**Zip-Compression**

History:

The format was created by Phil Katz in 1989 and was first implemented in PKZIP as replacement for the ARC format. It was made an open format which can be used and implemented by anyone.

It became very popular among PC users since it could minimize the disk-space needed and also store files together in an archive. This was also very efficient when transferring files from one computer to another using floppy or network.

Usage:

For lossless compression.

To pack files together in an archive which is easier to distribute.

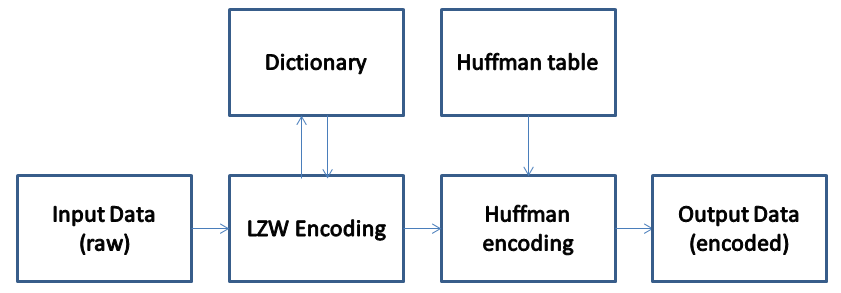
Compression algorithms:

Several different compression algorithms are used, such as Shrunk, Reduced, Imploded, DEFLATE etc. The most common used is the latter.

Prototype:

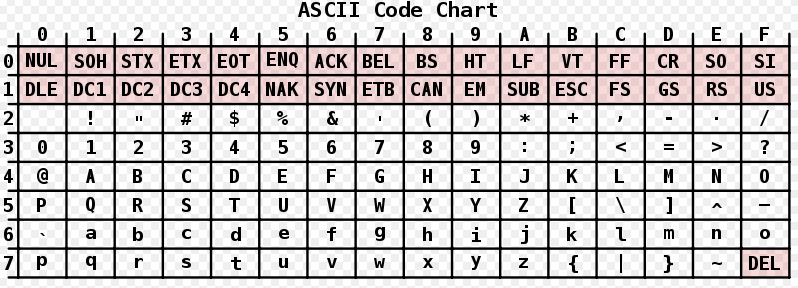
The prototype compression algorithm implemented is the LZW (Lempel-Ziv-Welch) which is a lossless data compression algorithm. It was created by Welch in 1983 as an improvement of the LZ78 algorithm. It is a fairly simple algorithm that relies on dictionaries for encoding and decoding.

The LZW became widely used when it became a part of the GIF image format I 1987 and it is still used in FreeBSD to this day.

Underneath the process can be seen:

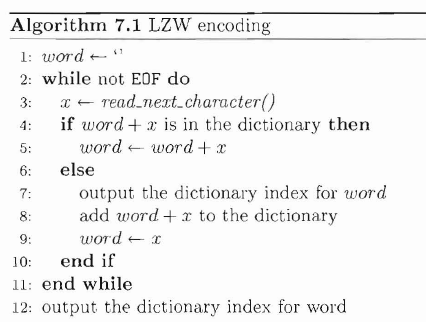
LZW algorithm:

The dictionary used for compression and decompression is created during encoding and decoding, but there is a start dictionary. In this prototype this start dictionary consists of ASCII values from 0x20 to 0x7E:



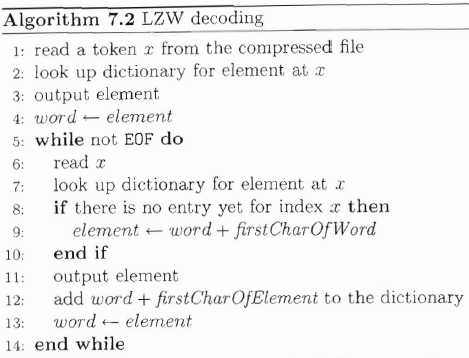
Encoding:

The pseudo code for the encoding algorithm:



Decoding:

The pseudo code for the encoding algorithm:

****

Implementation:

The implementation of the LZW prototype was made in MATLAB which makes the readout of different registers and variables easy. It consists of the following files:

create\_dictionary.m Creates the start dictionary.  
decode.m Decodes LZW encoded ‘file’.   
decode\_string.m Finds the element in dictionary at x.  
encode.m Encodes LZW from ‘file’.  
encode\_string.m Finds the dictionary position for word.  
test.m Test-file with encoding -> decoding. Outputs compression ratio.  
test.txt Txt-file containing some Shakespeare – is 3.35KB.

Huffman-coding:

Test: