BAB II

LANDASAN TEORI

2.1 Pengkodean Data

Data disimpan di dalam komputer pada *main memory* untuk diproses. Sebuah karakter data disimpan dalam *main memory* menempati posisi 1 byte. Komputer generasi pertama, 1 byte terdiri dari 4 bit, komputer generasi kedua, 1 byte terdiri dari 6 bit dan komputer generasi sekarang, 1 byte terdiri dari 8 bit. Suatu karakter data yang disimpan di *main memory* diwakili dengan kombinasi dari digit binary (*binary digit* atau bit) suatu kode biner dapat digunakan untuk mewakili suatu karakter.

Suatu komputer yang berbeda menggunakan kode biner yang berbeda untuk mewakili suatu karakter. Komputer yang 2 byte terdiri dari 4 bit, menggunakan kode binari yang berbentuk kombinasi 4 bit, yaitu *Binary coded decimal* (BCD). Komputer yang menggunakan 6 bit untuk 1 bytenya, menggunakan kode biner yang terdiri dari 6 kombinasi yaitu *Standard Binary Coded Decimal* (SBCDK). Komputer yang 1 byte terdiri 8 bit, menggunakan kode *Decimal Interchange Code* (DIC) atau *American Standard Code of Information Interchange* (ASCII).

2.1.1 *Binary Coded Decimal* (BCD)

BCD merupakan kode biner yang digunakan hanya untuk mewakili nilai digit desimal saja, yaitu angka 0 samapai dengan 9. BCD menggunakan

kombinasi dari 4 bit, sehingga sebanyak 16 (2 pangkat 4 = 16) kemungkinan kombinasi yang dapat diperoleh dan hanya 10 kombinasi yang digunakan.

Kode BCD yang orisinil sudah jarang dipergunakan untuk komputer generasi sekarang karena tidak dapat mewakili huruf atau simbol-simbol karakter khusus.

2.1.2 *Standard Binary Coded Decimal Interchange Code* (SBCDIC)

SBCDIC merupakan kode biner perkembangan dari BCD. BCD dianggap tanggung, karena masih ada 6 kombinasi yang tidak dipergunakan, 6 kombinasi yang tersebut tidak dapat digunakan untuk mewakili karakter yang lainnya. SBCDIC menggunakan kombinasi 6 bit, sehingga lebih banyak kombinasi yang bisa dihasilkan, sebanyak 64 kombinasi kode, yaitu 10 kode untuk digit angka, 26 kode untuk huruf alphabet dan sisanya karakter-karakter khusus yang dipilih.

Posisi bit di SBCDIC dibagi menjadi 2 area, yaitu 2 bit pertama disebut dengan *alphabet position* dan 4 bit berikutnya disebut dengan *numeric bit position***.**

2.1.3 *Extended Binary Coded Decimal Interchange Code* (EBCDIC)

EBCDIC terdiri dari kombinasi 8 bit yang memungkinkan untuk mewakili karakter sebanyak 256 kombinasi karakter. Pada EBCDIC, *high order* bits atau 4 bit pertama disebut dengan zone bit dan *low order* bits atau 4 bit kedua disebut dengan *numeric bits*.

2.1.4 *American Standard Code For Information Interchange* (ASCII) 7 bit

Kode ASCII yang standar menggunakan kombinasi 7 bit, dengan kombinasi sebanyak 127 dari 128 kemungkinan kombinasi.

Kode ASCII 7 bit terdiri dari dua bagian, yaitu *control characters* dan *information characters*. *Control characters* merupakan karakter-karakter yang digunakan untuk mengontrol pengiriman atau transmisi dari data, sedangkan *information characters* merupakan karakter-karakter yang mewakili data.

2.1.5 *American Standard Code For Information Interchange* (ASCII) 8 bit

Karakter-karakter grafik yang tidak dapat diwakili oleh ASCII 7 bit, dapat diwakili dengan ASCII 8 bit karena lebih banyak memberikan kombinasi karakter.

2.2 Kompresi Data

Kompresi data adalah metode pengkodean data untuk merepresentasikan data dengan panjang atau jumlah bit yang lebih kecil dibandingkan dengan kode aslinya, sehingga lebih efisien. Kompresi data merupakan suatu cara yang dipakai dalam sistem komputer untuk menyusutkan ukuran suatu data atau file menjadi lebih kecil dari ukuran sebenarnya, sehingga kompresi ini dapat menjadi solusi untuk mengatasi penyimpanan dan pengiriman data dalam ukuran yang besar karena dapat mempersingkat waktu pengiriman dan menghemat penggunaan media penyimpanan. Pada proses kompresi sekumpulan data ditransformasikan dengan algoritma atau teknik tertentu sehingga menghasilkan data yang berukuran kecil dari ukuran aslinya.

Pada akhir 40-an dimana dimulainya tahun Teori Informasi, ide pengembangan metode coding yang efisien baru dimulai dan dikembangkan. Dimulainya penjelajahan Ide dari entropy, information content dan redundansi. Salah satu ide yang popular adalah apabila probabilitas dari simbol dalam suatu pesan diketahui, maka terdapat cara untuk mengkodekan simbol, sehingga pesan memakan tempat yang lebih kecil.

Model pertama yang muncul untuk kompresi sinyal digital adalah *Shannon Fano Coding*. *Shannon* dan *Fano* (1948) terus-menerus, mengembangkan algoritma ini yang menghasilkan *codeword* biner untuk setiap simbol (unik) yang terdapat pada data file.

Huffman coding (1952) memakai hampir semua karakteristik dari *Shannon Fano coding*. *Huffman coding* dapat menghasilkan kompresi data yang efektif dengan mengurangkan jumlah redundansi dalam mengkoding simbol. Telah dapat dibuktikan, bahwa *Huffman Coding* merupakan metode *fixed-length* yang paling efisien.

Pada limabelas tahun terakhir, *Huffman Coding* telah digantikan oleh *arithmetic coding*. *Arithmetic coding* melewatkan ide untuk menggantikan sebuah simbol masukan dengan kode yang spesifik. Algoritma ini menggantikan sebuah aliran simbol masukan dengan sebuah angka keluaran *single floating-point*. Lebih banyak bit dibutuhkan dalam angka keluaran, maka semakin rumit pesan yang diterima.

Algoritma Dictionary-based compression menggunakan metode yang sangat berbeda dalam mengkompres data. Algoritma ini menggantikan string

variable-length dari simbol menjadi sebuah *token*. *Token* merupakan sebuah indek dalam susunan kata di kamus. Apabila *token* lebih kecil dari susunan kata, maka *token* akan menggantikan *prase* tersebut dan kompresi terjadi.

Kompresi data atau lebih dikenal sebagai pemampatan data umumnya diterapkan pada mesin komputer, hal ini dilakukan karena setiap simbol yang dimunculkan pada komputer memiliki nilai bit-bit yang berbeda. Misal pada ASCII setiap simbol yang dimunculkan memiliki panjang 8 bit, misal kode A pada ASCII mempunyai nilai desimal = 65, jika dirubah dalam bilangan biner menjadi 010000001. Pemampatan data digunakan untuk mengurangkan jumlah bit-bit yang dihasilkan dari setiap simbol yang muncul. Dengan pemampatan ini diharapkan dapat mengurangi (memperkecil ukuran data) dalam ruang penyimpanan.

Saat ini terdapat berbagai tipe algoritma kompresi, antara lain :

Huffman, Arithmetic Coding, LIFO, LZHUF, LZ77 dan variannya (LZ78, LZW, GZIP), Dynamic Markov Compression (DMC), Block-Sorting Lossless, Run-Length, Shannon-Fano, PPM (Prediction by Partial Matching), Burrows-Wheeler Block Sorting dan Half Byte.

Suatu teknik kompresi data dapat dinilai dari kinerjanya, yang pada umumnya ditinjau dari 2 segi, yaitu rasio kompresi dan kecepatan waktu prosesnya. Rasio kompresi adalah ukuran kuantitas data setelah proses kompresi dibandingkan dengan kuantitas aslinya.

Rasio Kompresi (R) = ukuran data hasil kompresi asli / ukuran data Asli

Dari persamaan diatas dapat dilihat bahwa makin besar nilai rasio kompresinya, maka makin efektif metode kompresi tersebut.

Penilaian terhadap suatu metode kompresi berdasarkan nilai rasio kompresinya, secara garis besar memenuhi kriteria sebagai berikut :

1. metode kompresi dikatakan efektif apabila rasio kompresi data lebih besar dari 1 (R > 1) yang berarti tujuan dari proses kompresi telah tercapai karena ukuran data nilai terkompresinya jauh lebih kecil dari pada ukuran data aslinya.
2. metode tersebut dikatakan gagal jika nilai ($R \leq 1$) yang menunjukan bahwa ukuran data yang terkompresi sama (R=1) atau lebih besar (R<1) dari pada data aslinya.

Sedangkan kecepatan waktu proses dihitung dari lama proses pengkodean persimbol.

2.2.1 Faktor Penting Kompresi Data

Dalam kompresi data, terdapat empat faktor penting yang perlu diperhatikan, yaitu:

1. *Time Process* merupakan waktu yang dibutuhkan dalam menjalankan proses,
2. *Completeness* merupakan kelengkapan data setelah file-file tersebut dikompres,
3. *Ratio Compress* merupakan ukuran data setelah dilakukan kompresi,

4. *Optimaly* merupakan perbandingan apakah ukuran file sebelum dikompres sama atau tidak sama dengan file yang telah dikompres. Tidak ada metode kompresi yang paling efektif untuk semua jenis file.

2.2.2 Metode Kompresi Data

Pada umumnya kompresi data terdiri dari proses pengambilan stream simbol dan transformasikan menjadi kode. Jika kompresinya efektif stream kode hasilnya akan lebih kecil dari simbol aslinya. Keputusan untuk mengeluarkan kode tertentu untuk simbol didasarkan pada model.

Teknik kompresi data yang modern menggunakan pengkodean yang berdasarkan model dimana dihasilkan deretan kode yang merupakan bentuk terkompresi dari deretan simbol masukkan (*input*).

Secara sederhana model dapat diartikan sebagai sekumpulan data dan aturan yang digunakan untuk memproses simbol *input* dan menentukkan kode mana yang akan dikeluarkan sebagai *outputnya*. Dengan kata lain, model adalah cara untuk menghitung distribusi peluang simbol masukkan dalam konteks tertentu. Suatu program menggunakan model untuk menentukan secara akurat probilitas dari setiap simbol dan kode untuk menghasilkan kode yang tepat berdasarkan probabilitas tersebut,

Kompresi dapat diperoleh dengan membentuk bit yang lebih sedikit bagi simbol-simbol yang nilai peluangnya lebih besar dari pada yang bernilai peluang lebih kecil.

Model yang dikenal adalah model statis dan model adaptif, pada model statis peluang bagi tiap simbol dalam alphabet telah ditentukan terlebih dahulu, peluang ini dapat ditentukan dengan menghitug frekuensi dalam teks contoh yang dianggap dapat mewakili, pada awal pengiriman, model statis ini dikompresikan terlebih dahulu antara *encoder* dan *decoder*, dan selanjutnya yang dapat digunakan untuk banyak pesan.

Pada model adaptif, peluang bagi simbol-simbol tersebut dapat berubah berdasarkan frekuensi simbol yang mengetahui dari simbol-simbol masukkan terlebih dahulu sebagaimana model statis, baik *encoder* dan *decoder* mengetahui model yang akan digunakan.

2.3 Teknik Kompresi Data

Teknik kompresi data dapat digolongkan menjadi dua kelompok utama yaitu : *Lossy* dan *Lossless*. Teknik kompresi secara *lossy* dimaksudkan dengan teknik kompresi data dengan menghilangkan ketelitian data utama guna mendapatkan data sekecil mungkin (kompresi data sebesar mungkin). Teknik kompresi data secara *lossless* yaitu teknik kompresi data dengan mengurangkan jumlah data yang terjadi duplikasi (memiliki simbol yang sama) sebelum terjadi kompresi.

2.3.1 Lossy Compression

Teknik kompresi lossy ini tidak menghilangkan redundansi saja akan tetapi juga menghilangkan atau membuang bit-bit yang kurang dominan atau

dianggap kurang perlu sehingga akan menghasilkan jumlah bit yang kecil tentu saja ini tidak bisa dilakukan pada file teks atau program, karena setiap bagian file ini penting.

Kompresi lossy umumnya dilakukan pada data multimedia, seperti gambar, file suara, atau video. Sebagai contoh pada file suara (audio), dengan membuang data suara dengan frekuensi diluar selang 20 Hz – 20 KHz, cukup banyak pengurangan ukuran yang dicapai. Telinga manusia tidak dapat mendengarkan suara diluar selang tersebut sehingga data yang dibuang memang tidak ada gunanya.

Begitu juga dengan file gambar, karena file gambar tidak diutamakan ketelitian tetapi visualisasi objek yang dilihat mata manusia. Maka teknik kompresi lossy ini cocok digunakan file gambar.

Tingkat kualitas kompresi lossy dapat diatur sesuai tingkat yang dibagikan dimana rasio kompresi akan berkurang bila kualitas sinyal meningkat. Pada kompresi citra dan suara, tingkat kualitas hasil kompresi disesuaikan dengan batas kemampuan penilaian indera manusia.

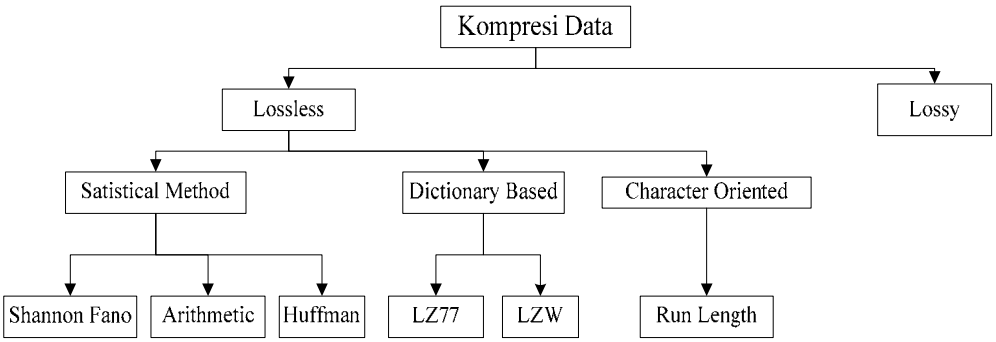
2.3.2 Lossless Compression

Kompresi Lossless digunakan untuk yang tidak bisa mengalami perbedaan atau perubahan dari aslinya, dengan kata lain kompresi lossless digunakan untuk representasi data yang harus bersifat lossless, tidak ada satu pun data dari bit data hilang / rusak, karena kerusakkan 1 bit saja akan mengakibatkan data menjadi tidak berguna. Teknik lossless umumnya akan mencari pola-pola yang terdapat

pada file yang ingin dikompresi, teknik ini akan menghasilkan rasio ukuran yang lebih kecil pada file yang memiliki tingkat pengulangan pola yang sedikit, misalnya karena mengandung banyak informasi yang unik, tidak akan diperoleh rasio yang baik, oleh karena itu teknik ini sangat cocok digunakan untuk mengkompresi data alphabet numreik baik berupa taks ataupun *numeric*.

Kompresi lossless umumnya memiliki rasio kompresi sekitar 50% tergantung jenis file yang dikompres. File teks biasanya memiliki rasio kompresi yang cukup tinggi sementara file grafis akan memiliki rasio kompresi yang rendah. Dilain pihak kompresi lossy dapat memiliki hingga 10% dengan hanya sedikit penurunan kualitas.

2.4 Skema Kompresi



Gambar 2.1 skema kompresi data

Secara umum kompresi data lossless umumnya diimplementasikan dengan menggunakan salah satu jenis pemodelan berikut : *Statistical Method*, *Dictionary Based,* dan *Character Oriented*, sekema metode kompresi ini dapat dilihat pada gamabar diatas.

2.4.1 Statistical Method

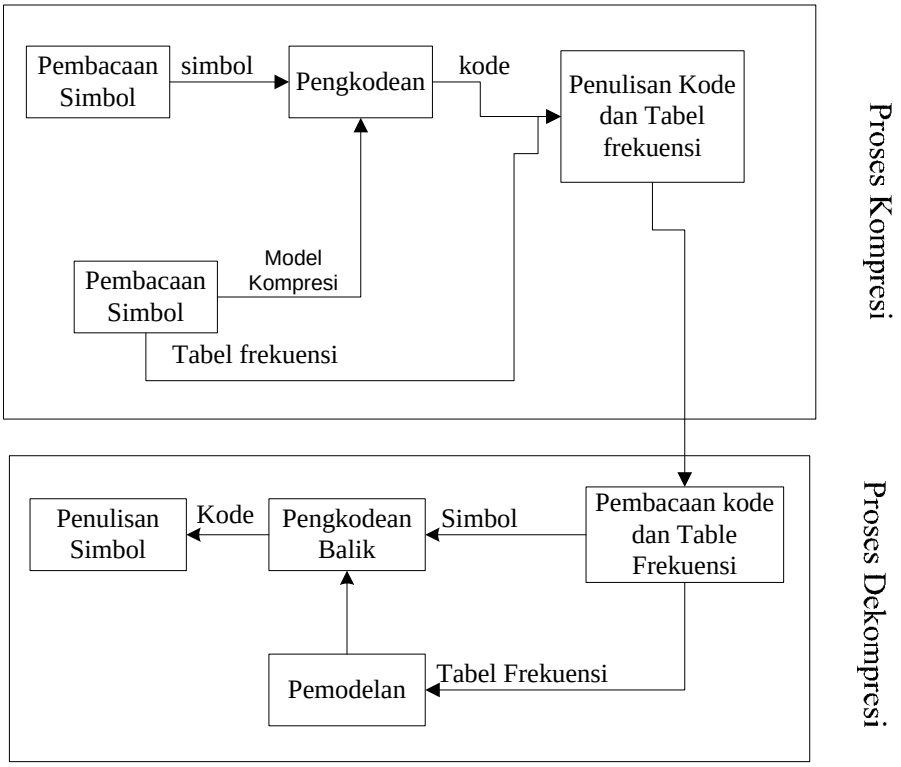
Metode statistik ini adalah suatu cara penyingkatan simbol atau kata dalam teks dengan cara memperpendek simbol/kata yang memliki frekuensi kemunculan terbanyak, karakter yang sering muncul jumlah bitnya dibuat lebih pendek dari yang jarang muncul, seperti pada algoritma *Shannon fano*, metode *Huffman*, serta methode *arithmetic coding*. Pada metode ini, aliran data yang dikompresi akan melewati dua fase yaitu : pemodelan dan pengakodean.

Pemodelan diawali dengan menghitung probabilitas kemunculan tiap simbol pada suatu input, kemudian membentuk suatu table translasi dengan mengikuti aturan sebagai berikut :

1. Tiap-tiap simbol dikodekan dengan kombinasi bit yang berbeda.
2. Simbol dengan probabilitas kemunculan tinggi dikodekan dengan jumlah bit yang lebih sedikit.
3. Walapun panjang kodenya berbeda, kode-kode tersebut dapat didekode secara unik.

Ada dua macam metode pemodelan pada metode statistik[14], yaitu metode pemodelan statis dan pemodelan adatif. Gambar 2.2 menunjukkan proses dari metode pemodelan statis.

Pada proses kompresi dengan model statis ini terdiri dari dua kali pembacaan. Yang pertama adalah membaca seluruh file data yang kemudian akan didapat model berupa table frekuensi kemunculan simbol, kemudian membangun kode-kode berdasarkan model yang ada.

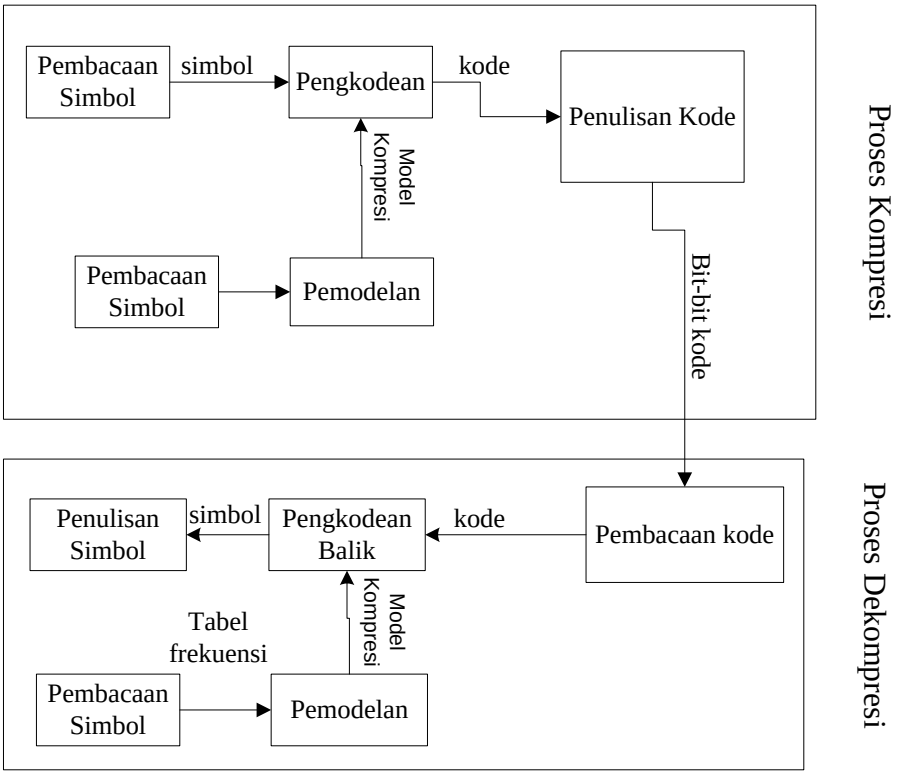


Gambar 2.2 proses kompresi dan dekompresi dengan model stastis

Pembacaan yang kedua adalah membaca satu per satu simbol file data sambil dikodekan berdasarkan kode yang telah terbentuk. Setelah itu model yang telah terbentuk dikirimkan untuk proses dekompresi.

Pada proses dekompresi dilakukan pembacaan bit-bit hasil kompresi sambil mencocokkan dengan table kode yang pada akhirnya akan dihasilkan simbol-simbol file data aslinya.

Sedangkan pada metode pemodelan adaptif, data tidak perlu dibaca terlebih dulu untuk menghitung statistiknya, tetapi statistik itu secara ber kelanjutan diubah setiap pembacaan dn pengkodena sebuah karakter baru.



Gambar 2.3 menunjukkan proses dari metode pemodelan adaptif

2.4.2 Dictionary Based

Teknik ini menggantikan kumpulan string dengan suatu kode yang didapat dari suatu kamus, setelah data dibaca, program mencari kumpulan simbol yang ada pada kamus. Jika suatu string yang cocok ditemukan, sebuah indeks atau pointer akan digunakan untuk mengganti simbol tersebut. Semakin panjang string yang cocok, semakin tinggi rasio kompresinya. Kompresi data dengan menggunakan pendekatan ini ialah suatu cara menyingkat penulisan simbol yang telah dikenal dalam bentuk kode yang lebih kompak seperti mengkodekan alamat suatu daerah dengan kode pos dan penyingkatan judul buku. Pada kompresi data

suatu daerah pendekatan ini selalu diperlukan tabel pengkodean yang cukup besar jumlahnya.

Metode kompresi dengan model dictionary based memiliki kelemahan karena keefektifan model ini sangat bergantung pada beberapa besar dictionary yang digunakan. Contoh pada metode lempel ziv.

2.4.2.1 Metode lempel ziv

Sejak tahun 1980 beberapa skema pendapat data secara umum menggunakan pemodelan statistik (*statistical modeling*). Tetapi pada tahun 1977 dan 1978, Jacob ziv dan Abraham lemvel menggunakan metode pemampatan dengan memakai metode kamus adatif (*adaptive dictionary method*). Dari dua orang tersebut tercipta suatu algoritma pemampatan data yang dinamakan metode lemvel & ziv. Dua algoritma tersebut menggunakan metode berbasis kamus (*Dictionary Based Method*) untuk menciptakan algoritma yang lebih menarik.

Pada metode ini terdapat satu aturan untuk penguraian barisan simbol-simbol dari suatu himpunan alphabet menjadi bangan-bangan yang lebih pendek. Algoritman ini menguraikan pesan sumber menjadi sekumpulan segmen-segmen dengan panjang yang bertambah secara berangsur-angsur.

2.4.3 Character oriented

Kompresi data dengan teknik ini merupakan suatu cara penyingkatan pengkodean simbol dengan melihat sifat-sifat simbol yang akan dikodekan tersebut. Misalnya dengan melihat kecendrungan suatu huruf akan diikuti oleh

huruf lamanya pada suatu teks, atau melihat pola bit yang dikirimkan. Kunci pemilihan karakter khusus ini bergantung pada beberapa faktor, termasuk kode data yang digunakan, jumlah skema kompresi yang digunakan, juga pengetahuan tentang data yang akan ditransmisikan atau disimpan. Dari ketiga faktor diatas yang terpenting adalah kode data yang akan digunakan. Contoh pada kompresi data dengan mengunakan metode *Run Length Encoding*.

2.4.3.1 Run Length Encoding

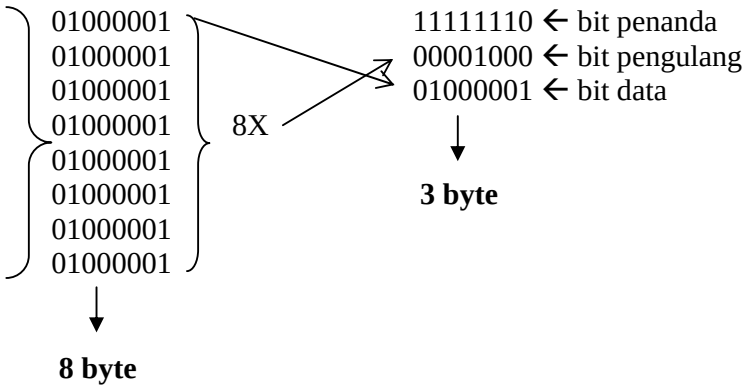
Run Length Encoding atau yang lebih dikenal RLE adalah salah satu algoritma yang termasuk kedalam klasifikasi algoritma lossless.

RLE merupakan algoritma yang sangat bergantung pada skema pengkodean, yang menggantikan *runs* symbol yang sama dengan *runs* yang lebih pendek. Tujuan utamanya adalah membuat file lebih pendek dengan menghilangkan perulangan.

Algoritma *Run length encoding* digunakan untuk memampatkan data yang berisi karakter-karakter berulang. Saat karakter yang sama diterima secara berderet empat kali atau lebih (lebih dari tiga), algoritma ini mengkompres data dalam suatu tiga karakter berderetan. Algoritma *Run Length* paling efektif pada file-file grafis, dimana biasanya berisi deretan panjang karakter yang sama.

Metode yang digunakan pada algoritma ini adalah dengan mencari karakter yang berulang lebih dari 3 kali pada suatu file untuk kemudian diubah menjadi sebuah bit penanda (simbol) diikuti oleh sebuah bit yang memberikan informasi jumlah karakter yang berulang dan kemudian ditutup dengan karakter

yang dikompres, yang dimaksud dengan bit penanda (simbol) disini adalah deretan 8 bit yang membentuk suatu karakter ASCII. Jadi jika suatu file mengandung karakter yang berulang, misalnya AAAAAAAA atau dalam biner 01000001 sebanyak 8 kali, maka data tersebut dikompres menjadi 11111110 00001000 01000001. Dengan demikian kita dapat menghemat sebanyak 5 bytes. Agar lebih jelas algoritma *Run-Length* dapat digambarkan sebagai berikut :



Gambar 2.4 *Algoritma Run Length Encoding*

Deretan data sebelah kiri merupakan deretan data pada file asli, sedangkan deretan data sebelah kanan merupakan deretan data hasil pemampatan dengan algoritma *Run Length Encoding*.

Langkah-langkah algoritma RLE adalah sebagai berikut:

1. Ambil dua byte(atau symbol, angka, dan sebagainya) dari file.
2. Jika byte-byte tersebut sama, keluarkan, berikan karakter (contohnya @) dan byte pertama mulai menghitug dari byte yang kedua.
3. Ulangi untuk mengecek byte selanjutnya, naikkan *counter* jika byte tersebut sama dengan byte pertama atau berhenti jika tidak.

4. Lanjutkan hingga seluruh byte dalam file terkompresi.

Contoh lain misalkan terdapat string ‘ABBBBBBBBCDEEEEF’ (17 byte), maka dengan menggunakan kompresi RLE, file terkompres hanya terdiri dari 10 byte, yaitu A @ 8 B C D @ 4 E F.

Contoh lainnya yaitu, misalkan pada file bitmap (bmp) terdapat byte-byte 12,12,12,12,12,12,12,12,35,76,112,67,87,87,5,5,5,5,5,1,…. Maka dengan menggunakan kompresi RLE, file terkomprenya adalah @812,35,76,112,67,87,87,@55,1,…

String yang berulang dari data digantikan oleh karakter kontrol (@) diikuti oleh jumlah perulangan karakter dan karakter yang berulang tersebut. Karakter kontrol tidak selalu sama dari data satu ke dta lain.

2.5 Sekilas tentang Personal Digital Assistant (PDA) dan Windows CE

2.5.1 Pengertian PDA

PDA merupakan sebuah alat yang dapat digunakan untuk menyimpan sejumlah data dan infomasi penting, yang dapat dibawa kemana-mana karena bentuknya yang sangat praktis (*portable*). PDA merupakan *computer mini* yang juga dilengkapi dengan keyboard dan sebuah pena untuk menulisnya. Dalam mengakses data atau informasi PDA dapat melalui jalur internet, karena PDA itu sendiri sudahdi lengkapi dengan koneksi jaringan, dengan menggunakan sistem nirkabel. Jadi para pengguna PDA dapat mengakses data dan informasi dimana saja berada tanpa dibatasai oleh ruang, jarak dan waktu. Dalam proses penerimaan data dan informasi PDA akan berjalan dalam beberapa sistem, antara lain :

1. PDAs dengan sistem palm
2. PDAs dengan sistem palm V
3. PDAs dengan sistem pocket PC
4. PDAs dengan sistem IE
5. PDAs dengan sistem iPAQ pocket PC

Pada dasarnya bahwa fungsi dan kegunaan PDA sangat luas dan komplek.

2.5.2 Windows CE

Windows CE adalah sebuah Operating System / OS yang dipasang pada komputer saku untuk keperluan penunjang kegiatan sehari hari, bidang penjualan, pendukung kegiatan para professional, dsb. Secara umum kategori Windows CE dibagi menjadi beberapa jenis seperti Pocket PC 2000, Pocket PC 2002, Pocket PC 2003, dan Microsoft Windows CE for Handheld PC Professional Edition.

Beberapa fitur – fitur yang tersedia pada Windows CE pada kegiatan sehari hari seperti :

1. Akses email lewat GPRS
2. Akses internet, LAN, modem atau koneksi RAS
3. Koneksi ke alat lainnya melalui Bluetooth, Infrared dan WiF i

Keunggulan utama dari Windows CE / Pocket PC adalah kompatibilitas dengan Microsoft Windows dalam komunikasi data, penanganan transfer file, sinkronisasi, dan akses database sehingga Pocket PC dengan berbasis Windows CE sudah mulai dipakai pada beberapa PDA (Personal Digital Assistant) terbaru. Source code dan software yang digunakan ini menggunakan Embedded Visual

Basic 3.0. Teknik yang akan dibahas adalah teknik dengan asumsi bahwa programmer/anda sudah menguasai pemrograman OOP/Object Oriented Programming dan dasar dari MFC (Microsoft Foundation Class.

2.5.3 Menggunakan *ActiveSync*

Untuk membuat komunikasi antara Windows CE dan komputer desktop dalam pembuatan software, diperlukan suatu software dari Microsoft yang bernama ActiveSync. Dengan aplikasi tersebut pengguna Pocket PC dapat menginstall/menghapus aplikasi, membackup data, restore data, sinkronisasi email/file/kontak, remote access, dsb.

1. *Standard partnership* **:** Pocket PC akan melakukan sinkronisasi email/file/daftar kontak, dsb secara terus menerus dengan komputer PC/Notebook.
2. *Guest partnership* **:** Pocket PC hanya akan melakukan koneksi dengan komputer/notebook tanpa melakukan sinkronisasi.

Pada halaman ini, pengguna dari Pocket PC dapat melakukan sinkronisasi dengan komputer biasa/notebook atau dengan server yang dilengkapi dengan Microsoft Exchange Server. Jika untuk keperluan pribadi anda dapat memilih pilihan pertama.

a. Pilihan 1 **:** Hanya sinkronisasi dengan komputer yang sedang aktif
b. Pilihan 2 **:** Untuk melakukan sinkronisasi dengan 2 komputer

Dibagian ini, anda dapat memilih fitur apa saja yang akan dilakukan sewaktu Pocket PC melakukan sinkronisasi. Untuk sinkronisasi pertama kali, jangan lupa untuk meng'tick' koneksi sinkronisasi apa yang anda gunakan, jika craddle Pocket PC anda menggunakan kabel Serial COM, pilih *Allow serial cable or infrared connection to this COM port*. Jika menggunakan USB, pilih pilihan yang kedua. Lanjutkan proses instalasi sampai Pocket PC anda sudah selesai sinkronisasi, untuk penanda bahwa Pocket PC anda sudah terpasang dengan benar, anda dapat melihat icon ActiveSync dengan tanda warna hijau pada tray Windows (di pojok kiri bawah).

ActiveSync sangat diperlukan sekali agar Pocket PC tersebut dapat dihubungkan dengan komputer terutama untuk pembuatan aplikasi Pocket PC.

2.5.4 Aplikasi Pendukung Pembuatan Aplikasi Pocket PC

Untuk aplikasi dengan kasus Pocket PC tidak berhubungan dengan alat lain, Software Embedded Visual Basic juga dilengkapi dengan Windows Mobile Emulator yang dapat langsung menjalankan aplikasi yang sudah dibuat. Perlu diketahui bahwa compiler untuk Pocket PC berbeda dengan compiler Emulator, anda tidak dapat menjalankan aplikasi yang dicompile dengan target Emulator di Pocket PC dengan prosessor ARM/MIPS/SH3, demikian juga sebaliknya. Emulator ini berfungsi layaknya seperti PDA yang asli, tetapi tanpa fitur untuk berhubungan dengan alat lainnya yang menggunakan media Bluetooth atau WiFi.

Perangkat pendukung lainnya yang sering digunakan dalam pengembangan aplikasi berbasis Windows CE :

1. Remote File Viewer

   Aplikasi untuk menjelajahi file – file pada Pocket PC anda atau pada emulator.

2. Remote Registry Editor

   Aplikasi untuk melihat/merubah registry pada Pocket PC atau pada emulator.

Perangkat tersebut tersedia pada EVB di menu ‘Tools’. Jadi untuk membuat suatu software Pocket PC pada komputer, ada 3 komponen yang harus dipasang pada komputer anda :

1. Microsoft Visual Basic 3.0
2. Microsoft Pocket PC 2002 SDK

   Software Development kit untuk membuat sotware pada PDA yang berbasis Windows Mobile 2002.

3. Pocket PC 2002 SDK English Emulation Images

   Software Emulasi Windows Mobile 2002.