1 ALPHABETS AND ENCODINGS

An *alphabet* is a finite set of distinct symbols.

Some, but not necessarily all, sequences of symbols are "meaningful". That is, they have been selected to represent a symbol, object or concept. The assignment of meanings to a set of sequences defined on an alphabet is called an *interpretation*.

The 26 letters of the alphabet can be used to define *sequences* that we commonly called "words". Some words are meaningful, others are just garbled sequences of letters.

The 10 digits, "0" through "9" define sequences called the non-negative integers. In this case every sequence is meaningful since each defines some integer.

By adding the character "-" to the alphabet of digits, we can construct sentences that correspond to the full set of integers, both positive and negative.

2 ENCODINGS

An *encoding* is the assignment of a unique sequence of symbols from one alphabet to represent each symbol or object in a finite set. When a symbol in the set has been assigned a unique sequence of symbols chosen from the alphabet, that sequence is called a "codeword" for the symbol in the set.

The set of symbols to be encoded can itself be an alphabet. One way a sequence of symbols from that alphabet can be encoded using another alphabet is by concatenating the codewords for each symbol of the given alphabet as expressed in the other alphabet. The resulting encoded sequence is called a "message". For example, if 8 (from the alphabet consisting of the decimal digits) is encoded as "1000" (using the binary alphabet, {0, 1}) and 7 is encoded as "0111" then 87 is expressed by the message "10000111".

"Little Endian Notation" is used to identify a particular bit position in a binary sequence. This notation labels each bit position with an integer, beginning from the right, with the right most bit position being position 0. The bit position of a labelled register or bus is indicated with a subscript, expressed as an integer in parentheses.

There are two types of encodings:

1. **Variable Length Encoding**: Assigns codewords that may be sequences of different length.
2. **Fixed Length Encoding**: The codewords for all symbols have the same length.

Variable length encodings may lead to shorter messages, but may not be uniquely decipherable; that is, attempting to interpret the message may lead to more than one possible interpretation. In such cases, the encoding is called "ambiguous".

The following two examples illustrate variable length encodings of an alphabet of form symbols: {A, B, C, D }:

SYMBOL	AMBIGUOUS ENCODING	UNIQUELY DECIPHERABLE ENCODING
A	0	0
B	1	10
C	01	110
D	11	111

Using each of the ambiguous encoding scheme, the sequence 0110 can be interpreted in several ways, such as ABBA orADA. However, there is only one possible interpretation using the uniquely decipherable encoding scheme: AC.

Fixed length encodings are always uniquely decipherable, but the messages may be longer. To "decode " a message constructed using a fixed length encoding, one only needs to know the length of each codeword. The message is then decoded by partitioning it into segments of that length to obtain the individual codewords and identifying the interpretation of each one.

The binary alphabet {0, 1} is most important in digital systems, since information is represented by binary sequences; that is, sequences of symbols from a binary alphabet. In digital design, the term "encoding" is restricted to mean "binary encoding"; that is, the process of converting a non-binary sequence into a binary one. Similarly, ''decoding" is restricted to mean the process of converting a binary sequence into a non-binary one. In either case it is important to know what coding system is used.

2.1 Common Fixed Length Encodings

A binary alphabet does not lend itself to human interpretation readily, and so encodings must be defined for "user-friendly" alphabets using the binary alphabet. Some of the ways of encoding larger alphabets using the binary alphabet are:

- *Natural binary encoding*: Each unsigned decimal integer is represented by its base-2 value. The arithmetic techniques for converting between base-2 and base-10 provide the tools for transforming messages in one alphabet to messages in the other. Techniques to encode decimal to binary are described in many references. To decode binary into decimal, the evaluation of placeholder notation, to be described next, can be employed.
- *Natural Binary Decoding*: Values can be represented symbolically as a sequence of characters, called "placeholder notation." The significance of each character is determined by its position in the sequence and by the number, m, of characters in the alphabet. The value represented by the sequence:

  $$b_{n-1}b_{n-2}\ldots b_1b_0$$

  is obtained by evaluating the expression:

  $$m^{n-1}\cdot b_{n-1} + m^{n-2}\cdot b_{n-2} + \cdots m^1\cdot b_1 + m^0\cdot b_0$$

For example the binary sequence 1100101 represents the base-2 integer 1100101_2.

$$\begin{aligned} 1100101_2 &= 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 2^6 + 2^5 + 2^2 + 2^0 \\ &= 64 + 32 + 4 + 1 \\ &= 101_{10} \end{aligned}$$

Note that this encoding is a fixed-length-k encoding to encode the first 2^k non-negative decimal integers where all codewords have the same length k. For integers that require less than k bits, leading zeros can be added without changing the value being represented.

- *Binary Coded Decimal (BCD)*: Rather than encode every integer into its natural binary equivalent, only the symbols "0" through "9" are represented by their corresponding 4-bit natural binary value equivalents. The BCD representation of an unsigned integer is obtained by concatenating the corresponding binary codes of each digit together:

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

  For example: 9132 would be encoded as 1001 0001 0011 0010. (The spaces between certain bits are not part of the encoding, but are there to improve readability of the binary sequence by making apparent the individual codewords for each digit of the number.)

- *BCD Decoding*: From the right, the bits of a binary sequence are grouped in sets of four bits. If the leftmost grouping has fewer than four bits, then 0's are added to create a 4 bit segment. Each 4 bit segment should correspond to a valid BCD codeword. To decode the message simply replace each 4-bit codeword with the symbol it represents.

  For example: 11000110101 is partitioned into 110 0011 0101. Adding a "leading 0," results in 0110 0011 0101 = 635.

  If any 4-bit segment does not correspond to a valid codeword, then the message is considered "invalid."

- *Gray Encoding*: The following encoding of the 10 digits illustrates that it is not necessary that the codewords be related to the digit as a base-2 value:

DIGIT	BCD
0	0000
1	0001
2	0011
3	0010
4	0110
5	1110
6	1010
7	1011
8	1001
9	1000

This encoding employs codewords of the same length as BCD but, except for 0 and 1, the codewords do not represent the base-2 values of the corresponding digits. The noteworthy property of Gray codewords is that two adjacent codewords differ in only one bit position.

- *Gray Decoding*: Decoding of a binary sequence constructed from Gray codewords proceeds in a manner identical to that used for decoding BCD messages, except that the Gray codeword table is referenced to determine the meaning of each codeword.