \text{recomp (10's) \& chg sign} \Rightarrow \mathbf{-645.2_{10}}$$

$$\begin{array}{r}
 101.0_2 \\
 -001.1_2 \\
 \hline
 \end{array}
 \Rightarrow
 \begin{array}{r}
 000.1_2 \\
 101.0_2 \\
 +110.0_2 \\
 \hline
 1011.1_2
 \end{array}
 \Rightarrow \text{throw away carry} \Rightarrow \mathbf{011.1_2}$$

$$\begin{array}{r}
 001.1_2 \\
 -101.0_2 \\
 \hline
 \end{array}
 \Rightarrow
 \begin{array}{r}
 000.1_2 \\
 001.1_2 \\
 +010.1_2 \\
 \hline
 100.1_2
 \end{array}
 \Rightarrow \text{recomp (2's) \& chg sign} \Rightarrow \mathbf{-011.1_2}$$

F. Logical Complement

1. 1's complement is the only complement method used in binary logic (Boolean algebra)

III. Binary Codes

A. Definitions

1. BIT: Binary Digit
2. BYTE: Commonly 8 bits
3. NIBBLE: 1/2 BYTE (4 bits)
4. \$.25: 2 bits
5. CODE: Set of n bit strings whose members represent numbers, letters, or other objects
 - a) n bits produce 2^n unique combinations

Table 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

B. Decimal Codes - Represent numeric values

1. Weighted Codes

a) A code where each digit position has a weight - total value determined by adding weights

b) Examples: BCD, 84-2-1, 2421, Biquinary

(1) BCD represents 0-9 in binary using 4 bits ($2^4 = 16$ possible combinations), 10 used, 6 unused

(a) BCD representation of 255 = 0010 0101 0101₂

(b) Binary representation of 255 = 11111111₂

(2) Packed BCD puts 2 BCD digits in one 8-bit byte

2. Non-weighted codes

a) Value determined by other than column weight

b) Example: Excess-3

(1) Excess-3 is merely BCD with 3 added

3. Self-Complementing Codes

a) Examples are Excess-3, 84-2-1, 2421

b) A code in which 1's complementing a binary representation yields a value whose decimal equivalent is the 9's complement of the original decimal value

(1) For example in the Excess-3 code:

$$0100_2 = 1_{10} \quad \text{and} \quad 1011_2 = 8_{10}$$

(2) For example in the 84-2-1 code:

$$0111_2 = 1_{10} \quad \text{and} \quad 1000_2 = 8_{10}$$

C. Gray (Reflected) Codes

Table 3 Some Typical 4 Bit Reflected Codes (Gray Codes)

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

1. Used to interface digital and mechanical systems

a) For example, detecting the position of a rotating shaft

2. Adjacent code numbers differ by one bit

3. Eliminates ambiguity on mechanical boundaries, for example

	Binary			Gray	
00	n	n	00	n	n
01	n		01	n	
10		n	11		
11			10		n

4. Creating A Gray Code

Start	Reflect	Expand	Reflect	Expand
1	1	01	01	001
0	0	00	00	000
	<hr/>	<hr/>		
	0	10	10	010
	1	11	11	011
			<hr/>	<hr/>
			11	111
			10	110
			00	100
			01	101

Another of several possible variations on the above:

Start	Reflect	Expand	Reflect	Expand
1	1	11	11	011
0	0	10	10	010
	<hr/>	<hr/>		
	0	00	00	000
	1	01	01	001
			<hr/>	<hr/>
			01	101
			00	100
			10	110
			11	111

D. Alphanumeric Codes

1. Represent letters, digits, symbols, control characters

Table 4 Codes For Selected Alphanumeric Characters

Character	7-Bit	8-Bit	12-Bit Punch
	ASCII Code	EBCDIC Code	Card Code
A	100 0001	1100 0001	12,1
B	100 0010	1100 0010	12,2
C	100 0011	1100 0011	12,3
D	100 0100	1100 0100	12,4
E	100 0101	1100 0101	12,5
F	100 0110	1100 0110	12,6
G	100 0111	1100 0111	12,7
H	100 1000	1100 1000	12,8
I	100 1001	1100 1001	12,9
J	100 1010	1101 0001	11,1
K	100 1011	1101 0010	11,2
L	100 1100	1101 0011	11,3
M	100 1101	1101 0100	11,4
N	100 1110	1101 0101	11,5
O	100 1111	1101 0110	11,6
P	101 0000	1101 0111	11,7
Q	101 0001	1101 1000	11,8
R	101 0010	1101 1001	11,9
S	101 0011	1110 0010	0,2
T	101 0100	1110 0011	0,3
U	101 0101	1110 0100	0,4
V	101 0110	1110 0101	0,5
W	101 0111	1110 0110	0,6
X	101 1000	1110 0111	0,7
Y	101 1001	1110 1000	0,8
Z	101 1010	1110 1001	0,9
0	011 0000	1111 0000	0
1	011 0001	1111 0001	1
2	011 0010	1111 0010	2
3	011 0011	1111 0011	3
4	011 0100	1111 0100	4
5	011 0101	1111 0101	5
6	011 0110	1111 0110	6
7	011 0111	1111 0111	7
8	011 1000	1111 1000	8
9	011 1001	1111 1001	9

2. ASCII - American Standard Code for Information Interchange

- a) Is a 7 bit code
- b) Values of characters and numbers are sequential and contiguous

3. EBCDIC - Extended BCD Interchange Code

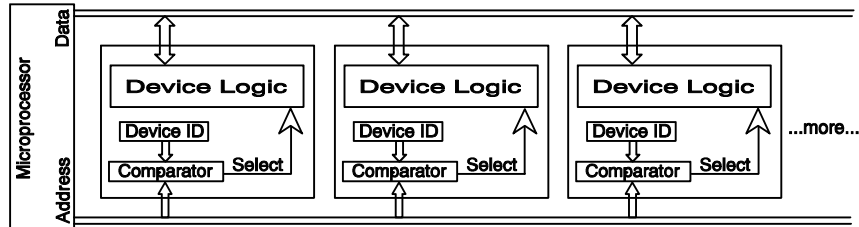
- a) Another code

4. Punch Card Code, 12 bit

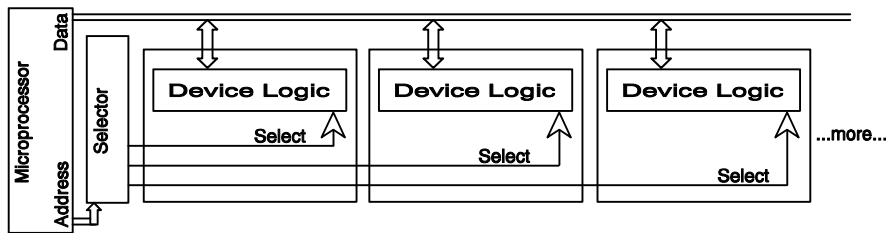
a) Hole => 1 No Hole => 0

E. Control Codes

1. Device selection using address encoded device ID



2. Device selection using address decoded device ID



3. Action control in a device register

15 - 6	5 - 3	2	1	0
message nbr	speed select	stop bits	parity type	check parity

F. Code Distance

1. String Distance - Number of bits by which any two strings differ

0001_2 and 1001_2 Have a distance 1

2. Code Distance - The smallest distance between any two strings in the code set

a) BCD is a distance 1 code

G. Error Detection Codes

1. Require a minimum code distance of 2
2. One bit parity codes are distance 2 codes that can detect an odd number of bit errors

Table 5 Parity Bit Codes

Message	Parity Bit Needed (for odd parity)	Parity Bit Needed (for even parity)
0000	1	0
0001	0	1
0010	0	1
0011	1	0
0100	0	1
0101	1	0
0110	1	0
0111	0	1
1000	0	1
1001	1	0
1010	1	0
1011	0	1
1100	1	0
1101	0	1
1110	0	1
1111	1	0

3. Biquinary Code

- a) Distance 2; Always 5 zeros and 2 ones; Receiver can count them as an error test;

4. As code distance increases so does ability to detect errors

5. Some codes also permit error correction

- a) Erroneous string can be set equal to the string to which it is the closest, for example

If the following string is received 1100101_2 and
 two members of the distance 3 code are 1100100_2
 and 1100011_2

the received string is at distance 1 from the first member and distance 2 from the second, both illegal distances. It is closest to the first member and may be corrected to that value.

(1) Not quite that simple

6. Longitudinal Parity Checking

- a) Transmitter counts the number of 1's sent and appends that count to the message
- b) Receiver also counts and compares its count to the count appended by the transmitter

7. Checksum Codes

- a) Transmitter adds the numeric value each byte transmitted and appends that sum to the message
- b) Receiver also adds and compares its sum with the sum appended by the transmitter

IV. Binary Storage and Registers

A. Computer Storage

1. Computers store bits in bi-stable (equally stable storing a 0 or a 1) binary cells
 - a) Flip-flop, capacitor, magnetic spot, optical spot, ferrite core (magnetic)
 - (1) Magnetic and optical storage is non-volatile
2. Registers
 - a) Generically, a group of parallel binary cells
 - b) Content is dependent upon interpretation of data, for example,
 $100\ 0001_2$: ASCII => A, EBCDIC => undefined, Binary => 65_{10}
 - c) Computers operate by transferring data between registers

Section 1 Exercises

- 1-1. Write the first 20 decimal digits in base 5.
- 1-2. Add and multiply the following numbers in the given base without converting to decimal:
- a) 3132_4 and 23_4 b) 5321.5_6 and 53.3_6 c) 75_8 and 426_8 d) AB_{13} and 77_{13}
- 1-3. Convert 1234.5_{10} to base 2, base 3, base 4, base 7, base 8, and base 16.
- 1-4. Convert the following numbers to binary: 31.0625_{10} , 10^3_{10} , 123.123_{10} , and 1023_{10}
- 1-5. Convert the following numbers to decimal: 110.101_2 , 111100.1101_2 , 1101.1110_2 , 11110111.100000_2 .
- 1-6. Convert the following numbers from the given base to the bases indicated:
- a) 795.63_{10} to binary, octal, and hexadecimal
b) 110111.110_2 to decimal, octal, and hexadecimal
c) 777.66_8 to decimal, binary, and hexadecimal
d) $ABCD.EF_{16}$ to decimal, octal, and binary
- 1-7. Convert the following numbers to decimal:
- a) 1111001.011_2 b) 2222_3 c) 1220.2_4 d) 4321_5
e) 0.526_7 f) 50_6 g) 7.8_9 h) $19AB_{12}$
- 1-8. Obtain the 1's and 2's complement of: 1101010_2 , 1110001_2 , 0001_2 , 1000_2 , 000000_2 .
- 1-9. Obtain the 9's and 10's complement of: 56789_{10} , 9999_{10} , 98760_{10} , 100000_{10} , 000000_{10} .
- 1-10. Find the 10's complement of 98765_{11} .
- 1-11. Perform the subtraction with the following numbers first using radix complements, then using diminished radix complements. Check each answer using straight subtraction.
- a) $6734_{10} - 291_{10}$ b) $3570_8 - 2600_8$ c) $7A53_{11} - A864_{11}$ d) $20_3 - 1000_3$
- 1-12. Perform the subtraction with the following numbers first using 2's complements, then using 1's complements. Check each answer using straight subtraction. It is irrelevant whether the numbers are signed or unsigned. If the answer is negative, represent it with its positive binary value preceded by a minus sign.
- a) $1101_2 - 110_2$ b) $1110_2 - 1000_2$ c) $0010_2 - 1001_2$ d) $100_2 - 11000_2$
- 1-13. Using the following two weighted codes to represent the decimal digits, determine all possible tables so that the 9's complement of each decimal digit is obtained by changing 1's to 0's and 0's to 1's.
- a) 3, 3, 2, 1 b) 4, 4, 3, -2

- 1-14. Represent 4328_{10}
- a) in BCD
 - b) in excess-3 code
 - c) in 2, 4, 2, 1 code
 - d) as a binary number
- 1-15. Obtain a weighted binary code for the base-12 digits using weights of 5421.
- 1-16. Determine the even-parity bit generated when a message consists of the ten decimal digits in the 8, 4, -2, -1 code.
- 1-17. Obtain a self-complementing binary code to represent all base-6 digits.
- 1-18. Using the minimum number of bits, assign a binary code in some orderly manner to the 52 playing cards.
- 1-19. Write your first name, middle initial, and last name in an eight-bit code made up the seven ASCII bits from the ASCII code table in the book and an odd parity bit in the most significant position. Include blanks between names and a period after the middle initial.
- 1-20. Show the configuration of a 32-bit register when its content represents
- a) the number 234_{10} in binary
 - b) the number 234_{10} in BCD
 - c) the characters C25D in 8-bit, odd parity ASCII. Parity is assigned for each byte and is the leftmost bit.
- 1-21. A 12-bit register contains 001101101001. What is its value if it represents
- a) three decimal digits in BCD
 - b) three decimal digits in excess-3 code
 - c) three decimal digits in 2, 4, 2, 1 code

Section 1 Exercise Answers

1-1. 0, 1, 2, 3, 4, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30, 31, 32, 33, 34

1-2. SUM PRODUCT

- | | | |
|----|------------|---------------|
| a) | 3221_4 | 212022_4 |
| b) | 5415.2_6 | 510221.23_6 |
| c) | 523_8 | 41076_8 |
| d) | 155_{13} | $639C_{13}$ |

1-3. 10011010010.1_2 , 1200201.1_3 , 103102.2_4 , 3412.3_7 , 2322.4_8 , $4d2.8_{16}$

1-4. 11111.0001, 1111101000, 1111011.000111, 1111111111

1-5. 6.625, 60.8125, 13.875, 247.5

1-6. decimal binary octal hexadecimal

- | | | | | |
|----|-------------|---------------------------|------------|-----------|
| a) | 795.63 | 1100011011.10... | 1433.50... | 31b.A1... |
| b) | 55.75 | 110111.110 | 67.6 | 37.C |
| c) | 511.84... | 111111111.11011 | 777.66 | 1FF.D8 |
| d) | 43981.93... | 1010101111001101.11101111 | 125715.736 | ABCD.EF |

1-7. a) 121.375_{10} b) 80_{10} c) 104.5_{10} d) 586_{10}
 e) $0.772..._{10}$ f) 30_{10} g) $7.8..._{10}$ h) 3155_{10}

1-8. 1's: 0010101, 0001110, 1110, 0111, 111111
 2's: 0010110, 0001111, 1111, 1000, 000000

1-9. 9's: 43210, 0000, 01239, 899999, 999999
 10's: 43211, 0001, 01240, 900000, 000000

1-10. 12345_{11}

1-11. a) 6443_{10} b) 770_8 c) -2911_{11} d) -210_3

1-12. a) 111_2 b) 110_2 c) -111_2 d) -10100_2

1-13. 3321 3321 3321
0 0000 0000 0000
1 0001 0001 0001
2 0010 0010 0010
3 0011 0011 0100
4 0101 1001 0101
5 1010 0110 1010
6 1100 1100 1011
7 1101 1101 1101
8 1110 1110 1110
9 1111 1111 1111

3321 3321 3321
0 0000 0000 0000
1 0001 0001 0001
2 0010 0010 0010
3 0100 1000 1000
4 1001 0101 1001
5 0110 1010 0110
6 1011 0111 0111
7 1101 1101 1101
8 1110 1110 1110
9 1111 1111 1111

443-2 443-2
0 0000 0000
1 0011 0011
2 1001 1001
3 0010 0010
4 1000 0100
5 0111 1011
6 1101 1101
7 0110 0110
8 1100 1100
9 1111 1111

443-2 443-2
0 0000 0000
1 0011 0011
2 0101 0101
3 0010 0010
4 1000 0100
5 0111 1011
6 1101 1101
7 1010 1010
8 1100 1100
9 1111 1111

1-14. a) 0100 0011 0010 1000 b) 0111 0110 0101 1011
c) 0100 0011 0010 1110 d) 1 0000 1110 1000

1-15. 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1011, 1100, 1101, 1110

1-16. 00000, 10111, 00110, 00101, 10100, 11011, 01010, 01001, 11000, 01111

1-17. 0 1 2 3 4 5
000, 001, 010, 101, 110, 111

1-18. Two bits for suit, four bits for the numbers 0010 - 1010, and J = 1011, Q = 1100, K = 1101, A = 1110

1-19. J 01001010
O 01001111
E 01000101
 00100000
M 11001101
 10101110
 00100000
S 11010011
M 11001101
I 01001001
T 01010100
H 11001000

1-20. a) 0000 0000 0000 0000 0000 0000 1110 1010
b) 0000 0000 0000 0000 0000 0010 0011 0100
c) 01000011 00110010 10110101 11000100

1-21. a) 369 b) 36 c) Not Valid