KOMPRESI CITRA MENGGUNAKAN METODE KOMPRESI *HALF-BYTE*

TUGAS AKHIR

Oleh :

NANANG KOESHARWANTO

0210960044-96

PROGRAM STUDI ILMU KOMPTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2009

KOMPRESI CITRA MENGGUNAKAN METODE KOMPRESI *HALF-BYTE*

TUGAS AKHIR

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer
dalam bidang Ilmu Komputer

Oleh:
NANANG KOESHARWANTO
0210960044-96

PROGRAM STUDI ILMU KOMPTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2009

KATA PENGANTAR

Segala puji syukur penulis panjatkan kehadirat Allah SWT, atas berkat rahmat dan karunia-Nya sehingga skripsi dengan judul : "Kompresi Citra Menggunakan Metode Kompresi *Half-Byte*" dapat terselesaikan.

Adapun maksud dan tujuan penyusunan skripsi ini adalah untuk memenuhi salah satu syarat dalam memperoleh gelar Sarjana Komputer pada Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya Malang.

Dalam proses penyusunan skripsi ini tidak sedikit kesulitan dan rintangan yang dihadapi, sehingga pada kesempatan ini pula penulis menyampaikan banyak rasa terima kasih kepada :

1. Nurul Hidayat, S.Pd, M.Sc, selaku pembimbing pertama yang telah banyak membantu dan meluangkan waktunya untuk membimbing penulis.
2. Edy Santoso, S.Si, M.Kom, selaku pembimbing kedua yang telah banyak membantu dan meluangkan waktunya untuk membimbing penulis hingga terselesainya skripsi ini.
3. Wayan Firdaus Mahmudy, S.Si., MT, selaku dosen penasehat akademik dan Ketua Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.
4. Dr. Agus Suryanto, M.Sc., selaku Ketua Jurusan Matematika FMIPA Universitas Brawijaya.
5. Segenap bapak dan ibu dosen di Fakultas MIPA Universitas Brawijaya yang selama ini membimbing dan memberikan bekal ilmu kepada penulis.
6. Seluruh staf dan karyawan di Fakultas MIPA Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi ini.
7. Rekan-rekan yang telah membantu penulis hingga dapat terselesaikannya skripsi ini.
8. Serta pihak yang secara langsung maupun tidak langsung telah membantu penulis demi kelancaran penulisan skripsi ini.

Penulis menyadari sepenuhnya bahwa penyusunan tugas akhir ini masih jauh dari sempurna dan oleh karenanya dengan segala kerendahan hati penulis menerima tanggapan baik berupa kritik maupun saran yang membangun.

Akhirnya penulis berharap agar skripsi ini dapat memberikan sumbangan serta manfaat bagi semua pihak yang berkepentingan.

Malang, 05 Pebruari 2009

Penulis

KOMPRESI CITRA MENGGUNAKAN METODE KOMPRESI *HALF-BYTE*

ABSTRAK

Gambar adalah bentuk file yang praktis dan tersaji menarik. Terdapat beberapa permasalahan dalam penyajian bentuk gambar atau citra antara lain kebutuhan ruang penyimpanan yang berkapasitas besar, biaya penyimpanan dan operasional yang besar serta diperlukannya waktu yang cukup lama dalam pengaksesannya. Oleh karena itu, diperlukan suatu metode untuk mengatasi permasalahan tersebut. Permasalahan tersebut dapat diatasi dengan banyak cara, salah satunya ialah dengan memperkecil (mengkompresi) ukuran gambar atau citra. Teknik kompresi adalah teknik untuk mereduksi jumlah bit yang diperlukan untuk penyimpanan data citra, sehingga ukuran citra yang terkompresi lebih kecil dibanding citra asal, tetapi tidak mengalami penurunan kualitas citra yang signifikan dengan demikian maka kapasitas ruang penyimpanan yang diperlukan akan menjadi lebih kecil serta waktu dan biaya pengaksesan menjadi lebih kecil pula. Saat ini ada banyak metode kompresi yang telah ada, salah satunya adalah metode kompresi *Half Byte*. Penelitian ini membahas kinerja metode kompresi *Half Byte*. Cara kerja metode *Half Byte* ialah dengan memanfaatkan bit sebelah kiri pada data (dalam bentuk heksadesimal) atau nilai yang sering sama secara berurutan. Sehingga data yang sering sama secara berurutan tersebut dapat dikompresi dengan cukup memakai satu kali bit sebelah kiri dan diikuti oleh bit sebelah kanan pada urutan data yang sama bit sebelah kirinya. Pada penelitian ini diujikan 12 sampel gambar, lalu kinerjanya diukur berdasarkan rasio ukuran file hasil kompresi terhadap file awal. Disimpulkan bahwa dalam hal rasio hasil kompresi, metode *Half Byte* mampu mencapai nilai rasio maksimal hingga 50 atau separuh dari ukuran file gambar aslinya, serta keberhasilan kompresi dengan menggunakan metode *Half Byte* tidak dipengaruhi oleh resolusi gambar dan ukuran file tetapi bergantung terhadap banyaknya variasi nilai warna yang sama yang berurutan.

IMAGE COMPRESSION USING
***HALF-BYTE* COMPRESSION METHOD**

ABSTRACT

Picture is a practical and interesting file. There are some problem in presentating picture or image, big capacities of storage space requirements, stock holding cost and operational, and required of a lot time in accessing it. Therefore, required a method to overcome the problems. Those problems can be solved in any ways, one of them is minimizing (compression) picture or image measure. Compression is a method to reduce number of bits needed in storaging data image, and the result of this method is image size is smaller rather than origin image, but doesn't have degradation of quality of image. So, the storage space capacities required, time and expense of access becomes. Nowdays, there are many compression methods, one of them is *Half Byte* compression method. This research studies performance of *Half Byte* of compression method. The work of *Half Byte* method is exploiting bit left side at data (in the form of heksadesimal) or value that is often same alternately. So the often data same sequentially can be compression enoughly using one bit left side and followed by bit right side at the same data sequence of bit left side. This research tested 12 picture samples, then its performance is measured based on result of file size ratio compression to initial file. Concluded that in the compression ratio results, *Half Byte* method can reach maximum ratio value until 50 or half from the original picture file size. The success of compression by using *Half Byte* method is not influenced by resolution of picture and file size but hinging on the number various the same colour value successively.

LEMBAR PENGESAHAN

KOMPRESI CITRA MENGGUNAKAN METODE KOMPRESI *HALF-BYTE*

Oleh:
NANANG KOESHARWANTO
0210960044-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 05 Pebruari 2009
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Nurul Hidayat, S.Pd, M.Sc	**Edy Santoso, S.Si, M.Kom**
NIP. 132 300 240	NIP. 132 304 307

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, M.Sc
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	: Nanang Koesharwanto
NIM	: 0210960044-96
Jurusan	: Matematika
Program Studi	: Ilmu Komputer
Penulis tugas akhir berjudul	: Kompresi Citra Menggunakan Metode Kompresi *Half Byte*

Dengan ini menyatakan bahwa :

1. Isi dari tugas akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.
2. Apabila dikemudian hari ternyata Tugas Akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 05 Pebruari 2009
Yang menyatakan,

Nanang Koesharwanto
NIM. 0210960044-96

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR *EVENLIST*

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Data atau *file* di dalam komputer memiliki berbagai macam bentuk atau tipe, antara lain *file image* atau citra, teks, aplikasi, file terkompresi dan lain sebagainya. Adapun bentuk *file* yang praktis dan tersaji menarik adalah berupa gambar.

Terdapat beberapa permasalahan dalam penyajian bentuk gambar atau citra antara lain kebutuhan ruang penyimpanan yang berkapasitas besar, biaya penyimpanan dan operasional yang besar serta diperlukannya waktu yang cukup lama dalam pengaksesannya. Oleh karena itu, diperlukan suatu metode untuk mengatasi permasalahan tersebut.

Permasalahan tersebut dapat diatasi dengan banyak cara, salah satunya ialah dengan memperkecil (mengkompresi) ukuran gambar atau citra. Teknik kompresi adalah teknik untuk mereduksi jumlah *bit* yang diperlukan untuk penyimpanan data citra, sehingga ukuran citra yang terkompresi lebih kecil dibanding citra asal, tetapi tidak mengalami penurunan kualitas citra yang signifikan dengan demikian maka kapasitas ruang penyimpanan yang diperlukan akan menjadi lebih kecil serta waktu dan biaya pengaksesan menjadi lebih kecil pula.

Metode pengkompresian data telah berkembang dan masing-masing metode memiliki kelebihan dan kekurangan. Kemampuan dari tiap-tiap metode pengkompresian data umumnya diukur dengan parameter ukuran atau kapasitas dan kecepatan kompresi.

Penelitian yang dilakukan penulis berdasarkan pada beberapa penelitian kompresi yang telah dilakukan oleh (1) Nur Hayati (2004), yang melakukan penelitian mengenai kompresi citra menggunakan metode kompresi *JPEG* yang memiliki tiga langkah utama, yakni Transformasi Kosinus Diskrit (*DCT*), Kuantisasi dan *Entropy Encoder*. Pada tahap *Entropy Encoder*, Hayati (2004) menggunakan metode *Run Length Encoding*.

Transformasi Kosinus Diskrit (*DCT*) adalah suatu proses pengubahan data citra menjadi deretan nilai *integer*. Kuantisasi adalah proses pembobotan dari karakteristik *DCT* dilakukan dengan membagi setiap komponen *DCT* dengan suatu bobot, kemudian

dibulatkan ke suatu nilai. Sedangkan *Entropy Encoder* adalah proses pengambilan niliai hasil kuantisasi dan mengkodekan ke biner sehingga proses kompresi lebih mudah dilakukan.

Hasil yang diperoleh dalam penelitian tersebut ialah rasio ukuran kompresi (perbandingan ukuran citra asal dan citra hasil kompresi) yang hasilnya berbeda-beda untuk tiap citra, tergantung dari jumlah warnanya. Rasio ukuran paling kecil bernilai 4,5% dan rasio terbesar adalah 71,56%.

(2) Herry Sujaini dan Yessi Mulyani, 2000 (tugas kuliah S-2 pada mata kuliah Jaringan Komputer). Dimana metode kompresinya menggunakan metode komresi *Run Length*, *Half Byte* dan *Hufmann* untuk sebarang *file*.

Pada penelitian ini penulis akan mencoba mengkhususkan kompresi *image*/citra berformat *bmp* menggunakan *entropy code Half Byte* yang proses kompresinya adalah dengan memanfaatkan *bit* sebelah kiri pada data (dalam bentuk heksadesimal) atau nilai yang sering sama secara berurutan. Sehingga data yang sering sama secara berurutan tersebut dapat dikompresi dengan cukup memakai satu kali *bit* sebelah kiri dan diikuti oleh *bit* sebelah kanan pada urutan data yang sama *bit* sebelah kirinya.

1.2 Rumusan Masalah

Dalam penelitian ini penulis menekankan perumusan masalah pada bagaimana hasil kompresi *file* citra dengan menggunakan metode kompresi *Half Byte*.

1.3 Batasan Masalah

Batasan-batasan pembahasan dalam penelitian ini ialah:

1. Sampel citra yang digunakan hanya format *bmp* dan berjumlah dua belas buah.
2. Efektivitas dibatasi pada perbandingan ukuran *file* antara citra asal dengan *file* hasil kompresi.

1.4 Tujuan Penelitian

Tujuan dari penulisan penelitian ini adalah

1. Perancangan dan implementasi kompresi *file* citra menggunakan metode kompresi *Half Byte*.
2. Menganalisa dan menghitung tingkat kompresi berdasarkan rasio perbandingan ukuran *file* citra hasil kompresi dengan *file* citra aslinya.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah didapatkannya kapasitas atau ukuran *file* citra yang lebih kecil sehingga meringankan suatu proses yang menggunakan citra serta memperkecil kebutuhan akan jumlah media penyimpanan yang diperlukan.

1.6 Metodologi Penelitian

Metodologi pelaksanaan kegiatan penelitian ini dilaksanakan dengan :

1. Kajian pustaka dengan mempelajari konsep metode kompresi *Half Byte* dan prosesnya.
2. Implementasi algoritma kompresi metode kompresi *Half Byte* pada citra.

1.7 Sistematika Penulisan

Sistematika penulisan penelitian ini dibagi menjadi 5 (lima) bab, yakni:

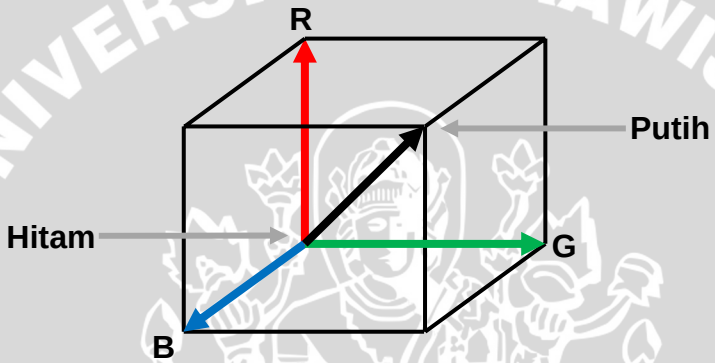
- BAB I PENDAHULUAN
  Dalam bab ini membahas tentang latar belakang permasalahan, rumusan masalah, tujuan, batasan masalah yang diambil, metodologi yang digunakan dan sistematika pembahasan.
- BAB II TINJAUAN PUSTAKA
  Membahas tentang teori dasar sebagai penunjang dari permasalahan yang diambil.

- BAB III METODOLOGI DAN PERANCANGAN
  Bab ini berisi tentang metodologi dan perancangan metode kompresi *Half Byte*.
- BAB IV HASIL DAN PEMBAHASAN
  Dalam bab ini berisi tentang pembahasan analisa kemampuan metode kompresi *Half Byte* kemudian dilakukan proses rasio berdasarkan ukuran hasil kompresinya.
- BAB V KESIMPULAN DAN SARAN
  Merupakan bab penutup yang berisi kesimpulan dan saran.

BAB II
TINJAUAN PUSTAKA

2.1 Citra (*Image*)

Menurut Mandala (2003) citra merupakan dimensi yang berisi informasi warna dan tidak bergantung pada waktu. Sebuah *Personal Computer* (*PC*) hanya dapat mengenali sebuah citra melalui deretan angka digital. Deretan angka tersebut mewakili posisi titik citra. Titik (atau yang disebut dengan piksel) citra menggambarkan posisi koordinat dan mempunyai intensitas atau tingkat keabuan yang dapat dinyatakan dengan bilangan. Intensitas ini menunjukan warna citra, melalui penjumlahan (*Red, Green* dan *Blue/ RGB*). Jika setiap tingkat keabuan citra diwakili oleh 1 byte atau 8 *bit*, maka tingkat keabuan dihitung 2^8 atau 256 nilai yaitu 0 sampai dengan 255. Dimana 0 mewakili tingkat intensitas gelap dan 255 mewakili tingkat intensitas terang.



Gambar 2.1 Koordinat *RGB*
Sumber : Mandala, 2003

Gambar 2.1 menjelaskan antara citra hitam putih (*monokrom*) dengan citra berwarna (*multispektral*) terdapat perbedaan pada kanal yang digunakan dalam perwakilan warna. Pada citra hitam putih hanya memakai 1 kanal, dimana f(x,y) merupakan fungsi tingkat keabuan dari hitam ke putih, x menyatakan variabel baris dan y variabel kolom. Citra berwarna memakai 3 kanal warna yang mewakili komponen warna yaitu merah, hijau dan biru (*RGB*).

Satu piksel diwakili 24 *bit* yang dibagi menjadi 8 *bit* per segmen. Setiap segmen tersebut mewakili salah satu warna dari *RGB* (Hayati, 2004).

2.2 Kompresi Citra

Pada umumnya citra berwarna mengandung jumlah data yang cukup besar baik dilihat dari jumlah pikselnya maupun dilihat dari besarnya nilai integer serta ditemukan kemungkinan dimana suatu piksel dengan piksel-piksel tetangganya mempunyai intensitas yang sama, hal ini menyebabkan pemborosan tempat penyimpanan. Oleh karena itu, dilakukan proses kompresi data tersebut hingga dapat meningkatkan efisiensi dalam penyimpanan.

Secara umum, kompresi atau pemampatan data merupakan proses mengubah deretan angka (simbol) masukan menjadi keluaran yang dikehendaki. Jika kompresi efektif, hasil berkas (*file*) keluaran akan lebih kecil dari berkas aslinya (Nelson, 1996).

Model kompresi atau pemampatan data dapat dibagi menjadi dua yaitu *statistical model* dan *dictionary-based model* (Hayati, 2004). *Statistical model* adalah model yang mengkodekan setiap simbol masukan dengan suatu kode tertentu. Kompresi dengan model statistik diterapkan pada algoritma *Huffman*. Kode tersebut ditentukan berdasarkan statistik, frekuensi kemunculan yang tinggi dikodekan dengan kode yang lebih pendek, sedangkan simbol dengan frekuensi rendah dikodekan dengan kode yang lebih panjang. *Dictionary-based model* adalah suatu model yang mengkodekan setiap simbol masukan dengan suatu alamat simbol tersebut disimpan. Model kompresi ini bekerja dengan *pointer* atau indek yang menghubungkan masukan dengan simbol yang disimpan. Model kompresi ini terdapat dalam metode *Run-Length Encoding* dan *Half-Byte.*

2.3 Proses Kompresi Citra Dengan Metode *Half Byte*

Metode kompresi *Half Byte* merupakan suatu metode kompresi dengan prosesnya adalah memanfaatkan *bit* sebelah kiri yang sering sama secara berurutan (Sujaini, 2000). Misalnya pada suatu *file* yang berisi data teks bertuliskan ”bilangan”, dalam heksadesimal akan diterjemahkan pada tabel 2.1.

Tabel 2.1 Konversi Heksadesimal

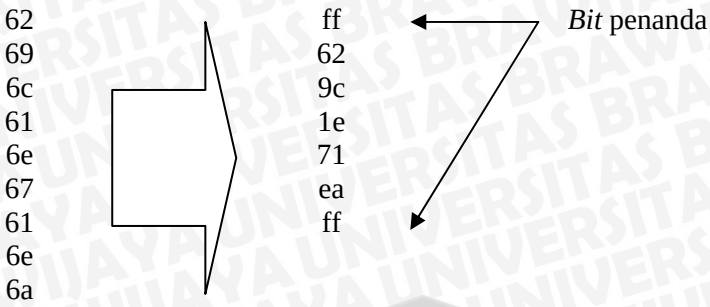
Karakter	Heksadesimal
b	62
i	69
l	6c
a	61
n	6e
g	67
a	61
n	6e

Jika diperhatikan karakter-karakter tersebut memiliki *bit* sebelah kiri yang sama yaitu ’6’. Gejala seperti inilah yang dimanfaatkan oleh metode kompresi *Half Byte*.

Saat karakter yang *bit* kirinya sama secara berderet, maka algoritma ini mengkompres data tersebut diawali dengan ”*bit* penanda” kemudian *bit* pertama dari deretan yang sama diikuti dengan pasangan *bit* kanan dari deretan tersebut dan ditutup dengan *bit* penanda.

Bit penanda (*marker bit*, dalam penelitian ini disingkat ‘mb’), berupa suatu *byte* yang boleh dipilih secara acak asalkan digunakan secara konsisten pada seluruh *bit* penanda pemampatan. *Bit* penanda disini berfungsi penanda bahwa karakter selanjutnya adalah karakter pemampatan atau akhir pemampatan. Dalam penelitian ini, penentuan *bit* penanda dilakukan dengan mencari frekuensi nilai warna yang paling sedikit yang terdapat dalam sebuah citra.

Metode kompresi *Half Byte* digambarkan lebih jelas pada gambar 2.2.



Gambar 2.2 Proses metode kompresi *Half Byte*
Sumber : Sujaini, 2000

Deretan data sebelah kiri merupakan deretan data pada *file* asli, sedangkan deretan data sebelah kanan merupakan deretan data hasil pemampatan dengan metode *Half Byte*.

2.4 Efektivitas

Dalam melihat efektivitas suatu metode kompresi dapat dinilai dengan berbagai cara, salah satunya adalah dengan mengukur rasio ukuran kompresinya atau perbandingan ukuran *file* asal dengan *file* hasil kompresi. Secara sederhana dapat dituliskan pada persamaan 2.1.

$$RK = \left(1 - \left[\frac{Ukuran\ file\ kompresi}{Ukuran\ file\ asli}\right]\right) x\ 100 \quad \text{.....(2.1)}$$

Dari persamaan 2.1 didapatkan bahwa semakin besar prosentase rasio ukuran kompresi, berarti hasil kompresi semakin baik (Nelson, 1996).

BAB III
METODOLOGI DAN PERANCANGAN

3.1 Rancangan Penelitian

Rancangan umum merupakan urutan langkah yang terstruktur dan dituliskan secara sistematis yang akan dikerjakan untuk penyelesaian suatu masalah. Pada penelitian ini dilakukan dua metode pokok (kompresi dan dekompresi *file*) dengan langkah-langkah pada tiap metodenya.

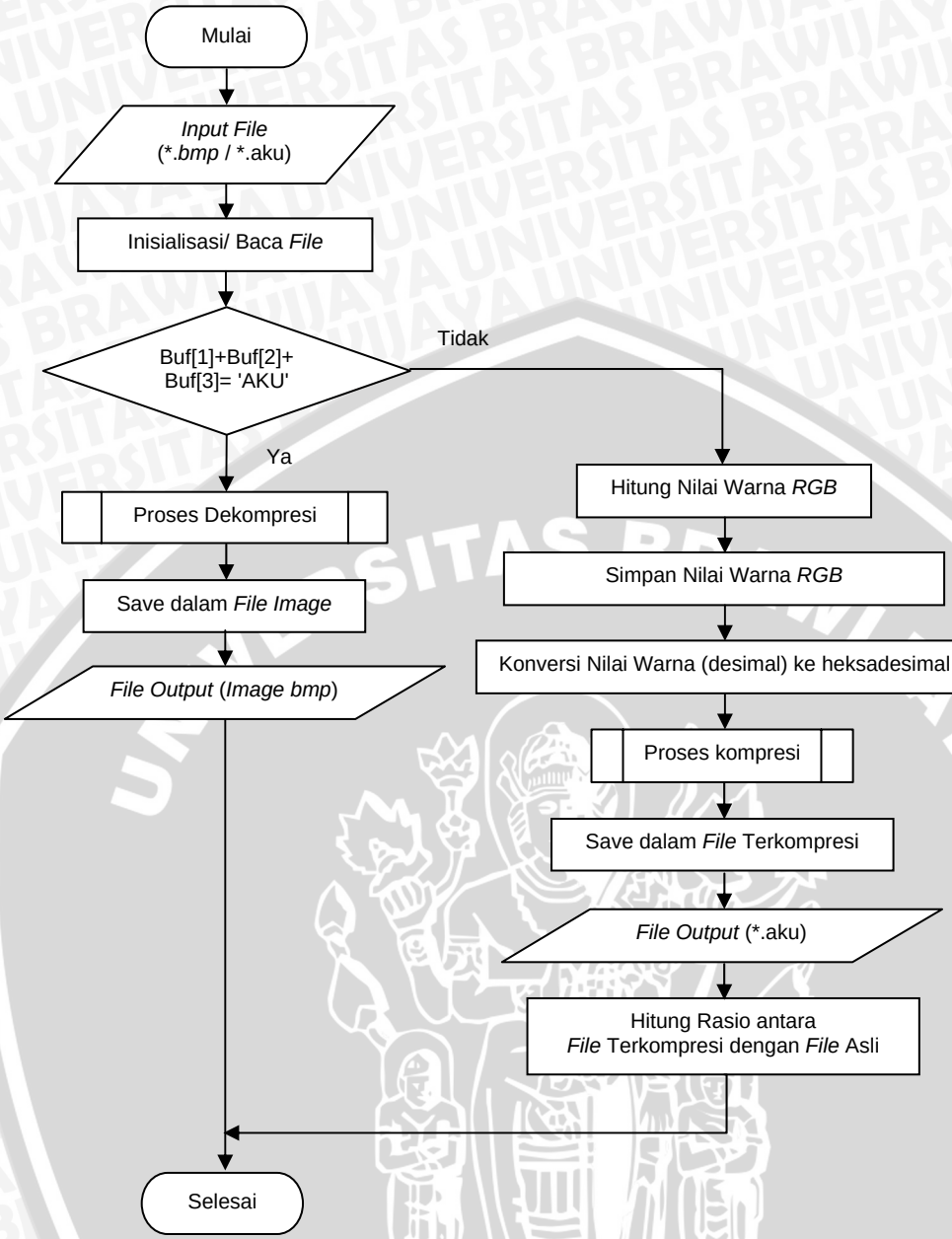
❖ **Kompresi *File***

1. Memilih *file* citra dengan format *bmp* dengan resolusi tidak lebih dari 300 x 300.
2. Menghitung nilai warna *Red*, *Green* dan *Blue* di tiap piksel citra.
3. Penyimpanan nilai tiap warna *RGB*.
4. Pengkodean nilai tiap warna (desimal) ke heksadesimal.
5. Proses kompresi *Half byte*.
6. Menyimpan *file* hasil kompresi.
7. Menghitung rasio kapasitas atau ukuran *file*.

❖ **Dekompresi *File***

1. Memilih *file* (terkompresi) dengan format ektensi tertentu (dalam penelitian ini menggunakan ekstensi *.aku).
2. Membaca isi *file*, dan memastikan bahwa data *file* berupa array dengan awal data berisi karakter 'AKU' dan *bit* penanda.
3. Proses dekompresi *Half Byte*.
4. Menyimpan *file* hasil dekompresi.

Berdasarkan langkah-langkah tiap metode kompresi dan dekompresi *file* dalam rancangan penelitian ini, maka dapat dibuat *flowchart* yang ditunjukkan pada gambar 3.1. Untuk pembahasan selanjutnya, peneliti akan membahas metode kompresi terlebih dahulu kemudian dilanjutkan dengan metode dekompresi.



Gambar 3.1 *flowchart* rancangan penelitian

3.2 Kompresi *File*

3.2.1 *File Input*

File input atau data yang dipergunakan sebagai data awal dalam tugas akhir ini ialah data citra yang bertipe atau memiliki format *bit*map (*.*bmp*). Resolusi data citra yang digunakan dibatasi tidak lebih dari 300 x 300 piksel. Penghitungan sumbu x dan sumbu y pada citra dimulai pada posisi kiri atas citra. Apabila ada data citra *input* yang memiliki resolusi lebih dari 300 x 300 piksel, maka data citra tersebut akan di-*crop* atau dipotong kelebihannya. Pemotongan dilakukan dengan membuang citra pada sumbu x ≥ 300 dan atau sumbu y ≥ 300. Gambar 3.2 adalah citra asal dengan resolusi 360 x 316 (3.2.a) dan citra setelah pemotongan dengan resolusi 300 x 300 (3.2.b)

a b

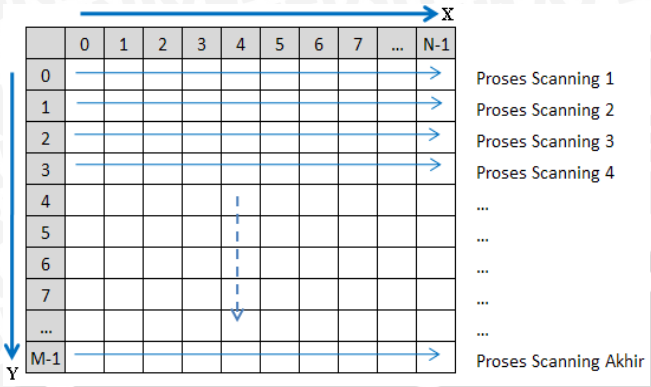
Gambar 3.2 Citra asal (a) dan citra hasil pemotongan (b)

3.2.2 Penghitungan Nilai Warna

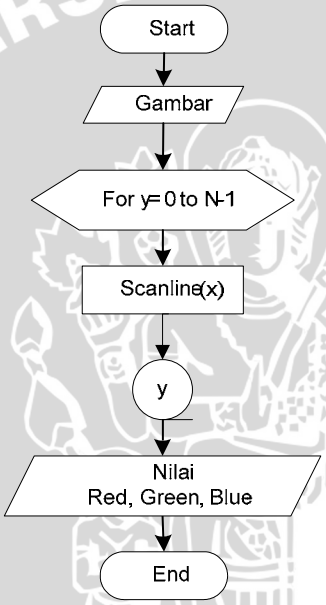
Setiap piksel data citra memiliki komposisi 3 (tiga) warna penyusun. Masing-masing ialah *red*/ merah, *green*/ hijau dan *blue*/ biru. Dan setiap warna memiliki nilai antara 0 – 255.

Untuk mendapatkan nilai warna *RGB*, dilakukan proses *scanning color* di setiap piksel yang dilakukan per baris (sumbu x) dimulai dari koordinat (0,0). Kemudian berjalan hingga akhir sumbu x pada citra atau lebar citra. Setelah proses *scanning* pada sumbu x selesai dengan y = 0, dilanjutkan dengan y = 1 yang diawali pada x = 0 berjalan hingga akhir sumbu x pada citra. Hal ini berlanjut hingga akhir sumbu y pada citra atau tinggi citra. Setelah seluruh citra

selesai di *scanning* maka akan didapatkan nilai *RGB* di tiap pikselnya. Gambar 3.3 adalah ilustrasi proses *scanning color* dan *flowchart*nya ditunjukkan pada gambar 3.4.

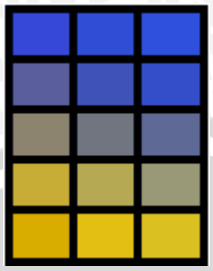


Gambar 3.3 Ilustrasi proses *scanning color* pada citra



Gambar 3.4 *Flowchart* Pembacan intensitas warna dengan fungsi *scanline*

Gambar 3.5 adalah contoh penghitungan nilai warna pada citra yang mempunyai resolusi 3 x 5 piksel (diperbesar ± 3000%) dengan menggunakan proses *scanning color*.



a

Nilai R		
54	49	46
74	54	51
152	106	74
212	206	176
216	226	218

b

Nilai G		
72	79	80
81	79	80
140	111	89
178	185	170
172	188	191

c

Nilai B		
216	215	211
170	197	214
105	130	160
31	63	105
3	10	30

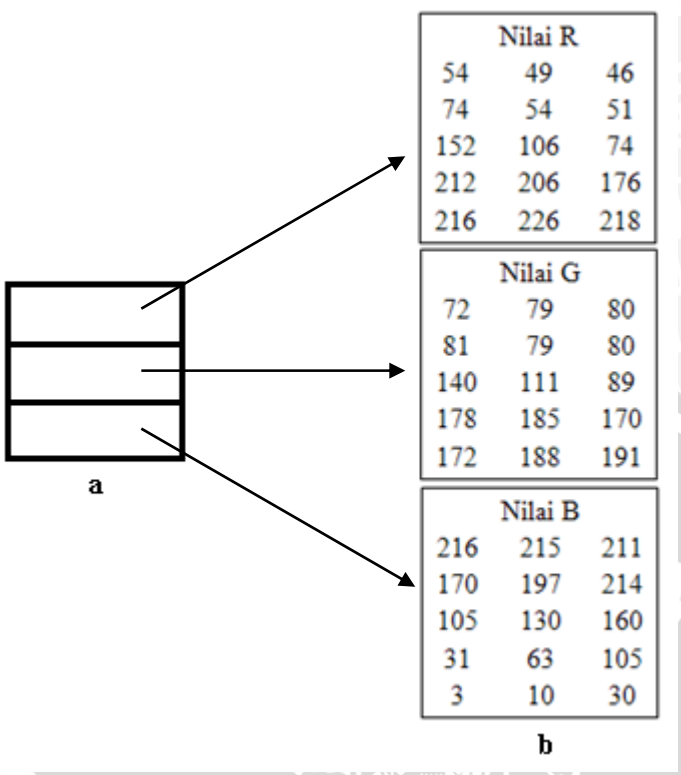
d

Gambar 3.5 Contoh perhitungan nilai warna
(a) Citra beresolusi 3 x 5 piksel.
(b,c,d) Nilai warna *RGB* ditiap piksel.

3.2.3 Penyimpanan Nilai Warna

Setelah langkah perhitungan nilai dengan menggunakan proses *scanning color* selesai, nilai pada masing-masing warna disimpan dalam suatu *record* (tipe data terstruktur yang berisi sejumlah data dan masing-masing data dapat berbeda tipe (Kadir, 2000)), dimana *record* tersebut diisi dengan data nilai ketiga warna tersebut.

Ketiga data nilai warna yang disimpan dalam *record* tersebut berbentuk *array* 2 dimensi (*array of array*). Gambar 3.6 adalah gambaran dari penyimpanan data warna ke dalam *record*.



Gambar 3.6 Gambaran suatu *record* yang berisi data
Nilai warna berupa *array* dua dimensi
(a) *Record*, (b) Isi *record*

3.2.4 Pengkodean Nilai Warna

Langkah pengkodean nilai warna pada array dilakukan dengan merubah nilai warna (desimal) ke dalam bilangan heksadesimal. Proses kompresi *Half Byte* memproses data dengan bilangan heksadesimal. Karena proses kompresi *Half Byte* memanfaatkan *bit* sebelah kiri yang sering sama secara berurutan.

Untuk merubah bilangan desimal ke dalam bilangan heksadesimal dilakukan dengan cara membagi bilangan desimal dengan 16 dan kemudian didapatkan sisa pembagian (berupa 0 sampai dengan F). Tabel 3.1 memperjelas langkah konversi bilangan desimal ke bilangan heksadesimal.

Tabel 3.1 Tabel konversi desimal ke heksadesimal

Desimal	Heksadesimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	a
11	b
12	c
13	d
14	e
15	f

Contoh perhitungan konversi desimal dengan nilai 200 ke dalam bentuk heksadesimal.

218 : 16 = 13, sisa 10
13 → d
10 → a
Maka 200 dalam heksadesimal menjadi da.

3.2.5 Proses Kompresi *Half Byte*

Setelah data nilai dijadikan array 1 dimensi dan telah dirubah menjadi bilangan biner, maka langkah proses kompresi *Half Byte* dapat dilakukan. Langkah awal ialah membaca data apakah terdapat deretan data yang *bit* kirinya sama secara berurutan sebanyak tujuh data atau lebih, jika memenuhi lakukan pemampatan. Kemudian berikan *bit* penanda pada *file* pemampatan.

Selanjutnya tambahkan *bit* kiri data pertama dari *file* asli dan gabungkan *bit* kanan karakter yang sama ke *file* pemampatan. Lalu tutup dengan *bit* penanda pada *file* pemampatan.

Contoh kompresi *Half Byte* dalam bentuk tabel disajikan dalam tabel 3.2, untuk menyederhanakannya maka penulis memproses langsung data bilangan heksadesimal dengan *bit* penanda 'ff'. Dari tabel 3.2 data asli berukuran 30 byte dan data hasil kompresi berukuran 25 byte. Dengan demikian proses telah berhasil mengkompresi data sebanyak 5 byte.

Dalam kompresi *Half Byte* ada beberapa ketentuan selain yang telah ditunjukkan pada tabel 3.2, antara lain (ditunjukkan pada gambar 3.7, 3.8, dan 3.9) :

- Bila pada *file* asli ditemukan nilai yang sama dengan *bit* penanda, maka dalam *file* terkompresi hasus dituliskan nilai tersebut sebanyak dua kali secara beruntun.

***File* Asli**	***File* terkompresi**	**Keterangan**
ff	ff	*
f6	ff	mb
fc	f6	
fe	ce	
f6	6a	
fa	a2	
fa	02	
f2	ff	mb
f0		
f2		

Gambar 3.7 Kompresi *Half Byte*
dengan nilai yang sama dengan *bit* penanda

*) Nilai yang ditulis ulang karena sama dengan *bit* penanda

- Bila terjadi penggabungan *bit* kanan menghasilkan nilai yang sama dengan *bit* penanda, sehingga nilai tersebut

diduga sebagai *bit* penutup, maka deretan *file* tersebut tidak dikompresi.

***File* Asli**	***File* terkompresi**	**Keterangan**
ef	ff	mb
e8	ef	
ec	8c	
ef	ff	*
ef	ab	
ea	20	
eb	ff	mb
e2		
e0		

Gambar 3.8 Kompresi *Half Byte*
dengan nilai hasil kompresi sama dengan *bit* penanda

*) Nilai hasil kompresi sama dengan *bit* penanda

- Bila banyaknya nilai yang dapat dikompresi berjumlah genap, maka nilai terakhir tidak perlu dikompresi.

***File* Asli**	***File* terkompresi**	**Keterangan**
a8	ff	mb
a5	a8	
aa	5a	
ad	db	
ab	69	
a6	14	
a9	ff	mb
a1		
a4		
a2		*

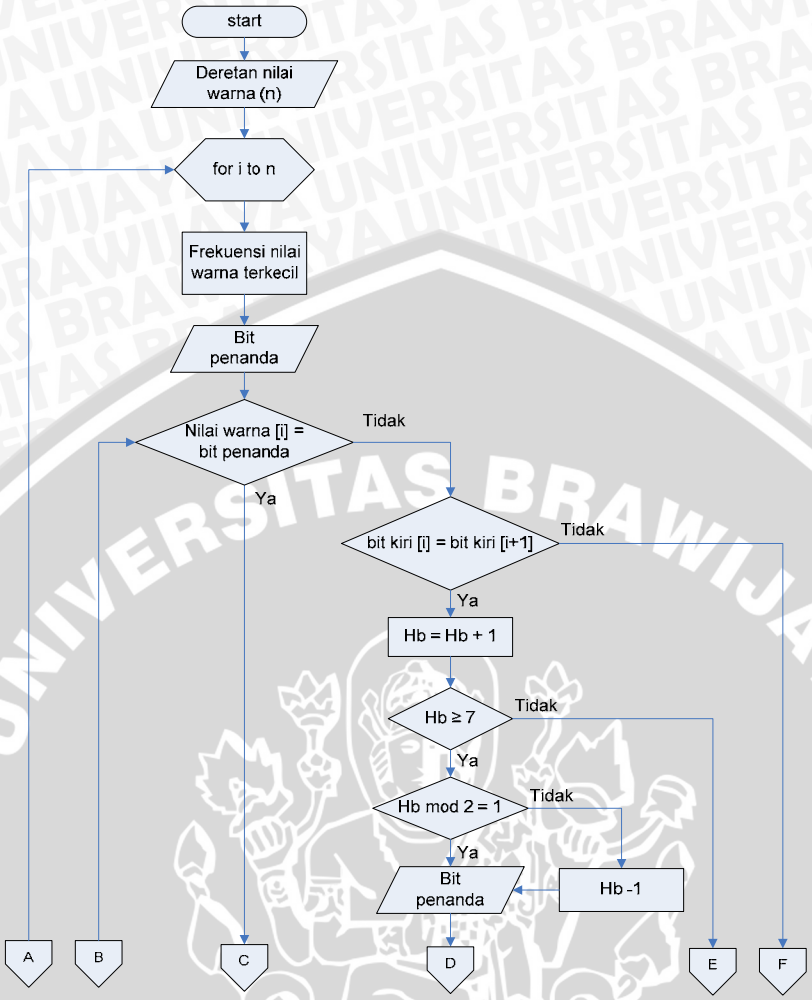
Gambar 3.9 Kompresi *Half Byte* dengan banyaknya nilai yang dapat dikompresi berjumlah genap

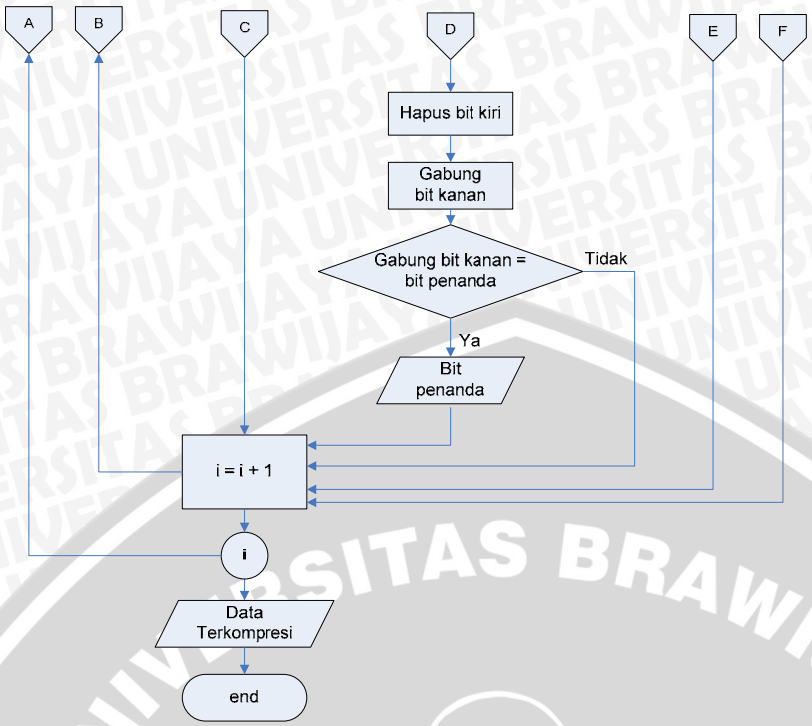
*) Nilai yang tidak dikompresi

Tabel 3.2 Kompresi *Half Byte*

Hekadesimal	Hasil Kompresi	Keterangan
62	ff	mb
69	62	*File* terkompresi
6c	9c	
61	1e	
6e	71	
67	ee	
61	ff	mb
6e	00	
6e	00	
00	01	
00	43	
01	55	
43	ff	mb
55	17	*File* terkompresi
17	be	
1b	f0	
1e	34	
1f	15	
10	2d	
13	ff	mb
14	30	
11	3f	
15	60	
12	6f	
1d	6f	
30		
3f		
60		
6f		
6f		

Dari pembahasan mengenai kompresi *Half Byte* dapat disusun *flowchart* yang ditunjukkan pada Gambar 3.10.


start
Deretan nilai warna (n)
for i to n
Frekuensi nilai warna terkecil
Bit penanda
Nilai warna [i] = bit penanda
Tidak
Ya
bit kiri [i] = bit kiri [i+1]
Tidak
Ya
Hb = Hb + 1
Hb ≥ 7
Tidak
Ya
Hb mod 2 = 1
Tidak
Ya
Bit penanda
Hb -1
A
B
C
D
E
F



Gambar 3.10 *Flowchart* Proses Kompresi

3.2.6 Penyimpanan Data *File* Terkompresi

Setelah data selesai dikompresi dengan proses kompresi *Half Byte*, maka langkah selanjutnya ialah penyimpanan *file* hasil kompresi. Tujuannya ialah memberikan label nama pada *file* tersebut dan dapat dilihat ukuran *file*nya. Adapun format penyimpanan dalam penelitian ini diatur dengan aturan :

- Berekstensi *.aku.
- Data ke-1 sampai ke-3 diisi karakter 'AKU' yang berfungsi sebagai sandi atau kode bahwa *file* ini adalah *file* yang terkompresi dengan metode *Half Byte*.
- Data ke-4 diisi dengan *bit* penanda.
- Data ke-5 dan selanjutnya adalah data *file* hasil kompresi.

3.2.7 Perhitungan Rasio Ukuran Kompresi

Setelah proses penyimpanan selesai, maka akan diketahui berapa ukuran *file*nya. Ukuran *file* ini yang akan diproses dalam perhitungan rasio dengan *file* aslinya.

Dengan menggunakan persamaan 2.1, diberikan contoh perhitungannya dengan contoh ukuran *file* terkompresi berukuran 20 dan ukuran *file* asli berukuran 25.

$$RK = \left(1 - \left[\frac{\text{Ukuran } file \text{ kompresi}}{\text{Ukuran } file \text{ asli}}\right]\right) x\,100$$

$$RK_1 = \left(1 - \left[\frac{20}{25}\right]\right) x\,100$$

$$RK_1 = (1 - [0{,}8]) \times 100$$

$$RK_1 = (0{,}2) \times 100$$

$$RK_1 = 20\ \%$$

3.3 Dekompresi *File*

3.3.1 *File Input* dan Pembacaan Data *File*

File input yang dipakai dalam proses dekompresi adalah *file* terkompresi atau *file* hasil kompresi metode *Half Byte* dengan format ekstensi *.aku.

Langkah selanjutnya adalah membaca isi *file*. Isi data ke-1 sampai ke-3 harus dipastikan adalah karakter 'AKU' (sebagai sandi atau kode bahwa *file* ini adalah *file* yang terkompresi dengan metode *Half Byte*). Data ke-4 adalah *bit* penanda. Data ke-5 dan selanjutnya adalah data *file image* yang terkompresi.

Sebagai gambaran isi data, ditunjukkan pada tabel 3.3.

Tabel 3.3 Isi data *file* terkompresi
dengan metode *Half Byte*

Data ke	1	2	3	4	5	…	n
Nilai	A	K	U	mb	Data terkompresi		

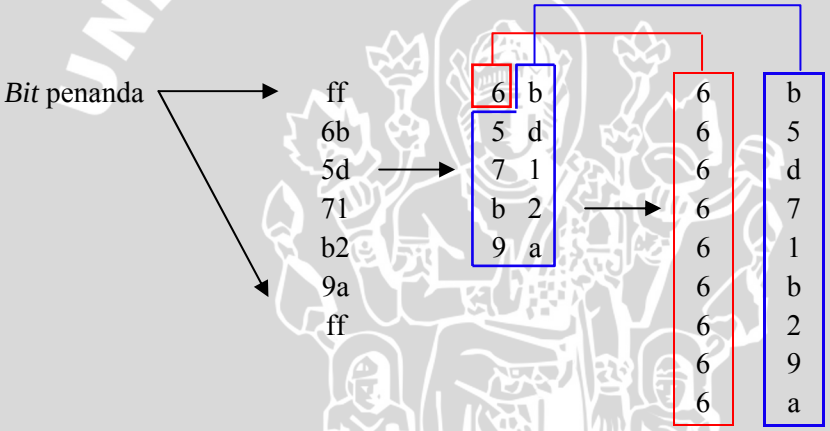
3.3.2 Proses Dekompresi *Half Byte*

Dalam melakukan proses pengembalian data terkompresi ke data asli (berupa gambar *bmp*), dilakukan beberapa langkah, yakni :

- Melihat seluruh data *file* dimulai dengan data ke-5.
- Jika data bukan merupakan *bit* penanda, maka data langsung dimasukkan ke dalam *file* pengembalian.
- Jika data merupakan *bit* penanda, dilakukan beberapa langkah, yakni:

    - Data kedua (data setelah *bit* penanda) ditambahkan ke dalam *file* pengembalian.
    - Pada data berikutnya, gabungkan *bit* kiri (sebagai *bit* kanan) dengan data *bit* kedua sebelah kiri (sebagai *bit* kiri). Kemudian *bit* kanannya (sebagai *bit* kanan) dengan data *bit* kedua sebelah kiri (sebagai *bit* kiri). Hasil dari penggabungan data ditambahkan pada *file* pengembalian.
    - Lakukan penggabungan *file* hingga ditemukan *bit* penanda yang berfungsi sebagai *bit* penutup.

Untuk lebih jelas, Gambar 3.11 menunjukkan ilustrasi proses pengembalian *file* terkompresi.



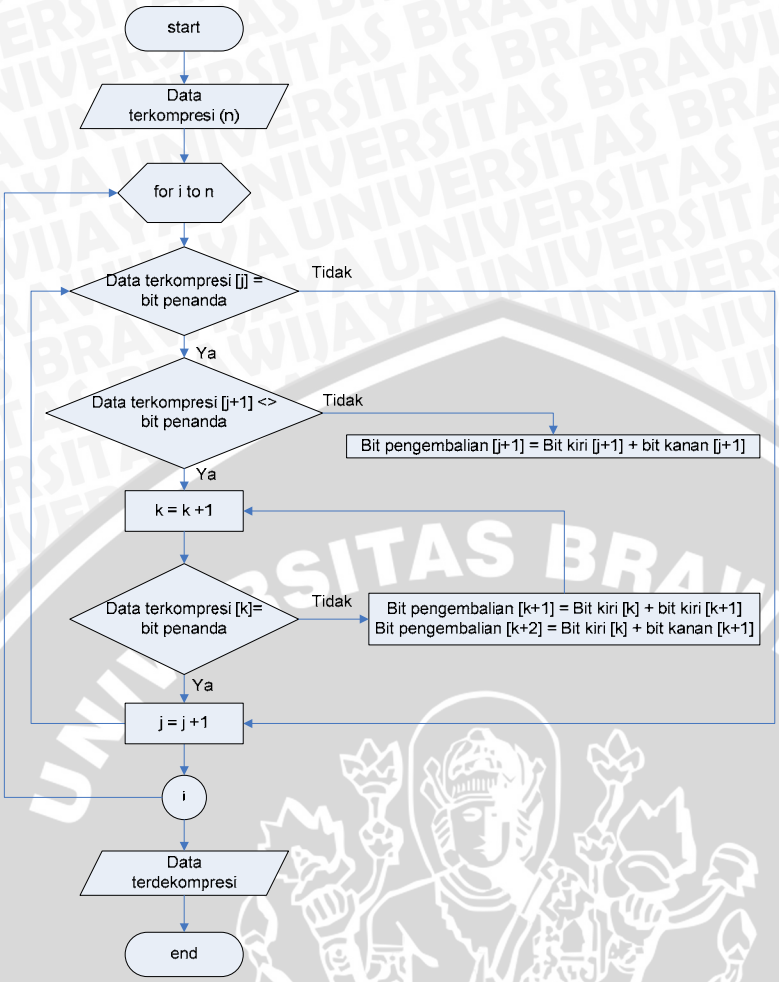
Gambar 3.11 Proses dekompresi data

Dengan menggunakan data pada tabel 3.2, tabel 3.4 merupakan tabel dekompresinya.

Tabel 3.4 Dekompresi *File*

***File* Terkompresi**	**Keterangan**	**Hasil Dekompresi**
ff	mb	62
62	*File* terkompresi	69
9c		6c
1e		61
71		6e
ee		67
ff	mb	61
00		6e
00		6e
01		00
43		00
55		01
ff	mb	43
17	*File* terkompresi	55
be		17
f0		1b
34		1e
15		1f
2d		10
ff	mb	13
30		14
3f		11
60		15
6f		12
6f		1d
		30
		3f
		60
		6f
		6f

Gambar 3.12 merupakan *flowchart* cari pembahasan langkah-langkah proses dekompresi.



Gambar 3.12 *Flowchart* proses dekompresi

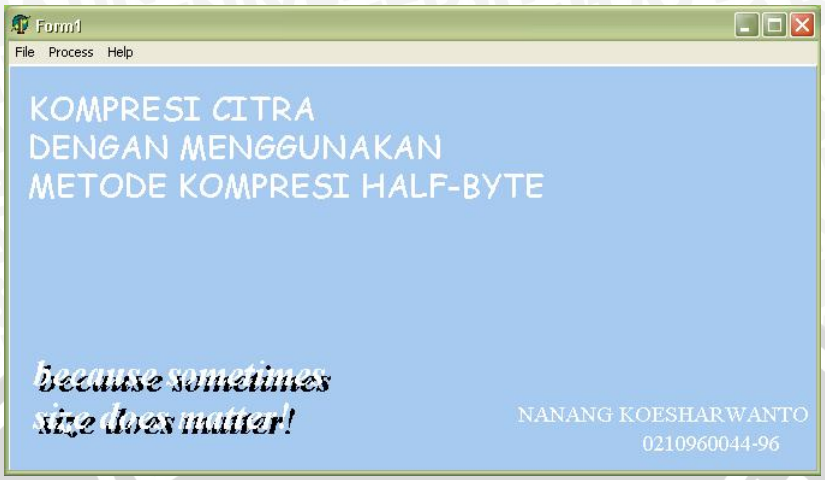
3.3.3 Penyimpanan *File* Terdekompresi

Setelah seluruh data selesai didekompresi, langkah selanjutnya adalah menyimpan data tersebut dalam bentuk aslinya (*image bmp*).

3.4 Rancangan Tampilan

3.4.1 Tampilan Awal

Saat program dijalankan, maka akan muncul tampilan di layar monitor berupa *form* yang ditunjukkan pada Gambar 3.13

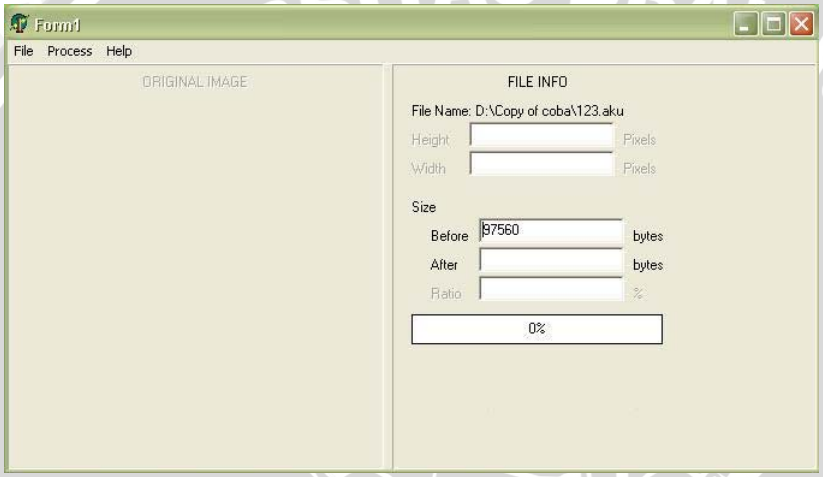


Gambar 3.13 Tampilan Awal

3.4.2 Buka *File*

Hal pertama yang dilakukan dalam perancangan program ini adalah dengan membuka *file* dengan tipe atau format gambar (*.*bmp*) dan tipe terkompresi (*.aku). Saat proses buka *file image* dijalankan maka di layar monitor akan muncul form seperti pada gambar 3.14 dan disertai dengan informasi nama dan lokasi *file* berada, tampilan gambar, panjang dan lebar (bila *file* berbentuk *bmp*) serta ukuran *file*. Pada gambar 3.15 adalah proses buka *file* terkompresi (*.aku). Pada gambar tersebut hanya ditampilkan informasi nama dan lokasi *file* berada serta ukuran *file*.

Gambar 3.14 Form Buka *File Image* (*bmp*)

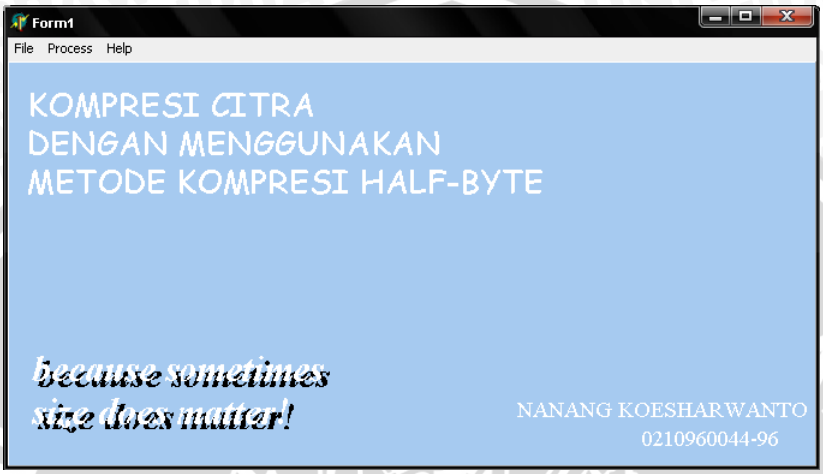


Gambar 3.15 Form Buka *File* Terkompresi (*.aku)

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Implementasi program dibuat dengan menggunakan *Borland Delphi 6.0*. Tampilan utama dari aplikasi kompresi citra dengan menggunakan metode kompresi *Half Byte* dapat dilihat pada Gambar 4.1.

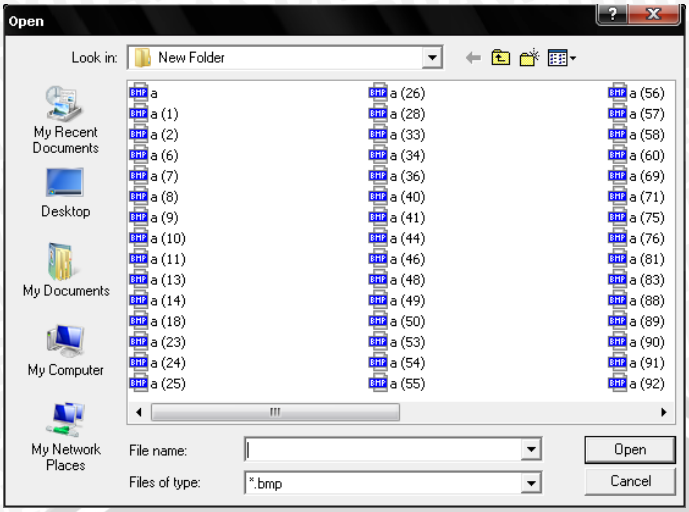


Gambar 4.1 Tampilan utama aplikasi

4.1.1 *Input* Data

Input data terdiri dari data gambar berekstensi **.bmp* atau data *file* terkompresi berekstensi **.aku* serta penjelasan masing-masing data beserta tampilan:

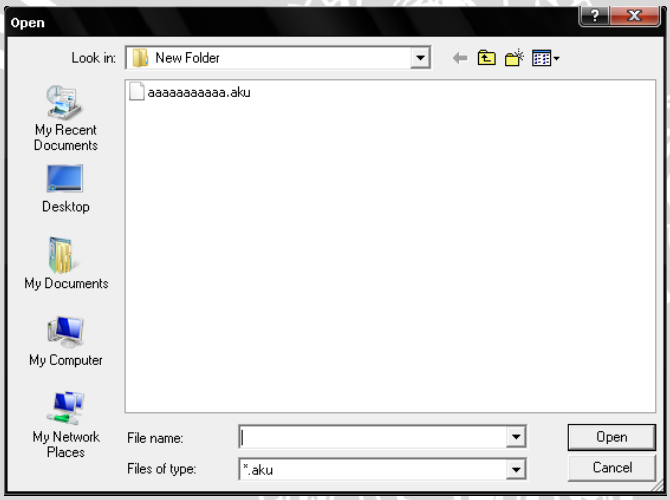
1. Data
   a. Gambar berekstensi **.bmp*
      Form open file image (Gambar 4.2) diatur hanya menampilkan data gambar berekstensi **.bmp* saja. Hal ini dilakukan untuk mempermudah pencarian *file image* yang diperlukan atau yang diinginkan.



Gambar 4.2 *Form Open File Image*

b. *File* terkompresi berekstensi *.aku

Form open file Half Byte (Gambar 4.3) diatur hanya menampilkan data gambar berekstensi *.aku saja. Hal ini dilakukan untuk mempermudah pencarian *file* yang akan didekompresi.



Gambar 4.3 *Form Open File Half Byte*

4.1.2 Deskripsi Program

4.1.2.1 Struktur Data

Struktur data yang ada dalam implementasi ini dapat dilihat pada Gambar 4.4.

```
Type
  pRGBTripleArray = ^TRGBTripleArray;
  TRGBTripleArray = array[word] of TRGBTriple;
  TColor = record
             R,G,B     : real;
           end;
  C                    : array of array of TColor;
  Row1,Row2            : pRGBTripleArray;
  Buf                  : array[1..5000000] of Char;
  BufC                 : array[1..5000000] of Char;
```

Gambar 4.4 Struktur Data

Struktur data pada Gambar 4.4 dijelaskan dalam Tabel 4.1.

Tabel 4.1 Keterangan struktur data

pRGBTripleArray	*Pointer* pada variabel TRGBTripleArray
TRGBTripleArray	Menyimpan data warna *RGB*
TColor	Menyimpan setiap data yang dimiliki oleh masing-masing warna *RGB*
C	Matriks *array* untuk menyimpan data warna
Row1,Row2	Berisi data berurutan berupa pointer variabel TRGBTripleArray
Buf	*Array* yang digunakan untuk menyimpan data kompresi sementara
BufC	*Array* yang digunakan untuk menyimpan data akhir hasil kompresi dan dekompresi

4.1.2.2 Prosedur

Prosedur-prosedur yang digunakan dalam proses kompresi dan dekompresi *file* antara lain prosedur inisialisasi data, pemberian kode data *file*, penentuan *bit* penanda, kompresi *Half Byte*, dekompresi *file*, penyimpanan *file* terkompresi, penyimpanan *file* dekompresi dan data warna bentuk matrik.

4.1.2.2.1 Prosedur Inisialisasi Data

Proses awal yang dilakukan pada data *input* ialah menginisialisasinya dengan cara membaca isi *file* secara keseluruhan dan menyimpannya secara sementara dalam sebuah array. *Evenlist* 4.1 menunjukkan prosedur dari inisialisasi data *file* :

```
1  AssignFile(FromF, OpenDialog1.FileName);
2  Reset(FromF, 1);
3  Fsize := FileSize(FromF);
4  repeat
5  BlockRead(FromF, Buf, SizeOf(Buf), NumRead);
6  until (NumRead = 0);
7  CloseFile(FromF);
```

Evenlist 4.1 Prosedur Inisialisasi Data

4.1.2.2.2 Prosedur Pemberian Kode Data *File*

Pada awal data hasil kompresi diberikan sebuah kode atau tanda berupa karakter ’AKU’, tujuannya sebagai penanda bahwa *file* tersebut adalah dikompresi dengan metode *Half Byte*. *Evenlist* 4.2 menunjukkan prosedur dari pemberian kode data *file* :

```
1  BufC[1]:='A';BufC[2]:='K';BufC[3]:='U';
```

Evenlist 4.2 Prosedur Pemberian Kode Data *File*

4.1.2.2.3 Prosedur Penentuan *Bit* Penanda

Bit penanda (*marker bit*) ditentukan dengan mencari data dalam *file* yang jumlahnya paling sedikit. Kemudian *bit* penanda diletakkan setelah kode ‘AKU’. *Evenlist* 4.3 menunjukkan prosedur dari penentuan *bit* penanda :

```
1    for uu:=0 to 255 do
2    Begin
3      SigmaAscii[uu]:=0;
4    end;
5    SigmaMin:=1000000;
6    for i:=0 to Fsize-1 do
7    begin
8      Inbuf:=integer(Buf[1+i]);
9      inc(SigmaAscii[Inbuf]);
10   end;
11   for uu:=0 to 255 do
12   begin
13     if SigmaAscii[uu]<SigmaMin then
14     begin
15       SigmaMin:=SigmaAscii[uu];
16       BitPenanda:=uu;
17     end;
18   end;
19   BufC[4]:=chr(BitPenanda);
```

Evenlist 4.3 Prosedur Penentuan *Bit* Penanda

4.1.2.2.4 Prosedur Kompresi *Half Byte*

Proses kompresi *Half Byte* diawali membaca data yang telah diinisialisasi, kemudian mencari tujuh atau lebih deretan data yang 4 *bit* sisi kirinya sama. Deretan tersebut kemudian dikompresi dengan dibuka dan ditutup dengan *bit* penanda. Proses berulang hingga akhir data *file*. *Evenlist* 4.4 menunjukkan prosedur dari kompresi *Half Byte*:

```
1    ii:=4;
2    BitKiriS:=(inttohex(Integer(Buf[FSize]),2));
3    Delete(BitKiriS,2,1);
4    if BitKiriS='4' then
5      Buf[FSize+1]:='P'
6    else
7      Buf[FSize+1]:='O';
8    i:=1;
9    SameHB:=1;
10   JadiPenanda:=false;
11   BitKiriS:=(inttohex(Integer(Buf[i]),2));
12   Delete(BitKiriS,2,1);
13   While i<=FSize do
```

```
14    Begin
15      if (SameHB=1) and (Buf[i]=chr(BitPenanda)) then
16      JadiPenanda:=True;
17      BitKananS:=(inttohex(Integer(Buf[i]),2));
18      Delete(BitKananS,1,1);
19      i:=i+1;
20      BitKiriT:=(inttohex(Integer(Buf[i]),2));
21      Delete(BitKiriT,2,1);
22      if (BitKiriT=BitKiriS) and (JadiPenanda=false) then
23        Begin
24          SameHB:=SameHB+1;
25          if (SameHB mod 2)=1 then
26          Begin
27            BitKananT:=(inttohex(Integer(Buf[i]),2));
28            Delete(BitKananT,1,1);
29            if StrtoIntDef('$'+BitKananS+BitKananT,255) =
                BitPenanda then JadiPenanda:=True;
30          end;
31        end
32      else
33        begin
34          i:=i-SameHB;
35          if (SameHB<7) or (JadiPenanda=True) then
36          begin
37            for uu:=i to i+SameHB-1 do
38            begin
39              ii:=ii+1;
40              BufC[ii]:=Buf[uu];
41              if Buf[uu]=chr(BitPenanda) then
42              begin
43                ii:=ii+1;
44                BufC[ii]:=Buf[uu];
45              end;
46            end;
47          end
48          else
49          begin
50            ii:=ii+1;
51            Bufc[ii]:=chr(BitPenanda);
52            ii:=ii+1;
53            Bufc[ii]:=Buf[i];
54            genap:=1;
55            for uu:=i+1 to i+SameHB-1 do
56            begin
57              case genap of
```

```
58              1:
59              begin
60                BitKiriS:=(inttohex(Integer(Buf[uu]),2));
61                Delete(BitKiriS,1,1);
62                genap:=0;
63              end;
64              0:
65              begin
66                BitKiriT:=(inttohex(Integer(Buf[uu]),2));
67                Delete(BitKiriT,1,1);
68                ii:=ii+1;
69                Bufc[ii]:=char(StrtoIntDef('$'+BitKiriS+
                                 BitKiriT,255));
70                genap:=1;
71              end;
72              end;
73            end;
74            ii:=ii+1;
75            Bufc[ii]:=chr(BitPenanda);
76            if genap=0 then
77            begin
78              ii:=ii+1;
79              Bufc[ii]:=char(Integer(Buf[i+SameHB-1])) ;
80            end;
81          end;
82          i:=i+SameHB;
83          SameHB:=1;
84          JadiPenanda:=false;
85          BitKiriS:=(inttohex(Integer(Buf[i]),2));
86          Delete(BitKiriS,2,1);
87        end;
88    end;
89    FCBSize:=ii;
90    FCSaveSize:=FCBSize;
```

Evenlist 4.4 Prosedur Kompresi *Half Byte*

4.1.2.2.5 Prosedur Dekompresi *File*

Dekompresi *file* juga diawali dengan membaca data yang telah diinisialisasi, kemudian mencari *bit* penanda. Jika ditemukan, maka data selanjutnya adalah data yang terkompresi, dan *bit* penanda kedua merupakan *bit* penutup. Pada data yang terkompresi dilakukan

pemisahan 4 *bit* kiri dan kanannya sehingga didapatkan *file* aslinya. *Evenlist* 4.5 menunjukkan prosedur dari dekompresi *file* :

```
1     i:=0;
2     BitPenanda:=Integer(Buf[4]);
3     ii:=4;
4     Repeat
5       inc(ii);
6       If Integer(Buf[ii])=BitPenanda then
7       begin
8         If Integer(Buf[ii+1])=BitPenanda then
9         begin
10          inc(i);
11          BufC[i]:=chr(BitPenanda);
12          inc(ii);
13        end
14        else
15        begin
16          inc(i);
17          BufC[i]:=Buf[ii+1];
18          BitKiri:=(inttohex(Integer(Buf[ii+1]),2));
19          Delete(BitKiri,2,1);
20          ii:=ii+2;
21          while (Integer(Buf[ii])<>BitPenanda)
22                and (ii<FSize) do
23          begin
24            BitKanan:=(inttohex(Integer(Buf[ii]),2));
25            Delete(BitKanan,2,1);
26            inc(i);
27            BufC[i]:=chr(StrtoIntDef('$'+BitKiri+
28                  BitKanan,255));
29            BitKanan:=(inttohex(Integer(Buf[ii]),2));
              Delete(BitKanan,1,1);
30            inc(i);
31            BufC[i]:=chr(StrtoIntDef('$'+BitKiri+
32                  BitKanan,255));
33            inc(ii);
34          end;
35        end;
36      end
37      else
38      begin
39        inc(i);
40        BufC[i]:=Buf[ii];
41      end;
```

```
42      Until ii>=FSize;
43      FCSize:=i;
44      end;
45  end;
```

Evenlist 4.5 Prosedur Dekompresi *File*

4.1.2.2.6 Prosedur Penyimpanan *File* Terkompresi

Data *file* yang telah dikompresi disimpan dalam array, kemudian data array tersebut disimpan dalam *file* yang diberi ekstensi tertentu, dalam penelitian ini menggunakan ekstensi *.aku. *Evenlist* 4.6 menunjukkan prosedur dari penyimpanan data *file* terkompresi:

```
1     begin
2       AssignFile(ToF, SaveDialog1.FileName);
3       Rewrite(ToF);
4      for ii:=1 to FCSaveSize do
5       begin
6         Write(ToF, bufC[ii]);
7       end;
8       CloseFile(ToF);
9     end;
```

Evenlist 4.6 Prosedur Penyimpanan *File* Terkompresi

4.1.2.2.7 Prosedur Penyimpanan *File* Dekompresi

Proses penyimpanan data hasil dekompresi disimpan dalam bentuk aslinya, yakni dalam bentuk gambar *bmp*. *Evenlist* 4.7 menunjukkan prosedur dari penyimpanan *file* dekompresi :

```
1     begin
2       AssignFile(ToF,SaveDialog2.FileName);
3       Rewrite(ToF);
4      for ii:=1 to FCSize do
5       begin
6         Write(ToF, bufC[ii]);
7       end;
8       CloseFile(ToF);
9     end;
```

Evenlist 4.7 Prosedur Penyimpanan *File* Dekompresi

4.1.2.2.8 Prosedur Data Warna Bentuk Matrik

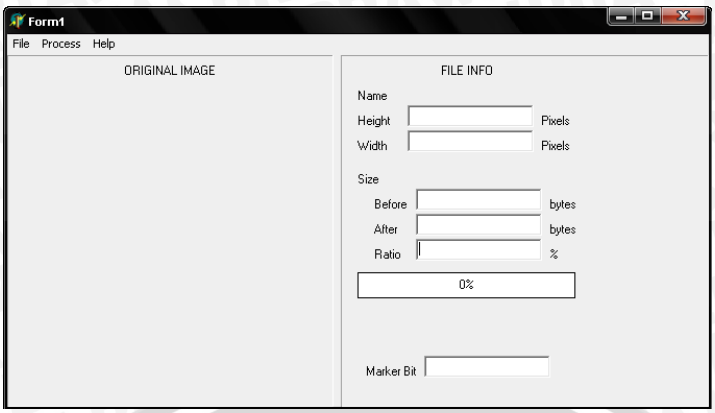
Proses penampilan data warna dalam bentuk matrik diawali dengan mengambil nilai warna ditiap piksel, yang kemudian disimpan dalam array. Kemudian ditampilkan dalam memo berbentuk matrik. *Evenlist* 4.8 menunjukkan prosedur data warna dalam bentuk matrik :

```
1     for x:=0 to N-1 do
2     begin
3         Row1 := BitMap1.ScanLine[x];
4         for y:=0 to M-1 do
5         begin
6             C[x][y].R:=Row1[y].RGBtRed;
7             C[x][y].G:=Row1[y].RGBtGreen;
8             C[x][y].B:=Row1[y].RGBtBlue;
9             RR := RR + FloattoStr(C[x][y].R)+ '     ';
10            GG := GG + FloattoStr(C[x][y].G)+ '     ';
11            BB := BB + FloattoStr(C[x][y].B)+ '     ';
12        end;
13        Memo1.LInes.Add (RR);
14        RR := '';
15        Memo2.LInes.Add (GG);
16        GG := '';
17        Memo3.LInes.Add (BB);
18        BB := '';
19    end;
```

Evenlist 4.8 Prosedur Data Warna Bentuk Matrik

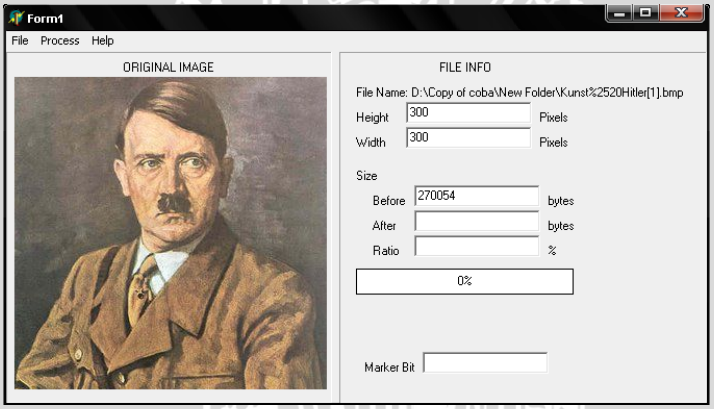
4.2 Pembahasan

Pada pembahasan ini**,** akan diuji cobakan dan dianalisa aplikasi yang telah dibuat. Gambar 4.5 adalah tampilan aplikasi setelah tampilan utama aplikasi dijalankan :



Gambar 4.5 Tampilan aplikasi

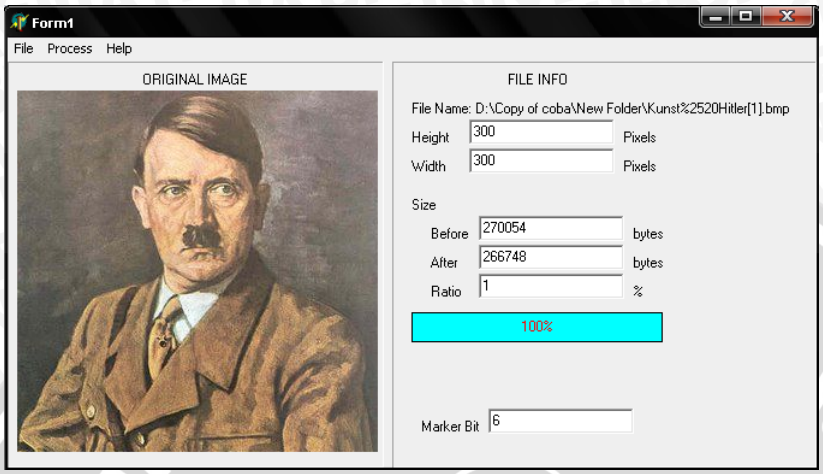
Langkah yang dilakukan selanjutnya adalah dengan mengklik menu *Open* yang terdapat dalam menu *File*. Dalam menu *Open*, terdapat dua sub menu, *Half Byte File* dan *Image File*. *Half Byte File* digunakan untuk membuka *file* terkompresi dengan metode *Half Byte* dan kemudian mendekompresi untuk mendapatkan *file* aslinya. Sedang *Image File* (ditunjukkan pada gambar 4.6) digunakan untuk mengambil gambar yang akan dikompresi. *Image File* akan dibahas terlebih dahulu.



Gambar 4.6 Tampilan *Open Image File*

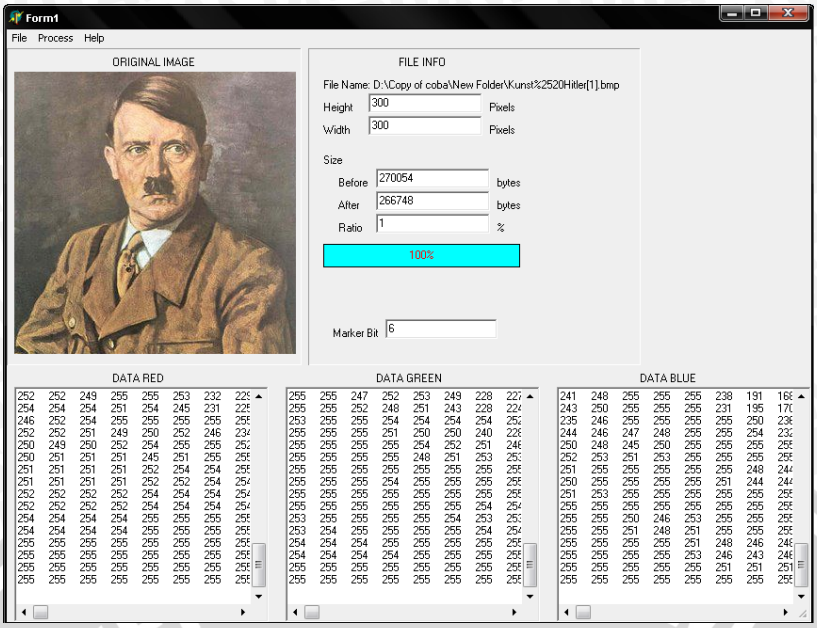
Setelah *Image File* diklik dan dipilih suatu gambar. Kemudian akan tampil gambar yang dipilih beserta informasi resolusinya (tinggi dan panjang) serta ukuran *file*nya.

Langkah selanjutnya ialah mengkompresinya dengan cara mengklik *Compress* dalam menu *Process*. Setelah diklik maka proses kompresi dilakukan dan hasilnya ditunjukkan pada gambar 4.7.



Gambar 4.7 Hasil Proses Kompresi

Gambar 4.8, menunjukkan adanya tambahan informasi berupa nilai warna *Red*, *Green* dan *Blue*. Untuk menampilkannya dengan menekan "*color value*" pada tab "*Process*".



Gambar 4.8 Nilai warna *RGB*

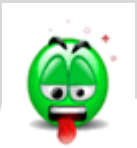
Setelah gambar dikompresi, selain kita mengetahui ukuran *file* terkompresi, kita juga dapat mengetahui nilai warna *RGB* ditiap piksel gambar. Untuk penyimpanan *file*, caranya dengan mengklik *Save As* → *Half Byte File* dalam menu *File* dan disimpan dengan nama *file* bebas.

Pada proses dekompresi *file* juga diawali dengan gambar 4.5. Kemudian klik *Open* → *Half Byte File* dalam menu *File*. Setelah dipilih *file* yang akan dikompresi, maka tampilan akan berisi hanya ukuran *file* tersebut. Kemudian klik *Decompress* dalam menu Process dan informasi bertambah dengan munculnya data ukuran *file* aslinya. Selanjutnya kita simpan dalam format *bmp* dengan nama *file* bebas.

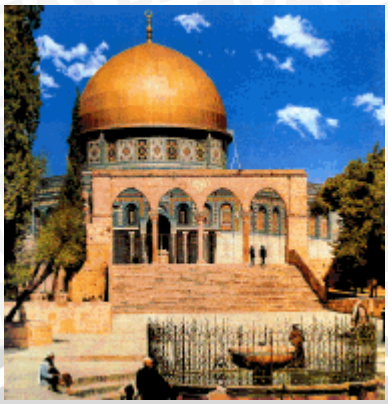
4.3 Analisa Hasil

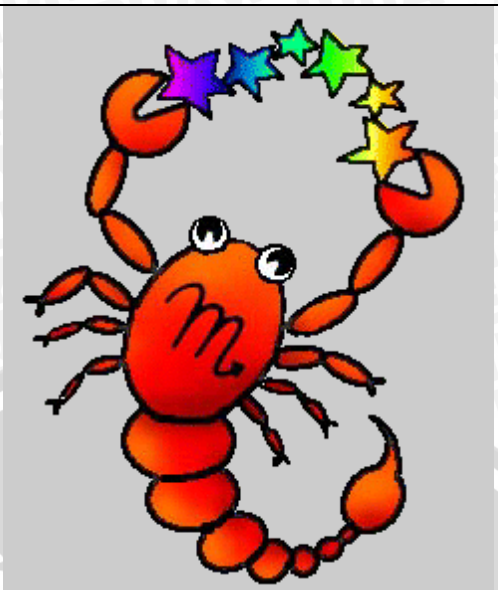
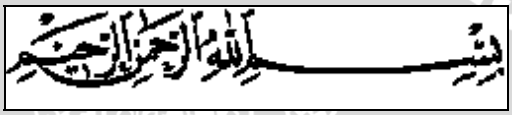
Pada analisa hasil akan dilakukan uji coba kompresi *file* dan mendekompresi kembali *file* yang telah terkompresi dengan menggunakan gambar dengan berbagai ukuran *file* dan resolusi. Dalam uji coba yang dilakukan, diujikan 12 buah gambar yang ditunjukkan dalam tabel 4.2.

Tabel 4.2 Daftar sampel gambar

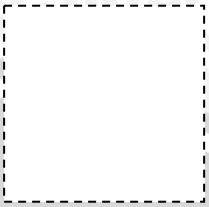
Nama *File*, Resolusi,	**Gambar**
Animasi 66 x 72	
George Bush 151 x 200	

Kaca 176 x 297	
Bendera 184 x 250	

Masjid 190 x 197	
Kaligrafi 210 x 300	

Scorpio 246 x 294	
Basmallah 254 x 51	

Cat 300 x 288	
Animal 300 x 300	

Hitam 100 x 100	
Putih 100 x 100	

Tabel 4.3 adalah tabel hasil dari uji coba dengan menggunakan sampel gambar pada tabel 4.2 dan menyimpannya dengan nama sama dengan *file* aslinya namun dengan ekstensi *.aku:

Tabel 4.3 Hasil uji coba kompresi sampel gambar

File Name (*.bmp)	***Resolution***	***Original File Size***	***Compressed File Size***	***Ratio***
Animasi	66 x 72	19.062	13.076	31
Hitam	100 x 100	30.054	15.052	50
Putih	100 x 100	30.054	15.055	50
Masjid	190 x 197	38.902	35.374	9
Basmallah	254 x 51	39.018	22.029	44
Scorpio	246 x 294	73.992	50.410	32
George Bush	151 x 200	91.254	87.408	4
Bendera	184 x 250	138.056	135.599	2
Kaca	176 x 297	156.870	97.560	38
Kaligrafi	210 x 300	189.654	117.519	38
Cat	300 x 288	259.254	249.090	4
Animal	300 x 300	270.054	260.928	3

Dari Tabel 4.3 dapat dilihat nilai ukuran *file* terkompresi beserta rasionya dari masing-masing gambar.

Gambar animasi, hitam, putih, basmallah, scorpio, kaca dan kaligrafi mendapatkan rasio yang cukup besar, yakni (berurutan) 31, 50, 50, 44, 32, 38 dan 38. Sedangkan gambar masjid, george bush, bendera, cat dan animal hanya memiliki rasio yang kecil, yakni kurang dari 10.

Rasio yang besar pada gambar animasi, hitam, putih, basmallah, scorpio, kaca dan kaligrafi didapatkan karena gambar-gambar tersebut memiliki sedikit variasi nilai warna yang sama yang berurutan. Sedangkan pada gambar masjid, george bush, bendera, cat dan animal variasi nilai warna yang sama yang berurutannya lebih banyak, sehingga menghasilkan rasio yang kecil.

Dengan melihat rasio pada gambar hitam dan putih (gambar yang memiliki satu warna), dapat disimpulkan bahwa, metode *Half Byte* memilki nilai rasio maksimal 50.

Berdasarkan uji coba yang telah dilakukan terhadap 12 gambar sampel, keberhasilan kompresi tidak dipengaruhi oleh resolusi gambar dan ukuran *file* tetapi bergantung terhadap banyaknya variasi nilai warna yang sama yang berurutan.

Tabel 4.4 adalah hasil uji coba proses dekompresi dari ke dua belas *file* gambar yang terkompresi.

Dari Tabel 4.4 diketahui bahwa semua sampel *file* gambar berhasil didekompresi dengan baik, ditunjukkan dengan samanya ukuran *file* dekompresi dengan *file* aslinya.

Tabel 4.4 Hasil uji coba dekompresi

File Name (*.aku)	***Compressed File Size***	***Decompressed File Size***	***Original File Size***
Animasi	13.076	19.062	19.062
George Bush	87.408	91.254	91.254
Kaca	97.560	156.870	156.870
Bendera	135.599	138.056	138.056
Masjid	35.374	38.902	38.902
Kaligrafi	117.519	189.654	189.654
Scorpio	50.410	73.992	73.992
Basmallah	22.029	39.018	39.018
Cat	249.090	259.254	259.254
Animal	260.928	270.054	270.054
Hitam	15.052	30.054	30.054
Putih	15.055	30.054	30.054

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dalam tugas akhir ini dari uji coba yang telah dilakukan adalah :

1. Pada tugas akhir ini telah dibuat model kompresi *Half Byte* untuk gambar *bmp* beserta dekompresinya.
2. Berdasarkan hasil uji coba yang telah dilakukan terhadap 12 gambar sampel (data ditunjukkan pada Tabel 4.3), keberhasilan kompresi dengan menggunakan metode *Half Byte* tidak dipengaruhi oleh resolusi gambar dan ukuran *file* tetapi bergantung terhadap banyaknya variasi nilai warna yang sama yang berurutan.
3. Rasio maksimal yang dapat dicapai dengan menggunakan metode *Half Byte* adalah sebesar 50. Ini ditunjukkan pada rasio gambar hitam dan putih (gambar yang memiliki satu nilai warna) dalam tabel 4.3
4. Berdasarkan Tabel 4.4, dapat diambil kesimpulan bahwa metode *Half Byte* mampu mengembalikan (dekompresi) *file* terkompresi ke dalam format gambar *bmp* sama dengan ukuran *file* aslinya. Ini menunjukkan bahwa metode kompresi *Half Byte* termasuk dalam *Lossless Compression* karena data hasil pengkompresian sama seperti data aslinya atau sumbernya.

5.2 Saran

Saran yang diharapkan dapat menjadi pengembangan penelitian ini :

1. Pengembangan metode *Half Byte* agar mampu :
   - mengkompresi gambar satu warna lebih dari 50%.
   - mendapatkan rasio yang besar saat mengkompresi gambar dengan banyak variasi warna.

DAFTAR PUSTAKA

- Gonzalez, RC. dan Wintz P. 1987. Applied Data Communication. John Willey & Son, Inc. Canada.
- Hayati, Nur. 2004. Studi Banding Teknik Kompresi Metode DCT dan Fraktal. Universitas Gajayana Malang. Malang.
- Kay, David C. & John R. Levine. 1993. Graphics File Formats. Windcrest® & McGraw-Hill. Kanada.
- Mandala, Jani F. 2003. Pemanfaatan Transformasi Wavelet Citra Wajah Sebagai Sistem Keamanan Kunci Kombinasi. Institut Teknologi Bandung. Bandung.
- Nelson, Mark and Gaily LJ. 1996. The Data Compression. M&T Books. New York.
- Proakis, J.G dan Manolakis. 1996. Digital Signal Processing Principles, Algorithms & Application. International Edition.
- Rowe, Lawrence A. 1998. Image/ Video Compression. University of California. California (http://www.BMRC. Berkeley. EDU/~larry).
- Suhono, H. S. 1997. Telekomunikasi Video Conferancing dan MPEG. Stiki. Malang.
- Sujaini, Herry dan Yessi Mulyani. 2000. Pemampatan File. Institut Teknologi Bandung. Bandung.
- Wolfgang, Ray. -----. JPEG Tutorial. IS&T.