Vol. 2, No. 1, Maret 2013, ISSN : 2089-9033

IMPLEMENTASI ALGORITMA KOMPRESI LZW PADA DATABASE SERVER

Hidayat[1], Wendi Zarman[2], Tri Pamungkas[3]
[1,2,3]Teknik Komputer Unikom, Bandung
[1]hidayat@unikom.ac.id

ABSTRAK

Paper ini membahas tentang implementasi algoritma kompresi LZW untuk mengkompresi data yang tersimpan dalam *web server*. Hal ini dilakukan untuk memperkecil ukuran data yang akan disimpan di *web server*. Algoritma *LZW* yang dirancang menggunakan bahasa pemrograman *PHP* dan sistem manajemen basis data *MySQL*. Proses kompresi dilakukan pada saat data (artikel) yang disimpan pada *web server* dan didekompresi setiap artikel tersebut akan dibaca.

Algoritma *LZW* yang merupakan *dictionary based compression* sangat efektif mengkompresi data teks yang memiliki banyak pola huruf, kata ataupun kalimat yang berulang, semakin banyak pembendaharaan gabungan string dalam *dictionary* tambahan pada suatu *plaintext* maka semakin baik hasil kompresi *plaintext* tersebut. Hasil implementasi algoritma *LZW* pada *web server* penyimpanan artikel menjadi bertambah banyak, terlihat dari nilai persentase penghematan penyimpanan setiap artikel berkisar antara 11,6656% sampai dengan 16,6656% untuk perhitungan nilai rata-rata persentase penghematan 200 sampai dengan 1.000 karakter pertama dari sepuluh buah artikel acak.

Kata Kunci: informasi, *internet*, kompresi, *LZW*, *dictionary*

1. PENDAHULUAN

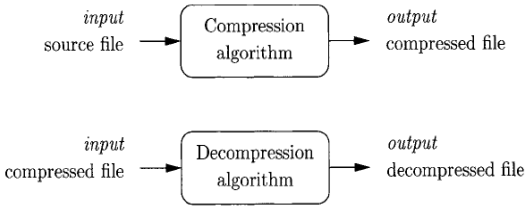
Banyaknya informasi yang disimpan dalam *database server* membutuhkan ukuran media penyimpanan yang besar. Keterbatasan kapasitas media penyimpanan data menyebabkan tidak semua jumlah informasi dapat disimpan pada media penyimpanan data tersebut. Solusi yang dapat dilakukan untuk mengatasi masalah tersebut diantaranya: menghapus beberapa data yang dianggap tidak penting, menambahkan ukuran media penyimpanan data, atau mengkompresi setiap informasi yang disimpan.

Dengan pertimbangan efisiensi biaya dan pentingnya dokumentasi data, maka penggunaan kompresi data dapat dijadikan solusi untuk mengatasi masalah di atas. Implementasi kompresi data memiliki kelebihan yaitu dapat mengurangi konsumsi ruang media penyimpanan data dengan memadatkan ukuran setiap data yang disimpan. Algoritma kompresi yang akan digunakan dalam penelitian ini adaah algoritma kompresi *Lempel-Ziv-Welch* (*LZW*) menggunakan bahasa pemrograman PHP.

2. TEORI PENUNJANG

Teknik Kompresi Data

Kompresi data dalam konteks ilmu komputer adalah suatu ilmu (dan seni) merepresentasikan informasi dalam bentuk yang padat[1].



Gambar 1. Diagram konteks kompresi dan dekompresi data secara umum[1]

Pada diagram konteks di atas ditunjukkan bahwa tahap kompresi adalah masukan berupa *file* asli kemudian dikompresi menggunakan algoritma kompresi tertentu dan menghasilkan *file* terkompresi, sedangkan tahap dekompresi adalah masukan berupa *file* terkompresi lalu didekompresi menggunakan algoritma yang sama ketika melakukan proses kompresi untuk menghasilkan *file* asli. Berdasarkan kebutuhan rekonstruksinya, skema kompresi data dapat dibagi ke dalam dua kelas besar: skema kompresi *lossless* dan skema kompresi *lossy*[2].

a. *Lossless compression*. Kompresi *lossless* umumnya digunakan untuk aplikasi yang tidak bisa mentolerir perbedaan antara data asli dan hasil rekonstruksi[2].
b. *Lossy compression*. data yang telah dikompresi menggunakan metode *lossy* umumnya tidak dapat dipulihkan atau direkonstruksi persis[2].

Adapun beberapa indikator yang perlu diperhatikan dalam setiap teknik kompresi adalah[1]:

a. Rasio kompresi, yaitu perbandingan antara ukuran data asli sebelum terkompresi dengan ukuran data yang telah terkompresi. Angka rasio

kompresi menunjukkan perbandingan ukuran yang dicapai dalam satu proses kompresi, semakin kecil nilai rasio kompresi maka hasil kompresi semakin memuaskan.

$$Rasio\ kompresi = \frac{ukuran\ setelah}{ukuran\ sebelum} \quad (1)$$

b. Faktor kompresi, yaitu: perbandingan antara ukuran data setelah terkompresi dengan ukuran data asli sebelum terkompresi. Angka faktor kompresi menunjukkan berapa kali kehandalan hasil kompresi yang dicapai dalam satu proses kompresi, semakin besar nilai faktor kompresi maka hasil kompresi semakin memuaskan.

$$Faktor\ kompresi = \frac{ukuran\ sebelum}{ukuran\ setelah} \quad (2)$$

c. Persentase penghematan, yaitu: persentase dari perbandingan antara selisih ukuran data dari sebelum terkompresi dan sesudah terkompresi dengan ukuran data sebelum terkompresi. Angka persentase penghematan menunjukkan seberapa besar penghematan yang dicapai dalam satu proses kompresi, semakin besar persentase penghematan maka hasil kompresi semakin memuaskan.

$$\%\ penghematan = \frac{uk.sblm - uk.stlh}{uk.sblm} \times 100\% \quad (3)$$

Algoritma LZW

LZW merupakan algoritma *dictionary based compression*, yaitu algoritma kompresi yang berbasiskan kamus. Pendekatan yang digunakan adalah mengidentifikasi adanya pola perulangan karakter[1]. Kelebihan teknik kompresi *LZW* adalah kecepatan waktu kompresi yang sangat singkat dengan tingkat kompresi yang cukup baik, yaitu persentase penghematan mencapai sekitar 60.2% ± 28.9[3]. Berikut ini adalah *pseudocode* proses kompresi dan dekompresi pada *LZW* secara umum[1]:

Algorithm 7.1 LZW encoding

```
1:  word ← ''
2:  while not EOF do
3:     x ← read_next_character()
4:     if word + x is in the dictionary then
5:        word ← word + x
6:     else
7:        output the dictionary index for word
8:        add word + x to the dictionary
9:        word ← x
10:    end if
11: end while
12: output the dictionary index for word
```

Algorithm 7.2 LZW decoding

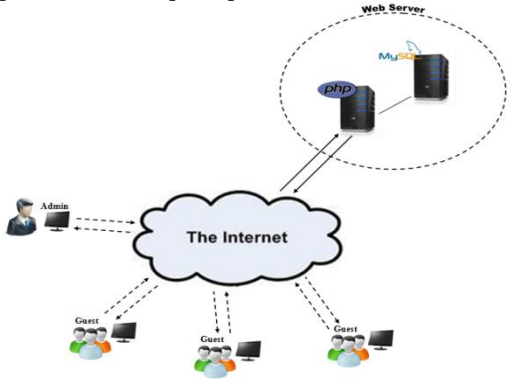
```
1:  read a token x from the compressed file
2:  look up dictionary for element at x
3:  output element
4:  word ← element
5:  while not EOF do
6:     read x
7:     look up dictionary for element at x
8:     if there is no entry yet for index x then
9:        element ← word + firstCharOfWord
10:    end if
11:    output element
12:    add word + firstCharOfElement to the dictionary
13:    word ← element
14: end while
```

Sistem Manajemen Basis Data MySQL

Data Base Management System (DBMS) adalah sekumpulan program yang digunakan untuk mendefinisikan, mengatur administrasi, dan memproses basis data dan aplikasi-aplikasi yang terkait dengan basis data. Sebuah DBMS adalah perangkat yang digunakan untuk membuat struktur dan beroperasi pada data yang berada di antara basis data[4]. MySQL adalah sebuah sistem manajemen basis data relasional yang digunakan pada arsitektur *client/server*. MySQL menjadi sistem basis data *open source* yang paling populer dan paling sukses di dunia. Popularitas ini sebagian besar dikarenakan karena kehandalan, kinerja, dan kemudahan penggunaannya [5]. Sebuah *RDBS* (Sistem basis data relasional) adalah penyimpanan data dan layanan pengambilan data berdasarkan model relasional data, seperti yang diusulkan oleh E.F. Codd pada tahun 1970. Sistem ini adalah mekanisme penyimpanan standar untuk data terstruktur[5].

3. PERANCANGAN

Pada penelitian ini dirancang implementasi teknik kompresi data LZW yang diterapkan pada *webserver*. Pada Gambar 2. ditunjukkan arsitektur implementasi kompresi pada *webserver*.



Gambar 2. Arsitektur implementasi.

Artikel yang diunggahkan oleh *User admin* akan dikompresi pada *webserver* untuk kemudian hasil kompresi tersebut disimpan dalam *database server*. Pada saat tejadi pembacaan artikel, maka

artikel dalam bentuk hasil kompresi akan didekompresi hingga muncul dalam bentuk teks aslinya.

Dictionary LZW

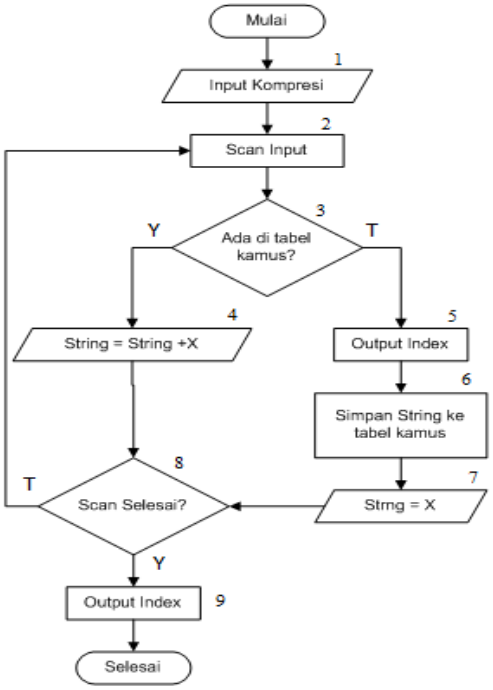
Dictionary LZW merupakan sekumpulan *record* kamus yang dibutuhkan pada perancangan sistem kompresi menggunakan algoritma *LZW*. Penentuan urutan *record* kamus sangatlah penting, karena urutan *record* kamus mempengaruhi hasil keluaran kompresi. Pengurutan didasarkan dari urutan karakter dasar yang paling umum hingga karakter yang jarang digunakan. Adapun tabel *dictionary LZW* ditunjukkan pada Tabel 1 (Lampiran). Jumlah *record* yang dibuat berjumlah 94 indeks. Data *Record* ini memuat karakter-karakter yang sering digunakan.

Pengkodean UTF-8 dan Character Set

Pengkodean *UTF-8* merupakan suatu proses mengkodekan hasil keluaran kompresi yang berbentuk bilangan heksadesimal, menjadi suatu urutan biner. Selanjutnya urutan biner tersebut ditranslasikan dalam bentuk karakter. Fungsi pengkodean *UTF-8* ini adalah agar sistem dapat melakukan *encoding* suatu nilai menjadi urutan biner dan *decoding* urutan biner tersebut menjadi nilai tertentu. Adapun tabel translasi karakter pengkodean UTF-8 ini ditunjukkan pada Tabel 2 (Lampiran).

Diagram Alir Kompresi LZW

Gambar 3. menunjukkan diagram alir kompresi LZW yang diterapkan pada sistem.



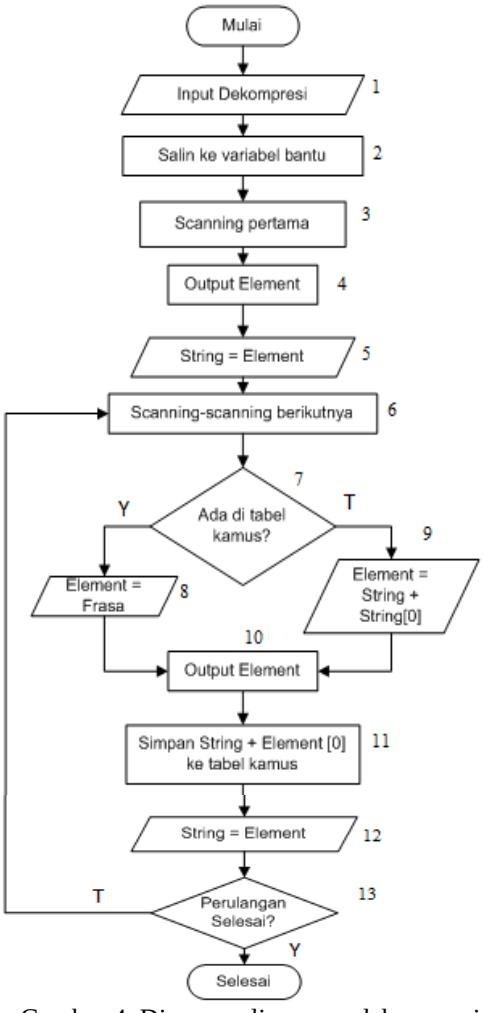
Gambar 3. Diagram alir proses kompresi *LZW*

Alur pada diagram alir di atas adalah sebagai berikut:

1. Data masukan berupa *plaintext*.
2. Pembacaan masukan perkarakter.
3. pembacaan tabel kamus pada *field* yang bersesuaian dengan *field* nilai *String*.
4 Jika terdapat dalam kamus maka *String* digabung dengan X (karakter yang sedang terbaca).
5 Menambahkan *Output* indeks *String* baru.
6 Menyimpan indeks *String* ke tabel kamus sebagai kamus tambahan yang baru.
7 Memasukkan X sebagai data *String*.
8 Memeriksa apakah semua karakter telah dibaca.
9 Mengeluarkan *Output* indeks *String* ke variabel *compressed*.

Diagram Alir Kompresi LZW

Gambar 4 menunjukkan diagram alir kompresi LZW yang diterapkan pada sistem.



Gambar 4. Diagram alir proses dekompresi LZW

Alur pada diagram alir di atas adalah sebagai berikut:

1. Data masukan berupa *compressedtext*).
2. Menyalin data masukan ke variabel *array*.
3. Scanning isi variabel array indeks pertama.

4. Mengeluarkan *Output Element* ke variabel '*plain*'.
5. Menyalin variabel *Element* ke variabel *String* baru.
6. Membaca isi variabel *array* indeks-indeks selanjutnya.
7. Memeriksa Adakah *record* dengan yang nilai dari *field* indeks yang merupakan nilai desimal dari kode isi variabel *array* indeks.
8. Jika ya, *Element* = Frase.
9. Jika tidak, *Element* = *String* sebelumnya digabung dengan karakter pertama dari *String*.
10. Mengeluarkan *Output Element* ke variabel '*plain*'.
11. Menggabungkan dan menyimpan *String* sebelumnya dengan karakter pertama dari *Element* ke tabel kamus sebagai kamus tambahan yang baru.
12. Memindahkan nilai *Element* ke *String*.
13. Memeriksa apakans emua isi *array* sudah dibaca, jika ya maka proses dekompresi selesai, jika tidak proses akan diulangi ke alur 6.

4. PENGUJIAN DAN ANALISA

Pengujian

Pengujian proses kompresi dan dekompresi LZW pada sistem ini dilakukan pada sejumlah teks artikel yang dengan jumlah karakter 200, 400, 600, 800 dan 1000. Masing-masing berjumlah sepuluh buah artikel.

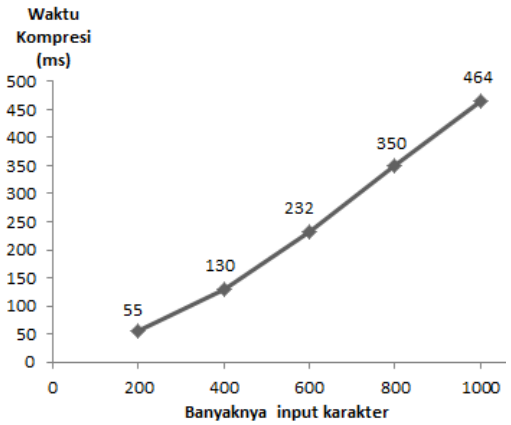
Tabel 3. menunjukkan rata-rata hasil pengujian kompresi dan dekompresi LZW dengan parameter pengujian *Compression Time* (CT), *Compression Ratio* (CR), *Compression Factor* (CF), *Saving Percentage* (SP), dan *Decompression Time* (DT).

Tabel 3. Rata-rata Hasil Pengujian

NO	INPUT	CT (ms)	CR (kali)	CF (kali)	SP (%)	DT (ms)
1	200	55	0,883	1,132	11,67	33
2	400	130	0,876	1,143	12,43	63
3	600	232	0,860	1,164	14,02	86
4	800	350	0,848	1,179	15,17	121
5	1000	464	0,833	1,201	16,70	169

Analisa

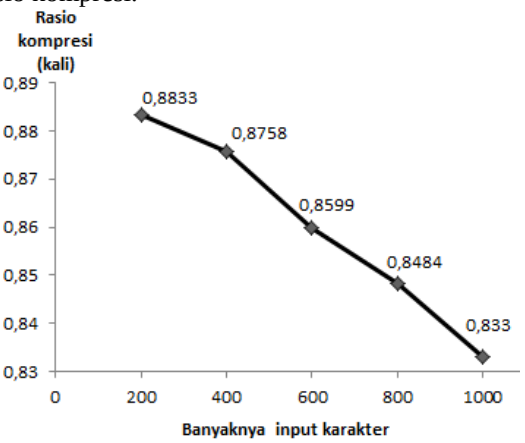
Grafik pada Gambar 5. menunjukkan hasil waktu kompresi.



Gambar 5. Grafik rata-rata waktu kompresi

Pada grafik di atas bahwa semakin bertambahnya jumlah karakter maka waktu kompresi semakin lama.

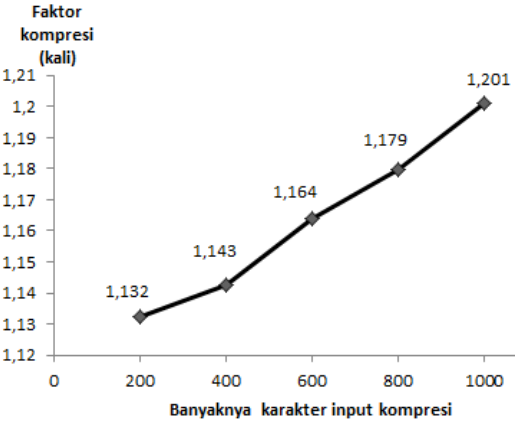
Grafik pada Gambar 6. menunjukkan hasil rasio kompresi.



Gambar 6. Grafik rata-rata rasio kompresi

Pada grafik di atas ditunjukkan bahwa rasio kompresi semakin bertambahnya jumlah karakter maka rasio kompresi semakin kecil.

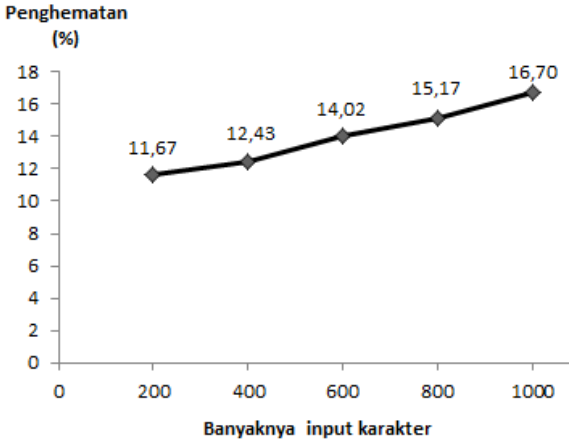
Grafik pada Gambar 7. menunjukkan hasil faktor kompresi.



Gambar 7. Grafik rata-rata faktor kompresi

Pada grafik di atas ditunjukkan bahwa faktor kompresi semakin bertambahnya jumlah karakter maka faktor kompresi semakin tinggi.

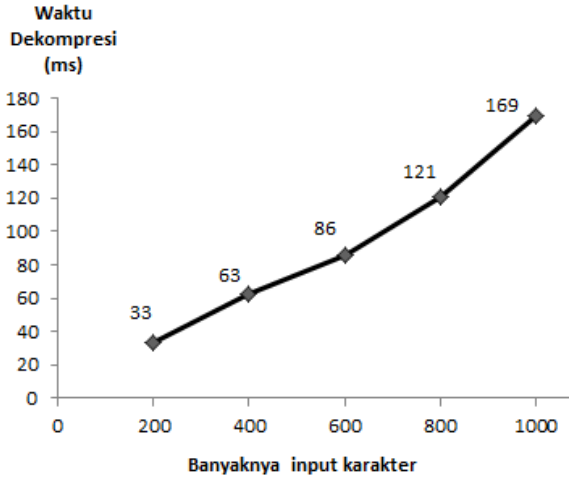
Grafik pada Gambar 8. menunjukkan hasil penghematan (%) kompresi.



Gambar 8. Grafik rata-rata penghematan (%)

Pada grafik di atas ditunjukkan bahwa penghematan kompresi semakin bertambahnya jumlah karakter maka penghematan (%) kompresi semakin tinggi.

Grafik pada Gambar 9. menunjukkan waktu dekompresi.



Gambar 9. Grafik rata-rata waktu dekompresi

Pada grafik di atas ditunjukkan bahwa waktu dekompresi lebih cepat dibanding waktu kompresi.

5. SIMPULAN DAN SARAN

Simpulan

Simpulan dari hasil penelitian yang telah dilakukan adalah sebagai berikut:

1. Teknik kompresi LZW dapat diimplementasikan untuk mengkompresi teks artikel yang akan disimpan dalam *web server*.
2. Hasil pengujian kompresi yang dilakukan pada artikel dengan jumlah karakter antara 200 sampai dengan 1000 adalah:
   a. waktu kompresi yang dibutuhkan untuk mengkompresi mencapai kisaran waktu antara 55 ms sampai dengan 464 ms.
   b. Rasio kompresi yang diperoleh adalah 0,883 kali sampai dengan 0,833 kali.
   c. Faktor kompresi yang mencapai kisaran 1,132 kali sampai dengan 1,201 kali.
   d. Penghematan hasil kompresi mencapai kisaran 11,67% sampai dengan 16,70%.
   e. Waktu dekompresi yang mencapai kisaran waktu antara 33 ms sampai dengan 169 ms. Data ini menunjukkan bahwa waktu dekompresi lebih cepat dibanding waktu kompresi.

Saran

Teknik kompresi data dapat juga diimplementasikan melalui sisi *client*. Hal ini akan dapat mengurangi beban *traffic* pada jaringan karena dengan implementasi kompresi di sisi *client*, maka data yang dikirim baik itu dari *client* ke *server* ataupun sebaliknya sudah dalam keadaan terkompresi.

DAFTAR PUSTAKA

[1] Pu, I. M. (2006). *Fundamental Data Compression.* Burlington: Elsevier.

[2] Sayood, K. (2006). *Introduction to Data Compression.* San Francisco: Morgan Kaufmann.

[3] Linawati, & Panggabean, H. P. (2004). Perbandingan Kinerja Algoritma Kompresi, LZW, dan DMC pada Berbagai Tipe File. *INTEGRAL*, 9.

[4] Taylor, A. G. (2003). *SQL for Dummies 5th Edition.* Indianapolis: Wiley Publishing.

[5] Bell, C. A. (2007). *Expert MySQL.* Berkeley: Apress.

Lampiran

Tabel 1. Kamus LZW

indeks	karakter	indeks	karakter	indeks	Karakter	indeks	Karakter
0	a	26	A	52	0	78	:
1	b	27	B	53	1	79	;
2	c	28	C	54	2	80	<
3	d	29	D	55	3	81	=
4	e	30	E	56	4	82	>
5	f	31	F	57	5	83	?
6	g	32	G	58	6	84	@
7	h	33	H	59	7	85	[
8	i	34	I	60	8	86	\
9	j	35	J	61	9	87	]
10	k	36	K	62	spasi	88	^
11	l	37	L	63	!	89	_
12	m	38	M	64	“	90	`
13	n	39	N	65	#	91	{
14	o	40	O	66	$	92	|
15	p	41	P	67	%	93	}
16	q	42	Q	68	&	94	~
17	r	43	R	69	‘		
18	s	44	S	70	(		
19	t	45	T	71	)		
20	u	46	U	72	*		
21	v	47	V	73	+		
22	w	48	W	74	,		
23	x	49	X	75	-		
24	y	50	Y	76	.		
25	z	51	Z	77	/		

Tabel 2. Translasi Set Karakter

id	kar	id	kar	id	kar	id	kar	id	kar	id	kar
1	U+0000	44	U+0037	87	U+0063	130	U+0096	173	U+00C4	216	U+00EF
2	U+0001	45	U+0038	88	U+0064	131	U+0097	174	U+00C5	217	U+00F0
3	U+0002	46	U+0039	89	U+0065	132	U+0099	175	U+00C6	218	U+00F1
4	U+0003	47	U+003A	90	U+0066	133	U+009A	176	U+00C7	219	U+00F2
5	U+0004	48	U+003B	91	U+0067	134	U+009B	177	U+00C8	220	U+00F3
6	U+0005	49	U+003C	92	U+0068	135	U+009C	178	U+00C9	221	U+00F4
7	U+0006	50	U+003D	93	U+0069	136	U+009D	179	U+00CA	222	U+00F5
8	U+0007	51	U+003E	94	U+006A	137	U+009E	180	U+00CB	223	U+00F6
9	U+0008	52	U+003F	95	U+006B	138	U+009F	181	U+00CC	224	U+00F7
10	U+000E	53	U+0040	96	U+006C	139	U+00A1	182	U+00CD	225	U+00F8
11	U+000F	54	U+0041	97	U+006D	140	U+00A2	183	U+00CE	226	U+00F9
12	U+0010	55	U+0042	98	U+006E	141	U+00A3	184	U+00CF	227	U+00FA
13	U+0011	56	U+0043	99	U+006F	142	U+00A4	185	U+00D0	228	U+00FB
14	U+0012	57	U+0044	100	U+0070	143	U+00A5	186	U+00D1	229	U+00FC
15	U+0013	58	U+0045	101	U+0071	144	U+00A6	187	U+00D2	230	U+00FD
16	U+0014	59	U+0046	102	U+0072	145	U+00A7	188	U+00D3	231	U+00FE
17	U+0015	60	U+0047	103	U+0073	146	U+00A8	189	U+00D4	232	U+00FF
18	U+0016	61	U+0048	104	U+0074	147	U+00A9	190	U+00D5	233	U+0100
19	U+0017	62	U+0049	105	U+0075	148	U+00AA	191	U+00D6	234	U+0101
20	U+0018	63	U+004A	106	U+0076	149	U+00AB	192	U+00D7	235	U+0102
21	U+0019	64	U+004B	107	U+0077	150	U+00AC	193	U+00D8	236	U+0103
22	U+001A	65	U+004C	108	U+0078	151	U+00AE	194	U+00D9	237	U+0104
23	U+001B	66	U+004D	109	U+0079	152	U+00AF	195	U+00DA	238	U+0105
24	U+0021	67	U+004E	110	U+007A	153	U+00B0	196	U+00DB	239	U+0106
25	U+0023	68	U+004F	111	U+007B	154	U+00B1	197	U+00DC	240	U+0107
26	U+0024	69	U+0050	112	U+007C	155	U+00B2	198	U+00DD	241	U+0108
27	U+0025	70	U+0051	113	U+007D	156	U+00B3	199	U+00DE	242	U+0109
28	U+0026	71	U+0052	114	U+007E	157	U+00B4	200	U+00DF	243	U+010A
29	U+0028	72	U+0053	115	U+007F	158	U+00B5	201	U+00E0	244	U+010B
30	U+0029	73	U+0054	116	U+0080	159	U+00B6	202	U+00E1	245	U+010C
31	U+002A	74	U+0055	117	U+0082	160	U+00B7	203	U+00E2	246	U+010D
32	U+002B	75	U+0056	118	U+0083	161	U+00B8	204	U+00E3	247	U+010E
33	U+002C	76	U+0057	119	U+0084	162	U+00B9	205	U+00E4	248	U+010F
34	U+002D	77	U+0058	120	U+0085	163	U+00BA	206	U+00E5	249	U+0110
35	U+002E	78	U+0059	121	U+0086	164	U+00BB	207	U+00E6	250	U+0111
36	U+002F	79	U+005A	122	U+0087	165	U+00BC	208	U+00E7	251	U+0112
37	U+0030	80	U+005B	123	U+0088	166	U+00BD	209	U+00E8	252	U+0113
38	U+0031	81	U+005D	124	U+0089	167	U+00BE	210	U+00E9	253	U+0114
39	U+0032	82	U+005E	125	U+008A	168	U+00BF	211	U+00EA	254	U+0115
40	U+0033	83	U+005F	126	U+008B	169	U+00C0	212	U+00EB	255	U+0116
41	U+0034	84	U+0060	127	U+008C	170	U+00C1	213	U+00EC	256	U+0117
42	U+0035	85	U+0061	128	U+008E	171	U+00C2	214	U+00ED		
43	U+0036	86	U+0062	129	U+0095	172	U+00C3	215	U+00EE		