SILABUS MATA KULIAH MICROPROCESSOR I
Nama Dosen: Yulius C. Wahyu Kurniawan, S.Kom.

Konsep Dasar Bilangan
Pengertian Base (Radix), Absolute Digit, Positional Value

Macam-macam Sistem Bilangan
Desimal, Oktal, Biner dan Heksadesimal

Konversi Sistem Bilangan dari yang satu ke yang lainnya

Operasi Aritmetika pada Sistem Bilangan Biner
Penjumlahan, dengan dan tanpa Carry
Pengurangan, terbagi atas:
- True Form, Complement Addition Binary atau Complement 1
- Double Complement Addition Binary atau Complement 2

Perkalian, terbagi atas:
- Metode desimal (biasa)
- Metode Shift (dari kiri)

Pembagian

Bilangan Bertanda dan Bilangan Tak Bertanda

Sistem Pengkodean
- Binary Coded Decimal (BCD) 8421 dan 2421
- Kode Excess 3 atau kode kelebihan 3, dimulai dari 0011 sampai 1100
- Kode Alphanumeric atau ASCII

Parity, berupa bit parity ganjil dan genap

Kode Humming dan Kode Gray
Konversi dari Kode Gray ke Biner dan sebaliknya

Aljabar Boole: invers, and, or. Teori Boole dan Teori De Morgan

Rangkaian Logika

Karnaugh Map

Konsep Dasar Sistem Bilangan

Sistem Bilangan selalu mencakup tiga hal:

BASE (RADIX)
Adalah maksimum angka atau simbol yang digunakan dalam sistem tersebut

ABSOLUTE DIGIT
Adalah jenis angka yang mempunyai nilai berbeda dalam sistem

POSITIONAL VALUE
Adalah nilai yang terkandung dalam suatu posisi

Sistem Bilangan Desimal dan Biner

Kita telah terbiasa menggunakan sistem bilangan desimal atau denary, yaitu sistem bilangan dengan basis 10, yang mempunyai 10 buah simbol, yaitu 0,1,2 … 9. Tetapi sistem ini tidak selalu merupakan pilihan terbaik untuk setiap aplikasi. Sistem biner yang lebih sederhana lebih cocok untuk digunakan dalam elektronika digital. Sistem biner merupakan sistem bilangan berbasis 2, dan hanya mempunyai dua buah simbol yaitu 0 dan 1. Berikut ini adalah perbandingan sistem bilangan desimal dan biner.

Sistem Bilangan Desimal
Base (Radix) : 10
Absolute Digit : 0,1,2 … 9
Positional Value : … 10^2 10^1 10^0 10^{-1} 10^{-2} …
Contoh:

$743{,}15 = 7 * 10^2 + 4 * 10^1 + 3 * 10^0 + 1 * 10^{-1} + 5 * 10^{-2}$
$123{,}25 = 1 * 10^2 + 2 * 10^1 + 3 * 10^0 + 2 * 10^{-1} + 5 * 10^{-2}$

Dalam sistem bilangan desimal atau denary, nilai yang terkandung dalam bilangan desimal diurutkan dalam ratusan, puluhan, satuan dan bilangan di belakang tanda koma. Kita bisa menguraikan bilangan desimal dalam bentuk eksponen basis 10 seperti pada contoh di atas.

Sistem Bilangan Biner
Operasi dalam sebuah komputer dilakukan dalam sistem bilangan biner.
Base (Radix) : 2
Absolute Digit : 0,1
Positional Value : … 2^2 2^1 2^0 2^{-1} 2^{-2} …
Contoh:

$00110 = 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$
$11010 = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$

Dalam sistem bilangan biner, dengan cara yang sama kita bisa menguraikan deretan bilangan biner dalam bentuk eksponen basis 2 seperti dalam contoh.

Setiap digit biner disebut bit; Bit paling kanan disebut dengan *Least Significant Bit* (LSB), dan bit paling kiri disebut Most Significant Bit (MSB)

Untuk membedakan bilangan pada sistem yang berbeda digunakan subskrip. Sebagai contoh, 9_{10} menyatakan bilangan sembilan pada sistem bilangan desimal, dan 01101_2 menunjukkan bilangan biner 01101.

Sistem Bilangan Oktal

Base (Radix) : 8
Absolute Digit : 0,1,2 … 7
Positional Value : … 8^2 8^1 8^0 8^{-1} 8^{-2} …

Bilangan oktal adalah sistem bilangan yang berbasis 8 dan mempunyai delapan simbol bilangan yang berbeda 0,1,2 … 7. Pada suatu bilangan oktal bisa diuraikan dalam eksponen basis 8.

Sistem Bilangan Heksadesimal

Base (Radix) : 16
Absolute Digit : 0,1,2 … 9, A, B, C, D, E, F
Positional Value : … 16^2 16^1 16^0 16^{-1} 16^{-2} …

Bilangan heksadesimal, sering disingkat dengan hex, adalah bilangan dengan basis 16 dan mempunyai 16 simbol yang berbeda. Dengan cara yang sama seperti sistem bilangan lainnya, kita bisa menguraikan satu bilangan heksadesimal ke dalam eksponen basis 16.

KONVERSI

Yang dimaksud konversi adalah cara mengubah atau mengalihkan bilangan ke sistem bilangan lainnya.

Konversi dari suatu sistem bilangan ke sistem bilangan desimal

Untuk melakukan konversi dari suatu sistem bilangan ke sistem bilangan desimal, kita dapat menguraikan bilangan tersebut ke dalam bentuk eksponen basis bilangan tersebut. Yang perlu kita perhatikan adalah basis bilangan yang digunakan.

Contoh:
Konversikan bilangan 11011_2 ke bilangan desimal

$$11011_2 = 2^4 + 2^3 + 2^1 + 2^0$$
$$= 16 + 8 + 2 + 1$$
$$= 27_{10}$$

Konversikan bilangan 756_8 ke bilangan desimal

$$756_8 = 7 * 8^2 + 5 * 8^1 + 6 * 8^0$$
$$= 448 + 40 + 6$$
$$= 494_{10}$$

Konversikan bilangan $31A_{16}$ ke bilangan desimal

$31A_{16} = 3 * 16^2 + 1 * 16^1 + 10 * 16^0$

$= 768 + 16 + 10$

$= 794_{10}$

Konversi dari sistem bilangan desimal ke sistem bilangan lainnya

Untuk melakukan konversi bilangan desimal ke sistem bilangan yang lain, dapat terbagi menjadi dua, yaitu konversi bilangan bulat dan pecahan.

Konversi desimal ke biner
Rumus.
Untuk Bilangan Bulat:

- Bilangan desimal dibagi 2 sampai habis.
- Bilangan binernya adalah sisa dari setiap hasil pembagian dengan urutan dari bawah ke atas.

Untuk Bilangan Pecahan:

- Kalikan bilangan pecahan desimal dengan 2 dan catat bilangan carry pada posisi bulatnya. Proses perkalian akan diteruskan sampai hasil perkalian sama dengan 1 atau sampai pada tingkat ketelitian yang diinginkan.
- Carry-carry yang diambil dalam urutan dari atas ke bawah merupakan pecahan binernya.

Untuk konversi bilangan desimal ke sistem bilangan lainnya, digunakan rumus yang sama, hanya saja basis bilangannya yang diubah.

Untuk lebih jelasnya, lihat contoh seperti dibawah ini:

$25{,}125_{10} = 11001{,}001_2$

Bilangan bulat desimal 25 dibagi 2 sampai habis, sisa pembagian adalah 11001.
Bilangan pecahan desimal 0,125 dikalikan 2, carry yang diambil adalah 001

Dengan cara yang sama, kita dapat mengubah bilangan desimal ke bilangan oktal ataupun bilangan heksadesimal seperti dibawah ini:

$480{,}69_{10} = 740{,}541_8$

$1094{,}25_{10} = 447{,}4_{16}$

Konversi sistem bilangan biner ke sistem bilangan oktal

Rumus:

- Kelompokkan tiga-tiga.
- Jika kurang dari tiga diberi 0 di depan untuk bilangan di depan koma dan di belakang untuk bilangan di belakang koma.

Contoh:
$10110111,101110_2$
kita akan kelompokkan tiga-tiga:
010 110 111, 101 $110_2 = 267,56_8$

Konversi sistem bilangan oktal ke sistem bilangan biner

Rumus:
Kembalikan nilai oktal sesuai dengan nilai biner yang bersesuaian

Contoh:
$745,23_8 = 111\ 100\ 101,\ 010\ 011_2$
$= 111100101,010011_2$

Konversi sistem bilangan biner ke sistem bilangan heksadesimal

Rumus:

- Kelompokkan empat-empat.
- Jika kurang dari empat diberi 0 di depan untuk bilangan di depan koma dan di belakang untuk bilangan di belakang koma.

Contoh:
$1011110111,01101000_2$
kita akan kelompokkan empat-empat:
0010 1111 0111, 0110 $1000_2 = 2F7,6816$

Konversi sistem bilangan heksadesimal ke sistem bilangan biner

Rumus:
Kembalikan nilai hex sesuai dengan nilai biner yang bersesuaian (lihat tabel)

Contoh:
$ABC,DE_{16} = 1010\ 1011\ 1100,\ 1110\ 1111_2$
$= 101010111100,11101111_2$

Tabel Konversi bilangan heksadesimal ke bilangan biner

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

Kalau kita perhatikan pada kolom 1 dan kolom 2, jika kita hilangkan bilangan 0 di depan untuk bilangan binernya, kita dapatkan tabel konversi bilangan oktal ke bilangan biner.

Latihan:

Ubah bilangan biner berikut ini menjadi bilangan desimal:

110	1110	10101	101101	111111

Ubah bilangan desimal berikut ini menjadi bilangan biner:

5	17	42	31	47

Ubah bilangan oktal berikut ini menjadi bilangan desimal

32	57	213	156

Ubah bilangan desimal berikut ini menjadi bilangan oktal

28	137	351	629

Ubah bilangan oktal berikut ini menjadi bilangan biner

27	210	555	6543

Ubah bilangan biner berikut ini menjadi bilangan oktal

010	110011	1011001	1010111000

Ubah bilangan heksadesimal berikut ini menjadi bilangan biner

2A	8D	C09	EF2	FFFF

Ubah bilangan biner berikut ini menjadi bilangan heksadesimal

11010110	110010	100101111111	1110101100110101

{Program Konversi Biner ke Desimal}

```
uses crt;

var
biner:string;

function convert(biner:string):real;
var
i,m,x,code,dum,hasil:integer;
begin
m:=0;
hasil:=0;
for i:=length(biner) downto 1 do
begin
val(biner[i],x,code);
dum:=x*trunc(exp(m*ln(2)));
hasil:=hasil+dum;
inc(m);
end;
convert:=hasil;
end;

begin
clrscr;
write('Input Bilangan Biner : ');
readln(biner);
write('Bilangan Desimalnya  : ',convert(biner):3:0);
readln;
end.
```

```
{Program Convert Decimal to Binery}

uses crt;

var
bil,sisa,hasil,i:integer;
a,x:string;

begin
clrscr;
write('Input Decimal Number   : ');
readln(bil);
sisa:=bil;
hasil:=bil;
a:='';
repeat
sisa:=hasil mod 2;
hasil:=hasil div 2;
str(sisa,x);
a:=a+x;
until hasil=0;
write('Equivalent in Binery   : ');
for i:=length(a) downto 1 do
begin
write(a[i]);
end;
readln;
end.
```

OPERASI ARITMETIKA PADA SISTEM BILANGAN BINER

Operasi Penjumlahan pada bilangan biner

Untuk melakukan penjumlahan pada bilangan biner, pada prinsipnya sama saja dengan penjumlahan pada bilangan desimal. Kita dapat menjumlahkan dua deretan bilangan biner dengan cara menyusunnya dan kita jumlahkan satu persatu dari atas ke bawah. Jika jumlahnya lebih besar dari bilangan basisnya, maka ada bilangan yang disimpan (carry). Carry ini yang kemudian dijumlahkan dengan digit di sebelah kirinya dan seterusnya. Dalam penjumlahan bilangan biner, carry akan timbul jika jumlah dari dua digit yang dijumlahkan adalah 2.
Berikut adalah aturan dasar untuk penjumlahan pada sistem bilangan biner.

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 with carry, yang dimaksud carry adalah bilangan yang disimpan 1.

Contoh:
Jumlahkan $11001_2 + 11011_2$

Untuk lebih jelasnya dapat kita gambarkan ke dalam tabel sbb.

Eksponen	2^5	2^4	2^3	2^2	21	2^0
	(32)	(16)	(8)	(4)	(2)	(1)
Bil. Ke 1		1	1	0	0	1
Bil. Ke 2		1	1	0	1	1
Carry	1	1		1	1	
Hasil	1	1	0	1	0	0

Jika lebih dari dua digit biner yang dijumlahkan, maka ada kemungkinan carry yang disimpan lebih besar dari 1. Sebagai contoh,

1 + 1 = 0, bilangan carry = 1
1 + 1 + 1 = 1, bilangan carry = 1
1 + 1 + 1 + 1 = (1 + 1) + (1 + 1)
= (0, carry 1) + (0, carry 1)
= (0, carry 2)
1 + 1 + 1 + 1 + 1 = 1 + (1 + 1) + (1 + 1)
= 1, carry 2
jadi dapat disimpulkan bahwa bilangan carry yang timbul dari penjumlahan bilangan biner dapat pula dijumlahkan terpisah dengan bilangan binernya, sebelum dijumlahkan dengan bilangan binernya.
0 + carry 2 = 1, carry 1
1 + carry 2 = 0, carry 2

Contoh:
Jumlahkan $10110_2 + 11101_2 + 11101_2$

Kita masih menggunakan tabel agar lebih jelas:

Eksponen	2^6	2^5	2^4	2^3	2^2	21	2^0
	(64)	(32)	(16)	(8)	(4)	(2)	(1)
Bil. Ke 1			1	0	1	1	0
Bil. Ke 2			1	1	1	0	1
Bil. Ke 3			1	1	1	0	1
Carry	1	2	2	2	1	1	
Hasil	1	0	1	0	0	0	0

Operasi Pengurangan pada bilangan biner

Pada bagian ini, kita hanya akan meninjau pengurangan bilangan biner yang memberikan hasil positif. Dalam hal ini, metode yang digunakan adalah sama dengan metode yang digunakan untuk pengurangan pada bilangan desimal. Dalam pengurangan bilangan biner, jika perlu dipinjam 1 (borrow) dari kolom di sebelah kirinya yang mempunyai derajat lebih tinggi.
Berikut adalah aturan dasar untuk pengurangan pada sistem bilangan biner.

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 pinjam (borrow) 1.

Operasi Pengurangan pada bilangan biner yang serupa dengan operasi pengurangan pada bilangan desimal dikenal dengan True Form.

Bilangan Biner Bertanda

Sejauh ini kita hanya melihat bilangan biner bernilai positif atau bilangan biner tak bertanda. Sebagai contoh bilangan biner 8 bit dapat mempunyai nilai antara:

0000 0000 sampai 1111 1111

yang semuanya bernilai positif. Untuk membedakan bilangan negatif dengan positif, tanda – diletakkan di sebelah kiri bilangan desimal. Dalam bilangan biner, untuk membedakannya, digunakan bit tanda bilangan (sign-bit) ditambahkan di sebelah kiri MSB. Jika bit tanda ditulis 0, maka bilangan tersebut positif dan jika ditulis 1, maka bilangan tersebut negatif. Pada bilangan biner bertanda yang terdiri 8 bit, bit yang paling kiri menunjukkan tanda dan 7 bit berikutnya adalah menunjukkan besarnya bilangan tersebut.

Contoh
[0]110 0111 = +(64 + 3 + 4 + 2 + 1) = +103
[1]101 0101 = -(64 + 16 + 4 + 1) = -85
dan seterusnya.

Maka dapat disimpulkan bahwa karena hanya tujuh bit yang menunjukkan besarnya, maka bilangan terkecil dan terbesar yang ditunjukkan bilangan biner bertanda yang terdiri dari 8 bit adalah:
[1]111 1111 = -127 dan
[0]111 1111 = +127

Secara umum bilangan biner tak bertanda yang terdiri dari n-bit mempunyai nilai maksimum $M = 2^n - 1$. Sementara itu untuk bilangan bertanda yang terdiri dari n-bit mempunyai nilai maksimum $M = 2^{(n-1)} - 1$. Sehingga untuk register 8 bit dalam mikroprosesor yang menggunakan sistem bilangan bertanda, nilai maksimum yang bisa disimpan adalah 127.

Complement Addition Binary & Double Complement Addition Binary

Bilangan negatif seringkali pula disajikan dalam bentuk komplemen 2 yang secara otomatis akan menghasilkan tanda bilangan yang benar. Hal ini digunakan untuk menghindari kesalahan dalam operasi aritmetika yang menggunakan bilangan positif dan bilangan negatif.
Komplemen 1 bagi bilangan biner adalah, 0 = 1 dan 1 = 0
Contoh:
Komplemen dari bilangan biner 10110 adalah 01001

Untuk bilangan positif, komplemen 2 adalah sama dengan bilangan aslinya, yaitu MSB untuk menunjukkan tanda bilangan dan bit-bit sisanya untuk menunjukkan besar bilangannya.

Untuk bilangan negatif, komplemen 2 diperoleh dengan cara menghitung terlebih dahulu komplemen 1 dari bilangan biner semula yang bertanda positif, kemudian menambahkan 1 ke LSBnya.

Lebih jauh, komplemen 2 dapat juga digunakan untuk mengurangkan dua bilangan biner. Jika ada dua bilangan biner A dan B, maka:
A – B = A + (-B) = A + (komplemen 2 dari B)
Hasil penjumlahan akan ditentukan dari tanda bilangan.
Jika hasilnya positif, bit tambahan di sebelah kiri tanda bilangan akan selalu muncul. Bit tambahan ini bisa diabaikan.
Jika hasilnya negatif, hasilnya harus langsung diubah menjadi bilangan bertanda dengan cara mencari komplemen 2, yang seperti sudah dijelaskan diatas, dapat dilakukan dengan cara mencari komplemen 1 terlebih dahulu dan kemudianmenambahkan 1 pada LSBnya.

Operasi Perkalian dan Pembagian pada bilangan biner

Perkalian pada bilangan biner mempunyai aturan sebagai berikut:

0 x 0 = 0
1 x 0 = 0
0 x 1 = 0
1 x 1 = 1

Perkalian bilangan biner dapat dilakukan seperti pada perkalian bilangan desimal. Terdapat dua metode untuk melakukan perkalian biner yaitu:

- Metode Desimal (seperti biasa, disusun dari kanan ke kiri)
- Metode Shift (disusun dari kiri ke kanan)

Kita dapat mencoba kedua metode di atas untuk membandingkan perkalian antara dua bilangan biner, yang tentunya akan menghasilkan bilangan yang sama.

Sedangkan untuk pembagian bilangan biner, dapat pula dilakukan dengan cara yang sama seperti pembagian pada sistem bilangan desimal. Unsur-unsur dalam pembagian biner antara lain: bilangan yang dibagi, pembagi, hasil bagi dan sisa bagi.

SISTEM PENGKODEAN

BCD – Binary Coded Decimal 8421 / 2421

Sampai saat ini kita hanya melihat perubahan dari bilangan desimal ke bilangan biner murni. Pada beberapa aplikasi, misalnya sistem berdasar mikroprosesor, seringkali terjadi lebih sesuai apabila digit bilangan desimal diubah menjadi 4 digit bilangan biner. Dengan cara ini, suatu bilangan desimal 2 digit akan diubah menjadi dua kelompok 4 digit bilangan biner, sehingga keseluruhannya menjadi 8 bit, tidak tergantung pada nilai bilangan desimalnya sendiri. Hasilnya sering disebut sebagai Binary Coded Decimal (BCD). Penyandian yang sering digunakan dikenal sebagai sandi 8421BCD. Selain itu juga dikenal penyandian lainnya dalam sistem BCD.

Tabel bilangan desimal dan 8421BCD – 2421BCD

Bilangan Desimal	8421BCD	2421BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1110
9	1001	1111

Sistem 8421BCD merupakan sandi yang paling banyak digunakan.

Contoh:

59_{10} = 0101 1001 (8421BCD)
843_{10} = 1000 0100 0011 (8421BCD)
59_{10} = 0101 1111 (2421BCD)
843_{10} = 1110 0100 0011 (2421BCD)

Kelemahan sistem BCD

Tidak dapat digunakan dalam operasi aritmetika diatas 9
Contoh:
6 + 5 = 11 sama dengan 1011 (bukan BCD)

Kode Excess-3

Sistem pengkodean lainnya adalah kode excess-3. Ciri dari kode ini adalah kelebihan tiga, sehingga 0 menjadi 3, 1 menjadi 4 dan seterusnya, sehingga kode ini diawali dengan digit 0011 sampai 1100.
Aturan kode excess-3.
Jika penjumlahan hasil tidak lebih dari 10 maka hasil dikurangi 3, tetapi bila hasil penjumlahan lebih dari 10 maka hasil ditambahkan dengan 3.

Kode AlphaNumeric

Kode-kode yang sudah kita bicarakan hanyalah kode-kode untuk menyatakan angka (bilangan) saja. Kode yang dapat digunakan baik untuk bilangan maupun karakter adalah sebagai berikut:

ASCII (American Standard Code for Information Interchange)
Kode ASCII terdiri dari 7 bit (seharusnya 8 bit, dengan bit yang paling kiri berisi 0) sehingga diperoleh $2^7 = 128$ kemungkinan untuk kode AlphaNumeric

EBCDIC (Extended Binery Coded Decimal Information Code)
Kode ini menggunakan 8 bit, $2^8 = 256$. Kode-kode lainnya pada dasarnya sama dengan kode ASCII, hanya tempatnya saja yang berubah.

PARITY

Untuk mengecek atau memeriksa dari kebenaran data / informasi yang ditransfer dalam microcomputer, diperlukan suatu tanda. Tanda untuk pengecekan ini dikenal dengan nama Parity. Untuk pengecekan data biasa digunakan bit parity, yaitu suatu bit yang ditambahkan pada data tersebut sehingga mendapatkan ketentuan yang diinginkan.
Ada dua macam bit parity yaitu:

- Parity ganjil (odd parity)
- Parity genap (even parity)

Pada parity ganjil, bit yang ditambahkan haruslah membuat jumlah digit 1 yang terdapat dalam data tersebut, termasuk bit paritynya, menjadi ganjil; sedangkan pada parity genap, jumlah digit satu harus genap.
Contoh:

BCD	Parity Ganjil	Parity Genap
1001	1001 **1**	1001 **0**
0100	0100 **0**	0100 **1**
1101	1101 **0**	1101 **1**
1100	1100 **1**	1100 **0**

Pengecekan parity dilakukan untuk mendeteksi bit yang salah.

KODE HUMMING

Untuk dapat mencetak dan membetulkan suatu data digunakan kode humming. Untuk pengkodean bilangan desimal, digunakan digit-digit BCD ditambah 3 bit lain sebagai parity. Jadi pada kode humming secara keseluruhan digunakan 7 bit untuk mengkodekan bilangan desimal.

Perhatikan contoh berikut ini:

	KODE HUMMING						
	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
BCD	**P1**	**P2**	8	**P4**	4	2	1
0	**0**	**0**	0	**0**	0	0	0
1	**1**	**1**	0	**1**	0	0	1
2	**0**	**1**	0	**1**	0	1	0
3	**1**	**0**	0	**0**	0	1	1
4	**1**	**0**	0	**1**	1	0	0
5	**0**	**1**	0	**0**	1	0	1
6	**1**	**1**	0	**0**	1	1	0
7	**0**	**0**	0	**1**	1	1	1
8	**1**	**1**	1	**0**	0	0	1
9	**0**	**0**	1	**1**	0	0	0

Dari contoh di atas, parity-parity P1, P2 dan P4 bertanggung jawab untuk membuat parity genap, dimana:

P1, adalah bit parity untuk bit 3, 5 dan 7
P2, adalah bit parity untuk bit 3, 6 dan 7
P4, adalah bit parity untuk bit 5, 6 dan 7

Jika terjadi kesalahan, maka parity P1, P2 dan P4 akan memberikan informasi bit yang salah sebagaimana tabel berikut ini:

P4	P2	P1	Bit yang salah
E	E	E	Tidak ada kesalahan
E	E	F	1
E	F	E	2
E	F	F	3
F	E	E	4
F	E	F	5
F	F	E	6
F	F	F	7

Keterangan:
E: BENAR; F: SALAH

Contoh:

Pengecekan Parity Ganjil.

1 0 1 0 1 1 1, maka BCDnya adalah 1111
P1 = 1 + 1 + 1 + 1 = Genap, maka SALAH
P2 = 0 + 1 + 1 + 1 = Ganjil, maka BENAR
P4 = 0 + 1 + 1 + 1 = Ganjil, maka BENAR
P4 P2 P1 = E E F, jadi bit 1 salah.

Pengecekan Parity Genap

1 0 0 0 0 0 1, maka BCDnya adalah 0001
P1 = 1 + 0 + 0 + 1 = Genap, maka BENAR
P2 = 0 + 0 + 0 + 1 = Ganjil, maka SALAH
P4 = 0 + 0 + 0 + 1 = Ganjil, maka SALAH
P4 P2 P1 = F F E, jadi bit 6 salah.

Kode Gray

Berikut ini tabel konversi desimal, biner dan gray.

Desimal	Biner	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

```
{Program Convert Decimal to Hexadecimal}

uses crt;

var
bil,sisa,hasil,i:integer;
a,x:string;

begin
clrscr;
write('Input Decimal Number   : ');
readln(bil);
sisa:=bil;
hasil:=bil;
a:='';
repeat
sisa:=hasil mod 16;
hasil:=hasil div 16;
str(sisa,x);
case sisa of
10: x:='A';
11: x:='B';
12: x:='C';
13: x:='D';
14: x:='E';
15: x:='F';
end;
a:=a+x;
until hasil=0;
write('Equivalent in Hexa     : ');
for i:=length(a) downto 1 do
begin
write(a[i]);
end;
readln;
end.
```

```
{Program Convert Decimal to Octal}

uses crt;

var
bil,sisa,hasil,i:integer;
a,x:string;

begin
clrscr;
write('Input Decimal Number   : ');
readln(bil);
sisa:=bil;
hasil:=bil;
a:='';
repeat
sisa:=hasil mod 8;
hasil:=hasil div 8;
str(sisa,x);
a:=a+x;
until hasil=0;
write('Equivalent in Octal    : ');
for i:=length(a) downto 1 do
begin
write(a[i]);
end;
readln;
end.
```

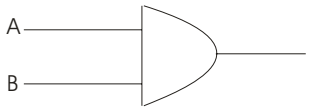
```
{Program Penjumlahan dua bilangan biner}

var
biner1,biner2,hasil,x:string;
code,digit1,digit2,i,carry,temp:integer;

begin
  write('Masukkan bilangan biner 1: ');
  readln(biner1);
  write('Masukkan bilangan biner 2: ');
  readln(biner2);
  carry:=0;
  for i:=length(biner1) downto 1 do
  begin
    val(biner1[i],digit1,code);
    val(biner2[i],digit2,code);
    temp:=digit1+digit2+carry;
    writeln('* Iterasi ke ',i);
    writeln('Penjumlahan Biner: ',digit1,' + ',digit2,' + carry ',carry,' = ',temp);
    if temp=2 then
    begin
      temp:=0;
      carry:=1;
    end
    else
    begin
      if temp=3 then
      begin
        temp:=1;
        carry:=1;
      end
      else
        carry:=0;
    end;
    str(temp,x);
    hasil:=hasil+x;
    if i=1 then
      if carry=1 then hasil:=hasil+'1';
  end;
  write('Hasilnya : ');
  for i:=length(hasil) downto 1 do
  write(hasil[i]);
  readln;
end.
```

GERBANG LOGIKA

Gerbang Logika adalah piranti dua keadaan yang memiliki output dua keadaan: output dengan 0 volt yang menyatakan logika 0 (rendah) dan output dengan tegangan tetap yang menyatakan logika 1 (tinggi). Gerbang logika dapat mempunyai beberapa input yang hanya terdiri dari dua kemungkinan 0 dan 1.
Dalam keadaan dasar, gerbang logika mempunyai dua input dan satu output. Bentuk-bentuk dasar gerbang logika beserta simbol dan tabel kebenarannya dapat dijelaskan sebagai berikut:

GERBANG AND (AND GATE)

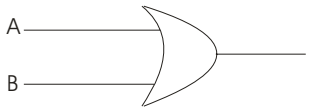
Gerbang AND digunakan untuk menghasilkan output 1 jika semua input adalah 1, jika tidak maka outputnya 0.



Simbol: F = A.B

GERBANG OR (OR GATE)

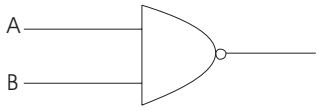
Gerbang OR akan memberikan output 1 jika salah satu dari inputnya adalah 1. Jika menginginkan output 0, maka semua inputnya harus 0.



Simbol: F = A+B

GERBANG NAND / NOT-AND (NAND GATE)

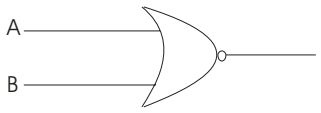
Gerbang NAND akan mempunyai output 0, bila semua inputnya adalah 1. Sebaliknya, jika ada sebuah input 0 pada sembarang input pada gerbang NAND, maka outputnya akan bernilai 1.



Simbol: F = $\overline{A.B}$

GERBANG NOR / NOT-OR (NOR GATE)

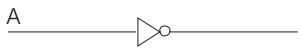
Gerbang NOR akan memberikan output 0 jika salah satu dari inputnya adalah 1. Jika diinginkan outputnya bernilai 1, maka semua input harus dalam keadaan 0.



Simbol: $F = \overline{A+B}$

Gerbang NOT (NOT GATE)

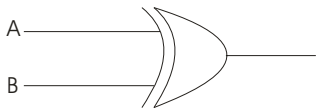
Gerbang NOT merupakan gerbang satu input yang berfungsi sebagai pembalik. Dikenal juga sebagai inverter, yaitu pembalik nilai. Jika inputnya 0, outputnya menjadi 1 dan sebaliknya.



Simbol: $\overline{A} = B$

Gerbang XOR (XOR GATE)

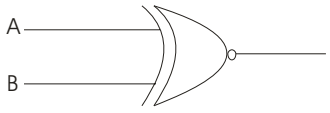
Gerbang XOR, dari kata Exclusive OR, akan memberikan output 1, jika masing-masing inputnya mempunyai nilai yang berbeda. Jika dilihat dari tabel kebenarannya, output pada gerbang XOR merupakan penjumlahan biner dari inputnya.



Simbol: $F = A+B$

GERBANG NXOR / NOT-XOR (NXOR GATE)

Gerbang NXOR merupakan ingkaran dari gerbang XOR. Hasil masing-masing outputnya adalah output gerbang XOR yang diinvers.



Simbol: $F = \overline{A+B}$

TABEL KEBENARAN (TRUTH TABLE)

Untuk menganalisa hasil output dari masing-masing gerbang, dibutuhkan suatu tabel kebenaran untuk dapat menentukan output yang dihasilkan.

A	B	AND	OR	NAND	NOR	XOR	NXOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

ALJABAR BOOLE

Jika kita perhatikan, simbol-simbol yang digunakan untuk mewakili gerbang logika mempunyai aturan khusus. Simbol-simbol ini nantinya kita sebut sebagai notasi boole yang menjadi dasar dari aljabar boole. Aljabar Boole digunakan sebagai alat untuk menganalisa segala sesuatu yang berhubungan dengan logika, termasuk juga untuk menganalisa rangkaian logika. Seperti yang sudah ditulis sebelumnya, notasi-notasi dari Aljabar Boole adalah sebagai berikut:

Fungsi AND dinyatakan dengan sebuah titik (.) sehingga sebuah gerbang AND yang mempunyai dua input A dan B, outputnya dapat dinyatakan sebagai:
F = A.B atau F = B.A
Sedangkan untuk gerbang AND dengan tiga input A, B dan C outputnya dapat ditulis:
F = A.B.C
Tanda titik sering tidak ditulis, sehingga notasi tersebut dapat ditulis sebagai:
F = ABC

Fungsi OR dinyatakan dengan sebuah tanda plus (+), sehingga gerbang OR dua input A dan B, outputnya dapat ditulis sebagai:
F = A+B atau F = B+A

Fungsi NOT dinyatakan dengan garis atas (overline) pada inputnya sehingga gerbang NOT dengan input A mempunyai output yang dapat dituliskan sebagai:
F = $\bar{A}$ (dibaca sebagai not A atau bukan A)

Fungsi XOR dinyatakan dengan tanda +. Untuk gerbang XOR dua input, outputnya dapat ditulis sebagai:
F = A+B

Untuk **Fungsi NAND, NOR dan NXOR**, kita tinggal menambahkan garis atas (overline) diatas notasi yang sudah dituliskan, karena gerbang-gerbang ini adalah ingkaran atau invers dari gerbang-gerbang AND, OR dan XOR.

Teori Boole

Dalam penggunaannya, kita nantinya akan sering menjumpai rangkaian logika yang terdiri dari beberapa gerbang. Beberapa Teori Boole berikut ini dapat digunakan untuk melakukan penyederhanaan terhadap suatu fungsi logika sehingga dapat menghasilkan fungsi logika yang sederhana.

1. **Komutative / Duality**
   $A+B = B+A$
   $A.B = B.A$

2. **Assosiative**
   $(A+B)+C = A+(B+C)$
   $(A.B).C = A.(B.C)$

3. **Distributive**
   $A.(B+C) = (A.B)+(A.C)$
   $A+(B.C) = (A+B).(A+C)$

4. **Identity**
   $A+A = A$
   $A.A = A$

5. **Negasi**
   $(\bar{A}) = \bar{A}$
   $(\bar{A}) = A$

6. $A+A.B = A$
   $A.(A+B) = A$

7. $0+A = A$
   $1.A = A$
   $1+A = 1$
   $0.A = 0$

8. $A+ \bar{A} = 1$
   $A. \bar{A} = 0$
   $A. \bar{A} B = 0$

9. $A+ \bar{A} B = A+B$
   $A.(\bar{A}+B) = A.B$

10. **Teori De Morgan**
    $(\overline{A+B}) = \bar{A}. \overline{B}$
    $(\overline{A.B}) = \bar{A} + \overline{B}$

Pembuktian Teori Boole dapat dilakukan dengan berbagai cara. Sebagai contoh, cara yang paling sering digunakan adalah pembuktian dengan tabel kebenaran.

Contoh:

Buktikan bahwa A+(B.C) = (A+B).(A+C)
Jawaban:

A	B	C	B.C	A+(B.C)	A+B	A+C	(A+B).(A+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Jadi A+(B.C) = (A+B).(A+C) … terbukti.

Buktikan bahwa A+$\bar{A}$.B = A+B
Jawaban:

A	B	$\bar{A}$.B	A+ $\bar{A}$.B	A+B
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

Jadi A+ $\bar{A}$.B = A+B … terbukti.

Penyederhanaan Fungsi Logika dengan Teori Boole

Teori Boole digunakan untuk menyederhanakan sebuah fungsi logika sehingga rangkaian logikanya akan menghasilkan gerbang-gerbang yang berjumlah minimum.

Contoh:

Sederhanakan fungsi-fungsi logika berikut:

$F = \overline{A}.B + A.B + \overline{A}.\overline{B}$

$F = A + A.\overline{B} + \overline{A}.B$

$F = \overline{A.B + A.B}$

Langkah – langkah merancang rangkaian logika

1. Nyatakan persoalan dengan jelas dan tentukan input / outputnya.
2. Siapkan tabel kebenaran dari persoalan tersebut.
3. Tentukan pernyataan logikanya.
4. Sederhanakan bila mungkin.
5. Gambar Rangkaian Logikanya.

Fungsi SOP (Sum of Product)
Disebut juga fungsi positif (minterm)
Output yang dihasilkan: 1

Fungsi POS (Product of Sum)
Disebut juga fungsi negatif (maxterm)
Output yang dihasilkan: 0