Università degli Studi di Catania
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA MAGISTRALE IN INFORMATICA

FLAC
Free Lossless Audio Codec

RELAZIONE PROGETTO MULTIMEDIA

Ganci Antonio (W82/000167)

Anno Accademico 2018/2019

1. Introduzione

Come per tutti i file, anche per l'audio sono definiti diversi formati, ossia strutture e algoritmi mediante il quale le informazioni sono archiviate.

Tipicamente in un formato audio digitale sono presenti le seguenti informazioni:

- *Intestazione:* contiene tutte le informazioni sulla frequenza di campionamento, la profondità di bit e il numero di canali. Può contenere altri metadati (brano, artista, ecc…). Se non presente, il formato si dice **headerless**.
- *Algoritmo di codifica e/o compressione:* tratta i dati grezzi per rappresentarli in modo da ottenere dei benefici. Se non presente, il formato si dice **raw**.
- *Dati:* le informazioni sul suono descritto contenuto nel file audio.

Una fondamentale distinzione tra formati audio che possiamo fare è sicuramente quella riguardante la compressione. La maggior parte dei formati audio, infatti, utilizza un algoritmo di compressione per ridurre lo spazio occupato in memoria e il tempo impiegato per la trasmissione del file. La compressione di un file può essere:

- *Lossy (con perdita):* quando l'informazione nel file compresso è minore di quella contenuta nel file di origine. Non è possibile ripristinare il file originale a partire dal file compresso.
- *Lossless (senza perdita):* quando l'informazione nel file compresso è identica a quella contenuta nel file di origine. È possibile ripristinare il file originale a partire dal file compresso.

Tra i formati che utilizzano un algoritmo di compressione lossy, il più utilizzato e conosciuto è sicuramente **MP3**, ma ne esistono molti altri tra cui **Ogg Vorbis** e **AAC**. Tra i formati lossless, spicca invece il formato **FLAC**.

In questo progetto si è analizzata la struttura del formato FLAC e si è implementata una sua versione semplificata.

2. FLAC (Free Lossless Audio Codec)

FLAC è un codec audio libero con compressione dati lossless, cioè senza perdita di qualità. Risulta quindi adatto sia all'ascolto con lettori di musica digitale, sia all'archiviazione su memorie di massa. Il formato FLAC è attualmente supportato da una buona parte dei software audio.

FLAC è specificamente progettato per comprimere dati audio, diversamente dalla maggior parte degli algoritmi di compressione lossless generici (ZIP), questo gli permette di raggiungere compressioni importanti, dell'ordine del 30-50% contro il 10-20% raggiunto da quelli tradizionali quando utilizzati per comprimere file audio.

FLAC supporta solo campioni in virgola fissa. Può gestire dati in PCM con profondità di bit da 4 a 24 bit, qualsiasi frequenza di campionamento da 1 Hz a 65535 Hz (con incrementi da 1 Hz) o da 10 Hz a 655350 Hz (con incrementi di 10 Hz) e un numero di canali da 1 a 8.

La codifica prevede diversi passaggi:

- *Suddivisione in blocchi:* l'input viene diviso in più parti contigue, anche variabili in grandezza (blocchi da 16 a 65535 samples).
- *Compattamento del flusso multicanale:* in questo step l'encoder FLAC si occupa di calcolare, nel caso di input stereo e surround, la media dei canali e la loro differenza. Il segnale a qualità migliore viene passato al processo successivo.
- *Previsione:* partendo dal primo blocco, avviene la previsione di quale possa essere il successivo con degli algoritmi matematici che tentano di ricostruire il segnale. FLAC usa quattro metodi per modellare il segnale di input: **Verbatim**, **Constant**, **Fixed Linear Prediction**, e **FIR Linear Prediction**. Verbatim è essenzialmente un predittore di ordine zero. Ciò significa che il segnale previsto per blocchi di questo tipo è zero e quindi il segnale residuo sarà il segnale originale. Ovviamente per questi blocchi la compressione è nulla. Il Verbatim stabilisce una baseline per gli altri predittori. Il predittore Costant viene utilizzato quando siamo di fronte ad un blocco "puro" (silenzio digitale), vale a dire un valore costante in tutto il blocco. Fixed Linear Prediction è una classe di predittori lineari fissi (audiopak e shorten) dello zero-terzo ordine a cui FLAC aggiunge un predittore del quarto ordine. Per una maggiore compressione (a costo di una codifica più lenta) FLAC supporta anche predittori lineari FIR fino al 32° ordine.

- *Codifica residua:* la codifica residua permette a FLAC di essere effettivamente un codec lossless. Si codifica senza perdita tutta la parte di segnale che si differenzia dalla ricostruzione matematica di predizione, e viene incorporata nel file finale. Il guadagno si ha sul fatto che il segnale residuo, proprio per la scrematura matematica, sarà minore in bytes rispetto al corrispettivo PCM. Attualmente FLAC definisce due metodi simili per la codifica del segnale di errore dalla fase di predizione. Il segnale di errore è codificato utilizzando Rice Coding (sottoinsieme semplificato di Golomb Code) in due modi: 1) l'encoder stima un singolo parametro Rice basato sulla varianza del residuo e si codifica l'intero residuo utilizzando questo parametro; 2) il residuo è suddiviso in diverse regioni (di lunghezza uguale) di campioni contigui e ogni regione è codificata con il proprio parametro Rice basato sulla media della regione.

Il flusso dati di un file FLAC ha la seguente struttura:

STREAM	
<32>	Marcatore "fLaC".
METADATA_BLOCK	Blocco obbligatorio di tipo STREAMINFO contenente le proprietà di base del flusso.
*METADATA_BLOCK**	Blocchi di metadati aggiuntivi.
FRAME+	Frames audio.

I primi 32 bits del flusso sono destinati ad un marcatore che contraddistingue tutti i file FLAC. Seguono un primo blocco di metadati obbligatorio in cui sono specificate le proprietà base del flusso quali frame rate, numero di canali, bit per campione, ecc., e una serie di blocchi di metadati aggiuntivi facoltativi. Per finire vengono memorizzati i frames audio. Ogni blocco di metadati ha la seguente struttura:

METADATA_BLOCK	
METADATA_BLOCK_HEADER	Header che specifica tipo e dimensione dei dati del blocco.
METADATA_BLOCK_DATA	

Nell'header è specificata la tipologia del blocco e la lunghezza in byte di quest'ultimo. È inoltre previsto il flag "last-metadata-block" che indica se il blocco in questione è l'ultimo blocco di metadati prima dei blocchi audio.

I dati della sezione ‘DATA’ devono corrispondere alla tipologia del blocco di metadati. Di seguito viene mostrata, oltre alla struttura dell’header, la struttura di un blocco dati di tipo STREAMINFO.

METADATA_BLOCK_HEADER	
<1>	Flag “last-metadata-block”: 1 se questo è l’ultimo blocco di metadati prima dei blocchi audio, 0 altrimenti.
<7>	BLOCK_TYPE: • 0: STREAMINFO • 1: PADDING • 2: APPLICATION • 3: SEEKTABLE • 4: VORBIS_COMMENT • 5: CUESHEET • 6: PICTURE • 7-126: Riservati • 127: Non valido, per evitare di far confusione con un codice di sincronizzazione frame.
<24>	Lunghezza (in byte) dei metadati (non include la dimensione del METADATA_BLOCK_HEADER).

METADATA_BLOCK_DATA	
METADATA_BLOCK_STREAMINFO *METADATA_BLOCK_PADDING* *METADATA_BLOCK_APPLICATION* *METADATA_BLOCK_SEEKTABLE* *METADATA_BLOCK_VORBIS_COMMENT* *METADATA_BLOCK_CUESHEET* *METADATA_BLOCK_PICTURE*	I dati del blocco devono corrispondere al tipo di blocco nell’intestazione di quest’ultimo.

Tutti le altre tipologie di blocco non sono qui descritte ma possono essere visualizzate sul sito di riferimento.

METADATA_BLOCK_STREAMINFO	
<16>	Dimensione minima del blocco (in samples) utilizzata nello stream.
<16>	Dimensione massima del blocco (in samples) utilizzata nello stream.
<24>	Dimensione minima del frame (in bytes) utilizzata nello stream.

<24>	Dimensione massima del frame (in bytes) utilizzata nello stream.
<20>	Sample rate in Hz. Sebbene siano disponibili 20 bit, la frequenza di campionamento massimo è 655350 Hz. Il valore 0 non è valido.
<3>	Numero di canali – 1. FLAC supporta da 1 a 8 canali.
<5>	Bit per sample – 1. FLAC supporta da 4 a 32 bits per sample.
<36>	Samples totali nel flusso. Per sample si intende un campione inter-canale, ovvero, un secondo di audio a 44.1 kHz avrà 44100 campioni indipendentemente dal numero di canali.
<128>	Firma MD5 dei dati audio non codificati. Ciò consente al decodificatore di determinare se esiste un errore nei dati audio.

Ogni frame audio ha la seguente struttura:

FRAME	
FRAME_HEADER	
SUBFRAME+	Un SUBFRAME per canale.
<?>	Zero-padding per l'allineamento dei byte.
FRAME_FOOTER	

L'header, come di norma, contiene tutte le informazioni sul frame (frame rate, bit per sample, ecc.). Gli ultimi 8 bits sono destinati ad un controllo CRC8 su tutto ciò che viene prima.

FRAME_HEADER	
<14>	Sync code '11111111111110'.
<1>	Riservato: • 0: Valore obbligatorio • 1: Riservato per un uso futuro.
<4>	Dimensione del blocco per inter-channel samples: • 0000: Riservato • 0001: 192 samples • 0010-0101: 576* (2^(n-2)) samples • 0110: 8 bit (blocksize-1) dalla fine dell'intestazione • 0111: 16 bit (blocksize-1) dalla fine dell'intestazione • 1000-1111: 256*(2^(n-8)) samples.
<4>	Sample rate: • 0000: prendilo dal blocco STREAMINFO • 0001: 88.2 kHz

	• 0010: 176.4 kHz • 0011: 192 kHz • 0100: 8 kHz • 0101: 16 kHz • 0110: 22.05 kHz • 0111: 24 kHz • 1000: 32 kHz • 1001: 44.1 kHz • 1010: 48 kHz • 1011: 96 kHz • 1100: sample rate di 8 bit (in kHz) dalla fine dell'header • 1101: sample rate di 16 bit (in kHz) dalla fine dell'header • 1110: sample rate di 16 bit (in decine di Hz) dalla fine dell'header • 1111: non valido, per evitare di far confusione (vedi BLOCK_TYPE in METADATA_BLOCK_HEADER).
<4>	Assegnazione dei canali: • 0000-0111: numero di canali indipendenti – 1 • 1000: left/side stereo: canale 0 è il sinistro, canale 1 è la differenza • 1001: right/side stereo: canale 0 è la differenza, canale 1 è il destro • 1010: mid/side stereo: canale 0 è la media, canale 1 è la differenza • 1011-1111: riservati.
<3>	Sample size in bits: • 000: prendilo dal blocco STREAMINFO • 001: 8 bits per sample • 010: 12 bits per sample • 011: riservato • 100: 16 bits per sample • 101: 20 bits per sample • 110: 24 bits per sample • 111 riservato.
<1>	Riservato: • 0: valore obbligatorio • 1: riservato per un uso futuro
<?>	Se la dimensione dei blocchi è variabile: • <8-56>: "UTF-8" a 36 bits Altrimenti: • <8-48>: "UTF-8" a 31 bits.
<?>	Se la dimensione dei blocchi in bits è 011x: • 8/16 bit (blocksize-1)
<?>	Se la dimensione dei blocchi in bits è 11xx: • 8/16 bit sample rate

<8>	CRC-8 di tutto ciò che viene prima.

Il campo subframe è un campo cruciale per il formato FLAC. Per subframe si intende un insieme di campioni codificati da un determinato canale. Tutte le subframe all'interno di un frame conterranno lo stesso numero di campioni.

SUBFRAME	
SUBFRAME_HEADER	
SUBFRAME_CONSTANT *SUBFRAME_FIXED* *SUBFRAME_LPC* *SUBFRAME_VERBATIM*	Il SUBFRAME_HEADER specifica quale.

Il SUBFRAME_HEADER contiene informazioni riguardanti principalmente la tipologia di subframe.

SUBFRAME_HEADER	
<1>	Zero bit padding, per prevenire problemi di sync
<6>	Tipologia di subframe: • 000000: *SUBFRAME_CONSTANT* • 000001: *SUBFRAME_VERBATIM* • 00001x: riservato • 0001xx: riservato • 001xxx: se (xxx <= 4) *SUBFRAME_FIXED;* altrimenti riservato • 01xxxx: riservato • 1xxxxx: SUBFRAME_LPC.
<1+k>	Flag "wasted bits-per-sample": • 0: nessun "wasted-bit-per-sample" nel subblock di origine • 1: k "wasted-bit-per-sample" nel subblock di origine.

La struttura delle quattro tipologie di subframe è definita come segue:

SUBFRAME_CONSTANT	
<n>	Valore costante del subblock, n = bits-per-sample del frame.

SUBFRAME_VERBATIM	
<n*i>	Subblock non codificato, n = bits-per-sample del frame, i = dimensione del blocco del frame.

SUBFRAME_FIXED	
<n>	Samples di “warm-up” non codificati, n = ordine del predittore * bits-per-sample del frame.
RESIDUAL	Codifica residua.

SUBFRAME_LPC	
<n>	Samples di “warm-up” non codificati, n = ordine del predittore lpc * bits-per-sample del frame.
<4>	Precisione in bit dei coefficienti del predittore lpc – 1, 1111 non valido.
<5>	Shift necessario dei coefficienti del predittore lpc, complemento a due.
<n>	Coefficienti del predittore non codificati, n = coeff. Precisione * lpc order, complemento a due.
RESIDUAL	Codifica residua.

Sia SUBFRAME_FIXED che SUBFRAME_LPC mantengono per ultimo il campo RESIDUAL. Il RESIDUAL corrisponde al segnale residuo ovvero alla differenza tra il segnale predetto e quello originale codificato attraverso Rice Coding. Di seguito sono mostrati i due diversi metodi per la codifica residua.

RESIDUAL	
<2>	Metodi per la codifica residua: • 00: Partitioned Rice Coding con parametro a 4 bit; • 01: Partitioned Rice Coding con parametro a 5 bit; • 10-11: riservati
R_C_M_P_RICE *R_C_M_P_RICE2*	

RESIDUAL_CODING_METHOD_PARTITIONED_RICE	
<4>	Ordine della partizione.
RICE_PARTITION+	Ci saranno due partizioni del 2° ordine.

RICE_PARTITION	
<4(+5)>	Parametri codificati: • 0000-1110: Parametro Rice

	• 1111: Escape code, ovvero la partizione è in formato binario non codificato utilizzando n bits-per-sample.
<?>	Codifica residua. Il numero dei samples (n) nella partizione è determinata come segue: • Se l'ordine della partizione è zero, n = dimensione del blocco del frame – ordine del predittore • Altrimenti, se questa non è la prima partizione del subframe, n = (dimensione del blocco del frame / (2^ordine della partizione)) • Altrimenti n = (dimensione del blocco del frame / (2^ordine della partizione)) – ordine del predittore.

RESIDUAL_CODING_METHOD_PARTITIONED_RICE2	
<4>	Ordine della partizione.
RICE2_PARTITION+	Ci saranno due partizioni del 2° ordine.

RICE2_PARTITION	
<5(+5)>	Parametri codificati: • 00000-11110: Parametro Rice • 11111: Escape code, ovvero la partizione è in formato binario non codificato utilizzando n bits-per-sample.
<?>	Codifica residua. Il numero dei samples (n) nella partizione è determinata come segue: • Se l'ordine della partizione è zero, n = dimensione del blocco del frame – ordine del predittore • Altrimenti, se questa non è la prima partizione del subframe, n = (dimensione del blocco del frame / (2^ordine della partizione)) • Altrimenti n = (dimensione del blocco del frame / (2^ordine della partizione)) – ordine del predittore.

L'ultimo elemento del frame è il FOOTER che contiene solamente il controllo CRC-16 su tutto ciò che viene prima.

FRAME_FOOTER	
<16>	CRC-16 di tutto ciò che viene prima.

3. Linear Predicted Coding

I risultati migliori, FLAC, li ottiene utilizzando un predittore LPC (Linear Predictive Coding). Si parla sempre di un predittore lineare di ordine massimo pari a 32 che a costo di una codifica più lenta, garantisce una compressione maggiore del 5-10% rispetto al fixed linear predictor. L'idea alla base di LPC è quella di provare differenti sets di coefficienti c_j e selezionare quello che minimizza la differenza

$$d_i = \left[a_i - \sum_{j=1}^{n} c_j a_{i-j} \right]^2 \quad (1)$$

In statistica, data una variabile casuale V che assume i valori v_i, si denota la probabilità che V abbia uno specifico valore v come P (V = v). Inoltre, si definisce valore atteso di V, E(V), come la somma di tutti i valori v_i di V, ciascuno moltiplicato per la sua probabilità.

$$E[V] = \sum_i v_i P(V = v_i)$$

L'autocorrelazione $R_V(d)$ di una variabile casuale V è la correlazione di V con una copia di sé stesso, shiftato di d posizioni. Autocorrelazione e valore atteso sono correlati dalla seguente relazione:

$$R_V(d) = E[V_i V_{i+k}]$$

Per trovare il set di coefficienti $\{c_j\}$ che minimizzino l'equazione (1), bisogna differenziare il valore atteso di d_i rispetto al coefficiente c_p, porre la derivata uguale a zero e fare tutto ciò per tutti i possibili valori di p, da 1 a n. Il risultato è

$$-2 \left[\left(a_i - \sum_{j=1}^{n} c_j \, a_{i-j} \right) a_{i-p} \right] = 0, \quad for \; 1 \leq p \leq n$$

Oppure

$$\sum_{j=1}^{n} c_j E\left[a_{i-j} a_{i-p}\right] = E\left[a_i a_{i-p}\right], \quad for\ 1 \le p \le n$$

Sostituendo i valori attesi con i coefficienti di autocorrelazione si ottiene un sistema di *n* equazioni lineari

$$\begin{bmatrix} R(0) & R(1) & R(2) & \dots & R(n-1) \\ R(1) & R(0) & R(1) & \cdots & R(n-2) \\ R(2) & R(1) & R(0) & \dots & R(n-3) \\ \vdots & \vdots & \vdots & & \vdots \\ R(n-1) & R(n-2) & R(n-3) & \dots & R(0) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ R(3) \\ \vdots \\ R(n) \end{bmatrix}$$

con gli n coefficienti *c* incognite. Tutto questo può essere scritto in maniera compatta come **R**C = P e può essere facilmente risolto invertendo la matrice R. Il punto è che l'inversione di una matrice richiede in generale $O(n^3)$ operazioni, ma non in questo caso. La matrice R è infatti una matrice "speciale". Come si può notare, ogni diagonale di R è formata da elementi uguali, inoltre R è simmetrica. Questo tipo di matrici sono chiamate matrici Toeplitz (da Otto Toeplitz) e possono essere invertite da alcuni efficienti algoritmi.

FLAC utiliza un metodo ricorsivo ed efficiente, noto come algoritmo Levinson-Durbin, per risolovere questo sistema di n equazioni. L'algoritmo è stato proposto da Norman Levinson nel 1947 e migliorato da James Durbin nel 1960. Nella sua forma attuale richiede solamente $3n^2$ moltiplicazioni.

4. Software

Il software consiste in un'implementazione semplificata in python dell'algoritmo di codifica e decodifica di un file FLAC.

L'algoritmo di codifica converte un file wav in formato FLAC. Il file in input, per semplicità, è soggetto a 'restrizioni' riguardanti frame rate (44100 Hz), profondità di bit (16 bit) e numero di canali (stereo). Dopo aver estratto tutte le informazioni essenziali dal file wav, l'algoritmo inizia a costruire il nuovo flusso audio. Per prima cosa costruisce l'header, in seguito comincia a suddividere tutti i campioni in frame di lunghezza fissa (4096 samples per frame). Ogni frame, come si è visto prima, può contenere quattro subframes differenti: Verbatim, Constant, Fixed e LPC.

In questa versione non è stato implementato il subframe LPC, per cui l'algoritmo costruisce solamente subframes Constant, Verbatim e Fixed per poi scegliere quello più piccolo.

Un subframe Constant è formato semplicemente dal primo valore del frame e dalla dimensione di quest'ultimo.

Un subframe Verbatim è costituito semplicemente dai valori originali del frame.

I subframes Fixed sono costruiti sulla base di una classe di predittori lineari dallo zero al quarto grado. Il fixed linear predictor è un semplice predittore che adatta un polinomio ai campioni audio. Questo tipo di predittore è molto più veloce del predittore LPC ma in genere da risultati peggiori. Ogni subframes Fixed è composto da due parti: i samples di warm-up e il residual code. I samples di warm-up sono tutti i samples non predetti, per esempio, se si usa un predittore di ordine 2, i samples di warm-up sono i primi 2 sample del frame. Il residual code è invece la codifica in Rice Code della differenza tra i valori predetti e quelli reali. La codifica Rice è una forma specifica della codifica Golomb ed è usata per codificare stringhe di numeri con una lunghezza di bit variabile per ogni numero. Se la maggior parte di numeri è piccola (è quello che ci si aspetta poiché stiamo codificando una differenza), è possibile ottenere una compressione abbastanza buona.

Per ogni frame, l'algoritmo costruisce ogni tipo di subframe e ne valuta la dimensione finale. Tra tutti i subframes viene selezionato solamente il più piccolo e viene aggiunto al flusso audio.

La decodifica è ovviamente il processo inverso. L'algoritmo prende in input un file FLAC, decodifica tutti i tipi di subframes presenti e restituisce il file wav originale.

5. Conclusioni

Per valutare la qualità del software si sono effettuati due tipi di test, il primo valuta la correttezza della codifica mentre il secondo la qualità della compressione. I test sono stati effettuati su 14 tracce stereo a 16 bit con frame rate pari a 44100 Hz, tutte in formato wav. È possibile trovare tracce, campioni e script per il test nella cartella "test" del progetto. Per determinare la correttezza del processo di codifica, al termine di quest'ultimo, si deve ottenere un file FLAC che sia leggibile da ogni player audio che supporti il formato e che sia possibile ripristinare, ovviamente senza perdita alcuna. Entrambe le tracce di test, date in input all'algoritmo sono state convertite correttamente e sono leggibili da qualsiasi player. Le stesse tracce sono state poi ripristinate correttamente tramite l'algoritmo di decodifica. A questo punto è stato effettuata una verifica aggiuntiva che consiste nella differenza punto a punto tra i campioni della traccia originale e quelli della traccia convertita. Per fare ciò, si è implementato un semplice script in python che, presi in input i samples della traccia originale e della traccia convertita (estratti tramite Audacity), restituisce la differenza punto a punto. Il risultato è pari a zero, ciò indica che i samples audio delle due tracce sono uguali e che quindi la codifica non ha prodotto alcuna perdita di dati. La qualità della compressione è stata valutata mettendo a confronto le tracce convertite con l'algoritmo di codifica e le tracce convertite con Audacity. I risultati sono mostrati nella seguente tabella che mostra le varie dimensioni in MB delle tracce di test prima e dopo la compressione con l'algoritmo di codifica e con audacity.

Numero traccia	Peso originale	Dopo compressione con algoritmo	Dopo compressione con Audacity	Compression Ratio algoritmo	Compression Ratio Audacity
Track1	0,37	0,15	0,12	2,47	3,64
Track2	9,92	6,5	6,01	1,53	1,65
Track3	3,64	1,62	1,08	2,25	3,37
Track4	2,36	1,12	0,69	2,11	3,41
Track5	3,16	1,38	0,90	2,29	3,51
Track6	2,32	0,84	0,61	2,77	3,80
Track7	2,36	0,73	0,68	3,24	3,49
Track8	2,04	0,68	0,54	3,01	3,78
Track9	2,80	0,89	0,72	3,15	3,87
Track10	2,48	0,79	0,64	3,12	3,88
Track11	2,48	0,70	0,63	3,57	3,93
Track12	4,64	1,82	1,27	2,55	3,65
Track13	3,44	1,27	0,93	2,71	3,71
Track14	3,44	1,22	0,91	2,82	3,78
Compression Ratio Medio con algoritmo			2,68		
Compression Ratio Medio con Audacity			3,53		

Tabella 1: Confronto tra i Comression Ratio dell'algoritmo di codifica e Audacity

Come si può notare dalla tabella, il compression ratio medio ottenuto con l'algoritmo di codifica è pari a 2.68 mentre, come ci si poteva aspettare, il compression ratio medio ottenuto con la compressione di Audacity è leggermente maggiore (3.53).

In conclusione, per quanto riguarda il software implementato, pur essendo una versione base, riesce comunque ad effettuare codifica e decodifica di un file wav in modo corretto ed in modo efficiente. In futuro si potrebbe, per prima cosa, implementare il subframe LPC per una compressione maggiore, in seguito si potrebbe estendere la codifica ad altri formati (non solo wav).

Per quanto riguarda il formato FLAC, si può sicuramente affermare che attualmente è il formato che presenta il miglior compromesso tra qualità e compressione. Inoltre, al momento, è l'unico non proprietario e con un'implementazione open source.

References

[1] Data Compression: The complete reference. Fourth edition. David Salomon

[2] SHORTEN: Simple lossless and near-lossless waveform compression. Tony Robinson. Cambridge University Engineering Department, December 1994

[3] https://xiph.org/flac/

[4] https://it.wikipedia.org/wiki/Free_Lossless_Audio_Codec

[5] https://www.nayuki.io

[6] https://github.com/serban/flac-encoder