

# BAB 1

## Pengenalan Pemrograman Komputer

### 1.1 Tujuan

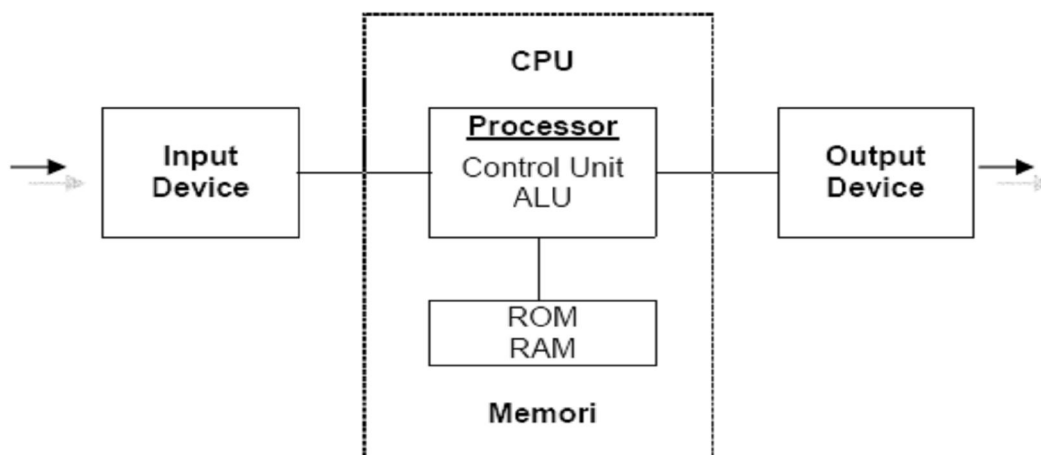
Bagian ini akan membahas dasar – dasar komponen dari komputer meliputi *hardware* (perangkat keras) dan *software* (perangkat lunak). Kami juga akan menyertakan gambaran global tentang bahasa pemrograman dan sirkulasi pemrograman. Akan dibahas pula pada akhir pembahasan ini mengenai sistem dan konversi numerik.

Pada akhir pembahasan, diharapkan pembaca dapat :

- Mengidentifikasi perbedaan komponen pada komputer
- Mengetahui tentang bahasa pemrograman komputer dan kategorinya
- Mengetahui alur kerja pembuatan program dan mengaplikasikannya pada pemecahan masalah
- Mempelajari tentang berbagai sistem numerik dan metode konversinya.

### 1.2 Pendahuluan

Kata komputer berasal dari bahasa Latin yaitu *Computare* yang artinya *menghitung*. Dalam bahasa Inggris disebut *to compute*. Secara definisi komputer diterjemahkan sebagai sekumpulan alat elektronik yang saling bekerja sama, dapat menerima data (input), mengolah data (proses) dan memberikan informasi (output) serta terkoordinasi dibawah kontrol program yang tersimpan di memorinya. Jadi cara kerja komputer dapat kita gambarkan sebagai berikut :



Gambar 1: Skema IO Komputer

Komputer memiliki dua komponen utama. Yang pertama adalah *hardware* (perangkat keras) yang tersusun atas komponen elektronik dan mekanik. Komponen utama yang lain yaitu *software* (perangkat lunak). Komponen ini terdiri atas data dan aplikasi – aplikasi komputer.

## 1.3 Komponen Dasar Komputer

### 1.3.1 **HARDWARE**

#### 1.3.1.1 Central Processing Unit (CPU)

Processor, merupakan bagian dari perangkat keras komputer yang melakukan pemrosesan aritmatika dan logika serta pengendalian operasi komputer secara keseluruhan. Prosesor terdiri atas dua bagian utama, yaitu ALU (Arithmetic Logic Unit) dan Control Unit. Kecepatan kerja prosesor biasanya ditentukan oleh kecepatan clock dari Control Unit-nya.

Contoh : jika prosesor memiliki frekuensi clock 350 MHz, berarti kecepatan pemrosesan satu instruksinya =  $T = 1/f = 1/(350 \times 10^6 \text{ Hz})$ , =  $0,286 \times 10^{-8}$  detik.

#### 1.3.1.2 Memori

**Memori** adalah media penyimpan data pada komputer. Memori, berdasarkan fungsinya dibagi menjadi dua yaitu :

##### a. **Primary Memory**

Dipergunakan untuk menyimpan data dan instruksi dari program yang sedang dijalankan. Seringkali disebut juga sebagai RAM. Karakteristik dari memori primer adalah :

- o Volatile (informasi ada selama komputer bekerja. Ketika komputer dipadamkan, informasi yang disimpannya juga hilang)
- o Berkecepatan tinggi
- o Akses random (acak)

##### b. **Secondary Memory**

Dipergunakan untuk menyimpan data atau program biner secara permanen. Karakteristik dari memori sekunder adalah

- o Non volatile atau persisten
- o Kecepatan relatif rendah (dibandingkan memori primer)
- o Akses random atau sekuensial

Contoh memori sekunder : floppy, harddisk, CD ROM, magnetic tape, optical disk, dll. Dari seluruh contoh tersebut, yang memiliki mekanisme akses sekuensial adalah magnetic tape

<b>Memori Utama (RAM)</b>	<b>Memori Sekunder (ROM)</b>	<b>Kategori</b>
Cepat	Lambat	Kecepatan
Mahal	Murah	Harga
Kecil	Besar	Kapasitas
Ya	Tidak	Volatile

Tabel 1: Perbandingan antara memori utama dan memori sekunder

### 1.3.1.3 Input Dan Output Device

Input-Output Device, merupakan bagian yang berfungsi sebagai penghubung antara komputer dengan lingkungan di luarnya. Dapat dibagi menjadi dua kelompok, yaitu :

#### a. Input Device (Piranti Masukan)

Berfungsi sebagai media komputer untuk menerima masukan dari luar. Beberapa contoh piranti masukan :

- o Keyboard
- o Mouse
- o Touch screen
- o Scanner
- o Camera

#### b. Output Device (Piranti Keluaran)

Berfungsi sebagai media komputer untuk memberikan keluaran. Beberapa contoh piranti keluaran :

- o Monitor
- o Printer
- o Speaker
- o Plotter

### 1.3.2 Software

Merupakan program-program komputer yang berguna untuk menjalankan suatu pekerjaan sesuai dengan yang dikehendaki. Program tersebut ditulis dengan bahasa khusus yang dimengerti oleh komputer. Program dapat dianalogikan sebagai instruksi yang akan dijalankan oleh prosessor. Software terdiri dari beberapa jenis, yaitu :

#### 1. *Sistem Operasi*, seperti DOS, Unix, Novell, OS/2, Windows.

Adalah software yang berfungsi untuk mengaktifkan seluruh perangkat yang terpasang pada komputer sehingga masing-masing dapat saling berkomunikasi. Tanpa ada sistem operasi maka komputer tidak dapat difungsikan sama sekali.

**2. Program Utility**, seperti Norton Utility, Scandisk, PC Tools.

Program utility berfungsi untuk membantu atau mengisi kekurangan/kelemahan dari sistem operasi, misalnya PC Tools dapat melakukan perintah format sebagaimana DOS, tapi PC Tools mampu memberikan keterangan dan animasi yang bagus dalam proses pemformatan. File yang telah dihapus oleh DOS tidak dapat dikembalikan lagi tapi dengan program bantu hal ini dapat dilakukan.

**3. Program Aplikasi**, seperti GL, MYOB, Payroll.

Merupakan program yang khusus melakukan suatu pekerjaan tertentu, seperti program gaji pada suatu perusahaan. Maka program ini hanya digunakan oleh bagian keuangan saja tidak dapat digunakan oleh departemen yang lain. Umumnya program aplikasi ini dibuat oleh seorang programmer komputer sesuai dengan permintaan/kebutuhan seseorang/lembaga/perusahaan guna keperluan interennya.

**4. Program Paket**

Merupakan program yang dikembangkan untuk kebutuhan umum, seperti :

- o Pengolah kata /editor naskah : Wordstar, MS Word, Word Perfect, AmiPro
- o Pengolah angka / lembar kerja : Lotus123, MS Excell, QuattroPro, dll
- o Presentasi : MS PowerPoint
- o Desain grafis : CorelDraw, PhotoShop

**5. Compiler.**

Komputer hanya memahami satu bahasa, yaitu bahasa mesin. Bahasa mesin adalah terdiri dari nilai 0 dan 1. Sangatlah tidak praktis dan efisien bagi manusia untuk membuat program yang terdiri dari nilai 0 dan 1, maka dicarilah suatu cara untuk menterjemahkan sebuah bahasa yang dipahami oleh manusia menjadi bahasa mesin. Dengan tujuan inilah, diciptakan *compiler*.

## 1.4 Sekilas Bahasa Pemrograman

### 1.4.1 Apa yang Disebut Bahasa Pemrograman?

Bahasa pemrograman adalah teknik komunikasi standar untuk mengekspresikan instruksi kepada komputer. Layaknya bahasa manusia, setiap bahasa memiliki tata tulis dan aturan tertentu.

Bahasa pemrograman memfasilitasi seorang programmer secara tepat menetapkan data apa yang sedang dilakukan oleh komputer selanjutnya, bagaimana data tersebut disimpan dan dikirim, dan apa yang akan dilakukan apabila terjadi kondisi yang variatif.

Bahasa pemrograman dapat diklasifikasikan menjadi tingkat rendah, menengah, dan tingkat tinggi. Pergeseran tingkat dari rendah menuju tinggi menunjukkan kedekatan terhadap "bahasa manusia".

## 1.4.2 Kategori Bahasa Pemrograman

### 1. Bahasa Pemrograman Tingkat Tinggi

Merupakan bahasa tingkat tinggi yang mempunyai ciri-ciri mudah dimengerti karena kedekatannya terhadap bahasa sehari – hari. Sebuah pernyataan program diterjemahkan kepada sebuah atau beberapa mesin dengan menggunakan **compiler**.

Sebagai contoh adalah : JAVA, C++, .NET

### 2. Bahasa Pemrograman Tingkat Rendah

Bahasa pemrograman generasi pertama. Bahasa jenis ini sangat sulit dimengerti karena instruksinya menggunakan bahasa mesin. Disebut juga dengan bahasa assembly merupakan bahasa dengan pemetaan satu – persatu terhadap instruksi komputer. Setiap intruksi assembly diterjemahkan dengan menggunakan **assembler**.

### 3. Bahasa Pemrograman Tingkat Menengah

Dimana penggunaan instruksi telah mendekati bahasa sehari – hari, walaupun masih cukup sulit untuk dimengerti karena menggunakan singkatan – singkatan seperti STO yang berarti simpan (STORE) dan MOV yang artinya pindah (MOVE). Yang tergolong dalam bahasa ini adalah Fortran.

## 1.5 Alur Pembuatan Program

Seorang programmer tidak melakukan pembuatan dan pengkodean program secara begitu saja, namun mengikuti perencanaan dan metodologi yang terstruktur yang memisahkan proses suatu aplikasi menjadi beberapa bagian.

Berikut ini langkah – langkah sistematis dasar dalam menyelesaikan permasalahan pemrograman :

1. Mendefinisikan masalah
2. Menganalisa dan membuat rumusan pemecahan masalah
3. Desain Algoritma dan Representasi
4. Pengkodean, Uji Coba dan pembuatan dokumentasi

Untuk memahami langkah dasar dalam pemecahan masalah dalam sebuah komputer mari kita mendefinisikan sebuah permasalahan yang akan diselesaikan langkah demi langkah sebagaimana metodologi pemecahan masalah yang akan dibahas selanjutnya. Masalah yang akan kita selesaikan akan didefinisikan pada bagian selanjutnya.

### 1.5.1 Definisi Permasalahan

Seorang programmer umumnya mendapatkan tugas berdasarkan sebuah permasalahan. Sebelum sebuah program dapat terdesain dengan baik untuk menyelesaikan beberapa permasalahan, masalah–masalah yang terjadi harus dapat diketahui dan terdefinisi

dengan baik untuk mendapatkan detail persyaratan input dan output.

Sebuah pendefinisian yang jelas adalah sebagian dari penyelesaian masalah. Pemrograman komputer mempersyaratkan untuk mendefinisikan program terlebih dahulu sebelum membuat suatu penyelesaian masalah.

Mari kita definisikan sebuah contoh permasalahan :

***"Buatlah sebuah program yang akan menampilkan berapa kali sebuah nama tampil pada sebuah daftar"***

### **1.5.2 Analisa Permasalahan**

Setelah sebuah permasalahan terdefinisi secara memadai, langkah paling ringkas dan efisien dalam penyelesaian harus dirumuskan.

Umumnya, langkah berikutnya meliputi memecahkan masalah tersebut menjadi beberapa bagian kecil dan ringkas.

**Contoh masalah :**

*Menampilkan jumlah kemunculan sebuah nama pada daftar*

**Input Terhadap Program :**

*Daftar Nama, Nama yang akan dicari*

**Output Dari Program :**

*Jumlah kemunculan nama yang dicari*

### **1.5.3 Desain Algoritma dan Representasi**

Setelah kita mengetahui dengan baik dan jelas mengenai permasalahan yang ingin diselesaikan, langkah selanjutnya yaitu membuat rumusan algoritma untuk menyelesaikan permasalahan. Dalam pemrograman komputer penyelesaian masalah didefinisikan dalam langkah demi langkah.

Algoritma adalah urutan langkah – langkah logis penyelesaian masalah yang disusun secara sistematis dan logis. Logis merupakan kunci dari sebuah algoritma. Langkah-langkah dalam algoritma harus logis dan bernilai benar atau salah.

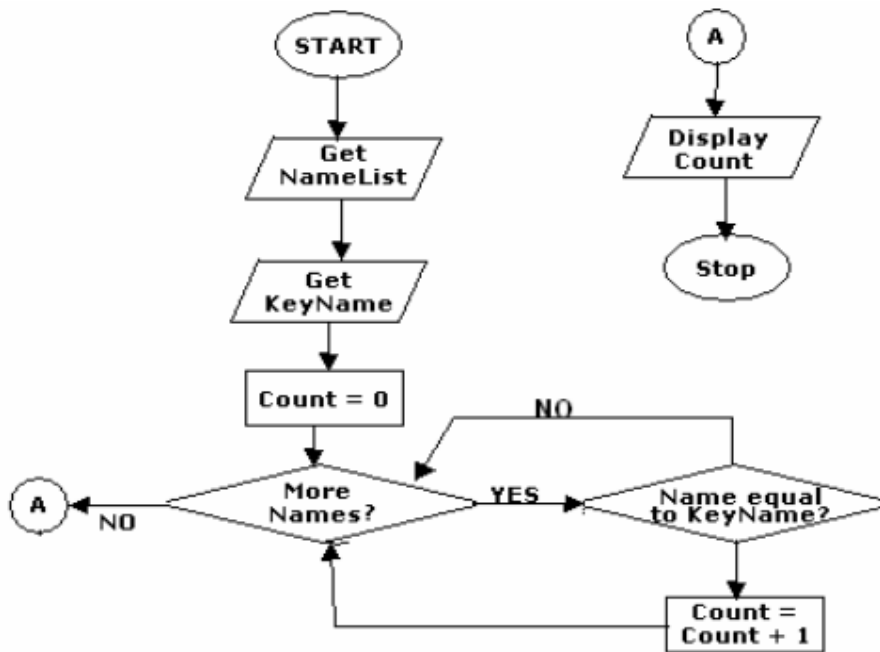
Algoritma dapat diekspresikan dalam bahasa manusia, menggunakan presentasi grafik melalui sebuah FlowChart (diagram alir) ataupun melalui PseudoCode yang menjembatani antara bahasa manusia dengan bahasa pemrograman.

Berdasarkan permasalahan yang terjadi pada bagian sebelumnya, bagaimanakah kita dapat memberikan solusi penyelesaian secara umum dalam sebuah alur yang dapat dengan mudah dimengerti?

**Mengekspresikan cara penyelesaian melalui bahasa manusia :**

1. Tentukan daftar nama
2. Tentukan nama yang akan dicari, anggaplah ini merupakan sebuah kata kunci
3. Bandingkan kata kunci terhadap setiap nama yang terdapat pada daftar
4. Jika kata kunci tersebut sama dengan nama yang terdapat pada daftar, tambahkan nilai 1 pada hasil perhitungan
5. Jika seluruh nama telah dibandingkan, tampilkan hasil perhitungan (output)

Mengekspresikan cara penyelesaian melalui FlowChart :



Gambar 2: Contoh Flowchart

Mengekspresikan solusi melalui Pseudocode :

```

listNama    =    Daftar Nama
keyNama     =    Nama yang dicari
hitung     =    0
    
```

```


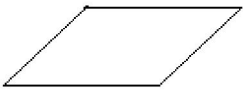
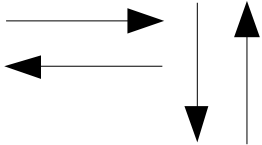
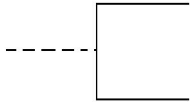
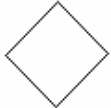
Untuk setiap nama pada Daftar Nama lakukan :
Jika nama == keyNama
Hitung = Hitung + 1
Tampilkan Hitung
    
```

### 1.5.3.1 Simbol Flowchart dan Artinya

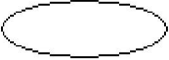

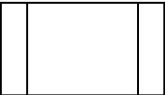
Flowchart adalah representasi grafis dari langkah – langkah yang harus diikuti dalam menyelesaikan suatu permasalahan yang terdiri atas sekumpulan simbol, dimana masing – masing simbol merepresentasikan kegiatan tertentu. Flowchart diawali dengan penerimaan input dan diakhiri dengan penampilan output.

Sebuah flowchart pada umumnya tidak menampilkan instruksi bahasa pemrograman, namun menetapkan konsep solusi dalam bahasa manusia ataupun notasi matematis.

Berikut ini akan dibahas tentang simbol–simbol yang digunakan dalam menyusun flowchart, kegiatan yang diwakili serta aturan yang diterapkan dalam penggunaan simbol tersebut :

<b>Simbol</b>	<b>Nama</b>	<b>Pengertian</b>
	<b>Simbol Proses</b>	Simbol ini digunakan untuk melambangkan kegiatan pemrosesan input. Dalam simbol ini, kita dapat menuliskan operasi-operasi yang dikenakan pada input, maupun operasi lainnya. Sama seperti aturan pada simbol input, penulisan dapat dilakukan secara satu per satu maupun secara keseluruhan.
	<b>Simbol Input – Output (IO)</b>	Merepresentasikan fungsi I/O yang membuat sebuah data dapat diproses (input) atau ditampilkan (output) setelah mengalami eksekusi informasi.
	<b>Simbol Garis Alir</b>	Simbol ini digunakan untuk menghubungkan setiap langkah dalam flowchart dan menunjukkan kemana arah aliran diagram. Anak panah ini harus mempunyai arah dari kiri ke kanan atau dari atas ke bawah. Anak panah ini juga dapat diberi label, khususnya jika keluar dari symbol percabangan.
	<b>Simbol Anotasi</b>	Merepresentasikan informasi deskriptif tambahan, komentar atau catatan penjelasan. Dalam simbol ini, kita dapat menuliskan komentar apapun dan sebanyak apapun, hal ini berguna untuk memperjelas langkah-langkah dalam flowchart. Garis vertical dan garis putus–putus dapat ditempatkan pada sisi kanan maupun kiri.
	<b>Simbol Percabangan</b>	Simbol ini digunakan untuk melambangkan percabangan, yaitu pemeriksaan terhadap suatu kondisi. Dalam simbol ini, kita menuliskan keadaan yang harus dipenuhi. Hasil dari pemeriksaan dalam simbol ini adalah YES atau NO. Jika pemeriksaan menghasilkan keadaan benar, maka jalur yang



<b>Simbol</b>	<b>Nama</b>	<b>Pengertian</b>
		harus dipilih adalah jalur yang berlabel Yes, sedangkan jika pemeriksaan menghasilkan keadaan salah, maka jalur yang harus dipilih adalah jalur yang berlabel No.
	<b>Simbol Terminator</b>	Terminator berfungsi untuk menandai awal dan akhir dari suatu flowchart. Simbol ini biasanya diberi label START untuk menandai awal dari flowchart, dan label STOP untuk menandai akhir dari flowchart. Jadi dalam sebuah flowchart pasti terdapat sepasang terminator yaitu terminator start dan stop.
	<b>Simbol Konektor</b>	Simbol konektor digunakan pada waktu menghubungkan suatu langkah dengan langkah lain dalam sebuah flowchart dengan keadaan on page atau off page. Konektor on page digunakan untuk menghubungkan suatu langkah dengan langkah lain dari flowchart dalam satu halaman, sedangkan konektor off page digunakan untuk menghubungkan suatu langkah dengan langkah lain dari flowchart dalam halaman yang berbeda. Konektor ini biasanya dipakai saat media yang kita gunakan untuk menggambar flowchart tidak cukup luas untuk memuat gambar secara utuh, jadi perlu dipisah-pisahkan. Dalam sepasang konektor biasanya diberi label tertentu yang sama agar lebih mudah diketahui pasangannya.
	<b>Simbol Prosedur</b>	Simbol ini berperan sebagai blok pembangun dari suatu program. Prosedur memiliki suatu flowchart yang berdiri sendiri diluar flowchart utama. Jadi dalam simbol ini, kita cukup menuliskan nama prosedurnya saja, jadi sama seperti jika kita melakukan pemanggilan suatu prosedur pada program utama (main program). Sama dengan aturan pada simbol percabangan, penulisan nama prosedur dilakukan secara satu per satu.

Tabel 2: Simbol dari Flowchart

### 1.5.4 Pengkodean, Uji Coba dan Pembuatan Dokumentasi

Setelah membentuk algoritma, maka proses penulisan program dapat dimulai. Menggunakan algoritma sebagai pedoman, maka kode program dapat ditulis sesuai bahasa pemrograman yang dipilih.

Setelah menyelesaikan seluruh kode program, langkah selanjutnya yaitu menguji program tersebut apakah telah berfungsi sesuai tujuannya untuk memberikan suatu solusi untuk menyelesaikan suatu masalah. Bilamana terjadi kesalahan – kesalahan logika atas program, disebut juga sebagai *bugs*, maka kita perlu untuk mengkaji ulang rumusan/algorithm yang telah dibuat, kemudian memperbaiki implementasi kode program yang mungkin keliru. Proses ini disebut dengan *debugging*.

Terdapat dua tipe kesalahan (*errors*) yang akan dihadapi seorang programmer. Yang pertama adalah *compile-time error*, dan yang kedua adalah *runtime error*.

*Compile-time errors* muncul jika terdapat kesalahan penulisan kode program. *Compiler* akan mendeteksi kesalahan yang terjadi sehingga kode tersebut tidak akan bisa dikompilasi.

Terlupakannya penulisan *semi-colon* (;) pada akhir sebuah pernyataan program atau kesalahan ejaan pada beberapa perintah dapat disebut juga sebagai *compile-time error*.

*Compiler* tidaklah sempurna sehingga tidak dapat mengidentifikasi seluruh kemungkinan kesalahan pada waktu kompilasi. Umumnya kesalahan yang terjadi adalah kesalahan logika seperti perulangan tanpa akhir. Tipe kesalahan ini disebut dengan *runtime error*.

Sebagai contoh, penulisan kode pada program terlihat tanpa kesalahan, namun pada saat anda menelusuri struktur logika kode tersebut, bagian yang sama pada kode tereksekusi berulang-ulang tanpa akhir. Pada kasus tersebut *compiler* tidak cukup cerdas untuk menangkap kesalahan tipe ini pada saat proses kompilasi. Sehingga saat program dijalankan, aplikasi atau bahkan keseluruhan komputer mengalami *hang* karena mengalami proses perulangan yang tidak berakhir. Contoh lain dari *run-time error* adalah perhitungan atas nilai yang salah, kesalahan penetapan kondisi dan lain sebagainya.

Untuk memudahkan dalam memeriksa suatu kesalahan suatu program ataupun memahami jalannya program, kita juga perlu membuat suatu dokumentasi dari program yang dibuat. Dokumentasi tersebut berisi informasi mulai dari tujuan dan fungsi program, algoritma, serta cara penggunaannya.

## 1.6 Sistem Numerik dan Konversi

Bilangan dapat disajikan dalam beberapa cara. Cara penyajiannya tergantung pada Basis (BASE) bilangan tersebut. Terdapat 4 cara utama dalam penyajian bilangan.

### 1.6.1 Sistem Bilangan Desimal

Manusia umumnya menggunakan bilangan pada bentuk desimal. Bilangan desimal adalah sistem bilangan yang berbasis 10. Hal ini berarti bilangan – bilangan pada sistem ini terdiri dari 0 sampai dengan 9. Berikut ini beberapa contoh bilangan dalam bentuk desimal :

$126_{10}$  (umumnya hanya ditulis 126)  
 $11_{10}$  (umumnya hanya ditulis 11)

### 1.6.2 Sistem Bilangan Biner

Bilangan dalam bentuk biner adalah bilangan berbasis 2. Ini menyatakan bahwa bilangan yang terdapat dalam sistem ini hanya 0 dan 1. Berikut ini contoh penulisan dari bilangan biner :

$1111110_2$   
 $1011_2$

### 1.6.3 Sistem Bilangan Oktal

Bilangan dalam bentuk oktal adalah sistem bilangan yang berbasis 8. Hal ini berarti bilangan–bilangan yang diperbolehkan hanya berkisar antara 0 – 7. Berikut ini contoh penulisan dari bilangan oktal :

$176_8$   
 $13_8$

### 1.6.4 Sistem Bilangan Heksadesimal

Bilangan dalam sistem heksadesimal adalah sistem bilangan berbasis 16. Sistem ini hanya memperbolehkan penggunaan bilangan dalam skala 0 – 9, dan menggunakan huruf A – F, atau a – f karena perbedaan kapital huruf tidak memiliki efek apapun. Berikut ini contoh penulisan bilangan pada sistem heksadesimal :

$7E_{16}$   
 $B_{16}$

<b>Heksadesimal</b>	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
<b>Nilai Dalam Desimal</b>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Tabel 3: Bilangan heksadesimal dan perbandingannya terhadap desimal

Berikut adalah perbandingan keseluruhan sistem penulisan bilangan :

<i>Desimal</i>	<i>Biner</i>	<i>Oktal</i>	<i>Heksadesimal</i>
126 <sub>10</sub>	1111110 <sub>2</sub>	176 <sub>8</sub>	7E <sub>16</sub>
11 <sub>10</sub>	1011 <sub>2</sub>	13 <sub>8</sub>	B <sub>16</sub>

Tabel 4: Contoh Konversi Antar Sistem Bilangan

## 1.6.5 Konversi

### 1.6.5.1 Desimal ke Biner / Biner ke Desimal

Untuk mengubah angka desimal menjadi angka biner digunakan metode pembagian dengan angka 2 sambil memperhatikan sisanya. Ambil hasil bagi dari proses pembagian sebelumnya, dan bagi kembali bilangan tersebut dengan angka 2. Ulangi langkah-langkah tersebut hingga hasil bagi akhir bernilai 0 atau 1. Kemudian susun nilai-nilai sisa dimulai dari nilai sisa terakhir sehingga diperoleh bentuk biner dari angka bilangan tersebut.

**Sebagai Contoh :**

$$126_{10} = ?_2$$

	<i>Hasil Bagi</i>	<i>Nilai Sisa</i>	
126 / 2 =	63	0	
63 / 2 =	31	1	
31 / 2 =	15	1	
15 / 2 =	7	1	
7 / 2 =	3	1	
3 / 2 =	1	1	
1 / 2 =	0	1	

Dengan menuliskan nilai sisa mulai dari bawah ke atas, didapatkan angka biner  $1111110_2$ .

Konversi bilangan biner ke desimal didapatkan dengan menjumlahkan perkalian semua bit biner dengan perpangkatan 2 sesuai dengan posisi bit tersebut.

**Sebagai Contoh :**

$$11001101_2 = ?_{10}$$

* Biner	1	1	0	0	1	1	0	1	11001101
* Desimal	128	64	0	0	8	4	0	1	205
* Pangkat	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		$X^{1-6}$

Angka desimal 205 diperoleh dari penjumlahan angka yang diarsir. Setiap biner yang bernilai 1 akan mengalami perhitungan, sedangkan yang bernilai 0 tidak akan dihitung karena hanya akan menghasilkan nilai 0.

### 1.6.5.2 Desimal ke Oktal/Heksadesimal dan Oktal/Heksadesimal ke Desimal

Pengubahan bilangan desimal ke bilangan oktal atau bilangan heksadesimal pada dasarnya sama dengan konversi bilangan desimal ke biner. Perbedaannya terletak pada bilangan pembagi. Jika pada konversi biner pembaginya adalah angka 2, maka pada konversi oktal pembaginya adalah angka 8, sedangkan pada konversi heksadesimal pembaginya adalah 16.

**Contoh konversi Oktal :**

$$126_{10} = ?_8$$

	<i>Hasil Bagi</i>	<i>Nilai Sisa</i>
$126 / 8 =$	15	6
$15 / 8 =$	1	7
$1 / 8 =$	`	1

Dengan menuliskan nilai sisa dari bawah ke atas, kita peroleh bilangan oktal  $176_8$

**Contoh konversi Heksadesimal :**

$$126_{10} = ?_{16}$$

	<i>Hasil Bagi</i>	<i>Nilai Sisa</i>
$126 / 16 =$	7	14 (E)
$7 / 16 =$		7

Dengan menuliskan nilai sisa dari bawah ke atas, kita peroleh bilangan Heksadesimal  $7E_{16}$

Konversi bilangan Oktal dan Heksadesimal sama dengan konversi bilangan Biner ke Desimal. Perbedaanya hanya terdapat pada penggunaan angka basis. Jika sistem Biner menggunakan basis 2, maka pada bilangan Oktal, basis yang digunakan adalah 8 dan pada bilangan Heksadesimal adalah angka 16.

**Contoh konversi Oktal :**

$$176_8 = ?_{10}$$

<i>Posisi</i>	<b>2</b>	<b>1</b>	<b>0</b>
<b>Digit Oktal</b>	<b>1</b>	<b>7</b>	<b>6</b>

$$6 \times 8^0 = 6$$

$$7 \times 8^1 = 56$$

$$1 \times 8^2 = 64$$

$$\text{TOTAL: } 126$$

**Contoh konversi Heksadesimal :**

$$7E_{16} = ?_{10}$$

<b>Posisi</b>	<b>1</b>	<b>0</b>	
<b>Digit Heksadesimal</b>	<b>7</b>	<b>E</b>	
			$14 \times 16^0 = 14$
			$7 \times 16^1 = 112$
			<b>TOTAL: 126</b>

**1.6.5.3 Biner ke Oktal dan Oktal ke Biner**

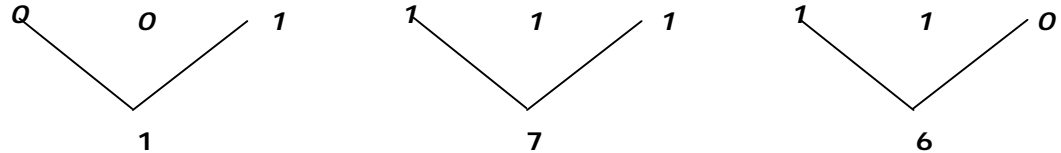
Untuk mengubah bilangan biner ke oktal, kita pilah bilangan tersebut menjadi 3 bit bilangan biner dari kanan ke kiri. Tabel berikut ini menunjukkan representasi bilangan biner terhadap bilangan oktal :

<i>Digit Oktal</i>	<i>Representasi Biner</i>
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

*Tabel 5: Bilangan oktal dan perbandingannya dalam sistem biner*

Sebagai contoh :

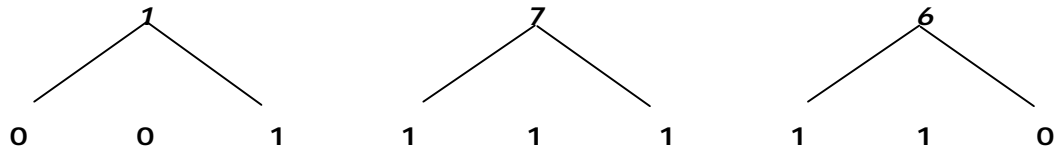
$$1111110_2 = \underline{?}_8$$



Mengubah sistem bilangan oktal menjadi bilangan biner dilakukan dengan cara kebalikan dari konversi biner ke oktal. Dalam hal ini masing–masing digit bilangan oktal diubah langsung menjadi bilangan biner dalam kelompok tiga bit, kemudian merangkai kelompok bit tersebut sesuai urutan semula.

Sebagai contoh :

$$176_8 = \underline{?}_2$$



#### 1.6.5.4 Biner ke Heksadesimal dan Heksadesimal ke Biner

Pengubahan bilangan Biner ke Heksadesimal dilakukan dengan pengelompokan setiap empat bit Biner dimulai dari bit paling kanan. Kemudian konversikan setiap kelompok menjadi satu digit Heksadesimal. Tabel berikut menunjukkan representasi bilangan Biner terhadap digit Heksadesimal :

<i>Digit Heksadesimal</i>	<i>Representasi Biner</i>
0	0000
1	0001
2	0010
3	0011
4	0100

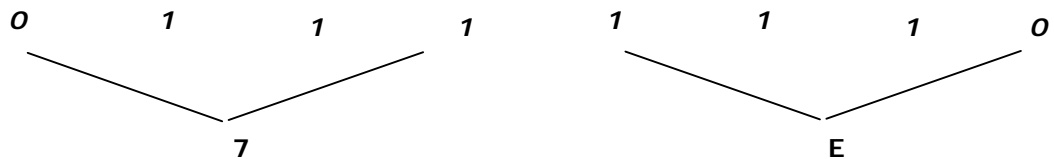


<i>Digit Heksadesimal</i>	<i>Representasi Biner</i>
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Tabel 6: Bilangan heksadesimal dan konversinya dalam biner

Sebagai contoh :

$$11111110_2 = \underline{7E}_{16}$$



Konversi bilangan Heksadesimal ke Biner dilakukan dengan membalik urutan dari proses pengubahan Biner ke Heksadesimal. Satu digit Heksadesimal dikonversi menjadi 4 bit Biner.

Sebagai contoh :

$$7E_{16} = \underline{11111110}_2$$



## 1.7 Latihan

### 1.7.1 Menyusun Algoritma

Dari permasalahan–permasalahan di bawah ini, susunlah sebuah algoritma untuk menyelesaikannya. Anda dapat menyusunnya dengan menggunakan pseudocode ataupun flowchart.

1. Memasak Roti
2. Menggunakan Komputer di Laboratorium
3. Menghitung rata–rata dari 3 buah bilangan

### 1.7.2 Konversi Sistem Bilangan

Konversikan bilangan – bilangan berikut ini :

1.  $1980_{10}$  ke sistem bilangan Biner, Heksadesimal dan Oktal
2.  $1001001101_2$  ke sistem bilangan Desimal, Heksadesimal dan Oktal
3.  $76_8$  ke sistem bilangan Biner, Heksadesimal dan Desimal
4.  $43F_{16}$  ke sistem bilangan Biner, Desimal dan Oktal