BAB II SISTEM-SISTEM BILANGAN DAN KODE

Didalam sistem-sistem digital informasi numerik biasanya dinyatakan dalam sistem bilangan biner (atau kode biner lain yang bersangkutan). Sistem biner telah diperkenalkan pada Bab I, dimana telah ditunjukkan pula kesamaan-kesamaan nya dengan system desimal. Beberapa system lain untuk menyatakan data numerik juga penting di dalam sistem-sistem digital, yakni sistem oktal, heksadesimal, Binary-coded-decimal (BCD), dan Excess-3. Berbagai macam system bilangan, hubungan-hubungannya, dan operasi-operasi aritmetik akan dibahas pada bab ini.

2.1 Konversi Biner ke Desimal

Setiap bilangan biner dapat dikonversi menjadi ekivalen desimalnya dengan cara menjumlahkan bobot-bobot pada bilangan biner yang mengandung bit 1, sebagai contoh :

$$1\,1\,0\,1\,1_{\,2}\ \text{(binary)}$$
$$2^4+2^3+0+2^1+2^0 = 16+8+0+2+1$$
$$= 27_{10}\ \text{(decimal)}$$

and

$$1\,0\,1\,1\,0\,1\,0\,1_{\,2}\ \text{(binary)}$$
$$2^7+0+2^5+2^4+0+2^2+0+2^0 = 128+0+32+16+0+4+0+1$$
$$= 181_{10}\ \text{(decimal)}$$

2.2 Konversi Desimal ke Biner

Ada beberapa cara untuk mengubah suatu bilangan desimal menjadi bilangan biner. Cara yang cocok dipakai untuk bilangan-bilangan kecil adalah kebalikan dari proses yang diuraikan pada sub bab 1.5. Bilangan desimalnya dengan mudah dapat dinyatakan sebagai suatu jumlah dari pangkat-pangkat dari bilangan 2 dan kemudian bit-bit 1 dan 0 dituliskan pada posisi-posisi yang sesuai. Sebagai contoh :

$$45_{10} = 32 + 0 + 8 + 4 + 0 + 1$$
$$= 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0$$
$$= 1\,0\,1\,1\,0\,1_2$$

Untuk bilangan-bilangan desimal yang lebih besar, cara diatas menghabiskan waktu. Suatu cara yang lebih mudah yaitu dengan melakukan pembagian berturut-turut dengan 2 dan menuliskan sisanya sampai diperoleh hasil 0. Perhatikan contoh berikut : bilangan desimal 25.375 dikonversi ke biner. Langkah yang pertama adalah memisahkan bilangan bulat dengan pecahan. Konversi ini dilakukan dengan secara berturut-turut membagi 25 dengan 2 dan menuliskan sisanya setiap pembagian sampai diperoleh hasil bagi 0.

25/ 2	= 12+ remainder of **1**	**1 (Least Significant Bit)**
12/ 2	= 6 + remainder of 0	0
6 / 2	= 3 + remainder of 0	0
3 / 2	= 1 + remainder of 1	1
1 / 2	= 0 + remainder of ***1***	***1 (Most Significant Bit)***
Result	25_{10} =	***1*** 1 0 0 **1**$_2$

Bagian pecahan dari bilangan (0.375) yang dikonversikan ke biner secara berturut-turut dikalikan dengan 2 dan seterusnya mengikuti prosedur seperti berikut ini :

0.375 x 2 = 0.75 = 0.75 + carry 0

0.75 x 2 = 1.50 = 0.50 + carry 1

0.50 x 2 = 1.00 = 0.00 + carry 1

$0.375_{10} = .011_2$

Akhirnya hasil selengkapnya untuk 25.375 dapat dituliskan sebagai gabungan dari konversi bulat dan pecahan :

$25.375_{10} = 11001.011_2$

2.3 Penjumlahan Biner

Penjumlahan bilangan biner dilakukan sama seperti penjumlahan bilangan-bilangan desimal. Dalam kenyataannya, penjumlahan biner lebih sederhana karena hanya ada lebih sedikit kasus yang dipelajari. Berikut adalah penjumlahan desimal:

```
  3 7 6
  4 6 1
-------
  8 3 7
```

Langkah-langkah yang sama berlaku pula pada penjumlahan biner, tetapi bagaimanapun juga hanya ada empat kasus yang terjadi pada penjumlaha biner pada setiap posisi yaitu :

0 + 0 = 0

1 + 0 = 1

1 + 1 = 0 + carry 1 ke dalam posisi berikutnya

1 + 1 + 1 = 1 + carry 1 ke dalam posisi berikutnya

Kasus terakhir terjadi apabila pada suatu posisi tertentu ada 2 bit yang dua-duanya 1 dan ada carry dari posisi sebelumnya. Berikut adalah contoh penjumlahan biner :

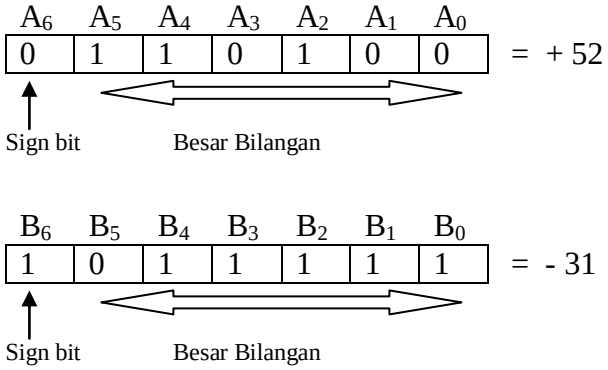
```
  0 1 1 (3)        1 0 0 1  (9)        1 1. 0 1 1  (3.375)
+ 1 1 0 (6)      + 1 1 1 1  (15)     + 1 0. 1 1 1  (2.750)
-----------      -------------       ---------------------
1 0 0 1 (9)      1 1 0 0 0  (24)     1 1 0. 0 0 1  (6.125)
```

Penjumlahan adalah operasi aritmetik yang paling penting dalam sistem digital. Operasi pengurangan, perkalian dan pembagian seperti yang dilakukan pada komputer dan kalkulator digital sesungguhnya hanya menggunakan penjumlahan sebagai operasi dasarnya.

2.4 Menyatakan Tanda Bilangan

Pada mesin-mesin biner, bilangan-bilangan biner dinyatakan oleh suatu set alat penyimpan biner (biasanya Flip-Flop). Misalnya, register FF 6 bit dapat menyimpan bilangan biner dari 000000 sampai 111111 (0 sampai 63 dalam desimal) . Ini menyatakan besarnya bilangan. Karena hampir semua komputer dan kalkulator digital menangani bilangan-bilangan positip maupun bilangan-bilangan negatip, suatu cara diperlukan untuk menyatakan tanda bilangan (+ atau -). Ini biasanya dilakukan dengan menambahkan bit lain pada bilangannya yang disebut bit tanda atau sign bit . Konvensi umum yang telah diterima adalah bahwa 0 pada sign bit menyatakan bilangan positip dan 1 pada sign bit menyatakan bilangan negatip. Ini ditunjukkan pada gambar 6. register A mengandung bit-bit 0110100. 0 pada bit paling kiri (A_6) adalah sign bit yang menyatakan positip. Enam bit yang lain menyatakan besarnya bilangan 110100_2, yang sama dengan 52 dalam desimal. Jadi bilangan yang disimpan dalam register A adalah +52. Demikian juga, bilangan yang disimpan dalam register B adalah -31, karena sign bitnya adalah 1 yang menyatakan negatip.

Sign bit digunakan untuk menunjukkan apakah bilangan biner yang disimpan adalah positip atau negatip. Untuk bilangan-bilangan positip, bit-bit selebihnya (selain sign bit) selalu digunakan untuk menyatakan besarnya bilangan dalam bentuk biner. Tetapi untuk bilangan-bilangan negatip ada tiga bentuk yang digunakan untuk menyatakan besarnya bilangan biner yaitu bentuk true-magnitude, bentuk komplemen ke 1, dan bentuk komplemen ke 2.



Gambar 2.1. Menyatakan Tanda Bilangan

True Magnitude Form

True magnitude form adalah representasi yang ditunjukkan pada gambar 2.1, dimana besar bilangan yang sebenarnya diberikan dalam bentuk biner. Bit pertama selalu merupakan sign bit.

Bentuk Komplemen ke 1

Bentuk komplemen ke 1 dari setiap bilangan biner diperoleh dengan mengubah setiap 0 di dalam bilangan tersebut menjadi 1, dan setiap 1 di dalam bilangan menjadi 0. Dengan kata lain mengubah setiap bit menjadi komplemennya. Misalnya komplemen ke 1 dari 101101 adalah 010010, dan komplemen ke 1 dari 011010 adalah 100101.

Apabila bilangan-bilangan negatip dinyatakan dalam bentuk komplemen 1, sign bitnya dibuat 1 dan besarnya dikonversikan dari bentuk biner sesungguhnya menjadi komplemen ke 1-nya. Sebagai contoh bilangan -57 akan dinyatakan sebagai berikut :

Sign bit

-57 = 1 111001 (true magnitude form)

= 1 000110 (bentuk komplemen ke 1)

Ingat bahwa sign bit tidak dikomplemenkan tetapi dipertahankan tetap sebagai 1 untuk menunjukkan bilangan negatip. Berikut beberapa contoh tambahan dari bilangan-bilangan negatip yang dinyatakan dalam bentuk komplemen ke 1.

- 14 = 10001 -7.25 = 1000.10

- 326 = 1010111001

Bentuk Komplemen Ke 2

Bentuk komplemen ke 2 dari suatu bilangan biner dibentuk dengan mengambil komplemen ke 1 dari bilangannya dan dengan menambahkan 1 pada posisi least significant bit. Prosedurnya ditunjukkan seperti di bawah ini untuk mengubah 111001 (desimal 57) menjadi bentuk komplemen ke 2-nya.

1 1 1 0 0 1 komplemenkan tiap bit untuk membentuk komplemen ke 1

↓ ↓ ↓ ↓ ↓ ↓

0 0 0 1 1 0

1 tambah 1 kepada LSB untuk membentuk komplemen ke 2

0 0 0 1 1 1

Jadi, dalam representasi komplemen ke 2-nya dari – 57 akan ditulis sebagai 1000111. Juga disini, bit yang paling kiri merupakan sign bit. 6 bit yang lain merupakan bentuk komplemen ke 2 dari besar bilangannya. Sebagai contoh lain komplemen ke 2 dari -14 ditulis 10010.

Ketiga bentuk dari menyatakan bilangan-bilangan negatip untuk -57 diikhtisarkan pada gambar 2.2.

1	1	1	1	0	0	1	True magnitude
1	0	0	0	1	1	0	Komplemen ke 1
1	0	0	0	1	1	1	Komplemen ke 2

Sign bit

Gambar 2.2. Tiga cara yang digunakan untuk menyatakan bilangan-bilangan biner negatip

Ketiga bentuk tersebut sekarang digunakan dalam sistem-sistem digital. Beberapa mesin-mesin digital menyimpan bilangan-bilangan negatip dalam true magnitude form, tetapi terlebih dahulu mengubahnya menjadi komplemen ke 1 atau komplemen ke 2 sebelum mengerjakan setiap operasi-operasi aritmetik. Mesin-mesin lain menyimpan bilangan-bilangan negatip dalam bentuk komplemen ke 1 dan komplemen ke 2. Pada hampir semua mesin-mesin digital modern, untuk operasi-operasi aritmetik bilangan-bilangan negatipnya ada dalam komplemen ke 1 atau bentuk komplemen ke 2. Saat ini representasi komplemen ke 2 paling banyak digunakan.

Harus di ingat bahwa dalam ketiga sistem, true magnitude, komplemen ke 1 dan komplemen ke 2, bilangan-bilangan positip selalu dalam bentuk biner sesungguhnya dan dengan sign bit 0. Perbedaannya terletak pada representasi bilangan-bilangan negatipnya.

Digunakannya bentuk-bentuk komplemen 1 dan komplemen 2 karena penggunaannya memungkinkan untuk melakukan operasi pengurangan hanya dengan menggunakan operasi penjumlahan. Ini penting karena berarti bahwa sebuah mesin digital dapat menggunakan rangkaian yang sama untuk dua-duanya, menjumlahkan dan mengurangkan, oleh karena itu menghemat tempat dan alat.

Mengubah Bentuk Komplemen Menjadi Biner

Untuk mengubah dari komplemen ke1 menjadi biner yang sebenarnya hanya diperlukan untuk mengkomplemenkan lagi setiap bit-nya. Untuk mengubah dari komplemen ke 2 menjadi biner yang sebenarnya hanya diperlukan untuk mengkomplemenkan setiap bit dan kemudian menambah 1 pada LSB nya.

2.5 Penjumlahan Pada Sistem Komplemen ke 2

Sistem komplemen ke 1 dan sistem komplemen ke 2 adalah sangat mirip. Tetapi bagaimanapun juga, sistem komplemen ke 2 adalah yang umum digunakan karena keuntungan yang terdapat pada pelaksanaan rangkaiannya.

Kasus I : Dua Bilangan Positip

Penjumlahan dari dua bilangan positip adalah langsung.

+ 9	→	0 1001	(yang ditambah)
+ 4	→	0 0100	(yang menambah)
+ 13		0 1101	

Sign bit

Perhatikan bahwa sign bit dari yang ditambahkan dan yang menambah dua-duanya adalah 0 dan sign bit dari jumlahnya adalah 0, yang menunjukkan bahwa jumlah tersebut adalah positip. Juga perhatikan bahwa yang ditambah dan yang menambah dibuat mempunyai jumlah bit yang sama. Ini harus selalu dilakukan dalam sistem komplemen ke 2.

Kasus II : Bilangan Positip dan Bilangan Negatip yang Lebih Kecil

Misal penjumlahan +9 dan -4. Ingat bahwa -4 akan ada dalam bentuk komplemen ke 2. Jadi, +4 (00100) harus diubah menjadi -4 (11100)

Sign bit

+ 9	→	0 1001	(yang ditambah)
- 4	→	1 1100	(yang menambah)
+5		10 0101	

Carry ini diabaikan, sehingga hasilnya adalah 00101 = +5

Perhatikan bahwa sign bit-sign bit tersebut juga ikut dalam proses penjumlahan. Ternyata sebuah carry dihasilkan pada posisi hasil penjumlahan terakhir. Carry ini selalu diabaikan, sehingga jumlah akhir sama dengan 00101 (+5)

Kasus III : Bilangan Positip dan Bilangan Negatip yang Lebih Besar

Contoh penjumlahan -9 dan +4

- 9		1 0111	(yang ditambah)
+4	→	0 0100	(yang menambah)
-5		1 1011	(jumlah = -5)

Disini jumlahnya mempunyai sign bit 1, yang menunjukkan suatu bilangan negatip. Karena jumlahnya adalah negatip, maka merupakan bentuk komplemen ke 2, sehingga empat bit terakhir (1011) menyatakan komplemen ke 2 dari 0101 (ekivalen dengan desimal 5). Jadi 11011 adalah ekivalen dengan -5.

Kasus IV : Dua Bilangan Negatip

- 9	→	1 0111	(yang ditambah)
- 4	→	1 1100	(yang menambah)
-13		11 0011	

Carry ini diabaikan, hasilnya adalah 10011 = -13

Sekali lagi hasil ini adalah negatip dan dalam bentuk komplemen ke 2 dengan sign bit 1.

Kasus V : Bilangan yang sama dan berlawanan

- 9	→	1 0111	(yang ditambah)
+9	→	0 1001	(yang menambah)
0		100000	

Carry ini diabaikan, sehingga hasilnya adalah 00000=+0

2.6 Pengurangan Dalam Sistem Komplemen Ke 2

Operasi pengurangan dengan menggunakan sistem komplemen ke 2 sesungguhnya melibatkan operasi penjumlahan dan sama sekali tidak berbeda dengan berbagai macam kasus yang telah dibahas pada sub bab 1.9. Pada saat mengurangkan satu bilangan biner dari bilangan biner yang lain, maka prosedurnya adalah sebagai berikut :

1. Cari komplemen ke 2 dari pengurang, termasuk dengan sign bit-nya. Apabila pengurangnya merupakan suatu bilangan positip, maka harus dirubah ke suatu bilangan negatip dalam bentuk komplemen ke 2. Apabila pengurangnya merupakan bilangan negatip, ini akan mengubahnya menjadi bilangan positip dalam bentuk biner sebenarnya.
2. Setelah menemukan komplemen ke 2 dari pengurang, tambahkan kepada yang dikurangi. Bilangan yang dikurangi tersebut dipertahankan dalam bentuk aslinya. Hasil dari penjumlahan ini merupakan selisih yang dicari. Sign bit dari selisih ini menentukan apakah tandanya + atau – dan apakah merupakan bentuk biner sesunguhnya atau bentuk komplemen ke 2.

Contoh :

Yang dikurangi (9) 01001

Pengurang (+4) 00100

Ubahlah pengurang menjadi komplemen ke 2-nya (11100). Sekarang tambahkan bilangan ini dengan yang dikurangi :

```
 + 9  →   0 1001
 - 4  →   1 1100
 ------   ------
 + 5  →   1 00101
            ↑
            └── diabaikan, sehingga hasilnya adalah 00101 = + 5
```

2.7 Perkalian Bilangan-Bilangan Biner

Perkalian bilangan biner dilakukan dengan cara yang sama dengan perkalian bilangan desimal, contoh :

1001 ⇐ yang dikalikan = 9_{10}
1011 ⇐ pengali = 11_{10}
1001
1001
0000
1001
1100011 ⇐ hasil akhir = 99_{10}

Hampir semua mesin-mesin digital hanya dapat menjumlahkan dua bilangana bilangan terse biner pada satu saat tertentu. Oleh karenanya penjumlahan hasil perkalian dilakukan dua demi dua; yaitu, yang pertama dijumlahkan dengan yang kedua, hasilnya dijumlahkan dengan yang ketiga, dan seterusnya.

Perkalian dalam sistem komplemen ke 2

Perkalian yang dilakukan sama seperti yang dijelaskan di atas dengan catatan bahwa bilangan yang dikalikan dan pengali dinyatakan dalam bentuk biner yang sebenarnya . Apabila dua bilangan yang dikalikan adalah positip maka dapat dikalikan sebagaimana mestinya. Tentu saja hasil kalinya adalah positip, dan diberi sign bit 0.

Apabila kedua bilangan tersebut negatip, terlebih dahulu dijadikan dalam bentuk komplemen ke 2. masing-masing diubah menjadi bilangan positip dan kemudian dikalikan. Hasilnya dipertahankan sebagai bilangan positip dan diberi sign bit 0.

Apabila salah satu dari kedua bilangan tersebut positip dan lainnya negatip, pertama-tama bilangan negatip diubah menjadi bilangan positip dengan mencari komplemen ke 2-nya. Hasilnya akan merupakan true magnitude form. Tetapi bagaimanapun juga, hasil kalinya harus negatip, maka hasilnya kemudian diubah menjadi bentuk komplemen ke-2 dan diberi sign bit 1.

2.8 Pembagian Biner

Proses untuk membagi suatu bilangan biner oleh bilangan biner lain adalah sama dengan proses yang diikuti untuk bilangan-bilangan desimal, contoh :

```
      0011          (9 : 3 = 3)
   ┌──────
11 │ 1001
   │ 011
   │ 0011
```

```
      0010.1        (10:4 = 2.5)
   ┌──────
100│ 1010.0
   │ 100
   │   100
   │   100
   │     0
```

Pembagian dari bilangan-bilangan bertanda dilakukan dengan cara yang sama seperti perkalian. Bilangan-bilangan negatip dijadikan positip dengan mengkomplementasikan dan kemudian baru melaksanakan pembagian. Apabila yang dibagi dan pembagi tandanya berlawanan, hasil baginya diubah menjadi bilangan negatip dengan menghitung komplementasi ke 2-nya dan diberi sign bit 1. Apabila yang dibagi dan pembagi tandanya sama, hasil baginya dibiarkan tetap positip dan diberi sign bit 0.

2.9 Sistem Bilangan Oktal

Sistem bilangan oktal sangat penting dalam bidang komputer digital. Sistem bilangan oktal mempunyai basis delapan, berarti bahwa bilangan ini mempunyai delapan yang mungkin : 0,1,2,3,4,5,6, dan 7. Jadi, setiap digit dari bilangan oktal dapat mempunyai harga dari 0 sampai 7. Posisi-posisi digit di dalam bilangan oktal mempunyai delapan bobot sebagai berikut :

8^3	8^2	8^1	8^0		8^{-1}	8^{-2}	8^{-3}
=512	=64	=8	=1	.	=1/8	=1/64	=1/512
Most **S**ignificant **D**igit (MSD)				Octal point			**L**east **S**ignificant **D**igit (LSD)

Konversi Oktal Ke Desimal

Contoh : $24.6_8 = 2 \times (8^1) + 4 \times (8^0) + 6 \times (8^{-1}) = 20.75_{10}$

Konversi Biner ke Oktal / Oktal ke Biner

Digit Oktal	0	1	2	3	4	5	6	7
Binary Ekivalen	000	001	010	011	100	101	110	111

Setiap digit oktal dinyatakan oleh tiga bit dari digit biner.

Contoh : $100\ 111\ 010_2 = (100)\ (111)\ (010)_2 = 4\ 7\ 2_8$

Pembagian Secara Berulang

Metode ini menggunakan pembagian berulang dengan 8.

Contoh konversi 177_{10} ke octal dan biner:

177/8	=	22 + sisa **1**	**1** ***(LSB)***
22/ 8	=	2 + sisa 6	6
2 / 8	=	0 + sisa 2	***2 (MSB)***

Hasil $177_{10} = 261_8$

Konversi ke Biner = 010110001_2

Keuntungan dari Sistem Oktal

Pada umumnya sistem oktal tersebut berguna apabila sejumlah besar informasi bit-bit biner akan ditulis, di display, atau disampaikan dari orang yang satu ke orang yang lain secara tertulis atau lisan. Misalnya lebih mudah dan lebih kecil kemungkinan salahannya menyampaikan bilangan biner 101011100101_2 sebagai 5345_8 (ekivalen oktalnya). Penerima informasi dapat dengan mudah mengubahnya menjadi biner.

2.10 Sistem Heksadesimal

Sistem heksadesimal menggunakan basis 16. Jadi memiliki 16 kemungkinan simbol digit. Sistem ini menggunakan digit-digit : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, dan F.

Tabel 2. Hubungan antara heksadesimal, desimal dan biner

Heksadesimal	Desimal	Biner
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Posisi-posisi digit di dalam bilangan heksadesimal mempunyai enambelas bobot sebagai berikut :

16^3	16^2	16^1	16^0		16^{-1}	16^{-2}	16^{-3}
=4096	=256	=16	=1	.	=1/16	=1/256	=1/4096
Most **S**ignificant **D**igit (MSD)				Hexadec. point			**L**east **S**ignificant **D**igit (LSD)

Konversi Heksadesimal ke Desimal

Contoh : $2AF_{16} = 2 \times (16^2) + 10 \times (16^1) + 15 \times (16^0) = 687_{10}$

Konversi Desimal ke Heksadesimal (dengan pembagian berulang)

Metode ini menggunakan pembagian 16 secara berulang.

Contoh : konversi 378_{10} ke heksadesimal dan biner:

378/16	= 23+ sisa **10**	**A *(LSB)***
23/ 16	= 1 + sisa 7	7
1 / 16	= 0 + sisa **1**	***1 (MSB)***

Hasil : $378_{10} = 17A_{16}$

Konversi ke Biner = $0001\ 0111\ 1010_2$

Konversi Biner ke Heksadesimal / Heksadesimal ke Biner

Hexadecimal Digit	0	1	2	3	4	5	6	7
Binary Equivalent	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal Digit	8	9	A	B	C	D	E	F
Binary Equivalent	1000	1001	1010	1011	1100	1101	1110	1111

Setiap digit heksadesimal dinyatakan dengan empat bit dari digit biner.

Contoh . $1011\ 0010\ 1111_2 = (1011)\ (0010)\ (1111)_2 = B\ 2\ F_{16}$

Konversi Oktal ke Heksadesimal / Heksadesimal ke Oktal

Contoh . Konversikan $5A8_{16}$ ke Oktal .

$5A8_{16} = 0101\ 1010\ 1000$ (Biner)

$= 2\ 6\ 5\ 0$ (Oktal)

2.11 Penjumlahan Heksadesimal

Penjumlahan heksadesimal dilakukan sama persis dengan penjumlahan desimal, yang perlu diperhatikan bahwa bilangan heksadesimal merupakan bilangan ber-basis 16. Contoh :

4	5	8		A	1	3		
6	7	1		4	9	5		
A	C	9		E	A	8		

2.12 Kode BCD

Apabila bilangan-bilangan, huruf-huruf, kata-kata dinyatakan dalam suatu grup simbol-simbol tertentu, ini disebut pengkodean, dan grup simbol-simbol tersebut dinamakan kode. Barangkali salah satu kode yang paling dikenal adalah kode Morse, dimana serangkaian titik dan garis menyatakan huruf-huruf alphabet.

Semua sistem digital menggunakan beberapa bentuk bilangan biner untuk operasi internalnya, tetapi untuk menyajikan hasilnya ke luar digunakan bilangan desimal. Ini berarti bahwa konversi-konversi antara sistem biner dan desimal sering dilakukan. Telah diketahui bahwa konversi antara desimal dan biner untuk bilangan-bilangan besar dapat panjang dan rumit. Oleh karena itu kadang-kadang digunakan cara-cara

pengkodean bilangan desimal lain, yang menggabungkan beberapa sifat dari sistem desimal dan sistem biner.

Binary-Coded-Decimal Code

Apabila setiap digit dari suatu bilangan desimal dinyatakan dalam ekivalen binernya, maka prosedur pengkodean ini disebut binary-coded-decimal (disngkat BCD). Karena digit desimal besarnya dapat mencapai 9, maka diperlukan 4 bit untuk mengkode setiap digit (kode biner untuk 9 adalah 1001).

Untuk menunjukkan kode BCD, ambil bilangan desimal 874, setiap digit dapat diubah menjadi ekivalen binernya sebagai berikut :

8 7 4

↓ ↓ ↓

1000 0111 0100

Sebagai contoh lain, ubahlah 94.3 menjadi representasi kode BCD-nya

9 4 . 3

↓ ↓ ↓

1001 0100 . 0011

Dengan demikian, kode BCD menyatakan setiap digit bilangan desimal dengan bilangan biner 4 bit. Jelaslah bahwa hanya digunakan bilangan-bilangan biner 4 bit dari 0000 sampai 1001.

Perbandingan Antara Kode BCD dan Kode Biner Langsung

Penting untuk diketahui bahwa bilangan BCD tidak sama dengan bilangan biner langsung. Kode biner langsung mengkodekan lengkap seluruh bilangan desimal dan menyatakan dalam biner; kode BCD mengubah tiap-tiap digit desimal menjadi biner secara individual (satu per satu). Sebagai contoh ambil bilangan desimal 137 dan bandingkan kode biner langsung dengan BCD-nya :

$137_{10} = 10001001_2$ (biner)

$137_{10} = 0001\ \ 0011\ \ 0111$ (BCD)

BCD digunakan dalam mesin-mesin digital apabila yang diberikan sebagai input atau di-display sebagai output adalah informasi digital. Voltmeter digital, pengukur frekuensi, kalkulator, dan jam digital semuanya menggunakan BCD karena mereka menyajikan informasi output dalam desimal.

BCD sering tidak digunakan dalam komputer digital berkecepatan tinggi, oleh karena dua alasan. Pertama, BCD membutuhkan lebih banyak bit dibanding kode biner langsung, oleh karena itu kurang efisien. Kedua, proses aritmetik untuk BCD lebih rumit dibanding biner langsung sehingga memerlukan rangkaian yang lebih kompleks. Semakin kompleks akan memperlambat kecepatan operasinya.

2.13 Penjumlahan BCD

Penjumlahan bilangan-bilangan desimal yang berbentuk BCD paling mudah dipahami melalui dua kasus yang dapat terjadi pada saat digit-digit desimal dijumlahkan.

Jumlah Samadengan Sembilan atau Kurang

Penjumlahan 5 dan 4 yang menggunakan BCD untuk menyatakan tiap-tiap digit :

5	0101 ←	BCD untuk 5
4	0100 ←	BCD untuk 4
9	1001 ←	BCD untuk 9

Contoh lain :

4 5	0100 0101 ←	BCD untuk 4 5
3 3	0011 0011 ←	BCD untuk 3 3
7 8	0111 1000 ←	BCD untuk 7 8

Pada contoh di atas tak satupun hasil penjumlahan dari digit-digit desimal melampaui 9, oleh karena itu tidak dihasilkan carry-carry desimal. Untuk kasus-kasus ini proses penjumlahan BCD adalah langsung dan sama dengan penjumlahan biner.

Jumlah Lebih Besar dari 9

Perhatikan penjumlahan BCD 6 dan 7 dalam BCD berikut ini :

6	0110	←	BCD untuk 6
7	0111	←	BCD untuk 7
13	1101		grup kode terlarang dalam BCD

Hasil 1101 tidak terdapat dalam kode BCD , ini merupakan salah satu grup kode 4 bit terlarang. Ini terjadi karena jumlah dari dua bit tersebut melampui 9. Apabila ini terjadi maka hasilnya harus dikoreksi dengan menambah 6 (0110) untuk menghindarkan enam grup terlarang.

6	0110	←	BCD untuk 6
7	0111	←	BCD untuk 7
13	1101		grup kode terlarang dalam BCD
	0110		ditambah 6 untuk koreksi
	001 0011		BCD untuk 13

Sebagai contoh lain :

4 7	0100	0111	BCD untuk 4 7
3 5	0011	0101	BCD untuk 3 5
	0111	1100	
		0110	ditambah 6
	1000	0010	jumlah BCD yang benar

2.14 Kode Excess-3

Kode excess-3 ada hubungannya dengan kode BCD dan kadang-kadang digunakan menggantikan BCD karena mempunyai keuntungan kentungan dalam operasi-operasi aritmetik tertentu. Pengkodean excess-3 untuk bilangan desimal dilaksanakan dengan cara yang sama seperti BCD kecuali bahwa angka 3 ditambahkan pada setiap digit desimal sebelum mengkodekan dalam biner. Misalnya, mengkode bilangan desimal 3 kedalam kode excess-3, pertama-tama kita harus

menambah 3 untuk memperoleh 7. Kemudian 7 dikodekan dalam kode biner 4-bit ekivalennya, yaitu 0111.

Sebagai contoh lain, ubahlah 46 menjadi representasi kode excess-3.

```
  4      6
 +3     +3          tambahkan tiga untuk setiap digit
---    ---
  7      9
  ↓      ↓
0111   1001        Diubah menjadi kode biner 4-bit
```

Tabel 3 mencantumkan representasi kode BCD dan kode excess-3 untuk digit-digit desimal. Perhatikanlah bahwa kedua kode tersebut hanya menggunakan 10 dari 16 kemungkinan grup-grup kode 4-bit. Tetapi bagaimanapun juga, kode exces-3 tidak menggunakan grup-grup kode yang sama. Untuk excess-3,grup-grup kode yang terlarang adalah 0000,0001,0010,1101,1111.

Tabel 3. Representasi kode BCD dan kode Excess-3

Desimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

2.15 Kode Gray

Kode Gray termasuk kelas kode yang disebut kode perubahan minimum atau minimum change code,dimana hanya mengubah satu bit dalam grup kodenya apabila pindah dari satu step ke step berikutnya. Kode Gray merupakan kode tak berbobot atau unweighted,yang berarti bahwa posisi-posisi bit dalam grup-grup kode tidak mempunyai bobot tertentu. Oleh karena itu, kode Gray tidak sesuai untuk operasi aritmetik tetapi digunakan pada alat-alat input/output dan pada beberapa jenis konvertor-konvertor analog ke digital.

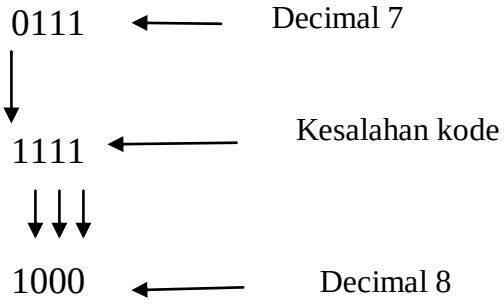
Tabel 4 menunjukkan representasi kode Gray untuk bilangan-bilangan desimal dari 0 sampai 15,bersama-sama dengan kode biner langsung. Apabila kita memperhatikan grup-grup kode Gray untuk setiap bilangan desimal, dapat dilihat bahwa pada setiap perpindahan dari satu bilangan desimal ke bilangan berikutnya hanya mengubah satu bit kode Gray. Misalnya, pada saat pindah dari 3 ke 4, kode Gray berubah dari 0010 dan 0110, dengan hanya kedua dari kiri yang berubah. Naik dari 14 ke 15 bit-bit kode Gray berubah dari 1001 ke 1000,dengan hanya bit terakhir yang berubah. Ini adalah karakteristik utama dari kode Gray. Bandingkanlah ini dengan kode biner, dimana pada setiap tempat mulai dari satu sama ke seluruh bit berubah pada saat naik dari satu step ke step berikutnya.

Kode Gray sering digunakan dalam situasi-situasi dimana kode-kode lain, seperti misalnya biner, dapat memberikan hasil-hasil yang salah atau meragukan dalam transisi-transisi dimana berubah lebih dari satu kode bit. Misalnya, dengan menggunakan kode biner untuk naik dari 0111 ke 1000 membutuhkan keempat bit berubah secara serentak. Tergantung kepada alat atau rangkaian yang menghasilkan bit, mungkin ada perbedaan berarti (signifikan) dalam waktu-waktu transisi dari bit-bit yang berbeda. Apabila demikian halnya, maka transisi dari 0111 menjadi 1000 dapat menghasilkan satu atau lebih keadaan-keadaan intermediate.

Tabel 4. Representasi kode Gray dan Biner

Desimal	Kode Biner	Kode Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Misalnya, apabila bit yang paling signifikan berubah lebih cepat dari yang selebihnya, akan terjadi transisi-transisi seperti berikut ini :



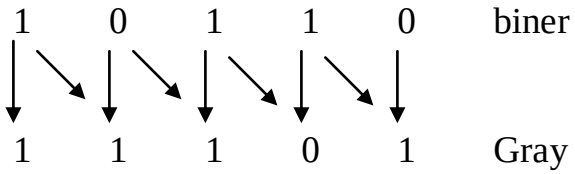
terjadinya 1111 hanya sesaat tetapi dapat menyebabkan kesalahan operasi dari elemen-elemen yang sedang dikontrol oleh bit-bit. Jelaslah bahwa dengan menggunakan kode Gray dapat meniadakan masalah ini, karena hanya terjadi satu perubahan bit per transisi dan siantara bit-bit tidak terjadi race.

Mengubah kode biner ke kode Gray :

- Bit pertama dari kode Gray samadengan bit pertama dari bilangan biner

- Bit kedua dari kode Gray samadengan exclusive-OR dari bit pertama dan kedua dari bilangan biner, yaitu akan samadengan 1 apabila bit-bit kode biner tersebut berbeda, 0 apabila sama.
- Bit kode Gray ketiga samadengan exclusive-OR dari bit-bit kedua dan ketiga dari bilangan biner, dan seterusnya.

Untuk menunjukkannya,marilah kita mengubah biner 10110 menjadi kode Gray :

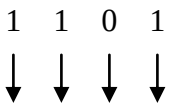


Mengubah dari Gray Ke Biner

Untuk mengubah dari Gray ke Biner diperlukan prosedur yang berlawanan dengan prosedur yang diberikan di atas.

1. Bit biner pertama adalah sama dengan bit kode Gray pertama
2. Apabila bit Gray kedua 0, bit biner kedua sama dengan yang pertama; apabila bit gray kedua 1, bit biner kedua adalah kebalikan dari bit biner pertama.
3. Langkah 2 diulang untuk setiap bit berikutnya.

Untuk lebih jelasnya perhatikan contoh berikut :



SOAL LATIHAN

Pilih salah satu jawaban dari soal berikut ini :

1. Konversikan $(63.25)_{10}$ ke biner.
   - 11111.11

- 111001.01
- 111111.01
- 111111.1
- NA

2. Konversikan $(43.8125)_{10}$ ke biner.
   - 101011.1101
   - 110101.1101
   - 101011.1011
   - 110101.1011
   - NA
3. Konversikan $(1001011.011)_2$ ke desimal
   - 73.0375
   - 75.375
   - 91.375
   - 75.573
   - NA
4. Konversikan $(110101.1011)_2$ to desimal
   - 53.6875
   - 53.6375
   - 52.6875
   - 55.6375
   - NA
5. Konversikan $(11001.1)_2$ to basis 8.
   - $(62.4)_8$
   - $(62.1)_8$
   - $(31.1)_8$
   - $(31.2)_8$
   - $(31.4)_8$
6. Konversikan $(25.6)_8$ ke biner.
   - $(10101.11)_2$

- $(11101.10)_2$
- $(10101.10)_2$
- $(10010.11)_2$
- $(11111.01)_2$

7. Konversikan $(35.1)_8$ ke basis 16.
   - $(17.4)_{16}$
   - $(1D.1)_{16}$
   - $(D1.2)_{16}$
   - $(E8.1)_{16}$
   - NA
8. Konversikan $(39.A)_{16}$ ke basis 8.
   - $(35.5)_8$
   - $(70.5)_8$
   - $(71.5)_8$
   - $(72.25)_8$
   - $(75.5)_8$
9. Konversikan $(485)_{10}$ ke basis 16.
   - $(1E5)_{16}$
   - $(231)_{16}$
   - $(5E1)_{16}$
   - $(15E)_{16}$
   - NA
10. Konversikan $(397)_{10}$ ke basis 3.
    - $(12310)_3$
    - $(121201)_3$
    - $(012211)_3$
    - $(112201)_3$
    - $(100202)_3$