Analisis dan Implementasi Kompresi File *Audio* Dengan Menggunakan Algoritma *Run Length Encoding* (RLE)

Aditya Rahandi[1], Dian rachmawati[2], Sajadin Sembiring [3]

Program Studi S1 Ilmu Komputer, FASILKOM-TI USU

Jalan Universitas No. 9 Kampus USU Medan 20155

[1]arahandi@gmail.com

[2]dee230783@gmail.com

[3]Sajadinbiring@gmail.com

***Abstrak*— Pengiriman *file* audio seringkali terhambat karena besarnya ukuran file yang akan dikirim. Salah satu cara untuk mengatasi hal tersebut adalah dengan memampatkan (kompresi) file tersebut sebelum dikirim. Penulis menggunakan algoritma *Run Length Encoding* yang bersifat *lossless* dalam pemampatan file audio. Masukan dalam sistem ini adalah *file audio* WAV dan MP3. Pada sistem ini terdapat tahap kompresi dan dekompresi. Tahap kompresi bertujuan untuk memampatkan ukuran *file audio,* sedangkan tahap dekompresi bertujuan untuk mengembalikan ukuran *file* audio ke ukuran semula.**

***Kata Kunci*— Audio, Kompresi, Dekompresi, Run Length Encoding, Lossless, WAV, MP3**

.

I. PENDAHULUAN

A. Latar Belakang

Pertukaran informasi dengan waktu yang singkat sangat dibutuhkan mengingat semakin cepat dan pesatnya perkembangan teknologi. Kecepatan pengiriman informasi dalam bentuk perpaduan teks, suara dan gambar menjadi bagian utama dalam pertukaran informasi. Kecepatan pengiriman ini sangatlah bergantung kepada ukuran dari informasi tersebut. Salah satu penyelesaian untuk masalah di atas adalah dengan melakukan pemampatan (kompresi) data baik itu teks, suara, dan citra sebelum dikirimkan dan kemudian penerima akan merekonstruksinya kembali menjadi data aslinya (dekompresi).

File Audio dengan format WAV datanya tidak terkompres sehingga seluruh sampel *audio* disimpan semuanya di *harddisk*. File *audio* ini jarang sekali digunakan di internet karena ukurannya yang relatif besar dengan batasan maksimal untuk file WAV adalah 2GB. MPEG (*Moving Picture Expert Group*)-1 *audio* layer III atau yang lebih dikenal dengan MP3, adalah salah satu dari pengkodean dalam *digital audio* dan juga merupakan format kompresi *audio*. Ukuran file MP3 lebih kecil dibandingkan dengan file WAV karena beberapa data yang tidak dapat didengar oleh manusia telah dihilangkan.

Dalam penelitian ini penulis menggunakan metode *Run Length Encoding* (RLE) untuk mengkompresi file *audio*. Prinsip kerja metode ini adalah mengelompokkan data yang sama dari *sample audio* dan menghitung frekuensi kemunculannya. Algoritma ini hanya efisien untuk data yang berisi kelompok data yang memiliki perulangan.

Melihat hal tersebut, maka penulis tertarik untuk mengangkat penelitian ini dengan judul Analisis dan Implementasi Kompresi File Audio dengan Menggunakan Algoritma *Run Length Encoding* (RLE).

B. Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah:

1) Bagaimana mengembangkan aplikasi kompresi data *audio* (WAV dan MP3) dengan menggunakan metode *Run Lenght Encoding* (RLE).
2) Bagaimana perbandingan rasio kompresi file WAV dan MP3.

C. Batasan Masalah

Adapun yang menjadi batasan masalah dalam penelitian ini adalah:

1) File *audio* yang digunakan berformat *.WAV berukuran maksimum 5 Mb dan file *. MP3 berukuran maksimum 10 Mb.
2) Bahasa pemrograman yang digunakan adalah Microsoft Visual Basic 6.0
3) Data yamg diuji masin-masing file sebanyak 3 buah.
4) Tools pembuatan Hasil Pengujian menggunakan *Crystal Report* .

D. Tujuan Penelitian

Tujuan penelitian ini adalah :

1) Mengetahui rasio kompresi file audio dengan menggunakan algoritma *Run Length Encoding* (RLE).
2) Membangun aplikasi yang dapat mengkompresi file audio.

E. Manfaat Penelitian

Manfaat yang dapat diperoleh dari penelitian ini adalah:

1) Diperoleh sebuah aplikasi yang dapat mengompresi file audio.
2) Sebagai pembanding untuk metode kompresi lainnya.

F. Metodologi Penelitian

Penelitian ini dilakukan dengan metodologi sebagai berikut:

1) Studi Literatur
   Pengerjaan skripsi ini dimulai dengan mengumpulkan informasi tentang kompresi/dekompres, algoritma kompresi *Run Length Encoding*, file *audio* WAV dan MP3 yang diperlukan menggunakan metode *Library Research.* Penulis mengumpulkan informasi sebagai referensi baik dari buku, paper, jurnal, makalah, forum, milis, dan sumber-sumber lain yang berkaitan dan beberapa referensi lainnya untuk menunjang pencapaian penelitian

2) Analisa dan Perancangan
   Pada tahap ini, dilakukan analisis permasalahan yang ada, batasan yang dimiliki, *flowchart* sistem kompresi serta kebutuhan lain yang diperlukan untuk arsitektur perangkat lunak serta merancang *user interface* dan struktur program kompresi..

3) Implementasi
   Implementasi program dilakukan dengan pengkodean menggunakan bahasa pemrograman Microsoft Visual Basic 6.0.

4) Pengujian
   Pada tahap ini dilakukan pengujian terhadap aplikasi yang dibangun sesuai dengan tujuan yang ingin dicapai, yaitu melakukan kompresi kedua jenis file *audio* di atas untuk mengetahui rasio kompresi.

5) Penyusunan laporan dan Kesimpulan akhir
   Metode ini akan dilaksanakan dengan melakukan pendokumentasian hasil analisa dan pengujian secara tertulis dalam bentuk laporan skripsi.

II. TINJAUAN PUSTAKA

A. Kompresi

Pada dasarnya semua data itu merupakan rangkaian *bit* 0 dan 1. Yang membedakan antara suatu data tertentu dengan data yang lain adalah ukuran dari rangkaian bit dan bagaimana 0 dan 1 itu ditempatkan dalam rangkaian bit tersebut. Misalnya data berupa *audio*, dalam data *audio* suatu rangkaian bit tertentu mewakili satu nada. Semakin kompleks suatu data, ukuran rangkaian bit yang diperlukan semakin panjang, dengan demikian ukuran keseluruhan data juga semakin besar [2].

Dalam penyimpanan dan pengiriman data (transmisi), selain isi dari data tersebut parameter yang tidak kalah pentingnya adalah ukurannya. Kerap kali data yang disimpan dalam suatu media penyimpanan berukuran sangat besar sehingga memerlukan tempat yang lebih banyak dan tidak efisien. Apalagi bila data tersebut akan dikirim, semakin besar ukurannya, waktu yang diperlukan untuk pengiriman akan lebih lama. Untuk itu, diperlukan kompresi data (*data compression*) untuk memperkecil ukuran suatu data tanpa merubah isi atau informasi yang terkandung dalam data tersebut.

B. Kompresi Data

Data tidak hanya disajikan dalam bentuk *audio* (bunyi, suara, musik) maupun *video*, tetapi juga dapat berupa teks dan citra. Keempat macam data tersebut sering disebut dengan multimedia. Pada umumnya representasi data digital membutuhkan memori yang besar, disisi lain kebanyakan data misalnya citra (*image*) mengandung duplikasi. Duplikasi ini dapat berarti dua hal yaitu pertama, besar kemungkinan suatu *pixel* dengan *pixel* lain tetangganya memiliki intensitas yang sama, sehingga penyimpanan setiap *pixel* memboroskan tempat. Kedua, citra banyak mengandung bagian (*region*) yang sama, sehingga bagian yang sama ini tidak perlu dikodekan berulang kali. Saat ini, kebanyakan aplikasi menginginkan representasi dengan memori yang lebih sedikit. Pemampatan data atau kompresi data (*data compression*) bertujuan meminimalkan kebutuhan memori untuk merepresentasikan data digital. Prinsip umum yang digunakan pada proses kompresi adalah mengurangi duplikasi data sehingga memori untuk merepresentasikan menjadi lebih sedikit daripada representasi data digital semula [9].

Data digital yang telah dikompresi dapat dikembalikan ke bentuk data digital semula (dekompresi) dimana hal ini tergantung pada aplikasi *software* yang mendukung kompresi tersebut. Ketika suatu aplikasi mampu ‘menghilangkan’ atau mengkompresi data yang tidak dibutuhkan maka aplikasi tersebut juga mampu mengembalikan data yang dihilangkan tersebut sehingga menjadi data digital semula (dekompresi) namun terdapat juga suatu aplikasi yang dapat mengkompresi namun ketika dekompresi dapat menggunakan aplikasi lain contohnya aplikasi winzip dengan aplikasi winrar.

C. Metode Kompresi Data

Metode pemampatan data atau kompresi data dapat dikelompokan dalam dua kelompok besar yaitu metode lossless dan metode lossy [11]. Yaitu:

1) *Lossless*

Lossless data kompresi adalah kelas dari algoritma data kompresi yang memungkinkan data yang asli dapat disusun kembali dari data kompresi. *Lossless* data kompresi digunakan dalam berbagai aplikasi seperti format ZIP dan GZIP. *Lossless* juga sering digunakan sebagai komponen dalam teknologi kompresi data lossy. Kompresi *Lossless* digunakan ketika sesuatu yang penting pada kondisi asli.

2) *Lossy*

Lossy kompresi adalah suatu metode untuk mengkompresi data dan men-dekompresinya, data yang diperoleh mungkin berbeda dari yang aslinya tetapi cukup dekat perbedaaanya. *Lossy* kompresi ini paling sering digunakan untuk kompres data multimedia (suara atau gambar diam). Sebaliknya, kompresi *lossless* diperlukan untuk data teks dan file, seperti catatan bank, artikel teks dan lainnya.Format kompresi *lossy* mengalami *generation loss* yaitu jika melakukan berulang kali kompresi dan dekompresi file akan menyebabkan kehilangan kualitas secara progresif. hal ini berbeda dengan kompresi data lossless. ketika pengguna yang menerima file terkompresi secara *lossy* (misalnya untuk mengurangi waktu *download*) file yang diambil dapat sedikit berbeda dari yang asli di-*level bit* ketika tidak dapat dibedakan oleh mata dan telinga manusia untuk tujuan paling praktis. Dalam beberapa sistem kedua teknik digabungkan, dengan mengubah *codec* yang digunakan untuk mengkompresi kesalahan sinyal yang dihasilkan dari tahapan prediksi. Keuntungan dari metode *lossy* atas *lossless* adalah dalam beberapa kasus metode *lossy* dapat menghasilkan file kompresi yang lebih kecil dibandingkan dengan metode *lossless* yang ada, ketika masih memenuhi persyaratan aplikasi. Metode *lossy* sering digunakan untuk mengkompresi suara, gambar dan *video*. karena data tersebut dimaksudkan kepada *human interpretation* dimana pikiran dapat dengan mudah "mengisi bagian-bagian yang kosong" atau melihat kesalahan masa lalu sangat kecil atau inkonsistensi. Idealnya *lossy* adalah kompresi transparan, yang dapat diverifikasi dengan tes ABX. Sedangkan *lossless* digunakan untuk mengkompresi data untuk diterima ditujuan dalam kondisi asli seperti dokumen teks. *Lossy* akan mengalami *generation loss* pada data sedangkan pada lossless tidak terjadi karena data yang hasil dekompresi sama dengan data asli [1].

D. Rasio Kompresi

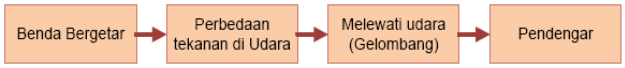
Rasio kompresi data adalah ukuran persentase data yang telah berhasil dimampaatkan. Secara matematis rasio pemampatan data ditulis sebagai berikut:

$$R = 100\% - ((K_o - K_1)/K_o) \times 100\ \%$$

Dimana, R = rasio kompresi
K_o = Ukuran file asli
K_1 = Ukuran file terkompresi

E. File Audio

Audio (suara) adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa sinyal analog dengan amplitudo yang berubah secara kontinyu terhadap waktu yang disebut frekuensi [1].



Gambar 1. Alur Gelombang Suara

Selama bergetar, perbedaan tekanan terjadi di udara sekitarnya. Pola osilasi yang terjadi dinamakan sebagai gelombang. Gelombang mempunyai pola sama yang berulang pada interval tertentu**,** yang disebut sebagai periode. Contoh suara periodik adalah instrumen musik, nyanyian burung sedangkan contoh suara non periodik adalah batuk, percikan ombak dan lain-lain.

F. Representasi Suara

Gelombang suara analog tidak dapat langsung direpresentasikan pada komputer. Komputer mengukur amplitudo pada satuan waktu tertentu untuk menghasilkan sejumlah angka. Tiap satuan pengukuran ini dinamakan "*sample*". *Analog To Digital Conversion* (ADC) adalah proses mengubah amplitudo gelombang bunyi ke dalam waktu interval tertentu (*sampling*), sehingga menghasilkan representasi digital dari suara. Dalam teknik *sampling* dikenal istilah *sampling rate* yaitu beberapa gelombang yang diambil dalam satu detik. Sebagai contoh jika kualitas CD Audio dikatakan memiliki frekuensi sebesar 44100 Hz, berarti jumlah *sampel* sebesar 44100 per detik [7].

Teknik *sampling* yang umum pada file *audio* seperti *Nyquist Sampling Rate* dimana untuk memperoleh representasi akurat dari suatu sinyal analog secara *lossless*, amplitudonya harus diambil *sample*-nya setidaknya pada kecepatan (*rate*) sama atau lebih besar dari 2 kali lipat komponen frekuensi maksimum yang akan didengar.
Misalnya untuk sinyal analog dengan *bandwith* 15Hz – 10 kHz→*sampling rate* = 2 x 10KHz = 20 kHz.

G. File WAV

WAV adalah format *audio* standar Microsoft dan IBM untuk *personal computer* (PC), biasanya menggunakan *coding* PCM (*Pulse Code Modulation*). WAV adalah data tidak terkompres sehingga seluruh sampel *audio* disimpan semuanya di *harddisk*. *Software* yang dapat menciptakan WAV dari *analog sound* misalnya adalah *Windows Sound Recorder*. File *audio* ini jarang sekali digunakan di internet karena ukurannya yang relatif besar dengan batasan maksimal untuk file WAV adalah 2 GB [6].

Parameter-parameter tersebut menyatakan *setting* yang digunakan oleh ADC *(Analog-to-Digital Converter)* pada saat data *audio* direkam. Biasanya laju sampel juga dinyatakan dengan satuan Hz atau kHz. Sebagai gambaran, data *audio* digital yang tersimpan dalam CD *audio* memiliki karakteristik laju sampel 44100 Hz, 16 bit per sampel, dan 2 kanal (*stereo*), yang berarti setiap satu detik suara tersusun dari 44100 sampel, dan setiap sampel tersimpan dalam data sebesar 16-bit atau 2 *byte*. Laju sampel selalu dinyatakan untuk setiap satu kanal. Jadi misalkan suatu data *audio* digital memiliki 2 kanal dengan laju sampel 8000 sampel/detik, maka sesungguhnya di dalam setiap detiknya akan terdapat 16000 sampel. Sebagaimana telah dijelaskan sebelumnya bahwa untuk *stream* data *audio* menggunakan *header* berupa struktur PCMWAVEFORMAT. PCM merupakan singkatan dari *Pulse Coded Modulation*, yaitu suatu metode yang digunakan untuk mengkonversikan sinyal *audio* dari bentuk analog ke bentuk digital.

H. File MP3

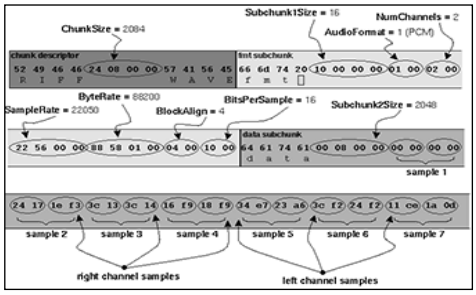
MPEG (*Moving Picture Expert Group*)-1 *audio* layer III atau yang lebih dikenal dengan MP3, adalah salah satu dari pengkodean dalam *digital audio* dan juga merupakan format kompresi *audio* yang memiliki sifat "menghilangkan". Istilah menghilangkan yang dimaksud adalah kompresi *audio* ke dalam format mp3 menghilangkan aspek-aspek yang tidak signifikan pada pendengaran manusia untuk mengurangi besarnya *file audio* [8].

\

Kompresi yang dilakukan oleh MP3 seperti yang telah disebutkan diatas, tidak mempertahankan bentuk asli dari sinyal input. Melainkan yang dilakukan adalah menghilangkan suara-suara yang keberadaannya kurang/tidak signifikan bagi sistem pendengaran manusia. Proses yang dilakukan adalah menggunakan model dari sistem pendengaran manusia dan menentukan bagian yang terdengar bagi sistem pendengaran manusia. Setelah itu sinyal input yang memiliki domain waktu dibagi menjadi blok-blok dan ditransformasi menjadi domain frekuensi. Kemudian model dari sistem pendengaran manusia dibandingkan dengan sinyal input dan dilakukan proses penyaringan yang menghasilkan sinyal dengan *range* frekuensi yang signifikan bagi sistem pendengaran manusia. Proses diatas adalah proses konvolusi dua sinyal yaitu sinyal input dan sinyal model sistem pendengaran manusia. Langkah terakhir adalah kuantisasi data, dimana data yang terkumpul setelah penyaringan akan dikumpulkan menjadi satu keluaran dan dilakukan pengkodean dengan hasil akhir *file* dengan format MP3. Kepopuleran dari MP3 yang sampai saat ini belum tersaingi disebabkan oleh beberapa hal. Pertama MP3 dapat didistribusikan dengan mudah dan hampir tanpa biaya, walaupun sebenarnya hak paten dari MP3 telah dimiliki dan penyebaran MP3 seharusnya dikenai biaya. Walaupun begitu, pemilik hak paten dari MP3 telah memberikan pernyataan bahwa penggunaan MP3 untuk keperluan perorangan tidak dikenai biaya. Keuntungan lainnya adalah kemudahaan akses MP3, dimana banyak *software* yang dapat menghasilkan file MP3 dari CD dan keberadaan *file* MP3 yang bersifat *ubiquitos* (kosmopolit).

I. Struktur dataFile Audio

Struktur data pada file *audio* berbeda-beda tergantung format *audio*-nya [10]. Misalnya file Wav memiliki struktur seperti pada Gambar. Kembali sama persis dari data yang telah dikompresi, dengan kata lain data asli tetap sama sebelum dan sesudah kompresi.



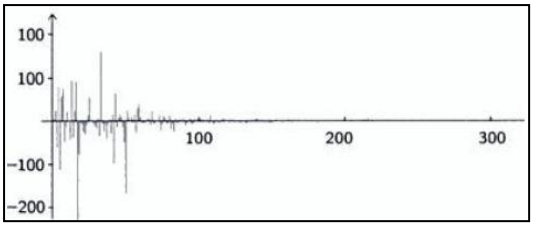
Gambar 2. Contoh struktur File WAVE Dalam Bentuk Hexa

Pada struktur file Wav di atas terdiri dari:

1) *Chunk Descriptor* yang terdiri dari data: 52 49 46 46 28 08 00 00 57 41 56 45.
2) *Fmt subChunk* yang terdiri data *subChunk1size*, *audioFormat*, *numChannel*, *sampleRate*, *byteRate* dan *BlockAlign*
3) Data *subChunk* yang terdiri dari data *subChunk2size* serta *sample-sample*

J. Pembacaan File Audio

Data audio yang di-encoding terdiri dari 576 baris frekuensi tiap channel dan bagian kecil yang disimpan sebagai 16 bit signed integer. Sebagai contoh diberi sebuah spektrum frekuensi pada file audio berformat wav yang dapat dilihat pada Gambar. Pada file suara yang terkuantisasi dilakukan *encoding* yang menghasilkan nilai *integer* yang merupakan nilai frekuensi dari sampel *audio.*



Gambar 3. Frekuensi file Audio

Sampel *audio* yang di-*encoding* dapat dilihat pada Gambar 4.

00 3e 1f 00 9a 00 1f 9a 00 3e 00 3f 00 3e 1f 00 5h 7d 00 -33 2f 5c 00
3e 1f 00 9a 00 1f 9f 00 3e 00 1f 00 3e 1f 00 5h 7d 02 -33 2f 5c 00 4d
2e 1f 00 9a 00 1f 9a 00 3e 00 1f 00 3e 1f 00 5h 7d 00 -33 2f 5c 00 3e
1f 00 9a 00 1f 9a 00 3e 00 1f 00 3e 1f 00 5h 7d 00 -23 2f 5c 00 3e 1f
12 9a 00 1f 9a 00 3e 00 1f 00 3e 1f 00 5h 7d 00 -33 2f 5c 00 3e 1f 00
7b 00 1b 9a 00 3e 00 1f 00 3e 1f 00 5h 7d 00 -33 2f 5c 00 3e 1f 00
9a 00 1f 9a 00 00 00 1f -12 3d 1f 00 5h 7d 00 -56 2f -01

Gambar 4. Encoding sampel *Audio*

K. Run Length encoding (RLE)

Berbeda dengan teknik-teknik sebelumnya yang bekerja berdasarkan karakter per karakter, teknik run length ini bekerja berdasarkan sederetan karakter yang berurutan. Run Length Encoding adalah suatu algoritma kompresi data yang bersifat lossless. Algoritma ini mungkin merupakan algoritma yang mudah untuk dipahami dan diterapkan untuk kompresi. Run Length Encoding (RLE) adalah algoritma kompresi yang sangat mendasar. Metode kompresi ini sangat sederhana, yaitu hanya memindahkan pengulangan byte yang sama berturut-turut (secara terus menerus).

L. Windows API (*Application Programming Interface*)

Windows API adalah sekumpulan fungsi-fungsi eksternal yang terdapat dalam file-file perpustakaan *Windows* (*library windows*) atau *file library* lainnya yang dapat digunakan dalam pembanguna program. Fungsi ini dapat menangani semua yang berhubungan dengan *Windows* seperti pengaksesan disk, *interface printer*, grafik Windows, kotak dialog, Windows *shell*, memainkan musik dan sebagainya [3].

Pada pemrograman Visual Basic, untuk bisa memutar sebuah file suara (WAV) atau file musik (MP3) harus menggunakan *Windows* API yaitu file winmm.dll. File ini terlebih dahulu dideklarasikan atau dikenalkan ke bahasa pemrograman yang sedang digunakan dengan cara sebagai berikut:

Declare Function mciSendString Lib "winmm.dll" alias _ "mciSendStringA"(ByVal lpstrCommand As String, ByVal lpstrReturnString As _ String,ByVal uReturnString as String,ByVal uReturnLength as long,byVal _ hwdnCallBack as long) As Long.

III. ANALISIS DAN PERANCANGAN

A. Analisis

Dalam penelitian ini dilakukan analisis dan perbandingan kompresi file *audio* yang bertipe WAV dengan MP3 dengan menggunakan metode *Run Length Encoding* (RLE). Proses kompresi data didasarkan pada kenyataan bahwa pada hampir semua jenis data selalu terdapat pengulangan pada komponen data yang dimilikinya, misalnya di dalam suatu data *audio* akan terdapat pengulangan penggunaan angka 0 sampai 9 atau huruf a sampai huruf z. Kompresi data melalui proses *encoding* bertujuan untuk menghilangkan unsur pengulangan ini dengan mengubahnya sedemikian rupa sehingga ukuran data menjadi lebih kecil.

Proses pengurangan unsur pengulangan ini dapat dilakukan dengan memakai beberapa teknik kompresi. Misalnya jika suatu komponen muncul berulang kali dalam suatu data, maka komponen tersebut tidak harus dikodekan berulang kali, tapi dapat dikodekan dengan menulis frekuensi muncul komponennya dan di mana komponen tersebut muncul. Teknik kompresi data lainnya, berusaha untuk mencari suatu bentuk kode yang lebih pendek untuk suatu komponen yang sering muncul.

Langkah di dalam proses kompresi dengan algoritma *Run-Length* adalah:

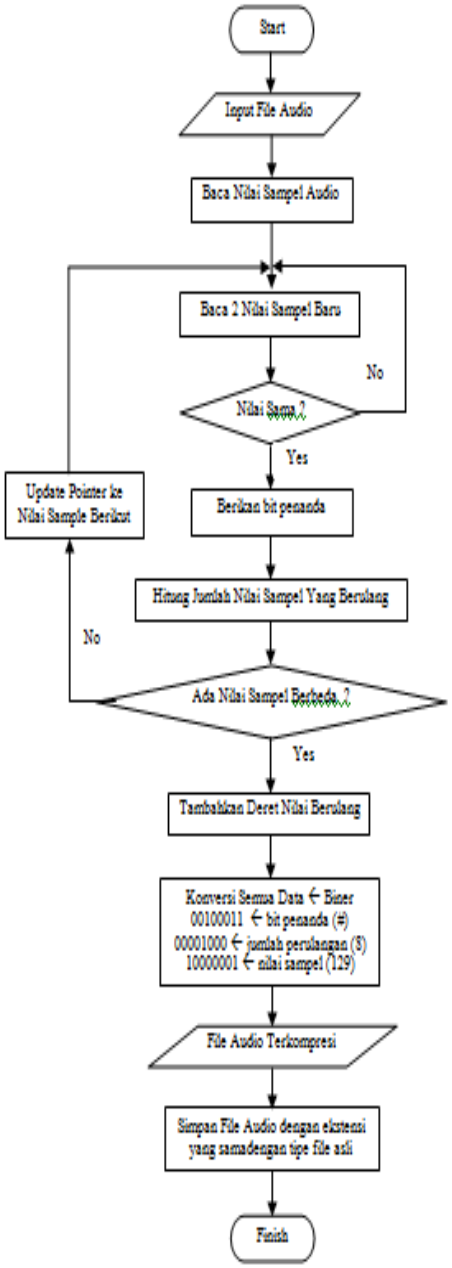
1) Pembacaan sampel *audio*
2) Kompresi data nilai jika ditemukan data nilai sampel yang sama secara berurutan lebih dari dua dengan cara:
    a. Berikan bit penanda # pada data kompresi,
    b. Tambahkan deretan bit untuk menyatakan jumlah nilai data yang sama berurutan.
    c. Tambahkan deretan bit yang menyatakan karakter yang berulang.
    d. Konversikan semua data kompresi ke dalam *hexa*.
3) Simpan file *audio* .
4) Hitung rasio kompresi.

Langkah dalam proses dekompresi algoritma Run Length encoding adalah :

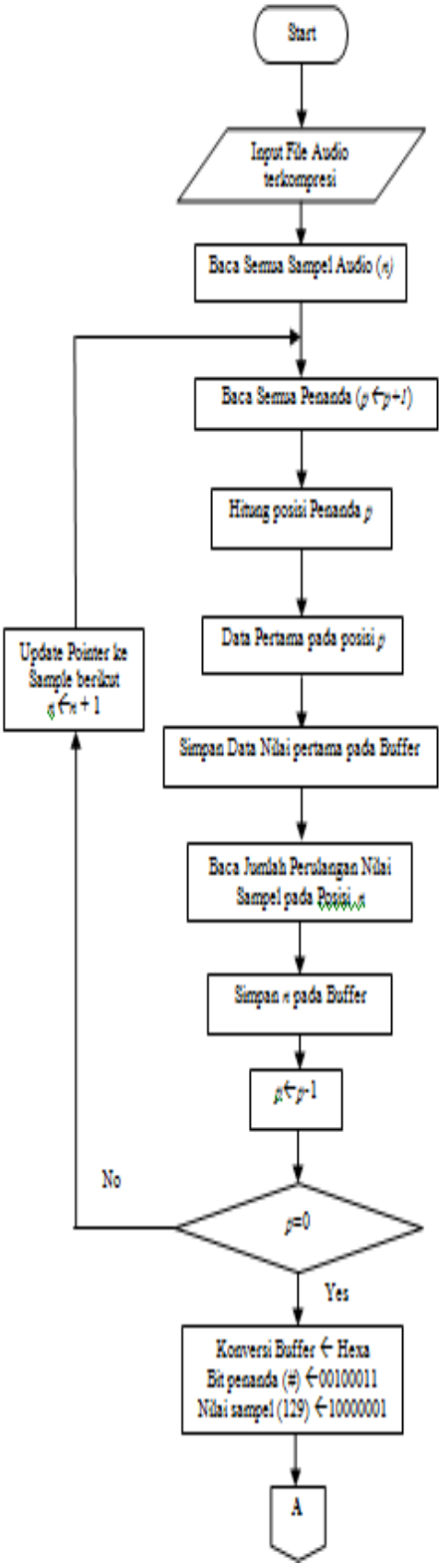
1) Buka file *audio* untuk membaca *header-header* dan *sample audio.*
2) Baca file *audio* untuk mendapatkan data *sample.*
3) Ambil nilai *sample audio* ke 1 sampai ke *n*

4) Lihat data pada hasil kompresi satu persatu dari awal sampai akhir, jika ditemukan *byte* penanda (#), lakukan proses pengembalian,
5) Baca posisi penanda setelah *byte* penanda, konversikan ke bilangan desimal untuk menentukan jumlah karakter yang berurutan.
6) Lihat data berikutnya, kemudian lakukan penulisan karakter tersebut sebanyak bilangan yang telah diperoleh pada data sebelumnya (*point* 5).
7) Ulangi langkah 4, 5 dan 6 sampai karakter terakhir.

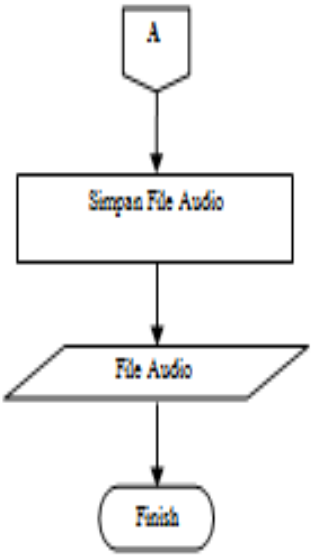
B. Flowchart



Gambar 5. Flowchart Kompresi



Gambar 6. Flowchart Dekompresi



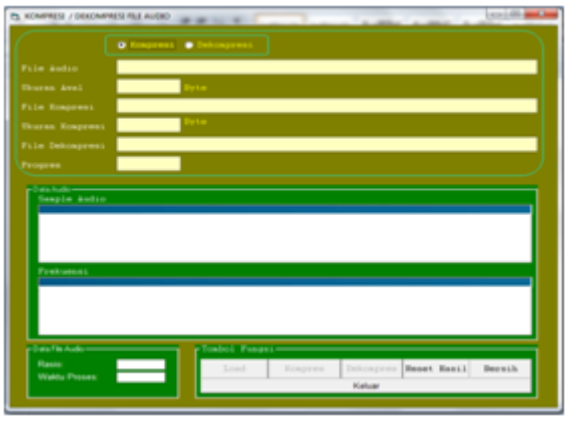
Gambar 7. Flowchart Dekompresi (Lanjutan)

IV. IMPLEMENTASI

Implementasi perangkat lunak adalah tampilan hasil rancangan dari penulisan kode program dimulai dari program Menu Utama, Kompresi/ Dekompresi serta Hasil pengujian.

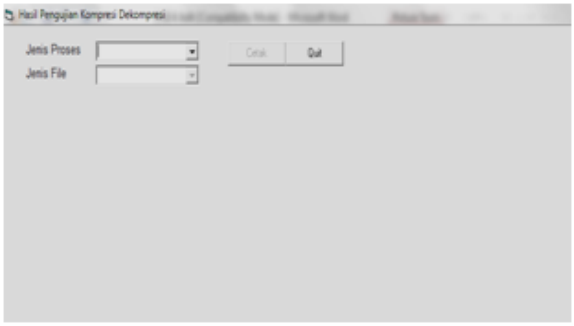
Gambar 8 Tampilan Menu Utama

Tampilan Menu Utama merupakan tampilan yang muncul setelah menjalankan program Utama. Tampilan ini berisi judul skripsi, gambar latar serta tampilan menu. Tampilan Menu terdiri dari Pilihan. Pada menu pilihan terdapat dua sub menu yaitu Kompresi dan Quit untuk menutup halaman menu utama. Didalam menu pilihan terdapat 2 sub menu yang dapat kita pilih, yaitu kompresi/dekompresi dan hasil pengujian.



Gambar 9. Tampilan Menu Kompresi/Dekompresi

Tampilan Kompresi merupakan tampilan berguna untuk melakukan proses kompresi dan dekompresi file *audio*. Tampilan Kompresi dapat dilihat pada Gambar 9. Tampilan Awal Kompresi adalah tampilan program kompresi yang masih kosong yang terdiri dari tombol *Load*, kompres, dekompres, reset hasil dan Bersih. Untuk melakukan kompresi file *audio*, terlebih dahulu pilih pilihan kompresi,seperti tampak pada gambar 4.3 diatas. Dengan dipilihnya pilihan kompresi, maka akan mengaktifkan tombol load.



Gambar 10. Tampilan Menu Hasil Pengujian

Pengujian sistem adalah pengumpulan data hasil proses kompresi dan dekompresi file audio dengan algoritma Run Length Encoding yang menampilkan waktu dan rsio kompresi. Tampilan Pencetakan Hasil Pengujian merupakan tampilan berguna untuk melakukan pencetakan hasil kompresi dan dekompresi file *audio*. Hasil Kompresi File MP3 adalah hasil pengujian kompresi file audio berformat MP3 dengan algoritma Run Length Encoding. Untuk dapat melihat hasil pengujian untuk kompresi , kita harus memilih ‘kompresi’ pada pilihan proses. Hasil Dekompresi File MP3 adalah hasil pengujian dekompresi file *audio* berformat MP3 dengan algoritma *Run Length Encoding*. Untuk dapat melihat hasil dekompresi file Mp3, kita hanya perlu mengganti pilihan menjadi ‘Dekompresi’ pada bagian jenis proses. Hasil Kompresi File WAV adalah hasil pengujian kompresi file *audio* berformat WAV dengan algoritma *Run Length Encoding*. Untuk dapat menampilkan hasil pengujian

kompresi file berformat WAV, kita harus mengganti pilihan jenis proses menjadi ‘kompresi’ dan mengganti pilihan jenis file menjadi ‘WAV’. Hasil Dekompresi File WAV adalah hasil pengujian dekompresi file *audio* berformat WAV dengan algoritma *Run Length Encoding*. Untuk dapat melihat tampilan hasil dekompresi file WAV, kita hanya perlu mengganti jenis proses menjadi dekompresi.

HASIL PENGUJIAN SISTEM

Proses: Kompresi

Jenis File: MP3

No	Nama File	UkuranAwal	UkuranAkhir	Rasio	Waktu
1	01. JADIKAN AKU RAJA.mp3	3,860,271.00	3,849,752.00	0.27	6.97
2	12 - The Power Of Love.MP3	5,900,288.00	5,889,489.00	0.18	7.43
3	Kom_Celine Dion - I Love You.m	5,225,082.00	5,225,861.00	0.01	7.54
4	CELINE_D___BEAUTY_AND_T	3,879,496.00	3,812,070.00	1.74	5.61
5	CHRISTIAN BAUTISTA - I Can L	3,456,191.00	3,452,339.00	0.11	4.71

Rasio Rata-rata: 0.46

Waktu Rata-rata: 6.45

Gambar 11. Tampilan Hasil Pengujian Sistem

V. KESIMPULAN & SARAN

Setelah merancang dan mengaplikasikan perangkat lunak kompresi file *audio* dengan menggunakan Algoritma *Run Length Encoding*, maka diperoleh hasil pengujian sistem adalah sebagai berikut:

1) Kompresi file MP3 dengan rasio rata-rata : 0.46 % dan waktu rata-rata: 6.45.
2) Dekompresi file MP3 dengan rasio rata-rata : 0.57 % dan waktu rata-rata: 5.66 detik.
3) Kompresi file WAV dengan rasio rata-rata : 13.83 % dan waktu rata-rata: 9.52 detik.
4) Dekompresi filer WAV dengan rasio rata-rata : 20.78 % dan waktu rata-rata: 4.23 detik.

Dari data di atas dapat ditarik kesimpulan bahwa :

1) Ukuran file sebelum kompresi sama dengan ukuran file sesudah didekompresi yang berarti bahwa tidak ada data yang hilang selama proses kompresi.
2) Waktu yang diperlukan untuk kompresi file WAV lebih besar dibandingkan dengan waktu kompresi file MP3.

Adapun saran-saran yang untuk penelitian maupun pengembangan berikutnya adalah:

1. Membandingkan rasio kompresi dengan metode kompresi *lossless* lainnya.
2. Melakukan kompresi dengan format audio yang lainnya antara lain format file MIDI serta format video.

REFERENSI

[1]. Binanto. 2010. *Multimedia Digital Dasar Teori +Pengembangan.Penerbit* ANDI: Yogyakarta.

[2]. Ferrianto, Gozali. 2008. Jurnal Ilmiah: Volume 4, Nomor 1, Agustus 2004, Halaman 37-52, ISSN 1412-0372 *Analisis Perbandingan Kompresi Data Dengan Teknik Arithmatic Coding dengan Run Length Encoding*. Jurusan Teknik Elektro. Universitas Trisakti. Jakarta.

[3]. Hadi, Rahadian. 1997. *Pemrograman Windows API dengan Microsoft Visual Basic*, Penerbit PT Elex Media Komputindo. Jakarta.

[4]. http://www.omninerd.com/articles/How_Audio_Compression_Works. Tanggal akses : 19 Peb 2010

[5]. Jogiyanto. 2005. *Analisis dan desain Sistem Informasi*. Penerbit: ANDI. Yogyakarta.

[6]. Meckah Merdiyan, Wawan Indarto. 2005. *Implementasi Algoritma Run Length, Half Byte Dan Huffman Untuk Kompresi File*. Seminar Nasional Aplikasi Teknologi Informasi 2005 (SNATI 2005). Yogyakarta, 18 Juni 2005

[7]. Pu, I. M. 2006. *Fundamental Data Compression*. Oxford : Elsevier..

[8]. Ruckert, Martin. 2005. Understanding MP3: *syntax, semantics, mathematics, and algorithms*. Birkhäuser.

[9]. Salomon, David dan Giovanni. 2010. *Handbook of Data Compression.* Fifth Edition. London : Springer – Verlag Limited.

[10]. Steve Rimmer.2001. *Advanced Multimedia Programming*. USA Windcrest/McGraw-Hill Inc.

[11]. Sutoyo, T dkk. 2009. *Teori Pengolahan Citra Digital*. Semarang : Andi