TEKNIK OTOMASI INDUSTRI JILID 2 untuk SMK

Agus Putranto, dkk.

BUKU SEKOLAH ELEKTRONIK bse

JILID 2

Agus Putranto, dkk.

Teknik Otomasi Industri

untuk

Sekolah Menengah Kejuruan

Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional

Agus Putranto, dkk.

Agus Putranto dkk

TEKNIK OTOMASI INDUSTRI

SMK

JILID 2

Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional

TEKNIK OTOMASI INDUSTRI

Untuk SMK

JILID 2

Penulis : Agus Putranto
Abdul Mukti
Djoko Sugiono
Syaiful Karim
Arie Eric Rawung
Sodikin Susa'at
Sugiono

Perancang Kulit : TIM

Ukuran Buku : 18,2 x 25,7 cm

PUT
t PUTRANTO, Agus
Teknik Otomasi Industri untuk SMK Jilid 2 /oleh Agus Putranto, Abdul Mukti, Djoko Sugiono, Syaiful Karim, Arie Eric Rawung, Sodikin Susa'at, Sugiono ---- Jakarta : Direktorat Pembinaan Sekolah Menengah Kejuruan, Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah, Departemen Pendidikan Nasional, 2008.
xii, 250 hlm
Daftar Pustaka : LAMPIRAN A.
Glosarium : LAMPIRAN B.
ISBN : 978-979-060-089-8
ISBN : 978-979-060-091-1

Diterbitkan oleh
Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional
Tahun 2008

KATA SAMBUTAN

Puji syukur kami panjatkan kehadirat Allah SWT, berkat rahmat dan karunia Nya, Pemerintah, dalam hal ini, Direktorat Pembinaan Sekolah Menengah Kejuruan Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah Departemen Pendidikan Nasional, pada tahun 2008, telah melaksanakan penulisan pembelian hak cipta buku teks pelajaran ini dari penulis untuk disebarluaskan kepada masyarakat melalui *website* bagi siswa SMK.

Buku teks pelajaran ini telah melalui proses penilaian oleh Badan Standar Nasional Pendidikan sebagai buku teks pelajaran untuk SMK yang memenuhi syarat kelayakan untuk digunakan dalam proses pembelajaran melalui Peraturan Menteri Pendidikan Nasional Nomor 12 tahun 2008.

Kami menyampaikan penghargaan yang setinggi-tingginya kepada seluruh penulis yang telah berkenan mengalihkan hak cipta karyanya kepada Departemen Pendidikan Nasional untuk digunakan secara luas oleh para pendidik dan peserta didik SMK di seluruh Indonesia.

Buku teks pelajaran yang telah dialihkan hak ciptanya kepada Departemen Pendidikan Nasional tersebut, dapat diunduh (*download*), digandakan, dicetak, dialihmediakan, atau difotokopi oleh masyarakat. Namun untuk penggandaan yang bersifat komersial harga penjualannya harus memenuhi ketentuan yang ditetapkan oleh Pemerintah. Dengan ditayangkannya *soft copy* ini akan lebih memudahkan bagi masyarakat untuk mengaksesnya sehingga peserta didik dan pendidik di seluruh Indonesia maupun sekolah Indonesia yang berada di luar negeri dapat memanfaatkan sumber belajar ini.

Kami berharap, semua pihak dapat mendukung kebijakan ini. Selanjutnya, kepada para peserta didik kami ucapkan selamat belajar dan semoga dapat memanfaatkan buku ini sebaik-baiknya. Kami menyadari bahwa buku ini masih perlu ditingkatkan mutunya. Oleh karena itu, saran dan kritik sangat kami harapkan.

Jakarta, 17 Agustus 2008
Direktur Pembinaan SMK

KATA PENGANTAR

Puji syukur kehadirat Alloh, dengan tersusunnya buku Teknik Otomasi Industri ini semoga dapat menambah khasanah referensi khususnya di bidang tekologi industri yang akhir-akhir ini mulai berkembang di Indonesia.

Isi buku ini sengaja disajikan secara praktis dan lengkap sehingga dapat membantu para siswa Sekolah Menengah Kejuruan (SMK), mahasiswa, guru serta para praktisi industri. Teknik Otomasi Industri yang selama ini dideskripsikan secara variatif dan adaptif terhadap perkembangan serta kebutuhan berbagai kalangan praktisi industri. Penekanan dan cakupan bidang yang dibahas dalam buku ini sangat membantu dan berperan sebagai sumbangsih pemikiran dalam mendukung pemecahan permasalahan yang selalu muncul didalam disain, pengendalian / pemgontrolan suatu sistem.

Oleh karena itu, buku ini disusun secara integratif antar disiplin ilmu yaitu teknik elektronika analog, elektronika daya,teknik digital, pemrograman dan elektronika daya yang saling mendukung sehingga skill yang diperlukan terkait satu dengan lainnya. Secara tuntas, kualitas maupun manajemen proses control standar yang berlaku di tingkat internasional termasuk didalam wilayah pembahasan.

Tim penulis mengucapkan terima kasih kepada berbagai pihak yang telah membantu materi naskah serta dorongan semangat dalam penyelesaian buku ini. Kami sangat berharap dan terbuka untuk masukan serta kritik konstruktif dari para pembaca sehingga dimasa datang buku ini lebih sempurna dan implementatif.

Tim Penulis

DAFTAR ISI

2.2 KOMPONEN LISTRIK DAN ELEKTRONIKA

BAB III. Dasar Teknik Digital

3.1 Aljabar boolean

Rangkaian digital memliki dua tingkatan diskrit. Salah satu contoh termudahnya adalah saklar dengan dua macam kemungkinan yaitu buka dan tutup. Aljabar Boolean adalah rumusan matematika untuk menjelaskan hubungan logika antara fungsi pensaklaran digital. Aljabar boolean meiliki dasar dua macam nilai logika. Hanya bilangan biner yang terdiri dari angka 0 dan 1 maupun pernyataan rendah dan tinggi.

3.2 Operasi logika dasar AND, OR dan NOT

Suatu fungsi logika atau operasi logika yang dimaksud dalam aljabar Boolean adalah suatu kombinasi variable biner seperti misalnya pada masukan dan keluaran dari suatu rangkaian digital yang dapat ditunjukkan bahwa di dalam aljabar Boolean semua hubungan logika antara variable variable biner dapat dijelaskan oleh tiga operasi logika dasar yaitu :

- Operasi NOT (negation)
- Operasi AND (conjuction)
- Operasi OR (disconjuction

Operasi operasi tersebut dijelaskan dalam tiga bentuk yaitu :

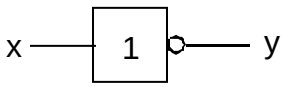
1. Tabel fungsi (tablel kebenaran) yang menunjukkan keadaan semua variabel masukan dan keluaran untuk setiap kemungkinan.
2. Simbol rangkaian untuk menjelaskan rangkaian digital.
3. Persamaan fungsi.

3.2.1 Operasi logika NOT

Fungsi NOT adalah membalik sebuah variable biner, misalnya jika masukannya adalah 0 maka keluarannya adalah 1. Gambar 3.1. memperlihatkan 3 macam bentuk penggambaran fungsi operasi NOT.

Tabel kebenaran Simbol rangkaian Persamaan fungsi

x	y
1	0
0	1



$y = \overline{x}$

a b c

Gambar 3.1 Operasi NOT

3.2.2 Operasi logika AND

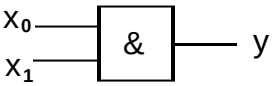
Operasi AND menghubungkan paling sedikit dua masukan variable dan dapat lebih variabel masukannnya mulai x_0, x_1 sampai x_n dan satu variabel keluaran y. Variabel keluaran akan berlogika 1 hanya jika semua masukannya x_0, x_1 sampai x_n dalam keadaan 1.
Gambar 3.2. Menggambarkan 3 macam penggambaran fungsi operasi logika AND.

Tabel kebenaran

x_1	x_0	y
0	0	0
0	1	0
1	0	0
1	1	1

a

Simbol rangkaian



b

Persamaan fungsi

$y = x_0 \wedge x_1$

c

Gambar 3.2 Operasi AND

3.2.3 Operasi logika OR

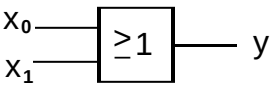
Operasi OR juga menghubungkan paling sedikit dua masukan variable dan dapat lebih variabel masukannnya mulai x_0, x_1 sampai x_n dan satu variabel keluaran y.
Variabel keluaran akan berlogika 0 hanya jika semua masukannya x_0, x_1 sampai x_n dalam keadaan 0.
Gambar 3.3. Menggambarkan 3 macam penggambaran fungsi operasi logika OR.

Tabel kebenaran

x_1	x_0	y
0	0	0
0	1	1
1	0	1
1	1	1

a

Simbol rangkaian



b

Persamaan fungsi

$y = x_0 \vee x_1$

c

Gambar 3.3 Operasi OR

3.3 Operasi logika kombinasi NAND, NOR dan Exclusive OR

3.3.1. Operasi logika NAND

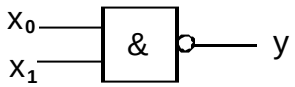
Operasi NAND merupakan kombinasi dua buah operasi logika dasar AND dan NOT. Masukan paling sedikit dua variable, dan dapat lebih variabel masukannnya mulai x_0, x_1 sampai x_n dan satu variabel keluaran y.
Variabel keluaran akan berlogika 0 hanya jika semua masukannya x_0, x_1 sampai x_n dalam keadaan 1.
Gambar 3.4. Menggambarkan 3 macam penggambaran fungsi operasi logika NAND.

Tabel kebenaran

x_1	x_0	y
0	0	1
0	1	1
1	0	1
1	1	0

a

Simbol rangkaian



b

Persamaan fungsi

$$y = \overline{x_0 \wedge x_1}$$

c

Gambar 3.4 Operasi NAND

3.3.2. Operasi logika NOR

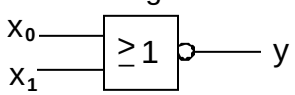
Operasi NOR merupakan kombinasi dua buah operasi logika dasar OR dan NOT. Masukan paling sedikit dua variable, dan dapat lebih variabel masukannnya mulai x_0, x_1 sampai x_n dan satu variabel keluaran y.
Variabel keluaran akan berlogika 1 hanya jika semua masukannya x_0, x_1 sampai x_n dalam keadaan 0.
Gambar 3.5. Menggambarkan 3 macam penggambaran fungsi operasi logika NOR.

Tabel kebenaran

x_1	x_0	y
0	0	1
0	1	0
1	0	0
1	1	0

a

Simbol rangkaian



b

Persamaan fungsi

$$y = \overline{x_0 \vee x_1}$$

c

Gambar 3.5. Operasi NOR

3.3.3. Operasi logika Exclusive OR

Operasi Excclusive OR biasanya disebut dengan EXOR menghubungkan dua masukan variable x_0 dan x_1 serta memiliki satu variabel keluaran y.

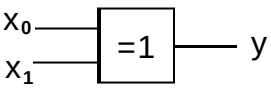
Gambar 3.6. Menggambarkan 3 macam penggambaran fungsi operasi logika Excclusive OR x_0 dan x_1

Tabel kebenaran

x_1	x_0	y
0	0	0
0	1	1
1	0	1
1	1	0

a

Simbol rangkaian



b

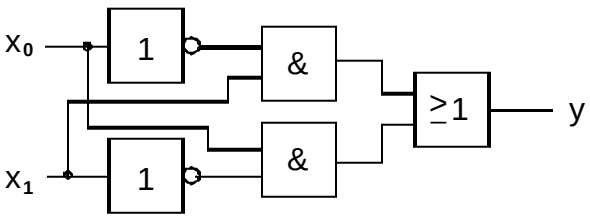
Persamaan fungsi

$y = x_0 \forall x_1$

c

Gambar 3.6. Operasi EXOR

Tabel kebenaran memprlihatkan bahwa ketika x_0 dan x_1 = 0 atau ketika ketika x_0 dan x_1 = 1, keluaran y akan berlogika 0. Variabel keluaran akan berlogika 1 hanya jika kondisi logika kedua masukannya x_0 dan x_1 berbeda.
Pada prakteknya, sebuah operasi EXOR dapat dibangun dari operasi logika AND, OR dan NOT seperti yang diperlihatkan pada gambar 3.7.



Gambar 3.7. Operasi EXOR yang dibangun dari operasi logika dasar.

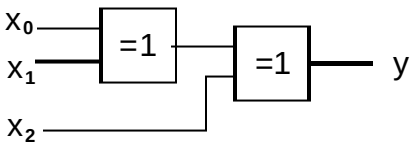
Pada Gambar 3.8. suatu operasi EXOR dapat dihubungkan bertingkat (kaskade) sehingga secara keseluruhan operasi EXOR tersebut menjadi memiliki tiga variable masukan x_0 , x_1 dan x_2 serta sebuah variable keluaran y.
Perilaku EXOR dengan tiga masukan tersebut ditunjukkan oleh table kebenaran di bawah ini.

Tabel kebenaran

x_2	x_1	x_0	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

a

EXOR dihungkan secara kaskade



b

Gambar 3.8. EXOR dengan tiga masukan

3.4. Multiplekser

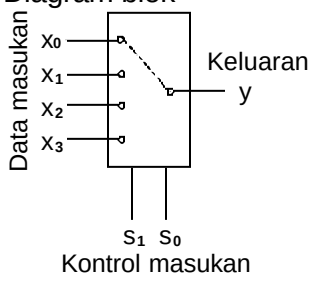
Multiplekser adalah suatu rangkaian logika yang memiliki banyak masukan dan satu keluaran. Fungsinya adalah seprti saklar pilih yang dapat dikontrol. Keluaran bergantung dari sinyal kontrol S_i, dan hanya satu dari masukan X_i yang tersambung ke keluaran. Dimana sinyal masukan yang terdiri dari lebih dari satu jalur diproses sehingga didapatkan satu keluaran.
Jika multiplexer memiliki 4 masukan x_0, x_1, x_2 dan x_3 maka sinyal kontrol yang diperlukan sebanyak dua masukan s_0 dan s_1 sehingga secara keseluruhan semua masukan multiplexer berjumlah 6 masukan.

Tabel kebenaran

Kontrol masukan		Keluaran
S_1	S_0	y
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

a

Diagram blok



b

Gambar 3.9. Multiplexer dengan empat masukan

3.5. Dekoder

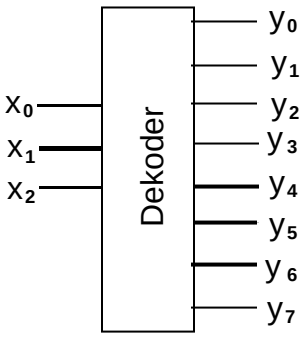
Dekoder adalah suatu rangkaian logika yang memiliki sedikit masukan dan banyak keluaran.

Tabel kebenaran

x_2	x_1	x_0	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

a

Diagram blok



b

Gambar 1.10. Dekoder tiga masukan delapan keluaran

Dekoder pada Gambar 3.10. memiliki tiga masukan x_0, x_1 dan x_2 dan delapan keluaran (y_0 – y_7). Bergantung dari kombinasi masukan. Keluaran akan berganti ke 0 maupun 1.
Kombinasi masukan dan keluaran yang dikeluarkan bergantung dari jenis atau tipe dekoder yang digunakan.
Dari tabel Gambar 1.10.a, kita ambil contoh pada keluaran y_6 menjadi 1 ketika input $x_0 = 0$, $x_1 = 1$, dan $x_2 = 1$.
Pada prakteknya, dekoder yang paling banyak dipergunakan adalah yang keluarannya dibalik.
Rumus umum dekoder adalah memiliki n maukan dan 2 pangkat n keluaran.

3.6. Flip-flop

3.6.1. RS Flip-flop

Flip-flop adalah suatu rangkaian bistabil dengan triger yang dapat menghasilkan kondisi logika 0 dan 1 pada keluarannya. Keadaan dapat dipengaruhi oleh satu atau kedua masukannya. Tidak seperti fungsi gerbang logika dasar dan kombinasi, keluaran suatu flip-flop sering tergantung pada keadaan sebelumnya. Kondisi tersebut dapat pula menyebabkan keluaran tidak berubah atau dengan kata lain terjadi kondisi memory. Oleh sebab itu flip-flop dipergunakan sebagai elemen memory.

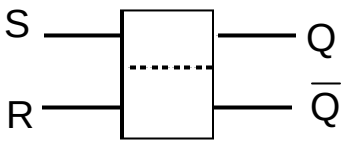
Rangkaian flip-flop yang paling sederhana adalah RS Flip-flop yang memiliki dua masukan yaitu R = Reset dan S = Set serta dua keluaran Q dan $\overline{Q}$.

Tabel kebenaran

S	**R**	**Q**	**$\overline{Q}$**
0	0	Tidak berubah	
0	1	0	1
1	0	1	0
1	1	Tidak tentu	

a

Simbol



b

Gambar 3.11. RS Flip-flop

Sesuai dengan namanya, keluaran flip flop Q = 1 dan Q = 0 pada saat S = 1 dan R = 0,dan reset ketika S = 0 dan R = 1 akan menghasilkan keluaran Q = 0 dan Q = 1. Kondisi tersebut adalah kondisi satbbil dari RS flip-flop.

Ketika kedua masukan R dan S berlogika 0, keluaran flip-flop tidak berubah tetap seperti pada kondisi sebelumnya. Tetapi ketika kedua masukan R dan S berlogika 1 maka keluaran flip-flop tidak dapat diramalkan karena kondisinya tidak tentu tergantung pada toleransi komponen dan tunda waktu temporal dan lain sebagainya dan kondisi tersebut dapat diabaikan.

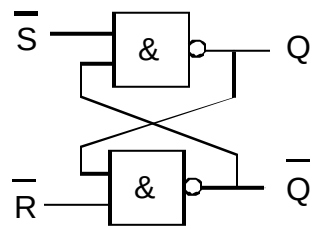
Pada prakteknya sebuah RS Flip-flop dapat dibangun dari rangkaian dua buah gerbang AND yang saling dihubungkan silang seperti ditunjukan pada Gambar 3.12.

Tabel kebenaran

$\overline{S}$	$\overline{R}$	**Q**	**$\overline{Q}$**
0	0	Tidak tentu	
0	1	0	1
1	0	1	0
1	1	Tidak berubah	

a

Rangkaian RS Flip flop



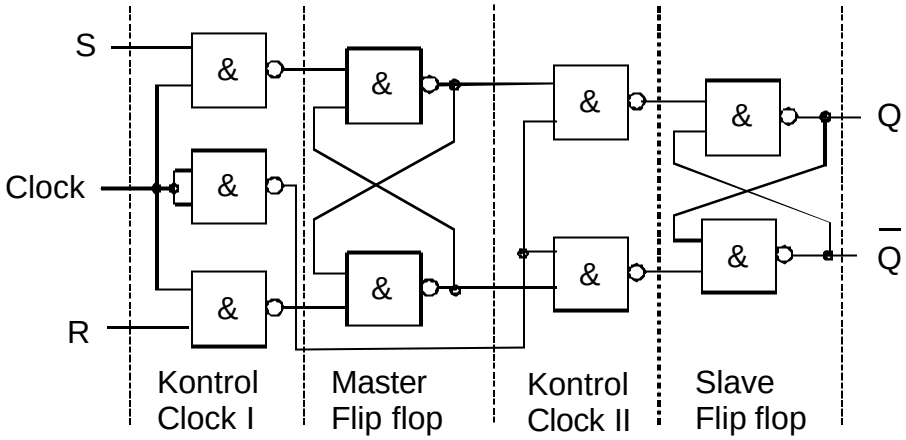
b

Gambar 3.12. Rangkaian RS Flip-flop dengan gerbang NAND

Berbeda dengan flip flop dengan Gambar 3.11, keluaran dari flip-flop adalah kebalikan dari flip-flop tersebut. Hal ini dapat dilihat dari adanya garis di atas variabel inputnya.

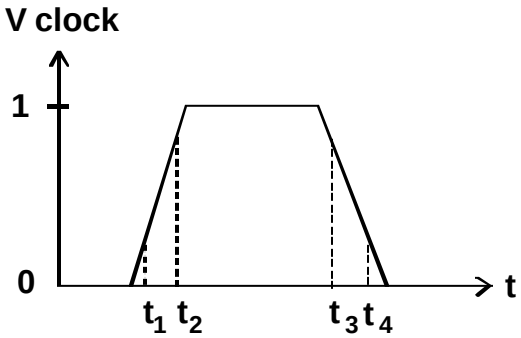
Lebih lanjut tipe yang sangat penting dari flip-flop adalah master slave flip-flop atau disebut juga dua memory yang pada dasarnya dibangun dari dua flip-flop yang terhubung secara seri. Jalur kontrol dapat diatur dari

sebuah clock melalui penambahan sebuah gerbang NAND. Gambar rangkaian dasrnya ditunjukkan dalam Gambar 3.13.



Gambar 3.13. Master-Slave Flip-flop menggunakan NAND

Pertama kita lihat pada master flip-flop. Jika masukan clock adalah 0 kedua keluaran dari kontrol clock I adalah 1. Ini artinya bahwa suatu perubahan keadaan pada masukan S dan R tidak berpengaruh pada master flip-flop. Flip flop tersebut mempertahankan keadaan. Di sisi lain jika masukan clock adalah 1 maka keadaan dari S dan R menentukan keadaan master flip-flop.
Slave flip flop memperlihatkan perilaku yang sama. Kadang kontrol clock adalah dibalik oleh sebuah inverter. Ini artinya bahwa clock 1 dari master flip flop menjadi 0 pada slve flip flop.
Operasi flip-flop ini dijelaskan lebih mudah dari sekuensial temporal dari pulsa clock seperti ditunjukan oleh Gambar 3.14.



Gambar 3.14. Sekuensial temporal untuk master slave flip flop

t1 : Ketika pulsa clock muncul dari 0 ke 1 terjadi toleransi daerah 0 ke arah 1 keluaran clock terbalik ke 0. Misalnya keluaran slave flip flop akan off dan mempertahankan kondisi.

t2 : Ketika pulsa clock muncul dari 0 ke 1 mencapai batas terendah dari toleransi daerah 1 masukan dari master flip flop adalah dapat diatur, misalnya master flip flop dipengaruhi oleh masukan R dan S.

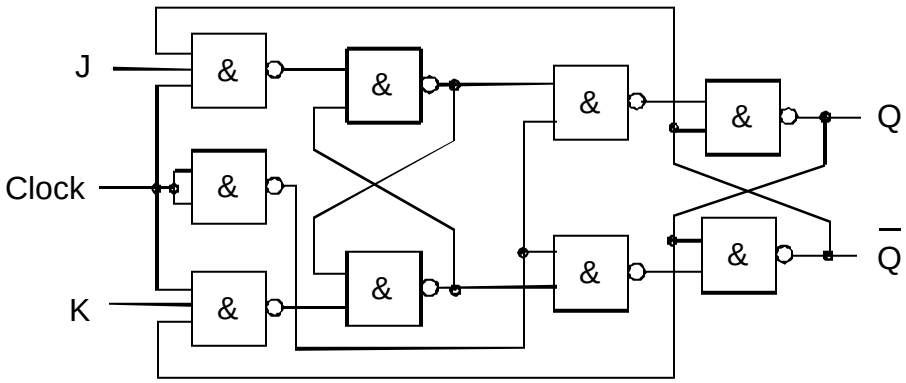
t3 : Ketika pulsa clock turun dari 1 ke 0 terjadi toleransi daerah 1 ke arah 0 masukan master flip flop kembali ditahan. Mmisalnya master flip flop menghasilkan keadaan baru.

T4 : Ketika pulsa clock turun dari 1 ke 0 mencapai batas tertinggi dari toleransi daerah 0 masukan dari master flip flop adalah dapat diatur, misalnya master flip flop dipengaruhi oleh masukan R dan S.

Hasilnya bahwa pengaruh masukan R dan S terjadi pada interval t_1 sampai t_2 data dikirim ke flip flop dan pada saat t_4 baru data dikirim ke keluaran. Selama masukan clock 0 data tersimpan di dalam flip flop.

3.6.2. JK Flip-flop

Pengembangan master slave flip flop pada prakteknya yang terpenting adalah Master slave JK flip flop yang dibangun dengan menyambungkan keluaran ke masukan gerbang seperti diperlihatkan Gambar 41.15.



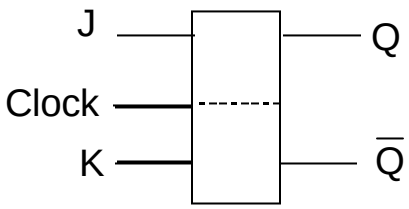
Gambar 3.15. Rangkaian JK Flip flop menggunakan NAND

Tabel Kebenaran

t_n		t_{n+1}	
K	**J**	**Q**	$\overline{\mathbf{Q}}$
0	0	Q	$\overline{Q}$
0	1	1	0
1	0	0	1
1	1	$\overline{Q}$	Q

a

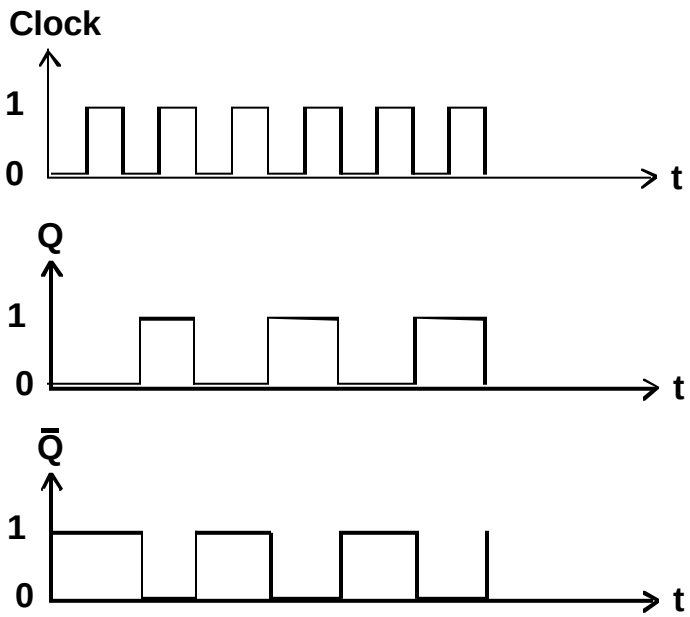
Simbol



b

Gambar 3.16. Tabel kebenaran dan symbol JK Flip flop

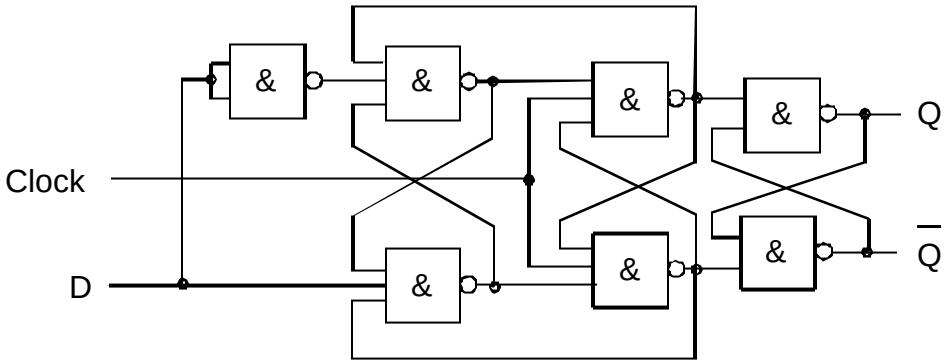
Keadaan masukan J = 1 dan K = 0 menghasilkan keluaran Q = 1 dan $\overline{Q}$ = 0 setalh pulsa clock. Untuk J = K = 1 keluaran akan selalu berubah setiap kali pulsa clock diberikan.



Gambar 3.17. Diagram pulsa JK flip flop ketika masukan J = K = 1

3.6.3. D Flip-flop

Suatu flip-flop yang mirip JK Master Ive flip-flop untuk J = K = 1 adalah dikenal dengan nama D flip-flop. Versi yang paling banyak dipergunakan dalam praktek diperlihatkan pada Gambar 3.18.



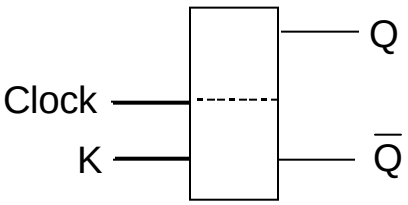
Gambar 3.18. Rangkaian D Flip flop menggunakan NAND

Tabel Kebenaran

t_n	t_{n+1}	
D	**Q**	$\overline{\mathbf{Q}}$
0	0	1
1	1	0

a

Simbol



b

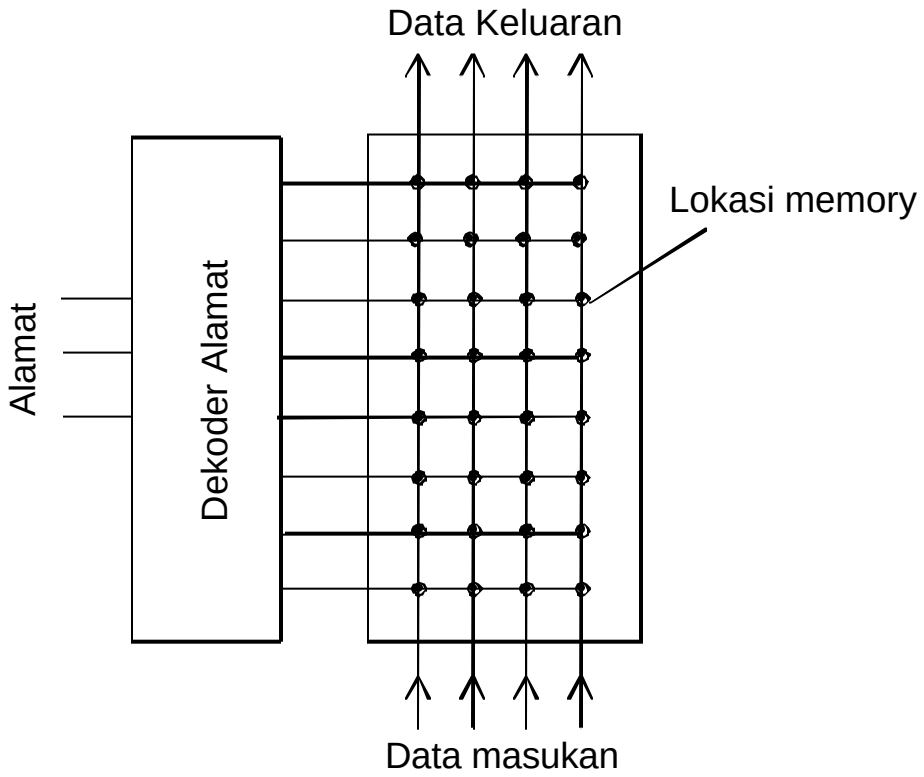
Gambar 3.19. Tabel kebenaran dan symbol D Flip flop

Kelebihan D flip-flop dibandingkan dengan JK flip-flop bahwa data masukan dikirim ke keluaran selama pulsa clock berubah dari o ke 1. Jika clock = 1 dan data masukan di D berubah, perubahan tersebut tidak lama berpengaruh terhadap keadaan keluaran. Suatu perubahan di D selama clock = 1 mengakibatkan pengaruh ke keluaran hanya pada perubahan 0 ke 1 berikutnya. Karena perlambatan internal memungkinkan dengan flip flop ini mengenal sebuah umpan balik misalnya dari $\overline{Q}$ ke D tanpa menghasilkan oscilasi. Karena kelebihan tersebut sering D flip flop ini disebut sebagai Delay flip-flop.

3.7. Memory

Elemen memory sangat menentukan dalam sistem mikrokomputer. Memory ini diperlukan untuk menyimpan program yang ada pada komputer dan data. Berbagai macam tipe memory dibedakan menurut ukuran , mode operasi, teknologi dan lain sebagainya dapat diperoleh

dipasaran. Memori dalam sistem mikrokomputer dapat juga dikatakan sebagai elemen penyimpanan matriks dua dimensi yang dibentuk dari flip – flop. Gambar 1.20. memperlihatkan suatu rangkaian dasar memory 8 X 4 bit. Setaiap titik simpul mewakili satu lokasi memory, satu bit (bit adalah singkatan dari binary digit atau angka dengan dua nilai 0 atau 1).



Gambar 3.20. Struktur dasar suatu memory

Jika data akan dituliskan ataupun dibaca dari suatu kombinasi antara 000 dan 111 yang juga disebut sebagai alamat yang yang dilalui oleh jalur alamat. Dekoder alamat digunakan untuk memilih satu diantara 8 jalur di dalam matriks dan isi dari jalur tersebut sehingga data words 4 bits dapat ditulis ataupun dibaca melalui data masukan ataupun data keluaran.

Penambahan jalur kontrol yang tidak ditunjukkan dalam Gambar 3.20. sangat diperlukan untuk mengontrol baca, tulis dan lain sebagainya.
Komponen memory konvensional di pasaran pada umumnya memiliki data work dengan ukuran 1, 2, 4 atau 8 bits yang mampu menyimpan data 1, 2, 4 atau 8 bit.
Jumlah jalur pada matrik biasanya 2 pangkat n, yang mana n adalah jalur alamat dari 2^n yang dapat dipilih. Suatu komponen memory seharusnya memiliki spesifikasi sebagai berikut :

Kapasitas 1kilobit = 1 k = 2^{10} bits = 1024 bits = 1024 elemen memory
Organisasi 256 x 4, contoh 256 = 2^8 jalur dari setiap 4 bits

Secara garis besar, memory dibagi menjadi 2 macam tipe :

1. Memory baca/tulis
   Memory ini memilki fungsi untuk menulis data yang nantinya akan di baca kembali. Jenis memori seperti ini disebut juga dengan RAM (Random Access Memory).

2. Memory hanya baca (Read Only Memory)
   Atau yang disingkat ROM. Data dapat diisikan ke dalam memory ketika proses pembuatan dan kemudian data dapat dibaca oleh pengguna.

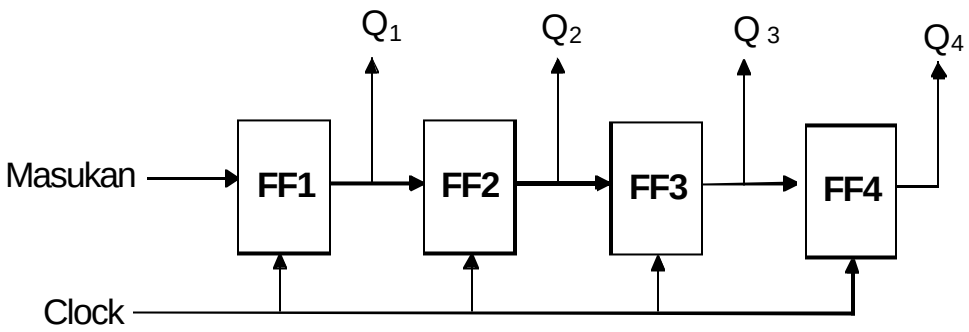
Memory dapat dibedakan berdasarkan teknologinya seperti misalnya bipolar, MOS (Metal Oxide Semiconductor) atau sepertihalnya RAM, perlu tidaknya merefresh simpanan data secara periodic baik dengan operasi dinamik maupun statis.
Pada jenis memory dinamik, elemen penyimpan pada prinsipnya adalah ssuatu kapasitor yang diisi dan direfresh secara periodic .
Pada jenis memory statis, elemen penyimpan data pada prinsipnya adalah suatu flip-flop yang tidak memerlukan refreshing.
Ada dua jenis ROM, yaitu :

- PROM (Programmable Read Only Memory) yang hanya dapat diprogram satu kali oleh pengguna.

- RePROM (Re Programmable Read Only Memory)
  Jenis RePROM ini dapat diprogram dan bila tidak diperlukan akan dapat dihapus diprogram lagi oleh para pengguna.

3.8. Register geser

Pada dasarnya merupakan koneksi seri dari Flip flop yang menggunakan clock untuk memindah data yang ada pada Flip flop sebelumnya dan dipindah ke data yang ada pada Flip flop selanjutnya.



Gambar 3.21. Diagram blok register geser

Mode Operasinya adalah sebagai berikut :

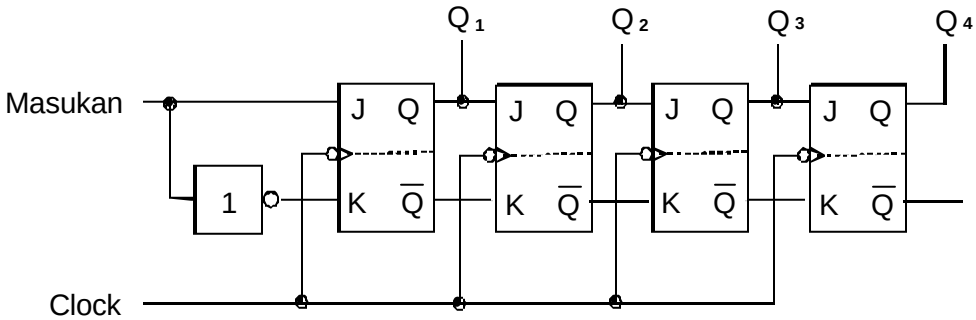
Dengan mengaumsikan sebelumnya bahwa Clock pertama, semua keluaran dari Q_1 sampai dengan Q_4 adalah 0 dan masukan input adalah 1. Setelah itu Data ini akan ditampilkan pada output Q_1 pada Clock pertama (t_{n+1}). Sebelum ke Clock ke 2, Input kembali menjadi 0. Dan pada saat clock kedua (t_{n+2}) keluaran Q1 menjadi 0 dan Q_2 menjadi 1. Setelah Clock t_{n+3} $Q_1 = 0$, $Q_2 = 0$ dan Q_3 menjadi 1. Setelah clock ke (t_{n+4}) , Q_4 menjadi kondisi 1.
Kemungkinan diatas dapat diilustrasikan pada tabel kebenaran berikut :

Tabel kebenaran register geser

Clock	t_n	t_{n+1}	t_{n+2}	t_{n+3}	t_{n+4}	t_{n+5}
Masukan	1	0	0	0	0	0
Q_1	0	1	0	0	0	0
Q_2	0	0	1	0	0	0
Q_3	0	0	0	1	0	0
Q_4	0	0	0	0	1	0

Pada tabel di atas dijelaskan ketika memasuki clock ke 5 semua keluaran kembali menjadi NOL.

Berikut ini adalah register geser dengan menggunakan JK Flip-flop :



Gambar 3.22. Register geser 4 bit menggunakan JK Flip-flop

Selama Shift regsiter tersebut hanya memasang 4 buah Flip-flop, maka informasi yang akan didapat hanya sebanyak 4 buah, oleh karena itulah dinamakan sebagai 4-bit Shift register atau register geser 4 bit.

Dengan Shift Register ini ada 2 kemungkinan dasar untuk membaca kembali informasi yang ada, yaitu :

1. Setelah clock ke 4 informasi telah masuk secara simultan yang ditampilkan pada keluaran Q_1 sampai dengan Q_4. Informasi ini

dibaca secara serial (satu setelah yang lainnya) dan dapat juga dibaca secara parallel.

2. Jika hanya Q_4 saja yang digunakan sebagai output keluran, data yang telah dimasukkan secara serial juga bisa dibaca secara serial.

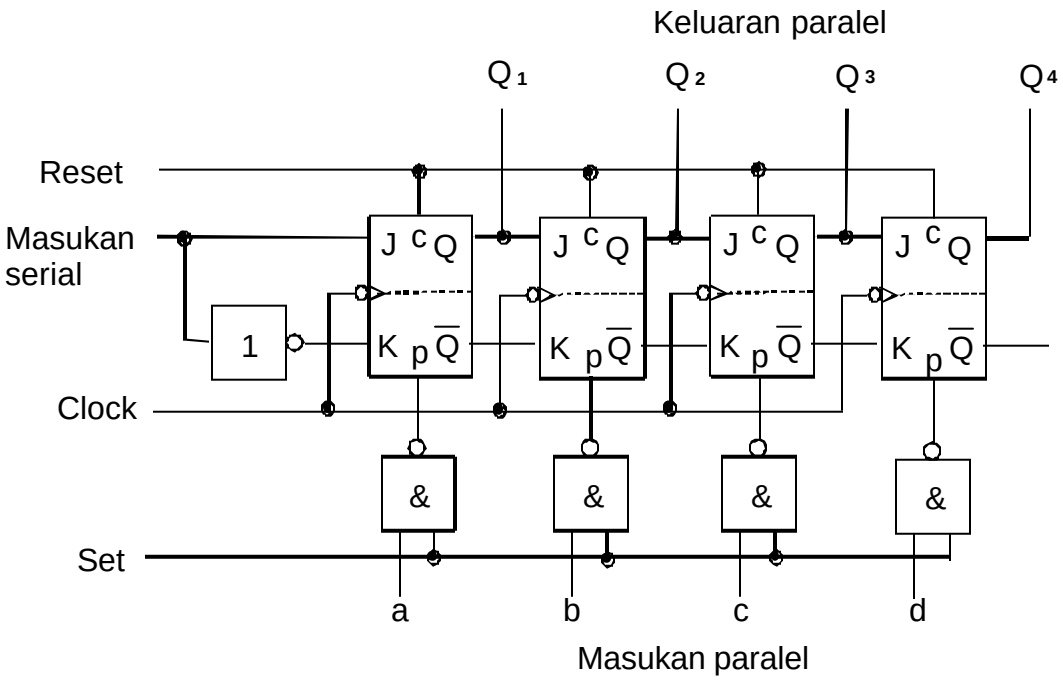
Shift register ini dapat digunakan sebagai penyimpanan sementara dan atau delay dari deretan informasi. Hal yang perlu diperhatikan setelah ini adalah aplikasi dari konversi serial / parallel maupun parallel / serial.

Dalam Operasi Parallel / serial data a sampai d dimasukkan bersamaan ke register dengan clock yang telah ditentukan. Keluaran serial akan muncul satu persatu pada indikator keluaran. Bagaimnapun juga jika data dimasukkan secara serial pada input dan sinkron dengan clocknya, maka setelah melengkapi barisan input, keluaran akan dapat dilihat secara parallel pada keluaran Q_1 sampai dengan Q_4 (Operasi Serial / Parallel).

Register geser diterapkan dengan fungsi yang berbeda-beda pada sistem komputer. Dimana macam-macam tipe yang digunakan adalah sebagai berikut :

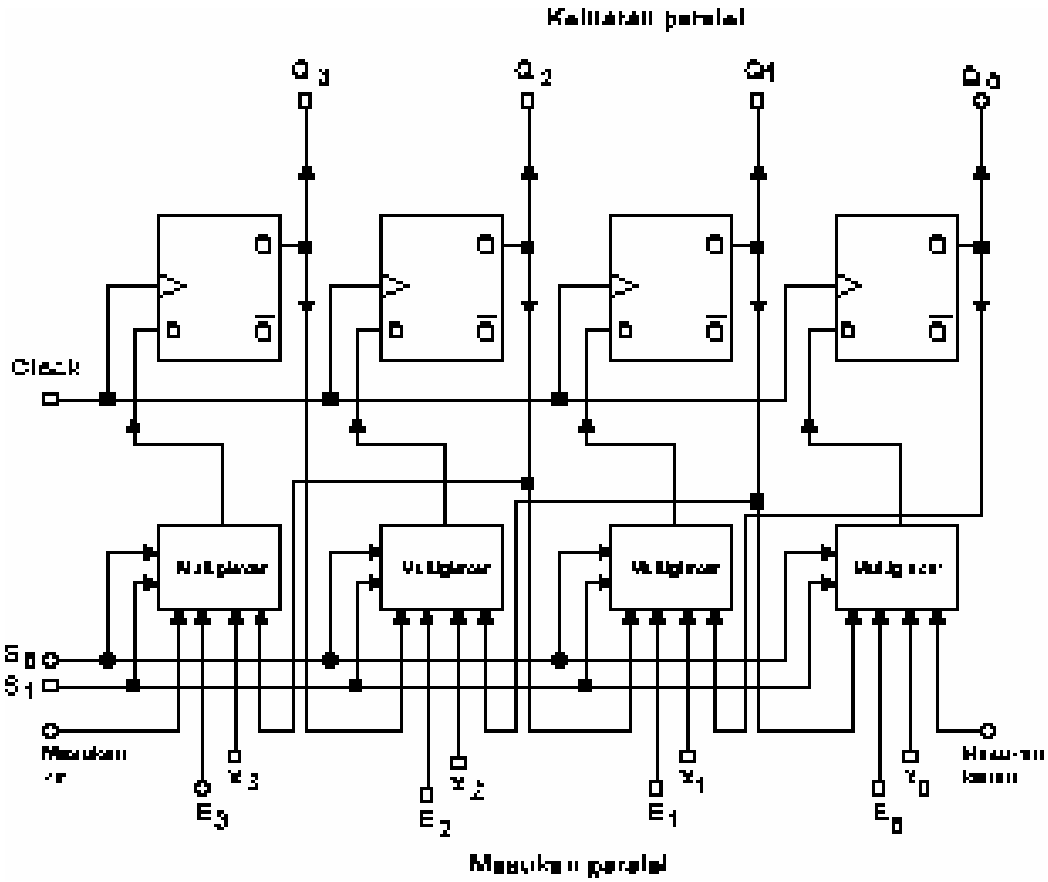
- Pergeseran data
- Masukan data serial dengan serial data keluaran
- Masukan data serial dengan keluaran data parallel
- Masukan data parallel dengan keluaran data seri
- Masukan data parallel dengan keluaran data parallel

Mode Operasi Parallel In / Parallel out dapat digunakan sebgai latihan untuk menngunakan register geser dengan mentransfer data pada masukan parallel ke data keluaran menggunakan pulsa yang telah ditentukan. Kemudian data ini akan tersimpan sementara sampai ada data yang dimasukkan. Kemudian Data pada register ini akan dihapus melalui input reset (operasi memori penyangga).



Gambar 3.23. Register geser untuk parallel/serial atau serial/parallel

Akhirnya, register geser yang digunakan pada sistem mikroprosesor sebagai memori penyangga.



Gambar 3.24. Register dengan multiplekser pada masukan D flip-flop

Prinsip dari operasi rangkaian ini ialah, dengan memakai input kontrol S_0, S_1, ke 4 multiplekser akan dapat dinyalakan salah satu dari ke 4 masukannya. Kemudian data yang telah dipilih pada input akan muncul pada keluaran. Contohnya , jika masukan paralel E_3 sampai E_0 dipilih maka data masukan akan dihadirkan secara parallel pada masukan D dari flip-flop. Dengan tepi clock positif selanjutnya, data dimasukkan ke flip-flop dan akan ditampilkan pada keluaran Q_3 sampai dengan Q_0. Data ini akan tersimpan hingga adanya pulsa clock yang membawa data baru pada E_3 s/d E_0 ke dalam register.

Dengan kombinasi kontrol S_0, S_1 yang lain. Input sebelah kanan pada multiplekser dapat dihubungkan ke Output. Data yang akan dimasukkan pada sebelah kiri rangkaian dapat dimasukkan secara serial ke dalam register. Prosesnya adalah sebagai berikut :

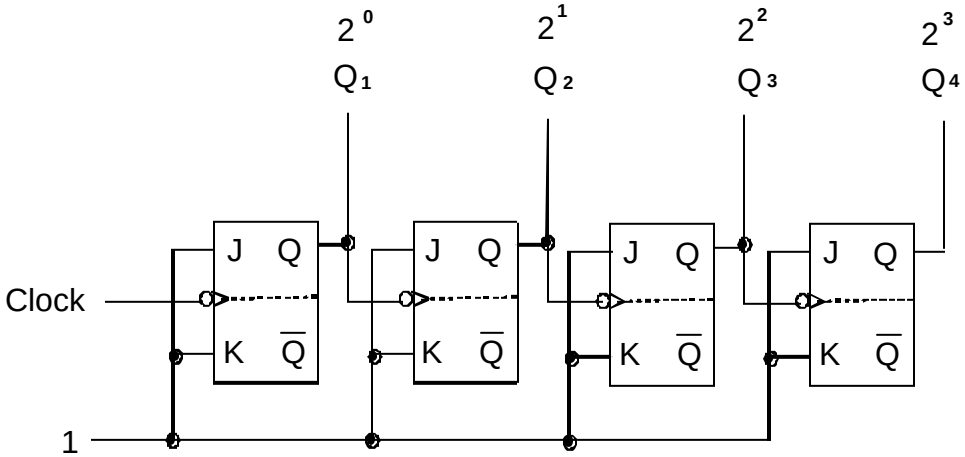
Jika kombinasi serial 1010 ada pada masukan sebelah kiri, maka pada saat clock pertama nilai 1 akan muncul pada keluran Q_0 dan pada masukan yang telah dipilih pada multiplekser selanjutnya. Pada saat clock kedua, keluaran akan menjadi $Q_0 = 0$ dan $Q_1 = 1$, sedangkan pada clock ketiga $Q_0 = 1$, $Q_1 = 0$, dan pada Clock ke 4 $Q_0 = 0$, $Q_1 = 1$, $Q_2 = 0$ dan $Q_3 = 1$.

Kombinasi masukan serial ini telah dibacakan ke register yang ada di sebelah kiri. Data serial yang ada pada masukan sebelah kanan akan di bawa secara analog. Masukan x_3 sampai x_0 tidak dimasukkan pada contoh ini. Sering untuk menghapus semua flip flop secara bersama sama adalah dengan cara mengeset semua masukan x_3 sampai x_0 ke logika 0. Jika masukan x semuanya dipilih melalui S_0, S_1 setelah pulsa clock berikutnya akan mengeset semua keluarn x_3 sampai x_0 ke logika 0.

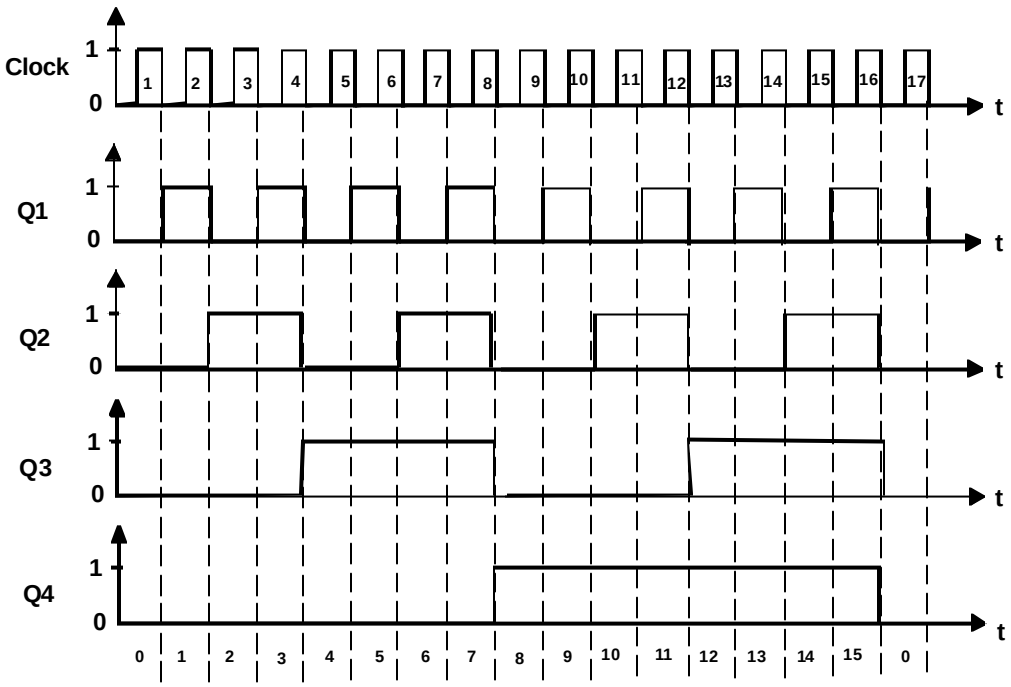
3.9. Counter

Counter adalah rangkaian digital yang didalamnya terdapat hubungan yang telah ditetapkan batasnya terhadap jumlah pulsa dan keadaan keluarannya. Komponen utama sebuah counter adalah flip-flop.

Mode operasi counter akan dijelaskan dengan bantuan pulsa diagram seperti tampak pada Gambar 3.26.
Sebelum clock pertama, keluaran Q_1 sampai dengan Q_4 adalah 0. Angka 0 disetarakan dengan kombinasi biner 0000. Setelah clock pertama bentuk bitnya menjadi 0001 yang diinterprestasikan sebagai angka 1.



Gambar 3.25. Rangkaian counter 4 bit



Gambar 3.26. Diagram pulsa counter 4 bit

Setelah clock kedua, 0010 akan muncul yang sesuai dengan angka 2, dan seterusnya. Seluruhnya terdapat 16 macam kombinasi yang sesuai dengan angka 0 sampai dengan 15. Setelah clock ke 16 seluruh keluaran akan kembali ke kondisi awal yaitu 0000. Untuk clock selanjutnya, proses diatas akan diulangi kembali.

Secara umum dapat disimpulkan bahwa n-bit counter dapat diasumsikan sebagai 2^n kombinasi keluaran yang berbeda-beda. Sejak angka 0 harus dialokasikan ke salah satu kombinasi ini, counter akan mampu menghitung hingga 2^{n-1} sebelum hitungan diulang kembali. Bila suatu counter terdiri atas 8 flip flop yang disusun seri, maka akan ada $2^8 = 256$ kombinasi biner yang berbeda yang berarti angka antara 0 sampai dengan 255.

Dengan mengubah sambungan memungkinkan untuk mereduksi kapasitas hitungan misalnya sebuah counter 4 bit yang memiliki 16 variasi keluaran dapat dibuat menjadi hanya 10 variasi keluaran. Counter akan menghitung o sampai 9 secara berulang ulang dan counter jenis ini disebut counter BCD.

Sering pula counter tidak hanya untuk menghitung naik dari 0000 ke 0001 dan seterusnya melainkan dapat pula dipergunakan sebagai penghitung turun dengan nilai awal adalah 1111 kemudian 1110 dan seterusnya sampai 0000 kemudian kembali ke 1111, counter yang seperti ini disebut counter down. Suatu counter akan berfungsi sebagai counter up atau

counter down dapat dipilih dengan sebuah kontrol untuk menentukan arah hitungan.

Beberapa counter dapat dibuat dari berbagai macam variasi. Kriteria penting suatu counter antara lain adalah :

- Arah hitungan (niak atau turun)
- Kontrol clock (serentak atau tak serentak)
- Kapasitas hitungan
- Kode hitunan
- Kecepatan menghitung
- Kemampuan counter untuk diprogram, yang artinya hitungan mulai berapa dapat diatur.

BAB IV DASAR ELEKTRONIKA DAYA

4.1. Sejarah ELektronika daya

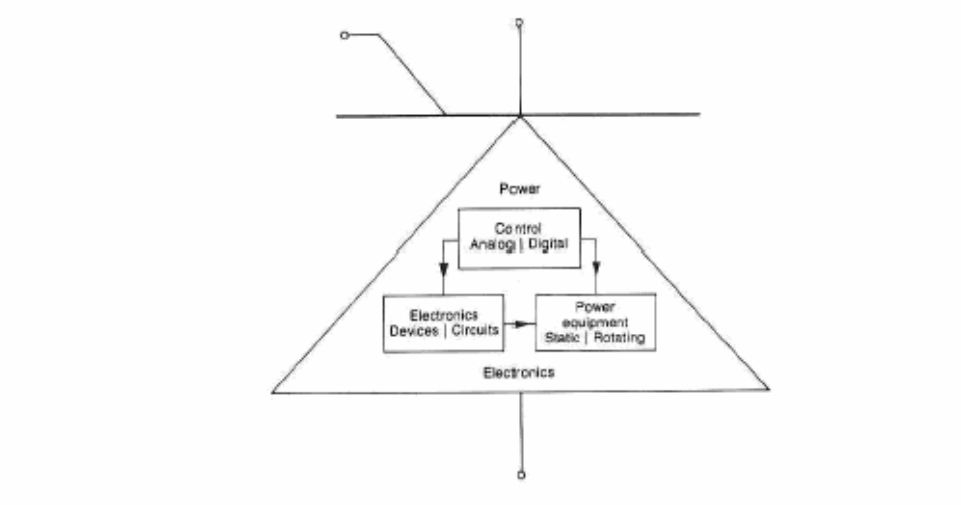
Bermula diperkenalkan penyearah busr mercuri 1900, metal tank, grid-cotrolled vacum tube, ignitron, phanotron dan thyratron semua ini untuk kontrol daya hingga tahun 1950. Tahun 1948 ditemukan transistor silikon , kemudian tahun 1956 ditemukan transistor pnpn triggering yang disebut dengan thyristor atau silicon controlled rectifier. Tahun 1958 dikembangkan thyristor komercial oleh general electric company. Sehingga sampai sekarang pengembangannya baik komponennya maupun aplikasinya sangat pesat.

4.2. Pengertian dan Prinsip kerja

Disiplin ilmu yang mempelajari penggunaan teknologi elektronika dalam konversi energi (daya) elektrik.
Mengapa energi (daya) elektrik perlu dikonversikan?

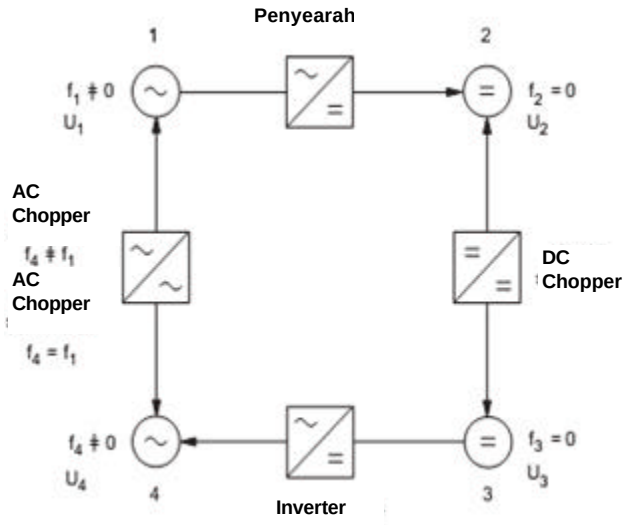
- Hampir semua peralatan listrik bekerja kurang efisien atau tidak bisa bekerja pada sumber energi (daya) elektrik yang tersedia.
- Banyak pembangkit energi (daya) elektrik nonkonvensional mempunyai bentuk yang tidak kompatibel dengan sumber energi (daya) elektrik lainnya.

Selama produksi, jaringan dan distribusi energi listrik yang digunakan secara umum baik tegangan satu phase dan tiga phase dapat konstan dan frekwensinya bisa stabil (50Hz atau 60Hz) dan tidak perlu adanya konversi bentuk tegangan, maka peralatan yang membutuhkan arus listrik tidak perlu membutuhkan tambahan system untuk penstabilan atau perubahan.
Penggunaan arus listrik pada peralatan industri banyak sekali yang harus dikendalikan, misalkan kecepatan putaran motor dapat diatur atau yang lainnya, maka perlu ada beberapa variable yang harus diatur (Tegangan atau Frekwensi) dan perubahan bentuk tegangan (konversi). Hal ini berlaku untuk tegangan satu atau tiga phase dan tegangan searah DC.



Gambar 4.1 Hubungan antara elektronika daya terhadap daya, elektronik dan kontrol

Tugas dari elektronika daya adalah merubah bentuk sumber energi listrik yang ada ke bentuk energi listrik yang diinginkan yang disesuaikan dengan beban yang dipergunakan.

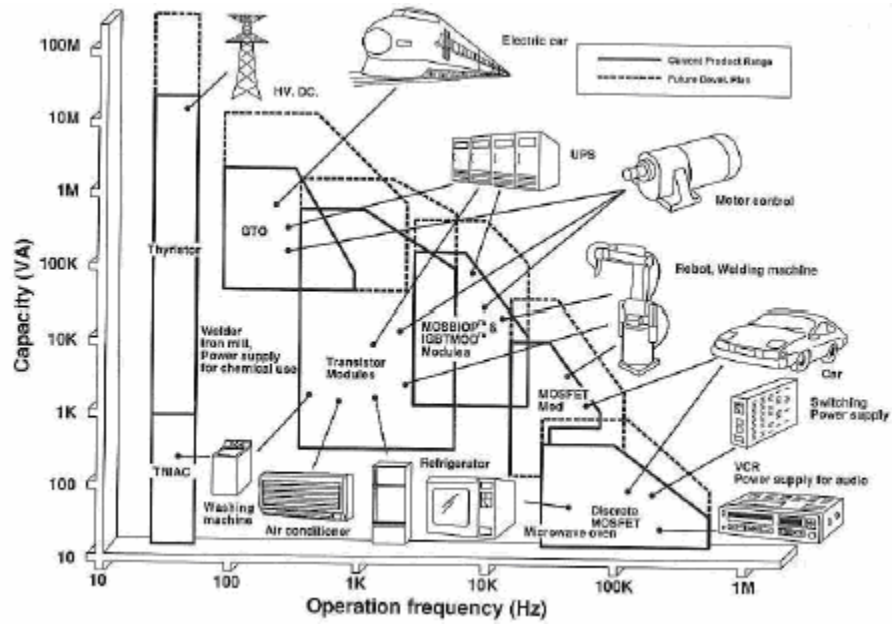


Gambar 4.2 Perubahan bentuk sumber energi listrik

Perbedaan dan perubahan energi listrik antara system tiga, satu phase arus bolak balik dan arus searah telah diatur dan diformulasikan pada DIN 41750.

a. Konverter AC ke DC (1 ? 2) Penyearah terkontrol
b. Konverter DC ke DC (2 ? 3). DC Chopper

c. Konverter DC ke AC (3 ? 4). Inverter
d. Konverter AC ke AC (4 ? 1). Kontroller tegangan AC (AC Chopper)
e. Konverter AC ke AC (4? 1). Kontroller tegangan AC (AC Chopper)



Gambar 4.3 Contoh aplikasi untuk elektronika daya

a) Teknik Penggerak
- Pemberian sumber tegangan pada mesin motor tiga phase yang variabelnya adalah tegangan dan frekwensi sehingga torsi dan kecepatannya dapat diatur. (Contohnya motor listrik di kereta listrik, motor pengatur posisi).
- Pemberian sumber tegangan pada motor DC dengan variabel tegangan pada ankernya dan lilitan. (Contohnya Motor pada kereta listrik).

b) Sumber Tegangan
- Catu daya (Contoh pada Personal Komputer)

c) Kebutuhan Rumah
- Pengatur terang redupnya lampu penerangan (Contoh dimmer ruang lampu dekorasi)

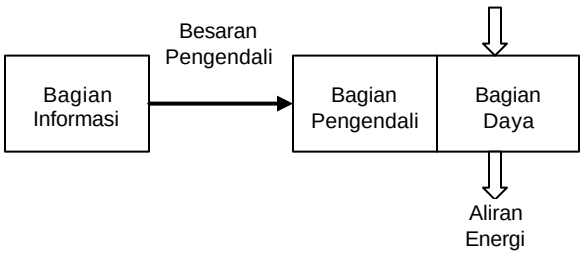
d) Kendaraan berat
- Pengapian elektronik
- Pembangkit pulsa untuk pengerak servo

- Penggerak stater generator
- Konverter dari 12 Volt ke 24 Volt DC
- Pengerak power string pada kemudi
- Transmisi automatik

Sekarang setiap peralatan listrik modern dan mesin modern yang menggunakan sumber listrik baik AC maupun DC selalu memerlukan elektronika daya untuk pengoperasiannya.

Dasar membangun peralatan elektronika daya

Peralatan yang ada elektronika dayanya dapat digambarkan secara umum ada tiga blok penting yaitu pada gambar dibawah



Gambar 4.4 Blok diagram dasar elektronika daya

1. Bagian Daya

Pada bagian daya berintefrensi langsung dengan aliran daya pada sumber elektrik. Bagian ini terbuat dari sebuah rangkaian spesial yang dapat sebagai penyearah, penyimpanan energi (C, L), Pengaman dan filter).

2. Bagian Pengendali

Bagian pengendali melakukan pengendali signal yang akan diumpankan pada bagian daya. Selah satu contohnya isi dari blok ini adalah signal penguat depan, pembalik potensial dan pemantau kesalahan.

3. Bagian Informasi

Sering sekali bagian Daya dan bagian Pengendali dilengkapi dengan bagian informasi, sehingga menjadi sebuah sistem pengendali dan pengaturan (open loop control dan close loop control) salah satu contoh pengaturan kecepatan motor listrik. Nilai besaran hasil koreksi eror langsung diumpankan ke bagian pengendali.

4.3. Komponen Elektronika Daya

Dari kerugian daya pada dasarnya diperbolehkan terjadi sebagian kecil terjadi pada semikonduktor pada saat posisi menahan arus listrik (off), komponen daya ini pada saat posisi mengalirkan arus listrik (on) langsung dengan aliran energi yang besar. Prinsip dasarnya seperti saklar mekanik. Pada dasarnya komponen hanya boleh secara ideal mengalirkan arus (U=0) dan pada saat menahan arus ideal (I=0).
Setiap kondisi kerja mengalirkan atau menahan dengan tegangan dan arus yang tinggi, maka timbul gangguan panas pada komponen, yang besarnya kerugian energi adalah (P= U * I ?0).
Saklar elektronik yang dilakukan oleh komponen elektronika daya secara nyata terdiri dari tiga tipe yaitu:

4.3.1. Satu Katup yang tidak dapat dikendalikan (Dioda)

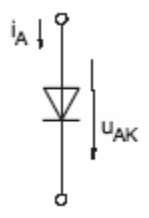
Dioda

Fungsi (Ideal)

- „ Membuka“ : i_{AK} >0, u_{AK} =0

- „ Menutup“ : i_{AK} =0, u_{AK} <0

Simbol



Gambar 4.5 Simbol Dioda

Data batas (Contoh)

U_{RRM} = 5000V, I_N = 4000A f_{max}=50 Hz

U_{RRM} = 2000V, I_N = 200A f_{max}=50 Hz

4.3.2. Pensaklaran Elektronik melalui sebuah Katup

Tipe komponen ini dapat disaklarkan hanya sambung dengan mengendalikan elektroda katupnya (gate). Dia akan tetap menghantarkan arus dari Anoda ke Katoda, jika arus pada katupnya diturunkan sampai nol, maka aliran arus berhenti. Komponen tersebut adalah :

Gambar 4.6 Simbol pensaklaran sebuah katup

Thyristor

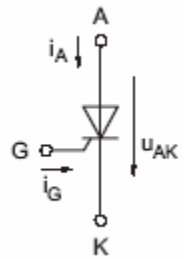
Fungsi secara ideal:

- „Terhubung“ arus akan mengalir dari anoda ke katoda melalui pengendalian arus $i_G>0$ pada kondisi $u_{AK}>0$.

- „ Tertahan“ sumber arus tetap mengalir dari anoda ke katoda jika $i_G=0$ dan selama $i_A>0$.

- „ Terputus“ tidak ada arus yang mengalir dari anoda ke katoda, hal ini akan terjadi, jika diset $i_G=0$ dan $i_A=0$. Sehingga arus dari anoda ke katoda terputus $i_A=0$.

Terputus arus yang mengalir dari anoda ke katoda melalui pengendalian arus $i_G<0$ $i_A>0$ adalah tidak mungkin terjadi.



Gambar 4.7 Simbol Thyristor

Sifat-sifat (ideal):

- Pada saat menghantar : $u_{AK} = 0$; $i_A > 0$
- Pada saat menutup : $i_A = 0$

Data batas (Contoh)

U_{DRM} = 8200V, I_N = 2400A f_{max}=50 Hz

U_{DRM} = 2500V, I_N = 2000A f_{max}=1,2 kHz

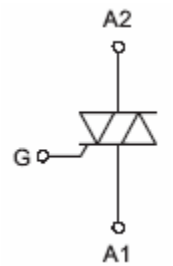
Triac

Thyristor atau SCR TRIAC mempunyai kontruksi sama dengan DIAC, hanya saja pada TRIAC terdapat terminal pengontrol (terminal gate). Sedangkan untuk terminal lainnya dinamakan main terminal 1 dan main terminal 2 (disingkat mt1 dan mt2). Seperti halnya pada DIAC, maka TRIAC pun dapat mengaliri arus bolak-balik, tidak seperti SCR yang hanya mengalirkan arus searah (dari terminal anoda ke terminal katoda). Lambang TRIAC di dalam skema elektronika, memiliki tiga kaki, dua diantaranya terminal MT1 (T1) dan MT2 (T2) dan lainnya terminal Gate (G) : Gambar dibawah memperlihatkan struktur dalam pada TRIAC : Triac adalah setara dengan dua SCR yang dihubungkan paralel. Artinya TRIAC dapat menjadi saklar keduanya secara langsung. TRIAC digolongkan menurut kemampuan pengontakan. TRIAC tidak mempunyai kemampuan kuasa yang sangat tinggi untuk jenis SCR. Ada dua jenis TRIAC, Low-Current dan Medium-Current.

Data batas (Contoh)

U_{DRM} = 800V, I_N = 8A f_{max}=50 Hz

U_{DRM} = 1000V, I_N = 40A f_{max}=50 Hz



Gambar 4.8 Simbol Triac

4.3.3. Pensaklaran Elektronik hubung dan putus melalui sebuah katup.

Tipe komponen ini adalah pengendalian elektrodenya dapat menghantarkan dan menyetop arus yang mengalir.

Sebagai bukti: kedua pengendalian ini dapat dilihat dari bentuk simbolnya.

Gambar 4.9 Simbol pensaklaran dua katup

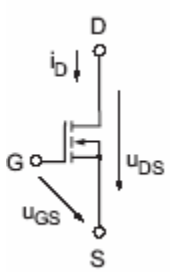
Komponen komponen tersebut yaitu:

Power MOSFET (n- Kanal)

Fungsinya (ideal):

- Pada saat menghantar:
  $u_{GS}>0$? $i_D>0$, $u_{DS}=0$

- Pada saat menutup:
  $u_{GS}<0$? $i_D=0$



Gambar 4.10 Simbol Power Mosfet (n-Kanal)

Data batas (Contoh):

$u_{DS\ max}$ = 1000V, i_{DN} =30A , f_{max} =100kHz

$u_{DS\ max}$ = 200V, i_{DN} =100A , f_{max} =50kHz

IGBT

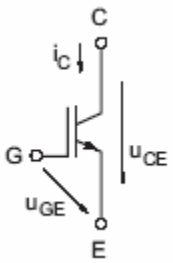
Fungsinya (ideal):

- Pada saat menghantar:
  $u_{GE}>0$? $i_C>0$, $u_{CE}=0$

- Pada saat menutup:
  $u_{GE}<0$? $i_C=0$

Data batas (Contoh):

$U_{CE\ max}$ = 1700V, i_{CN} =440A , f_{max} =20kHz



Gambar 4.11 Simbol IGBT

Transistor Daya Bipolar (BJT)

Fungsinya (ideal):

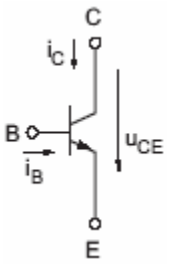
- Pada saat menghantar:
  $i_B>0$? $i_C>0$, $u_{CE}=0$

- Pada saat menutup:
  i_B=0 ? i_C=0

Data batas (Contoh):

$U_{CE\ max}$ = 1400V, i_{CN} =1000A , f_{max} =5kHz

$U_{CE\ max}$ = 1000V, i_{CN} =100A , f_{max} =50kHz



Gambar 4.12 Simbol Transistor Daya Bipolar (BJT)

GTO- Thyristor

Fungsinya (ideal):

- Pada saat menghantar:
  I_G>0 ? i_A>0, u_{AK}=0

- Pada saat menutup:
  I_G=0 ? i_A=0

Data batas (Contoh):

$U_{AK\ max}$ = 4500V, i_N =4000A , f_{max} =1..2kHz

$U_{AK\ max}$ = 6500V, i_N =1500A , f_{max} =1..2kHz

Komponen ini jika dibandingkan dengan IGBT dan MOSFET untuk dayanya jelas lebih tinggi hanya frekwensi kerjanya sedikit lebih rendah. Komponen ini dipergunakan pada rangkaian kereta api listrik dan penggerak mesin motor yang besar.

4.3.4. Perbandingan kinerja dari MOSFET, IGBT dan BJT.

Transistor IGBT (Insulated-Gate Bipolar Transistor) adalah piranti semikonduktor yang setara dengan gabungan sebuah transistor bipolar (BJT) dan sebuah transistor efek medan (MOSFET) Input dari IGBT adalah terminal Gate dari MOSFET, sedang terminal Source dari MOSFET terhubung ke terminal Basis dari BJT. Dengan demikian, arus drain keluar dan dari MOSFET akan menjadi arus basis dari BJT. Karena besarnya tahanan masuk dari MOSFET, maka terminal input IGBT hanya akan menarik arus yang kecil dari sumber. Di pihak lain, arus drain sebagai arus keluaran dari MOSFET akan cukuo besar untuk membuat BJT mencapai keadaan saturasi. Dengan gabungan sifat kedua elemen tersebut, IGBT mempunyai perilaku yang cukup ideal sebagai sebuah sakelar elektronik. Di satu pihak IGBT tidak terlalu membebani sumber, di pihak lain mampu menghasilkan arus yang besar bagi beban listrik yang dikendalikannya. Komponen utama di dalam aplikasi elekronika daya (power electronics) dewasa ini adalah sakelar zat padat (solid-state switches) yang diwujudkan dengan peralatan semikonduktor seperti transistor bipolar (BJT), transistor efek medan (MOSFET), maupun Thyristor. Sebuah sakelar ideal di dalam aplikasi elektronika daya akan mempunyai sifat-sifat sebagai berikut:

1). Pada saat keadaan tidak menghantar (OFF), sakelar mempunyai tahanan yang besar sekali, mendekati nilai tak berhingga. Dengan kata lain, nilai arus bocor struktur sakelar sangat kecil

2). Sebaliknya, pada saat keadaan menghantar (ON), sakelar mempunyai tahanan menghantar (R_on) yang sekecil mungkin. Ini akan membuat nilai tegangan jatuh (voltage drop) keadaan menghantar juga sekecil mungkin, demikian pula dengan besarnya daya lesapan (power dissipation) yang terjadi, dan

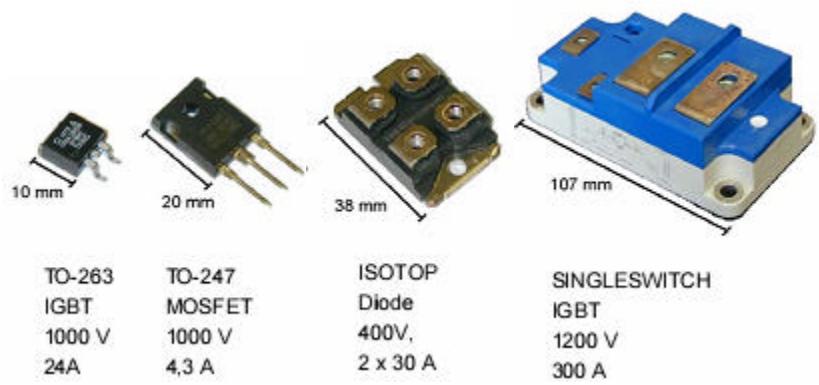
3). Kecepatan pensakelaran (switching speed) yang tinggi.

Sifat nomor (1) umumnya dapat dipenuhi dengan baik oleh semua jenis peralatan semikonduktor yang disebutkan di atas, karena peralatan semikonduktor komersial pada umumnya mempunyai nilai arus bocor yang sangat kecil. Untuk sifat nomor (2), BJT lebih unggul dari MOSFET, karena tegangan jatuh pada terminal kolektor-emitter, VCE pada keadaan menghantar (ON) dapat dibuat sekecil mungkin dengan membuat transitor BJT berada dalam keadaan jenuh (saturasi).

Sebaliknya, untuk unsur kinerja nomor (3) yaitu kecepatan switching, MOSFET lebih unggul dari BJT, karena sebagai divais yang bekerja berdasarkan aliran pembawa muatan mayoritas (majority carrier), pada MOSFET tidak dijumpai aruh penyimpanan pembawa muatan minoritas pada saat proses pensakelaran, yang cenderung memperlamnat proses pensakelaran tersebut. Sejak tahun 1980-an telah muncul jenis divais baru sebagai komponen sakelar untuk aplikasi elektronika daya yang disebut sebagai Insulated Gate Bipolar Transistor (IGBT). Sesuai dengan yang tercermin dari namanya, divais baru ini merupakan divais yang menggabungkan struktur dan sifat-sifat dari kedua jenis transistor tersebut di atas, BJT dan MOSFET. Dengan kata lain, IGBT mempunyai sifat kerja yang menggabungkan keunggulan sifat-sifat kedua jenis transistor tersebut. Terminal gate dari IGBT, sebagai terminal kendali juga mempunyai struktur bahan penyekat (insulator) sebagaimana pada MOSFET. Dengan demikian, terminal masukan IGBT mempunyai nilai impedansi yang sangat tinggi, sehingga tidak membebani rangkaian pengendalinya yang umumnya terdiri dari rangkaian logika. Ini akan menyederhanakan rancangan rangkaian pengendali (controller) dan penggerak (driver) dari IGBT. Di samping itu, kecepatan pensakelaran IGBT juga lebih tinggi dibandingkan divais BJT, meskipun lebih rendah dari divais MOSFET yang setara. Di lain pihak, terminal keluaran IGBT mempunyai sifat yang menyerupai terminal keluaran (kolektor-emitter) BJT. Dengan kata lain, pada saat keadaan menghantar, nilai tahanan menghantar (R_on) dari IGBT sangat kecil, menyerupai R_on pada BJT. Dengan demikian bilai tegangan jatuh serta lesapan dayanya pada saat keadaan menghantar juga kecil. Dengan sifat-sifat seperti ini, IGBT akan sesuai untuk dioperasikan pada arus yang besar, hingga ratusan amper, tanpa terjadi kerugian daya yang cukup berarti. IGBT sesuai untuk aplikasi pada perangkat Inverter maupun Kendali Motor Listrik (Drive).

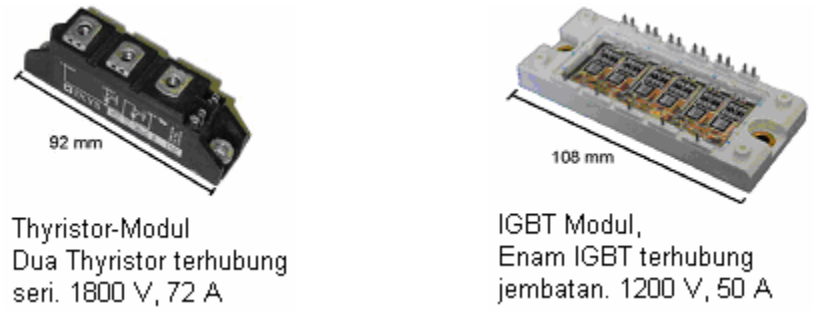
4.3.5. Bentuk Komponen

Sekarang semikonduktor dengan daya yang tinggi mempunyai bentuk yang standart. Bagi komponen yang daya tinggi sekali selalu mempunyai pendingin Untuk mentransfer disipasi panasnya sehingga tidak banyak kerugian daya.Contoh bentuknya dapat dilihat pada Gambar 4.13.



Gambar 4.13 Bentuk komponen elektronika daya

Ada yang telah dibuat beberapa komponen yang berbentuk modul yang sudah tersambung satu sama lainnya, ada yang dua komponen ada yang lebih, sebagai contohnya:



Gambar 4.14 Bentuk komponen elektronika daya berbentuk modul

Untuk komponen yang sebagai pemicu untuk mengendalikan elektrodenya atau gatenya telah tersedia, yang kadang ada yang sekaligus dua kanal untuk mentrigernya. Contoh bentuknya dapat dilihat pada gambar 4.15.

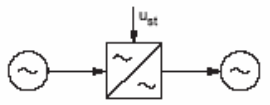
Dua kanal pemicu katup (gate)

Pemicu katup (gate) dengan Optokoppler

Gambar 4.15 Komponen pemicu elektroda atau gate

4.4. Contoh rangkaian elektronika daya

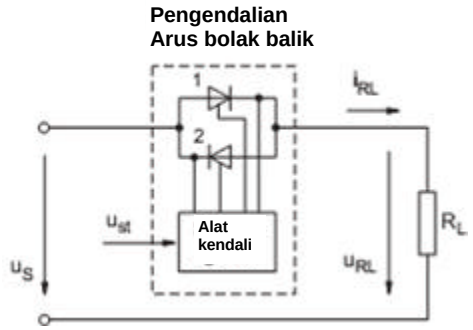
4.4.1. Konverter AC ke AC dengan Pengendalian pemotongan fase



Gambar 4.16 Blok diagram converter AC ke AC

Sumber arus balak balik yang sebagai sumber yaitu tegangan dan frekwensinya harus konstan dan yang terpakai pada tegangan beban dapat dirubah antara 0 = U_{RL} = U_S. Tingginya tegangan pada beban dapat diatur dengan tegangan kedali.

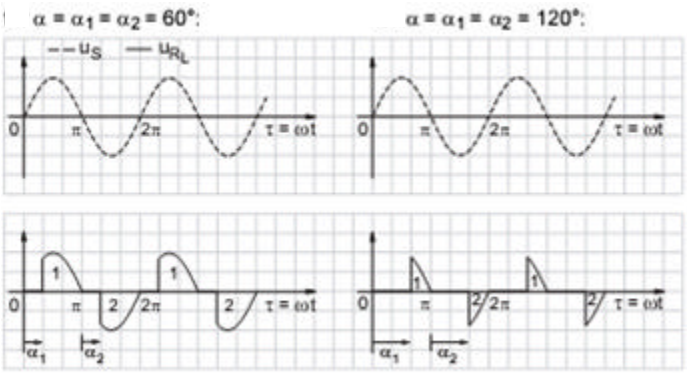
Contoh Pemakaian:
Dimmer, yaitu pengaturan terang gelapnya lamp
Mengendalikan kecepatan motor universal yang daya kecil.



Gambar 4.17 Blok Rangkaian converter AC ke AC

Catatan:
Pada daya yang kecil dapat diganti pada dua Thyristor diganti dengan Triac.

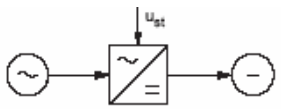
Tegangan sumber u_S dapat dihantarkan pada kedua katup 1 dan katup 2. Untuk katup 1 mengalirkan arus positip dan katup 2 mengalirkan arus negative ke beban i_{RL}.
Harga tegangan efektif pada beban dapat dirubah melalui pengaturan tegangan pada katup 1 dan katup 2 dengan cara setiap setengah gelombang ada penundaan.
Hasil penundaan waktu yang dapat dikatakan sudut gelombangnya yang terpotong, maka sering juga disebut penundaan sudut a (0 =a = p), pengendalian ini disebut pengendalian phase, kerena simetris antara gelombang positif dan negatifnya (a1 = a2).
Untuk tegangan keluaran pada beban akan maksimal u_{RL}= u_S jika diatur sudut a1= a2 = 0.



Gambar 4.18 Penundaan waktu pada tegangan u_S dan u_{RL}

Dua thyristor atau Triac harus ditriger setelah zero crossing agar tegangan MT1 dan MT2 cukup untuk merubah kondisi kerja Triac ketika ada arus gate.

4.4.2. Penyearah dengan pengendalian pemotongan fase



Gambar 4.19 Blok diagram converter AC ke DC (Penyearah)

Berdasarkan semikonduktor yang digunakan dan variasi tegangan keluarannya, penyearah satu atau tiga-fasa dapat diklasifikasikan menjadi :

- Penyerah tak terkendali.
- Penyearah terkendali.

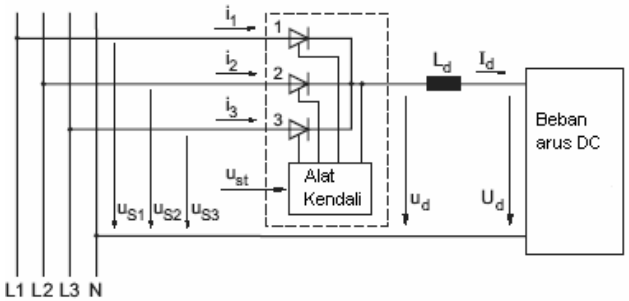
Umumnya semikonduktor penyearah terkendali menggunakan bahan semikonduktor berupa thyristor, atau menggunakan thyristor dan dioda secara bersamaan.
Berdasarkan bahan semikonduktar yang digunakan dan sistem kendalinnya penyearah satu atau tiga-fasa terkendali umumnya dapat dibedakan menjadi :

- Half wave Rectifiers
- Full wave Rectifiers-Full Controller
- Full wave Rectifiers-Semi Controller

Hal-hal yang menjadi masalah dalam teknik penyerahan antara lain adalah trafo penyearahan, gangguan-gangguan tegangan lebih atau arus lebih yang membahayakan dioda / thyristor, keperluan daya buta untuk beban penyearahan, harmonisa yang timbul akibat gelombang non sinus serta sirkit elektronik pengatur penyalaan

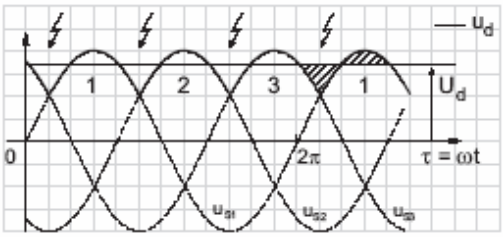
Kebutuhan arus searah dapat dibangun dengan sumber arus tiga phase yang nantinya akan dihasilkan tegangan Ud searah yang lebih baik juga arusnya Id.
Besarnya tegangan searah tergantung dari besarnya tegangan pengendali uST untuk mendapatkan besaran antara $0 = U_d = U_{dmax..}$
Pengendalian ini dilakukan pada katup dari thyristor.

Contoh Pemakaian:
Dalam aplikasinya, sirkit-sirkit penyearahan biasanya dilengkapi dengan sirkit. pengatur tambahan seperti pengatur tegangan pembatas arus dan-lain-lain sesuai dengan jenis pemakaiannya. Bidang gerak teknik penyearahan meliputi sistem-sistem pengatur putaran mesin DC pada mesin cetak kertas, tekstil, mesin las DC, pengisi baterai,sampai pada pengatur tegangan konstan generator sinkron (AVR).
Pemberian sumber tegangan pada Motor DC dengan jala jala 3 phase dan thyristor.



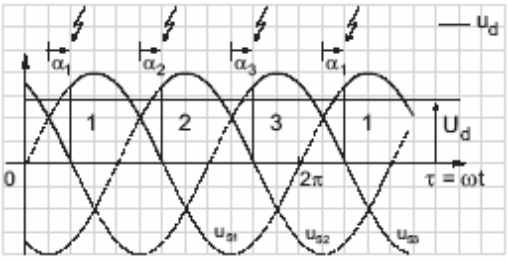
Gambar 4.20 Rangkaian Titik tengah tiga pulsa terkendali (M3C)

Akibat dari penyulutan pada katup 1, 2 dan 3 dapat dihasilkan tegangan searah dari tegangan antar phase dengan netral u_{S1}, u_{S2} dan u_{S3} dari sumber arus listrik 3 phase.
Keinginan dari harga aritmatik tegangan keluaran Ud mudah sekali untuk tercapai dengan pengendalian penundaan sudut.
Tegangan keluaran pada penyearah memungkinkan mendapatkan tegangan yang maksimal jika sudut penyulutan a = 0.



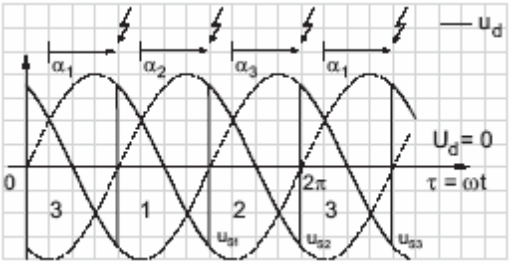
Gambar 4.21 Penyulutan sudut 0^o

Penundaan penyulutan pada Thyristor (a > 0) mengakibatkan menurunkan tegangan searah pada keluaraannya.



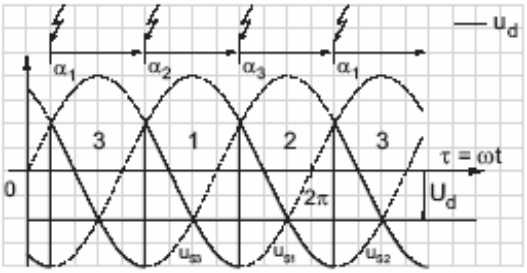
Gambar 4.22 Penyulutan sudut 30^o

Pada penyulutan dengan sudut a =90^o besar tegangan searah yang dihasilkan adalah nol (Ud= 0 V).



Gambar 4.23 Penyulutan sudut 90^o

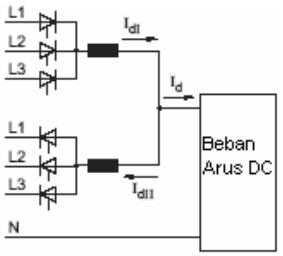
Energi yang dikirim pada tegangan keluaran dapat berubah kutupnya, jika besarnya sudut penyulutan melebihi 90^o (U_d<0 dan I_d>0)



Gambar 4.24 Penyulutan sudut 120^o

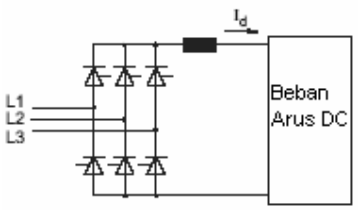
Hasil besar tegangan pada outputnya akan minus terhadap titik nol, jika penyulutan sudutnya melebihi >90^o.

Sering dalam aplikasi sumber arus mengunakan system perubahan polaritas, suatu contoh pada mesin motor DC yang putarannya dibuat suatu saat kekanan atau kekiri. Maka rangkaian dasarnya adalah sebagai berikut:



Gambar 4.25 Rangkaian Titik tengah enam pulsa terkendali (M6C)

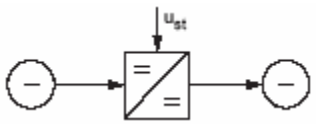
Ada sebuah bentuk lagi yang menghasilkan tegangan outputnya ganda, yang berarti tidak hanya pulsa positifnya saja yang disearahkan tetapi juga yang pulsa negatif, sehingga hasil tegangan outputnya lebih besar dan tegangan ripplenya lebih kecil. Bentuk ini sering disebut penyearah terkontrol jembatan tiga phase



Gambar 4.26 Rangkaian jembatan 6 pulsa terkendali (B6C)

4.4.3. Pengubah daya DC-DC (DC-DC Converter)

Tipe peralihan dari tegangan DC ke DC atau dikenal juga dengan sebutan *DC Chopper* dimanfaatkan terutama untuk penyediaan tegangan keluaran DC yang bervariasi besarannya sesuai dengan permintaan pada beban.

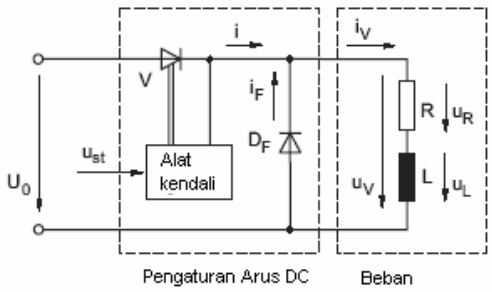


Gambar 4.27 Blok diagram konverter DC ke DC (DC Chopper)

Daya masukan dari proses DC-DC tersebut adalah berasal dari sumber daya DC yang biasanya memiliki tegangan masukan yang tetap. Pada dasarnya, penghasilan tegangan keluaran DC yang ingin dicapai adalah dengan cara pengaturan lamanya waktu penghubungan antara sisi keluaran dan sisi masukan pada rangkaian yang sama. Komponen yang digunakan untuk menjalankan fungsi penghubung tersebut tidak lain adalah switch (solid state electronic switch) seperti misalnya Thyristor, MOSFET, IGBT, GTO. Secara umum ada dua fungsi pengoperasian dari DC Chopper yaitu penaikan tegangan dimana tegangan keluaran yang dihasilkan lebih tinggi dari tegangan masukan, dan penurunan tegangan dimana tegangan keluaran lebih rendah dari tegangan masukan.

Prinsip dasar Pengubah DC-DC Tipe Peralihan

Untuk lebih memahami keuntungan dari tipe peralihan, kita lihat kembali prinsip pengubahan daya DC-DC seperti terlihat pada contoh gambar dibawah:



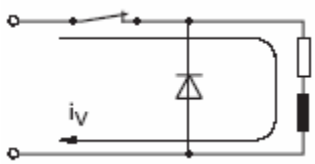
Gambar 4.28 Rangkaian Konverter DC ke DC (DC Chopper)

Pada gambar rangkaian ada sebuah katup V yang mengalirkan dan menyumbat arus yang mengalir, sehingga mempengaruhi besar tegangan pada beban.

1. Katup V menutup, kondisinya arus mengalir :

$u_v = U_o$

Diode D_F menutup tidak ada arus yang lewat karena katoda lebih positif dari pada anoda, tetapi arus iv mengalir naik ke beban dengan konstanta waktu $T = \frac{L}{R}$. Sehingga arus maksimum yang terjadi secara exponensial adalah: $i_{vmak=} \frac{U_0}{R}$

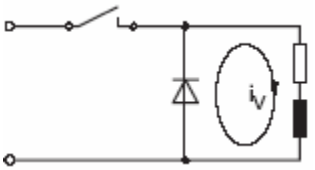


Gambar 4.29 Arus mengalir saat kondisi katup V menutup

2. Katup V membuka, kondisi arus tertahan

Setelah katup dibuka, arus mengalir i_v dari pembuangan dari induktor, sehingga ada komutasi arus dari anoda ke katoda (Dioda D_F)

$u_v = 0$



Gambar 4.30 Arus tertahan saat kondisi katup V membuka

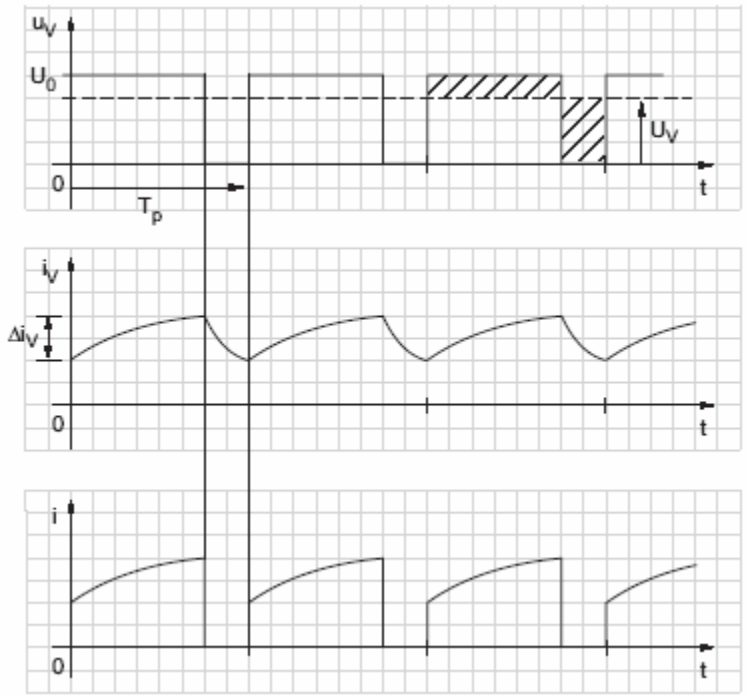
Arus pada beban i_v dihasilkan secara exponensial adalah nol.

Dalam satu periode terjadi tutup bukanya sebuah katup seperti saklar, pada beban terjadi potensial tegangan Uv dan memungkin arus yang mengalir iv pada beban. Dari terjadi perubahan tutup dan buka ada perbedaan arus terhadap waktu ? iv. Sehingga divisualisasikan dengan bentuk gelombang kotak (pulsa), dengan notasi frekwensi fp atau dirubah dengan fungsi waktu Tp. Sedangkan untuk periode positif adalah Tg dan periode pulsa negatif To. Sehingga tegangan output persamaannya dapat dituliskan sebagai berikut:

$$U_v = \frac{T_g}{T_g + T_o} \bullet U_o = \frac{T_g}{T_p} \bullet U_o \text{ atau}$$

$$\frac{U_v}{U_o} = \frac{T_g}{T_g + T_o} = \frac{T_g}{T_p}$$

Salah satu contoh gambar dibawah adalah bentuk gelombang pada tegangan dan arus pada beban terhadap waktu dengan T_g=0,75 T_p



Gambar 4.31 Bentuk gelombang tegangan dan arus pada beban

Untuk signal kendali ada beberapa cara memodulasi signalnya yaitu:

- Pulse wave modulation (PWM)

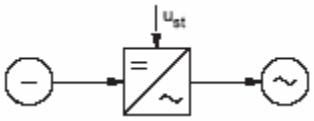
  Tp konstan, Tg variable

  Sangat lebar jarak pulsa, contoh : $\frac{1}{T_p} = f_{PWM} = 16kHz$

- Pengendali Pulsa
  T_g atau T_o konstan, T_p variable (jarang digunakan)
- Dua titik Pengaturan
  ?i konstan, T_p variable, system ini sering dipakai pada pengaturan

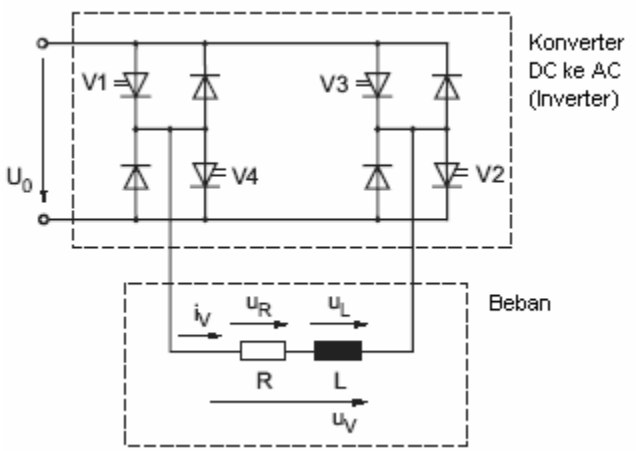
4.4.4. Pengubah daya DC ke AC satu fase (Konverter DC to AC)

Dengan beban tahanan murni atau induktif sebuah konverter harus menghasilkan tegangan output dan frekwensi yang konstan



Gambar 4.32 Blok diagram konverter DC ke AC

Salah satu aplikasinya adalah pada elektro lokomotip yang modern atau pengendalian Mesin Motor.



Gambar 4.33 Rangkaian pengubah tegangan DC ke AC dengan model jembatan

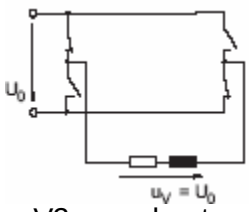
Dari rangkaian diatas ada empat thyristor mempunyai masing-masing satu katup V1, V2, V3 dan V4. Setiap katup dapat dikendalikan buka tutupnya. Sehingga didapatkan tiga macam besarnya tegangan keluaran antara lain:

1. V1 dan V2 menghantar, V3 dan V4 menutup

   $u_v = U_o$

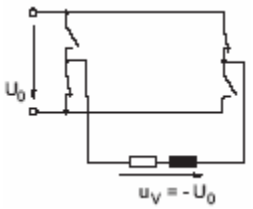
   Arus pada beban i_v naik ekponensial dengan konstanta waktu $T = \frac{L}{R}$

   dan hasilnya adalah $\frac{U_o}{R}$



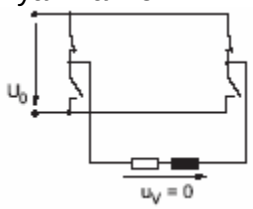
Gambar 4.34 Katup V1 dan V2 menghantar dan V3 dan V4 menutup

2. V3 dan V4 menghantar, V1 dan V2 menutup
   $u_v = -U_o$
   Arus pada beban naik eksponensial tetapi dengan polaritas negatif adalah $-\frac{U_o}{R}$



Gambar 4.35 Katup V3 dan V4 menghantar dan V1 dan V2 menutup

3. V1 dan V3 menghantar, V2 dan V4 menutup
   $u_v = 0$
   Arus pada beban mempunyai nilai nol.



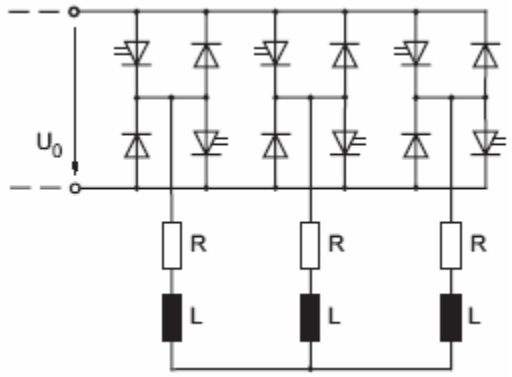
Gambar 4.36 Katup V1 dan V3 menghantar dan V2 dan V4 menutup

4. V2 dan V4 menghantar dan V1 dan V3 menutup
   $u_v = 0$
   Identis dengan pada nomer 3

Jika langkah 1, 2, 3 dan 4 bergantian dan dengan kendali pulsa pada katupnya, maka akan terjadi pada beban sebuah tegangan dan mempunyai kontanta waktu $T = \frac{L}{R}$. Dibawah ini salah satu contoh hasil tegangan keluran uv dengan kendali PWM.

Gambar 4.37 Bentuk tegangan keluaran

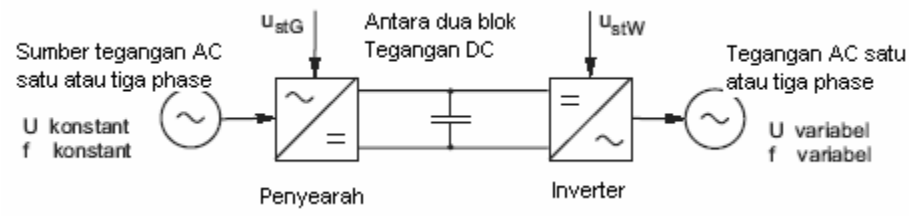
Pada system konverter DC ke AC yang menghasilkan tegangan tiga phase, prinsipnya sama dengan satu phase langkah prosesnya, tetapi terdiri tiga kolompok kombinasi katup pada thyristor. Bisa dilihat pada gambar dibawah ini.



Gambar 4.38 Rangkaian DC ke AC tiga phase

4.4.5. Pengubah daya AC ke AC (Konverter AC ke AC)

Pada sistim pengubah ini merupakan gabungan dari pengubah daya dari AC ke DC (penyearah) dan Pengubah daya dari DC ke AC. Sehingga dapat digambarkan seperti dibawah ini:



Gambar 4.39 Blok diagram konverter AC ke AC

BAB V. PENGUKURAN, PENGENDALI (KONTROL) DAN PENGATURAN

5.1. Definisi

Pengukuran berasal dari kata kerja mengukur dalam bahasa Inggris disebut *"measuring*" yang dalam bahasa Jerman dinamakan *"messen"*. Mengukur dalam ilmu teknik berarti mempersiapan, mentransfer dan menerangkan suatu informasi awal yang belum dimengerti oleh peralatan tertentu (dalam hal ini adalah sinyal) menjadi sinyal yang dapat diterima dan dimengerti oleh peralatan tersebut. Dengan mengukur akan diperoleh sinyal awal bisa berupa sinyal dengan besaran bukan listrik (misalnya : temperatur *"^{0}C; ^{0}F"*, tekanan *"bar; psi"* dan bisa berupa sinyal dengan besaran listrik (misalnya : tegangan *"Volt"*, arus *"Ampere"*, resistan/tahanan *"Ohm"*, dan lain-lainnya).

Untuk keperluan pengendalian, pengaturan dan supervisi dari suatu peralatan teknik biasanya diperlukan alat pendeteksi berupa alat ukur sinyal listrik, dimana pada awalnya sinyal ini biasanya mempunyai besaran fisika yang bisa diukur sesuai dengan harga besarannya. Besaran sinyal fisika ini kemudian diubah menjadi sinyal listrik oleh sensor/detektor melalui pengukuran.

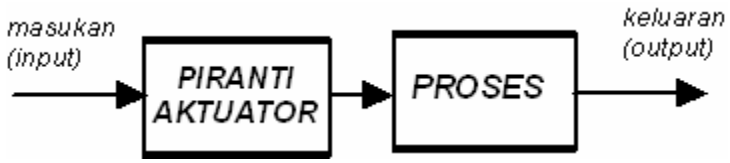
Sensor digunakan sebagai alat pendeteksi/pengukur sinyal bukan listrik menjadi sinyal listrik yang dalam istilah teknik pengaturan sebagai suatu blok pemberi sinyal harga terukur *(measuring value signal)* atau dalam bahasa Jerman biasa disebut *"messwertgeber".* Blok ini antara lain terdiri dari piranti absorbsi *"absorber device"*, piranti sensor *"sensing device*", dan elemen khusus yang diperlukan. Jadi pemberi sinyal harga terukur adalah suatu blok piranti sensor dengan keluaran sinyal listrik yang sudah terkalibrasi.

Pengendali (kontrol) juga memerlukan sensor, hanya saja sensor pada pengendali (kontrol) ini biasanya digunakan hanya sebagai masukan *(input)* saja, yang dalam bahasa teknik kontrol sebagai referensi atau besaran sinyal komando (*command signal value)* dan biasanya digunakan pada kontrol 2(dua) titik.
Yang jelas pada pengendali (kontrol) tidak menggunakan umpan balik (*feedback*) yang dalam bahasa Inggris disebut *"Open Loop Control"* atau dalam bahasa Jerman dinamakan *"Steurung"*.

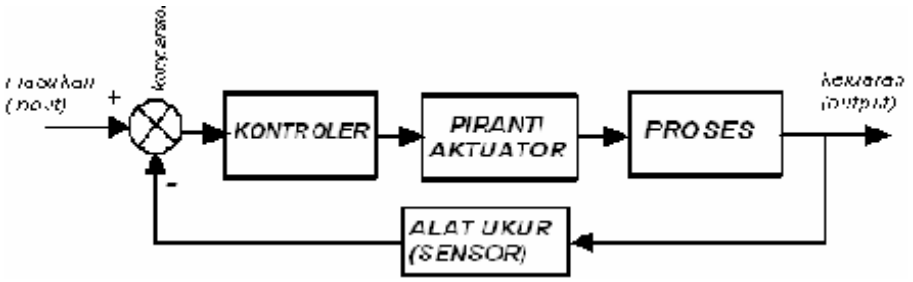
Pengaturan adalah mutlak harus menggunakan sensor untuk mendeteksi/mengukur keluaran yang akan dikembalikan sebagai umpan balik *(feed back)* untuk dibandingkan dengan masukan selaku referensi atau titik penyetelan *(setting point)*. Pengaturan dalam istilah bahasa Inggris disebut *"Closed Loop Control"* atau bahasa Jerman dinamakan *"Regelung".*

Lebih jelasnya dapat dilihat pada gambar 5.1 Proses yang dikontrol (Process to be controlled); gambar 5.2 Sistem kontrol loop terbuka (Open loop control system); gambar 5.3 Sistem control loop tertutup (Closed loop control system).

Gambar 5.1 Proses yang dikontrol



Gambar 5.2 Sistem control loop terbuka (Open loop control system)



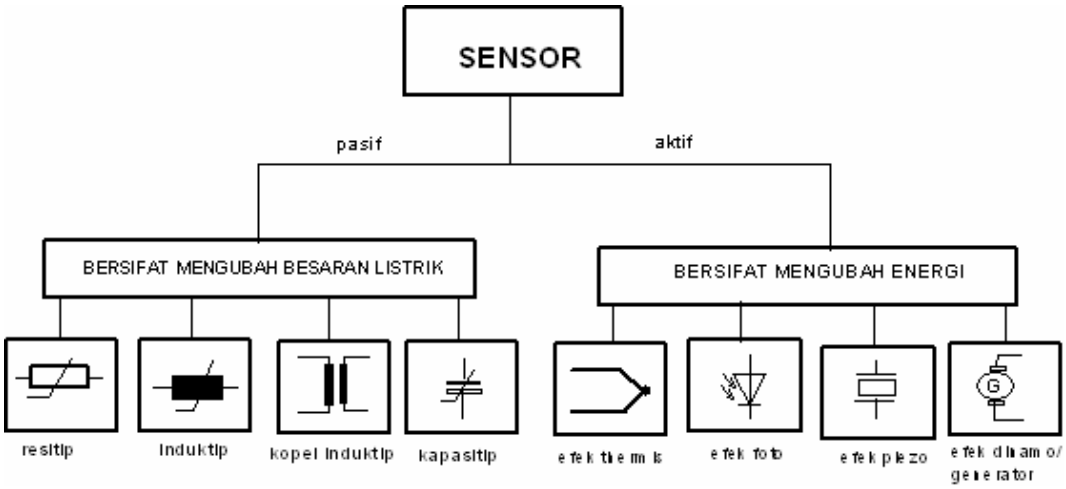
Gambar 5.3 Sistem control loop tertutup (Closed loop control system)

5.2 Sensor

Sebelum memahami dan menerapkan penggunaan sensor secara rinci maka perlu mempelajari sifat-sifat dan klasifikasi dari sensor secara umum.

Sensor adalah komponen listrik atau elektronik, dimana sifat atau karakter kelistrikannya diperoleh atau diambil melalui besaran listrik (contoh : arus listrik, tegangan listrik atau juga bisa diperoleh dari besaran bukan listrik, contoh : gaya, tekanan yang mempunyai besaran bersifat mekanis, atau suhu bersifat besaran thermis, dan bisa juga besaran bersifat kimia, bahkan mungkin bersifat besaran optis).

Sensor dibedakan sesuai dengan aktifitas sensor yang didasarkan atas konversi sinyal yang dilakukan dari besaran sinyal bukan listrik (non electric signal value) ke besaran sinyal listrik (electric signal value) yaitu : sensor aktif (active sensor) dan sensor pasif (passive sensor). Berikut gambar 5.4 Sifat dari sensor berdasarkan klasifikasi sesuai fungsinya.



Gambar 5.4 Sifat dari sensor berdasarkan klasifikasi

5.2.1 Sensor Aktif (active sensor)

Suatu sensor yang dapat mengubah langsung dari energi yang mempunyai besaran bukan listrik (seperti : energi mekanis, energi thermis, energi cahaya atau energi kimia) menjadi energi besaran listrik. Sensor ini biasanya dikemas dalam satu kemasan yang terdiri dari elemen sensor sebagai detektor, dan piranti pengubah sebagai *transducer* dari energi dengan besaran bukan listrik menjadi energi besaran listrik.

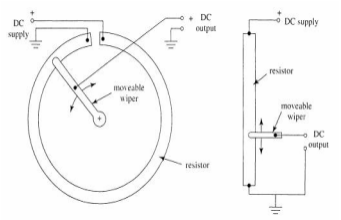
Sensor-sensor ini banyak macam dan tipe yang dijual di pasaran komponen elektronik (sebagai contoh : *thermocouple*, *foto cell* atau yang sering ada di pasaran LDR “Light Dependent Resistor”, *foto diode*, *piezo electric*, *foto transistor*, elemen *solar cell* , *tacho generator*, dan lain-lainnya). Prinsip kerja dari jenis sensor aktif adalah menghasilkan perubahan resistansi/tahanan listrik, perubahan tegangan atau juga arus listrik langsung bila diberikan suatu respon penghalang atau respon penambah pada sensor tersebut (contoh sinar/cahaya yang menuju sensor dihalangi atau ditambah cahayanya, panas pada sensor dikurangi atau ditambah dan lain-lainnya).

5.2.1.1 Sensor dengan Perubahan Resistansi

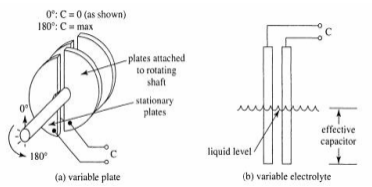
Sensor-sensor yang tergolong pada perubahan resistansi pada prinsipnya dapat mengubah besaran yang bersifat fisika menjadi besaran

yang bersifat resistansi/tahanan listrik.Tahanan listrik ini biasanya dalam bentuk sensor berupa komponen listrik atau elektronik yang akan berubah-ubah nilai tahanan listriknya bila diberikan energi secara fisika, misalkan diberi energi mekanis dapat berubah nilai tahanan listriknya, contoh: *potensiometer*, diberikan energi cahaya dapat berubah nilai tahan listriknya, contoh: *foto cell* atau biasa digunakan LDR : *"light dependent resistor*", diberikan energi panas atau dingin, contoh thermocouple, NTC *"negative temperature coefficient"*, PTC "*positive temperature coefficient*" dan lain-lainnya).

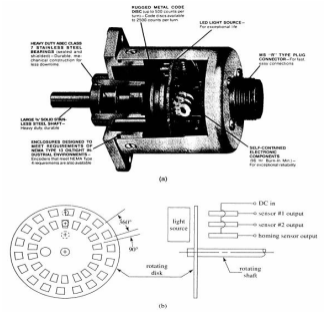
Berikut diberikan beberapa contoh gambar dari sensor dengan prinsip perubahan resistansi, temperatur, optik dan lain-lainnya.



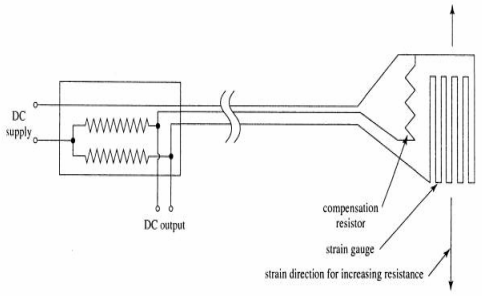
Gambar 5.5. Sensor potensiometer



Gambar 5.6. Sensor variable capasitor



Gambar 5.7. Sensor optical encoder

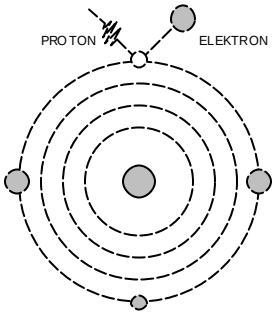


Gambar 5.8. Sensor strain gauge

5.2.1.1.1 Resistor Tergantung Cahaya (LDR "Light Dependent Resistor")

Suatu sensor yang akan berubah nilai resistan/tahanan listriknya jika pada permukaan dari LDR tersebut diberikan sinar yang lebih atau sebaliknya yaitu pada permukaan LDR dikurangi/dihalangi atas sinar yang menuju ke LDR tersebut. LDR tergolong sensor aktif dan sering disebut sebagai *foto cell*.

Secara prinsip dasar susunan atom semikonduktor tampak pada gambar 5.5 berikut ini.



Gambar 5.9 Susunan atom pada semikonduktor

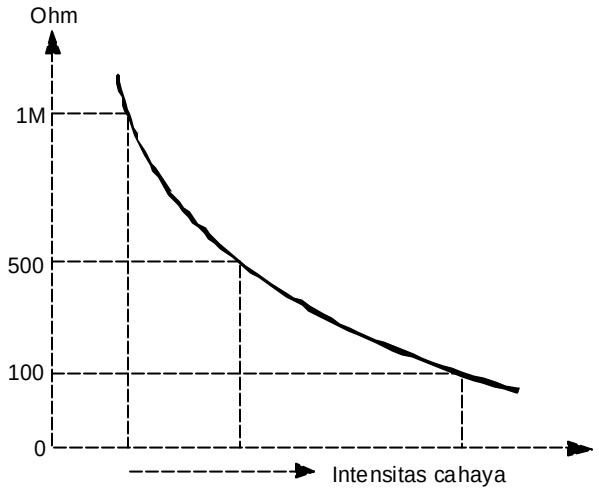
Dalam suatu sel *photoconductive* (*foto cell*) terdapat lapisan semi konduktor yang peka cahaya di antara dua kontak. Biasanya untuk maksud ini digunakan *cadmium-sulfid* , timbal dan seng. Jika suatu *proton* menumbuk sebuah elektron yang terdapat pada lintasan terluar dari sebuah atom dalam kisi-kisi kristal semi konduktor , maka elektro tersebut akan terlempar dari orbitnya disebabkan oleh energi proton dan akan bergerak bebas. Proses ini terjadi pada setiap tempat dalam kristal semi konduktor . Kenaikan jumlah elektron bebas terjadi diantara masing-masing atom, sehingga menaikkan konduktivitas (daya hantar). Ion positif dihasilkan pada setiap tempat , di mana elektron-elektron telah melepaskan diri dari atom.

Jika suatu sel *photoconductive* dilindungi dari cahaya, maka elektron-elektron bebas tadi akan ditarik ion-ion positif dan kembali pada posisi semula dalam atom.

Tahanan sel *photoconductive* akan turun (berkurang) jika dia disinari (kondisi terang) , sebaliknya tahanannya akan naik jika dia dilindungi dari cahaya (kondisi gelap).

Dari penjelasan prilaku LDR di atas maka Karakteristik LDR dapat digambarkan dalam

bentuk kurva resistansi/tahanan terhadap fungsi intensitas cahaya :

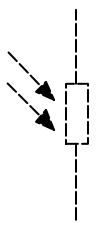


Gambar 5.10. Karakteristik LDR

Pada grafik di atas menunjukkan bahwa apabila intensitas cahaya makin kuat maka tahanan LDR makin kecil . Apabila dalam keadaan gelap LDR mempunyai tahanan yang sangat besar , dapat mencapai beberapa Mega Ohm . Tetapi bila seberkas cahaya jatuh padanya maka tahanannya akan menurun sebanding dengan intensitas cahaya tersebut. Makin kuat intensitas cahaya yang datang berarti makin besar tenaga yang diberikan maka berarti pula makin kecil tahanan LDR.

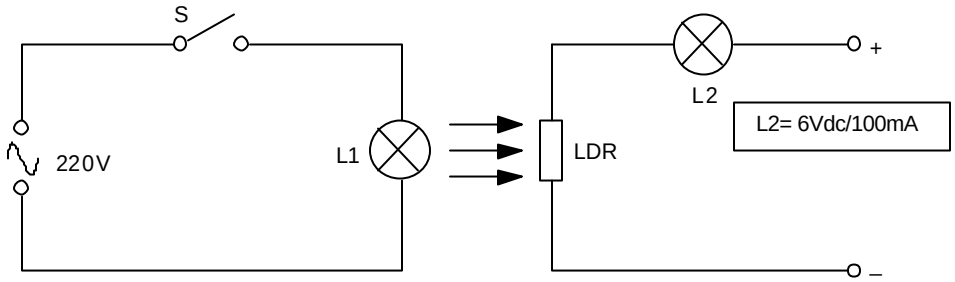
Sel-sel *photoconductive* dijumpai dalam beberapa bentuk yang mempunyai karakteristk berbeda-beda. Tahanan gelapnya berubah-ubah antara 10 sampai 1000 M ohm . Dengan intensitas penerangan yang tinggi dapat dicapai tahanan sekitar 100 Ohm .

Sel-sel photoconductive dapat bereaksi terhadap perubahan cahaya cepat hanya sampai sekitar 1/10.000 detik (10 Khz) .



Gambar 5.11 Simbol LDR (standar IEC)

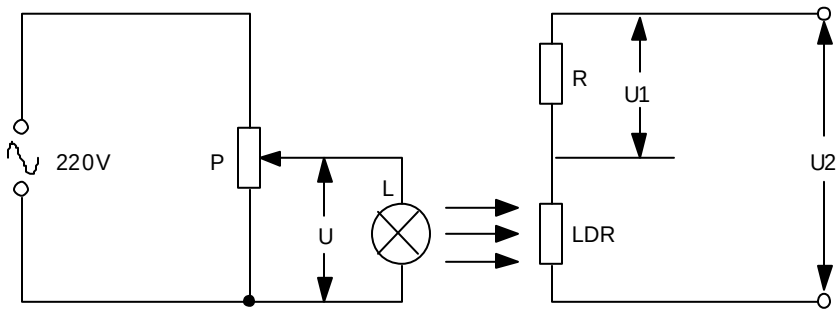
Contoh beberapa penerapan penggunaan sensor LDR adalah sebagai saklar (gambar 5.8) dan sebagai potensiometer, Suatu keuntungan yang diperoleh bila memanfaatkan LDR sebagai saklar adalah tanpa menimbulkan bunga api (loncatan busur api) .



Gambar 5.12. LDR sebagai saklar

Bila saklar S terbuka , maka lampu L_1 tidak menyala, LDR tidak mendapat cahaya gelap, berarti tahanan LDR menjadi besar . Hal ini mengakibatkan L_2 tidak menyala , walaupun menyala tapi sangat suram . Selanjutnya apabila saklar S ditekan (ON) , maka lampu L1 menyala dan menerangi LDR . Akibatnya tahanan LDR menurun atau menjadi sangat kecil . Hal ini mengakibatkan pula lampu L_2 menyala , karena seolah-olah saklar yang sedang ON .

Contoh lain adalah seperti ditunjukkan gambar 5.9 dimana LDR menyerupai sebuah potensiometer.



Gambar 5.13. LDR berfungsi sebagai potensiometer

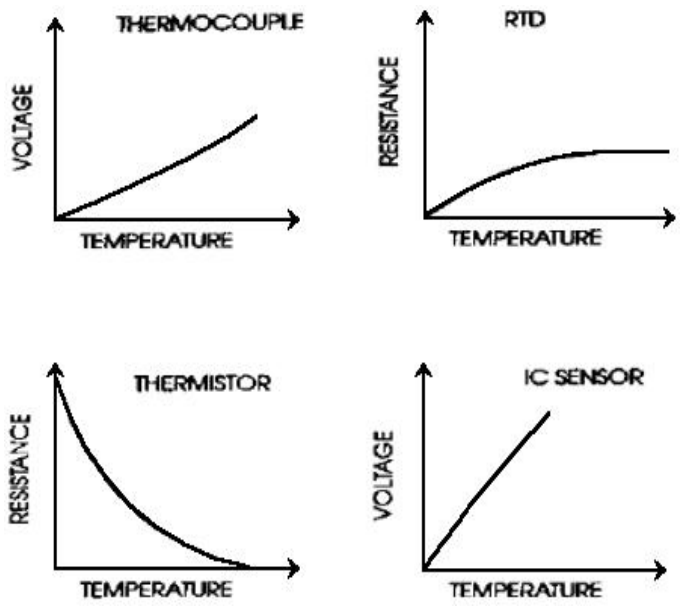
Perhatikan gambar di atas, terang dan gelapnya lampu L dapat diatur oleh potensiometer P. Perubahan intensitas cahaya akan mengakibatkan perubahan tahanan LDR. Selanjutnya akan mengakibatkan pula perubahan tegangan yang didrop oleh R (U_1). Hal seperti di atas , LDR menyerupai sebuah potensiometer .

5.2.1.2 Resistor Tergantung Cahaya (LDR "Light Dependent Resistor")

5.2.2 SENSOR THERMOCOUPLE

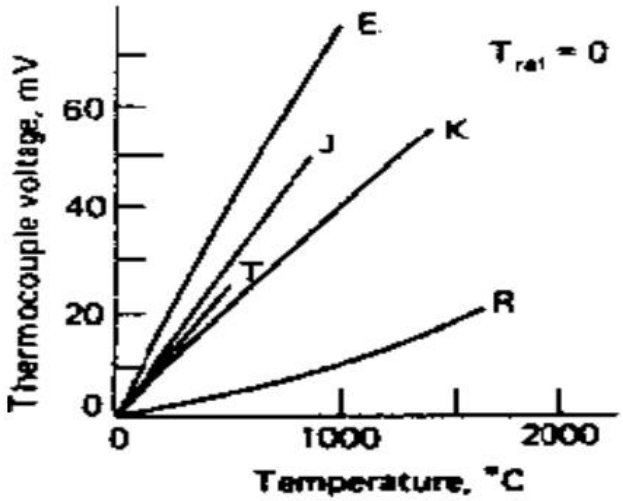
5.2.2.1 Sensor Suhu

Terdapat berbagai model dan jenis sensor suhu yang ada di pasaran, diantaranya PTC, NTC, PT100, LM35, thermocouple dan lain-lain. Berikut ini karakteristik beberapa jenis sensor suhu.



Gambar 5.14 Karakteristik beberapa jenis sensor suhu

Pada gambar diatas IC sensor dan thermocouple memiliki linearitas paling baik, namun karena dalam tugas ini suhu yang diukur lebih dari 100^{o}C, maka thermocouple yang paling sesuai karena mampu hingga mencapai suhu 1200^{o}C. Sedangkan IC sensor linear mampu hingga 135 ^{o}C. Output sensor suhu thermocouple berupa tegangan dalam satuan mili Volt. Berikut ini beberapa perilaku jenis thermocouple.



Gambar 5.15 Perilaku beberapa jenis thermocouple

Tabel 5.1 Karakteristik jenis thermoCouple.

Thermocouple Type	Names of Materials	Useful Application Range
B	Platinum30% Rhodium (+) Platinum 6% Rhodium (-)	2500 -3100F 1370-1700C
C	W5Re Tungsten 5% Rhenium (+) W26Re Tungsten 26% Rhenium (-)	3000-4200F 1650-2315C
E	Chromel (+) Constantan (-)	200-1650F 95-900C
J	Iron (+) Constantan (-)	200-1400F 95-760C
K	Chromel (+) Alumel (-)	200-2300F 95-1260C
N	Nicrosil (+) Nisil (-)	1200-2300F 650-1260C
R	Platinum 13% Rhodium (+) Platinum (-)	1600-2640F 870-1450C
S	Platinum 10% Rhodium (+) Platinum (-)	1800-2640F 980-1450C
T	Copper (+) Constantan (-)	-330-660F -200-350C

Sensor Temperatur PT100

Tipe ini termasuk jenis yang paling tua, yang konstruksinya terdiri dari satu tabung gelas yang mempunyai pipa kapiler kecil berisi vacuum dan cairan ini biasa berupa air raksa. Perubahan panas menyebabkan perubahan ekspansi dari cairan atau dikenal dengan *temperature to volumatic change* kemudian *volumetric change to level* secara simultan. Perubahan level ini menyatakan perubahan panas atau temperatur. Ketelitian jenis ini tergantung dari rancangan atau ketelitian tabung, juga penyekalannya.

Cara lain dari jenis ini adalah mengunakan gas tabung yang diisi gas yang dihubungkan dengan pipa kapiler yang dilindungi oleh spiral menuju ke spiral bourdon yang dipakai untuk menggerakkan pivot, selanjutnya menggerakkan pointer. Berikut adalah gambar dari sensor PT00.

Gambar 5.16 Sensor PT100

Tabel 5.2 Spesifikasi jenis sensor PT100

Thermocouple type	**Overall range** ^0C	**0.1** ^0C **resolution**	**0.025** ^0C **resolution**
B	20 to 1820	150 to 1820	600 to 1820
E	-270 to 910	-270 to 910	-260 to 910
J	-210 to 1200	-210 to 1200	-210 to 1200
K	-270 to 1370	-270 to 1370	-250 to 1370
N	-270 to 1300	-260 to 1300	-230 to 1300
R	-50 to 1760	-50 to 1760	20 to 1760
S	-50 to 1760	-50 to 1760	20 to 1760
T	-270 to 400	-270 to 400	-250 to 400

Tabel 5.3 Data Sheet

	Temperature	**Resistence**	**Voltage**
Sensor	PT100*,PT1000	N/A	N/A
Range	-200 to 800 ^0C	0 to 375 ?* 0 to 10 k?	0 to 115 mV 0 to 2.5 V*
Linierity	10 ppm	10 ppm	10 ppm
Accuracy @ 25 ^0C	0.01 ^0C *	20 ppm*	0.2%*
Temperature coeficien	3 ppm/ ^0C	3 ppm/ ^0C	100 ppm/ ^0C
RMS Noise (using filter)	0.01 ^0C	10 ppm	10 ppm
Resolution	0.001 ^0C	1 µ?	0.156µV
Conversion time per channel	720 mS**	720 mS**	180 mS
Number of input	4		
Connectors	4-pin miniDIN		
Input impedance	>>1 M?		
Overvoltage protection	$\pm 100V$		
Output	RS232, D9 female		
Environmetal	20 to 30 ^0C for stated accuracy, 0 to 70 ^0C overall, 20 to 90% RH		

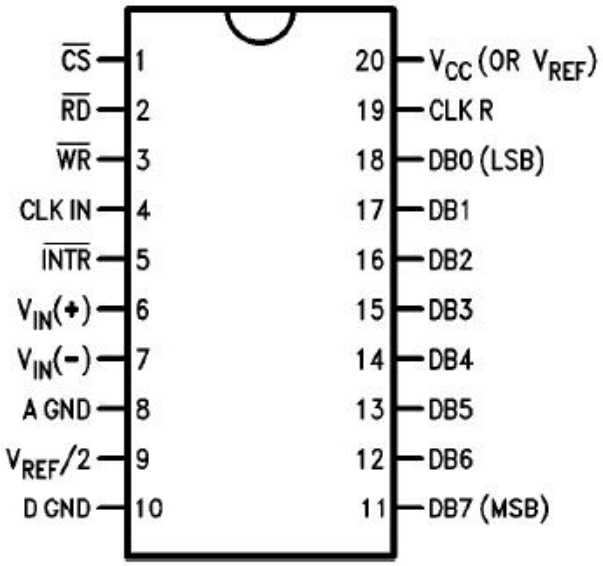
ADC (Analog Digital Converter)

ADC adalah suatu rangkaian yang mengkonversikan sinyal analog menjadi sinyal digital. Ada beberapa jenis rangkaian ADC antara lain Servo ADC, Successive Approximation dan Parallel Converter.

Ada banyak cara yang dapat digunakan untuk mengubah sinyal analog menjadi sinyal digital yang nilainya proposional. Jenis ADC yang biasa digunakan dalam perancangan adalah jenis successive approximation convertion atau pendekatan bertingkat yang memiliki waktu konversi jauh lebih singkat dan tidak tergantung pada nilai masukan analognya atau sinyal yang akan diubah.

IC ADC 0804 dianggap dapat memenuhi kebutuhan dari rangkaian yang akan dibuat. IC jenis ini bekerja secara cermat dengan menambahkan sedikit komponen sesuai dengan spesifikasi yang harus diberikan dan dapat mengkonversikan

secara cepat suatu masukan tegangan. Hal-hal yang juga perlu diperhatikan dalam penggunaan ADC ini adalah tegangan maksimum yang dapat dikonversikan oleh ADC dari rangkaian pengkondisi sinyal, resolusi, pewaktu eksternal ADC, tipe keluaran, ketepatan dan waktu konversinya. Berikut ini adalah diagram kopneksi dari IC ADC 0804 :



Gambar 5.17 Diagram koneksi dari IC ADC 0804

ADC 0804 adalah CMOS 8 bit Successive Approximation ADC. Pada modul ADC ini Vref untuk ADC 0804 adalah sebesar 2.5 Volt dan tegangan catu (Vcc) sebesar 5 Volt. ADC 0804 mempunyai karakteristik sebagai berikut :

a. Resolusi sebesar 8 bit
b. Conversion time sebesar 100 ms
c. Total unadjusted error : 1 LSB
d. Mempunyai clock generator sendiri (640 Khz)

Secara singkat prinsip kerja dari konverter A/D adalah semua bit-bit diset kemudian diuji, dan bilamana perlu sesuai dengan kondisi yang telah ditentukan. Dengan rangkaian yang paling cepat, konversi akan diselesaikan sesudah 8 clock, dan keluaran D/A merupakan nilai analog yang ekivalen dengan nilai register SAR.

Apabila konversi telah dilaksanakan, rangkaian kembali mengirim sinyal selesai konversi yang berlogika rendah. Sisi turun sinyal ini akan menghasilkan data digital yang ekivalen

ke dalam register buffer. Dengan demikian, keluaran digital akan tetap tersimpan sekalipun akan di mulai siklus kon konversi yang baru.

IC ADC 0804 mempunyai dua masukan analog, Vin (+) dan Vin (-), sehingga dapat menerima masukan diferensial. Masukan analog sebenarnya (Vin) sama dengan selisih antara tegangan-tegangan yang dihubungkan dengan ke dua pin masukan yaitu Vin= Vin (+) – Vin (-). Kalau masukan analog berupa tegangan tunggal, tegangan ini harus dihubungkan dengan Vin (+), sedangkan Vin (-) digroundkan. Untuk operasi normal, ADC 0804 menggunakan Vcc = +5 Volt sebagai tegangan referensi. Dalam hal ini jangkauan masukan analog mulai dari 0 Volt sampai 5 Volt (skala penuh), karena IC ini adalah SAC 8-bit, resolusinya akan sama dengan

$$\text{Resolusi} = \left(\frac{\text{tegangan skala penuh}}{2^n - 1}\right) = \frac{5\ \text{Volt}}{255} = 19{,}6\ \text{mVolt}$$

(n menyatakan jumlah bit keluaran biner IC analog to digital converter)

IC ADC 0804 memiliki generator clock intenal yang harus diaktifkan dengan menghubungkan sebuah resistor eksternal (R) antara pin CLK OUT dan CLK IN serta sebuah kapasitor eksternal (C) antara CLK IN dan ground digital. Frekuensi clock yang diperoleh di pin CLK OUT sama dengan :

$$f = \frac{0{,}91}{RC}$$

Untuk sinyal clock ini dapat juga digunakan sinyal eksternal yang dihubungkan ke pin CLK IN. ADC 0804 memilik 8 keluaran digital sehingga dapat langsung dihubungkan dengan saluran data mikrokomputer. Masukan (chip select, aktif rendah) digunakan untuk mengaktifkan ADC 0804. Jika berlogika tinggi, ADC 0804 tidak aktif (disable) dan semua keluaranberada dalam keadaan impedansi tinggi.

Masukan (write atau start convertion) digunakan untuk memulai proses konversi. Untuk itu harus diberi pulsa logika 0. Sedangkan keluaran (interrupt atauend of convertion) menyatakan akhir konversi. Pada saat dimulai konversi, akan berubah ke logika 1. Di akhir konversi akan kembali ke logika 0.

5.3 PERANCANGAN KONTROLER

A. PENDAHULUAN

Dalam perancangan Kontroler perlu meninjau hubungan antara output-input kontroler artinya termasuk derajat *(orde)* berapakah persamaan diferensialnya. Demikian juga dalam memilih *plant* diharapkan dapat menyesuaikan dengan kontroler yang dirancang.

Sebagai contoh untuk Kontroler dengan persamaan diferensial *orde nol* maka Kontroler yang dirancang adalah tipe *P (Proportional)*, orde satu Kontroler tipe *PI (Proportional + Integral)* dan untuk Kontroler dengan persamaan diferensial orde dua menggunakan Kontroler tipe *PID (Proportional + Integral + Differential)*. Kontroler secara teori berguna untuk mengendalikan *plant* mulai *orde satu*, *orde dua* atau lebih.

Plant adalah adalah seperangkat peralatan terdiri dari beberapa bagian mesin yang bekerja bersama-sama untuk melakukan operasi tertentu.

B. TIPE KONTROLER

$$\frac{U(s)}{E(s)} = Kp$$

1. *Kontroler Tipe-P (Proportional Controller)*

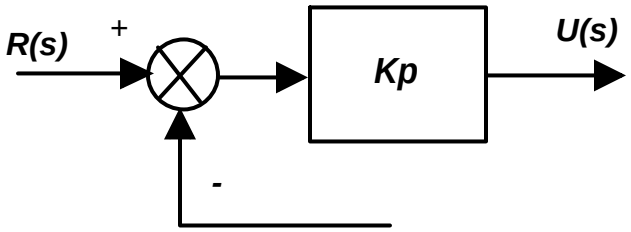
   Kontroler *tipe-P* adalah menyatakan hubungan antara *sinyal error* dan *sinyal kontrol*. Sehingga secara matematik dapat diberikan persamaan :

   U(t) = Kp.e(t)

   Atau dalam bentuk fungsi alih *(Transfer Function :"TF")* :

   Dimana *E(s) : Error signal; U(s) : Controller signal*; *Kp : Gain Over of Proportional.*

Sehingga secara diagram blok dapat dinyatakan :



Gambar 5.18 Diagram Blok Kontroler Tipe-P

2. Kontroler Tipe-I (Integral Controller)

Kontroler *tipe-I (Integral Controller)* adalah suatu kontroler yang level output kontrol *U(t)* diubah pada *rate* yang proporsional.

Karena :

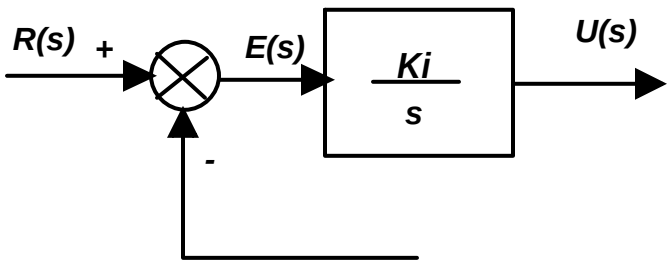
$$\frac{du(t)}{dt} = Ki.e(t) \quad ;\text{dan} \qquad U(t) = Ki\int e(t)dt$$

$$\frac{de}{dt} = s; \quad dan \quad \int edt = \frac{1}{s} \quad \text{dan } e(t) = 1(s)$$

Secara fungsi alih dapat dinyatakan dengan formulasi :

$$\frac{U(s)}{E(s)} = \frac{Ki}{s}$$

Secara diagram blok dapat digambarkan sebagai berikut :



Gambar 5.19 Diagram Blok Kontroler Tipe-I

3. Kontroler Tipe-PI (Proportional + Integral Controller)

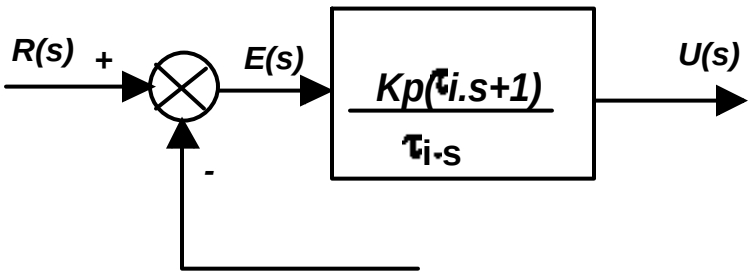
Kontroler ini menyatakan hubungan antara *sinyal error* dan *sinyal kontrol*, sehingga secara matematik dapat diberikan persamaan :

Dalam bentuk fungsi alih dapat diformulasikan :

$$U(t) = Kp[e(t) + \frac{1}{ti}\int e(t)dt]$$

Secara diagram blok dapat digambarkan sebagai berikut :

$$\frac{U(s)}{E(s)} = Kp(1 + \frac{1}{ti.s}) = \frac{Kp(ti.s+1)}{ti.s}$$



Gambar 5.20 Diagram Blok Kontroler Tipe-PI

Dimana ti : merupakan konstanta waktu untuk kontroler integral dan *Kp* factor penguatan proporsional.

4. Kontroler Tipe-PD (Proportional + Differential Controller)

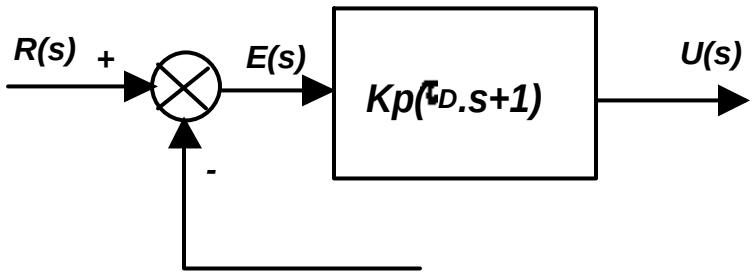
Merupakan kontroler yang menyatakan hubungan *sinyal eeror* dan *sinyal kontrol*, sehingga secara matematik dapat diformulasikan :

$$U(t) = Kp.e(t) + t_D \frac{d}{dt}e(t)$$

Dalam bentuk fungsi alih dapat dinyatakan :

$$\frac{U(s)}{E(s)} = Kp(t_D.s + 1)$$

Sehingga secara diagram blok dapat digambarkan sebagai berikut :



Gambar 5.21 Diagram Blok Kontroler Tipe-PD

Dimana τ_D :merupakan konstanta waktu Diferensial.

5. Kontroler Tipe-PD (Proportional + Integral + Differential Controller)

Kontroler tipe *PID* pada dasarnya dibedakan menjadi : 2(dua) macam yaitu :

⇒ Kontroler *PID* standart *(Standart PID Controller)*

⇒ Kontroler *PID* termodifikasi *(Modified PID Controller)*

5.1. Kontroler PID Standart

Hubungan antara *sinyal error* dan *sinyal kontrol* dapat dinyatakan :

$$U(t) = Kp[e(t) + \frac{1}{t_i}\int e(t)dt + t_D \frac{d}{dt}e(t)]$$

Dalam bentuk fungsi alih dapat diformulasikan :

$$\frac{U(s)}{E(s)} = Kp(1 + \frac{1}{t_i.s} + t_D s) = \frac{Kp(t_i.t_D.s^2 + t_i.s + 1)}{t_i.s}$$

5.2. Kontroler PID Termodifikasi (Modified PID Controller)

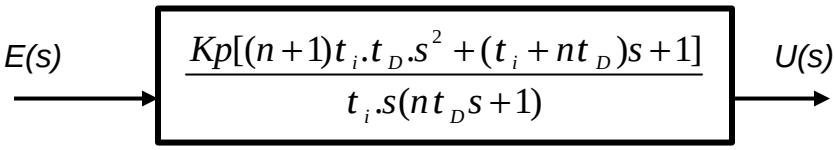
Hubungan *sinyal error* dan *sinyal kontrol* dalam bentuk fungsi alih adalah :

$$\frac{U(s)}{E(s)} = Kp(1 + \frac{1}{t_i.s} + \frac{t_D s}{n t_D s + 1})$$

Atau dinyatakan dalam bentuk persamaan lain :

$$\frac{U(s)}{E(s)} = \frac{Kp[(n+1)t_i.t_D.s^2 + (t_i + nt_D)s + 1]}{t_i.s(nt_Ds + 1)}$$

Secara diagram blok sistem dapat digambar sebagai berikut :

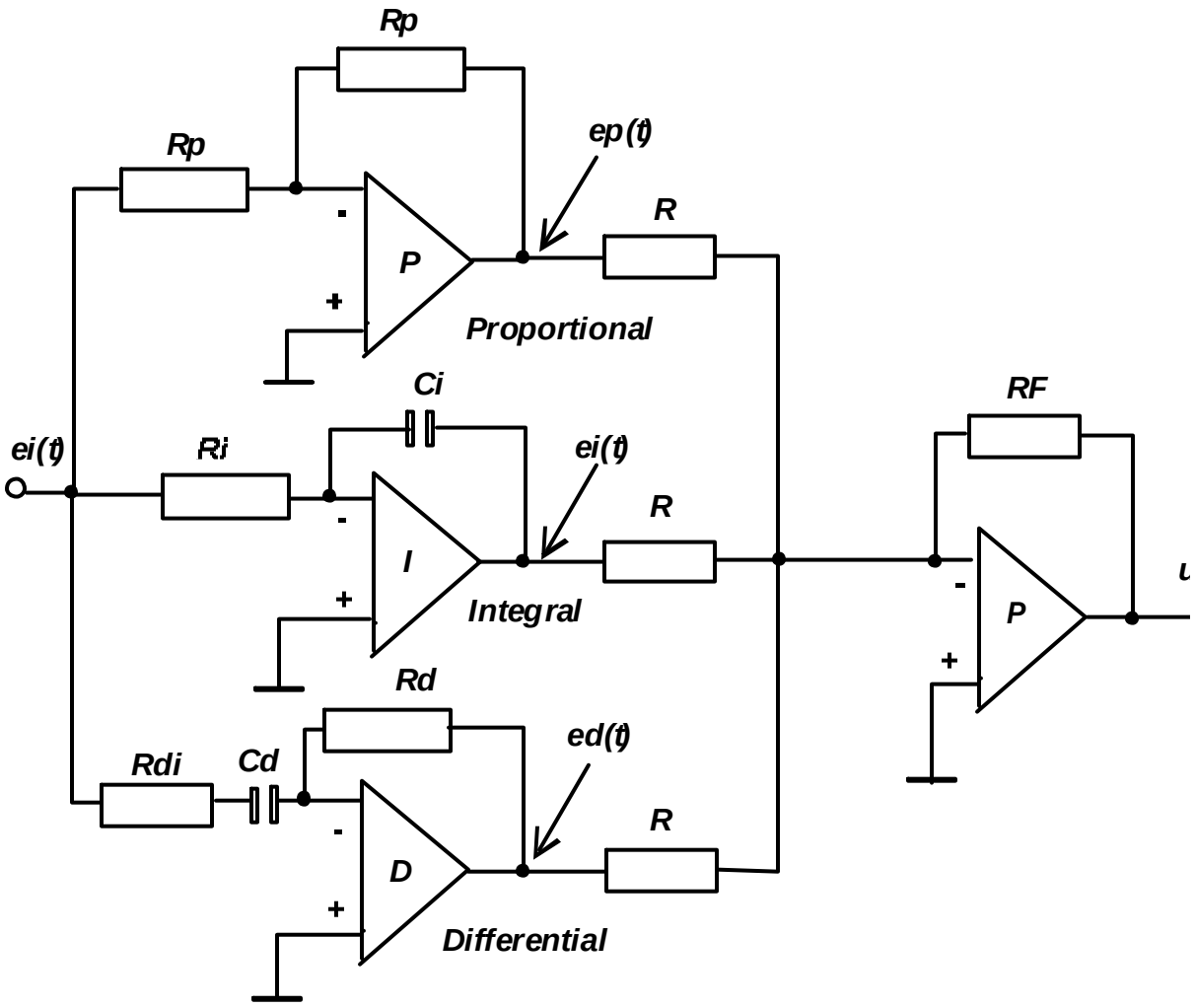


Gambar 5.22 Blok Diagram Transfer Function PID

C. APLIKASI RANGKAIAN KONTROLER PID DENGAN OP-AMP

Rangkaian Kontroler *PID* dengan menggunakan komponen analog *OP-AMP* dapat dibuat secara sederhanan yang dilengkapi dengan analisis fungsi alih secara matematik.

Rangkaian *Kontroler PID* ini berarti terdiri dari 3(tiga) buah rangkaian dasar *OP_AMP* yaitu masing-masing : *Kontroler P, Kontroler I dan Kontroler D.*



Gambar 5.23 Rangkaian Kontroler PID dengan menggunakan OP-AMP

1. Analisis Kontroler PID terbuat dari OP-AMP

Dari gambar OP-AMP. Halaman 6 tersebut dapat ditentukan persamaan :

$$U(t) = -\frac{RF}{R}[ep(t) + ei(t) + ed(t)] = -Kp[ep(t) + ei(t) + ed(t)]$$

Karena :

$$E_p(t) = -\frac{R_p}{R_p}e(t) = -e(t)$$

$$e_i(t) = -\frac{1}{R_i C_i}\int e(t) = -\frac{1}{t_i}\int e(t)dt$$

$$e_d(t) = -R_d C_d \frac{d}{dt}e(t) = -t_D \frac{d}{dt}e(t)$$

Maka persamaan di atas menjadi :

$$U(t) = -K_p[-e(t) - \frac{1}{t_i}\int e(t)dt - t_D \frac{d}{dt}e(t)]$$

$$U(s) = K_p[E(s) + \frac{1}{t_i s}E(s) + t_D sE(s)] = K_p[1 + \frac{1}{t_i s} + t_D s]E(s)$$

Sehingga persamaan Fungsi alih dari Kontroler *PID* tersebut adalah :

$$\frac{U(s)}{E(s)} = \frac{K_p[1 + \frac{1}{t_i s} + t_D s]E(s)}{E(s)} = K_p[1 + \frac{1}{t_i s} + t_D s] = \frac{K_p[t_i t_D s^2 + t_i s + 1]}{t_i s}$$

D. METODA PERANCANGAN KONTROLER PID

Seperti diketahui umumnya sistem plant di industri menggunakan Kontroler sistem orde tinggi. Namun biasanya orde tinggi ini direduksi menjadi orde satu atau orde dua. Untuk keperluan ini biasanya Kontroler PID banyak diimplementasikan di industri.

Beberapa metoda perancangan Kontroler PID yaitu :

1. Pendekatan waktu

a. Berdasarkan analitik dengan spesifikasi Respon orde I dan orde II.
b. Metoda Ziegler – Nichols.
c. Metoda Root Locus

2. Pendekatan Respon waktu :

a. Metoda analitik melalui Diagram Bode.
b. Metoda teknik Kompensator Lead/Lag melalui Diagram Bode.

3. Perancangan Adaptip.

Pada prinsipnya perancangan Kontroler PID adalah menentukan nilai dari parameter : *Kp; Ki* atau *ti* dan K_D atau t_D , sehingga respon sistem hasil desain sesuai dengan spesifikasi dan performansi yang diinginkan.

Tahapan pekerjaan perancangan Kontroler PID secara analitik antara lain :

(1). Menentukan model matematik *plant* (model reduksi dalam bentuk Orde satu atau Orde dua bila plant memiliki system Orde tinggi).

(2). Menentukan spesifikasi performansi :

-*Settling time* dan *% error steady state* untuk system Orde satu;

-*Settling time* dan *% over-shoot* serta % error steady state untuk pendekatan respon system Orde dua.

(3). Merancang Kontroler PID (tahapan akhir) yang meliputi :

-Pemilihan tipe kontroler (termasuk model *plant* dan tipe kontroler);

-Menghitung nilai parameter.

** CATATAN :*

⇒ Bila model *plant* Orde 0, maka dipilih Kontroler P, namun Kontroler P ini jarang digunakan karena terlalu sederhana dan hanya memenuhi salah satu dari kualitas.

⇒ Bila model *plant* Orde I, maka dipilih Kontroler PI dan bila model *plant* Orde II dipilih Kontroler PID.

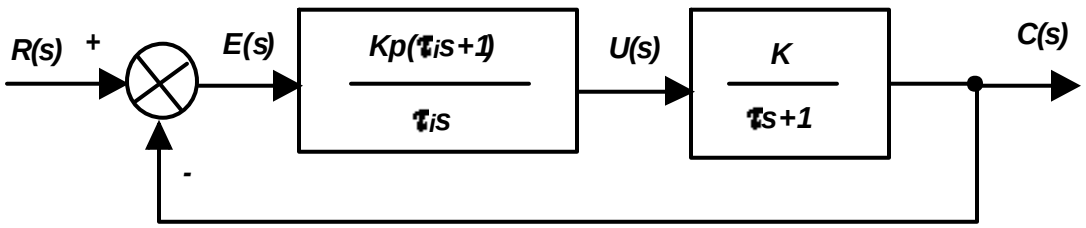
⇒ Bila Kontroler PD juga jarang digunakan karena memerlukan *Tuning* parameter yang presisi, memiliki respon yang sangat cepat dan cenderung tidak stabil.

E. CONTOH PERANCANGAN KONTROLER

1. Kontroler PI untuk Plant Orde I

Plant Orde I dikontrol oleh Kontroler PI, dengan *input R(s)* dan *output C(s)* dengan membentuk sinyal umpan balik ke *input R(s)*.

Maka system Kontroler PI dan Plant orde I tersebut dapat digambar secara diagram blok sebagai berikut :



Gambar 5.24 Diagram Blok Kontroler PI Plant Orde 1

Fungsi alih dari gambar diagram blok tersebut adalah :

$$C(s) = E(s)\frac{K_p(t_i s+1)}{t_i s}.(\frac{K}{ts+1})$$

Karena :

$$U(s) = E(s)\frac{K_p(t_i s+1)}{t_i s}$$

Dan

$$R(s) = E(s) + C(s) = E(s) + \frac{E(s)[K_p K(t_i s+1)]}{t_i s(ts+1)}$$

Berarti :

$$\frac{C(s)}{R(s)} = \frac{U(s).\frac{K}{ts+1}}{R(s)}$$

Maka Fungsi alih Kontroler PI dengan Plant tersebut adalah :

$$\frac{C(s)}{R(s)} = \frac{E(s).\frac{K_p(t_i s+1)}{t_i s}.\frac{K}{ts+1}}{E(s) + \frac{E(s)[K_p K(t_i s+1)]}{t_i s(ts+1)}} = \frac{E(s)[\frac{KK_p(t_i s+1)}{t_i s(ts+1)}]}{E(s)[1 + \frac{KK_p(t_i s+1)}{t_i s(ts+1)}]}$$

$$= \frac{KK_p(t_i s+1)}{t_i s(ts+1)}.\frac{t_i s(ts+1)}{t_i s(ts+1) + KK_p(t_i s+1)} = \frac{KK_p(t_i s+1)}{t_i s(ts+1) + KK_p(t_i s+1)}$$

Dari persamaan ini dapat disimpulkan bahwa *CLTF = C(s)/R(s)* mempunyai variasi yaitu :

-Bila *ti* = *t* , hasil desain system adalah Orde I

-Bila *ti* ¹ *t* , hasil desain system adalah Orde II.

-Desain Orde dari system tergantung pada pemilihan harga τi (Integral Time Constan).

a). Untuk $ti = t$, maka *CLTF* menjadi sebagai berikut :

$$\frac{C(s)}{R(s)} = \frac{K_pK(t_is+1)}{t_is(ts+1)+K_pK(t_is+1)} = \frac{K_pK}{(\frac{t_i}{K_pK}s)+1}$$

Atau dalam persamaan umum biasa dinyatakan :

$$\frac{C(s)}{R(s)} = \frac{1}{(t^*s+1)}$$

Dimana $t^* = \frac{t_i}{(K_p.K)}$ merupakan spesifikasi desaian dengan

gain $K_p.K = \frac{t_i}{(t^*.K)}$

b). Untuk $ti \neq t$, maka *CLTF* menjadi sebagai berikut :

$$\frac{C(s)}{R(s)} = \frac{K_pK(t_is+1)}{t_its^2+t_is+K_pKt_is+K_pK} = \frac{K_pK(t_is+1)}{t_its^2+(1+K_pK)t_is+K_pK}$$

$$\frac{C(s)}{R(s)} = \frac{(t_is+1)}{\frac{t_it}{K_pK}s^2+\frac{(1+K_pK)t_i}{K_p.K}s+1}$$

Atau dalam persamaan umum dapat dinyatakan :

$$\frac{C(s)}{R(s)} = \frac{(t_is+1)}{\frac{1}{\nu_n^2}s^2+\frac{2z}{\nu_n}s+1}$$

Dimana $\frac{1}{\nu_n^2} = \frac{t_it}{K_pK}$ dan $\frac{2z}{\nu_n} = \frac{(1+K_pK)t_i}{K_pK}$

2. Kesimpulan Orde I

a. Hasil desaian system Orde I adalah paling disukai dalam praktek, karena tidak memiliki "*Over-shoot*", *Zero Off-set (% Ess=0%)* dengan "*time constan*" yang baru t^*.

   Jadi *plant* Orde I dengan Kontroler *PI*, bila dipilih $ti = t$, system hasil desain adalah sistem Orde I dengan *time constan* t^* dan *Zero Off-set*.

b. Hasil desain system Orde II dengan *delay* dan *Zero Off-set*. Parameter sistem hasi desain antara lain : frekuensi natural *(*w_n*)*; koefisien redaman *(x)* dan *factor delay* (t_i*)*

   Jadi *plant* Orde I dengan Kontroler *PI* bila $ti \neq t$, sistem hasil desain adalah Orde II, *Zero Off-set* dengan parameter system w_n, x, dan t_i.

5.4 Kontroler Logika Fuzzy *(Fuzzy Logic Controller)*

Kontroler Logika *Fuzzy* (KLF) adalah suatu kontrol yang menggunakan metodologi digital dalam melakukan proses pengontrolan sistem. Proses pengontrolan dilakukan dengan strategi dan simulasi sistem fisik yang alami artinya hasil proses kontrol mendekati kondisi riil yang sesungguhnya [7].

Logika *Fuzzy (fuzzy logic)* adalah suatu logika yang menerapkan derajat kebenaran secara samar (*fuzzy)*, artinya logika *fuzzy* (*fuzzy logic*) mempunyai derajat kebenaran berbentuk linguistik yang menyertakan predikat kekaburan (*fuzziness*) sesuai proporsinya. Sebagai contoh derajat kebenaran suhu dinyatakan dalam sangat dingin, dingin, sedang, panas, sangat panas dan seterusnya. Bila dinyatakan dalam bilangan logika bolean *fuzzy* maka derajat kebenarannya diantara interval *0* sampai dengan *1*. Artinya derajat kebenaran bisa dirancang dari *"0"@*: *suhu sangat dingin*, *"0.3"@*: *suhu dingin* , *"0.5"@*: *suhu sedang* , *"0.7"@ suhu panas* sampai *"1.0"@ suhu sangat panas*.

Berbeda dengan logika konvensional, derajat kebenarannya dinyatakan secara pasti *(crispy)* yaitu salah satu dari 2(dua) pilihan. Definisi kebenarannya hanya berharga *"0"* atau *"1"* saja, tidak dapat dinyatakan antara suhu sangat dingin, dingin, sedang sampai panas. Dengan menggunakan sistem logika konvensional, dipastikan mengalami kesulitan jika diinginkan pembagian suhu mendekati kondisi sebenarnya.

5.4.1. Konsep Dasar Logika Fuzzy *(Basic Concepts of Fuzzy Logic)*

Konsep teori logika *fuzzy* pertama kali dikenalkan oleh Lotfi A. Zadeh (1965) melalui teori himpunan *fuzzy (fuzzy set).* Konsep ini didasari oleh kebutuhan untuk memperoleh metoda dalam memngembangkan analisis dan mempresentasikan dari masalah riil di lapangan yang serba tidak selalu tepat dan pasti.

Teori *fuzzy* ini pernah ditentang ilmuwan matematik Prof. W.Kahan kolega Prof. Lotfi A. Zadeh dengan pernyataan : *" What we need is more logical thinking, not less. The danger of fuzzy theory is that it will encourage the sort of imprecise thinking that has brought us so much trouble".*("Apa yang kita inginkan adalah lebih pada pemikiran yang logis, tidak kurang. Bahaya dari teori *fuzzy* adalah akan menimbulkan semacam pemikiran yang tidak presisi dan akan memberikan kepada kita banyak kesulitan")[15].

Pengembangan teori *fuzzy* ini banyak didukung para ahli seperti Prof. Ebrahim Mamdani (1974) dengan *"Succeeded to apply*

fuzzy logic for control in practice", kemudian diteruskan oleh Rutherford, dan Pedricz. Pada dekade (1965-1975) Ebrahim Mamdani, dari *Queen Mary College* London meneliti : *"Aplikasi fuzzy meliputi proses pada tangki pencampur"*, M. Sugeno dari *Tokyo Institute of Technology* dan Yamanakawa dari *Kyusu Institute of Technology* meneliti tentang :*"Komputer Fuzzy"*. Dari penelitian inilah teori fuzzy kemudian dapat diterima oleh masyarakat ilmiah sebagai terobosan dibidang kontrol cerdas [15].

Secara teori ada 4(empat) dasar konsep logika fuzzy yaitu : himpunan fuzzy, variabel linguistik, distribusi kemungkinan dan aturan *fuzzy JIKA-MAKA* [5].

5.4.2. Teori Himpunan Fuzzy *(Fuzzy Set Theory)*

Himpunan *fuzzy A* dalam semesta pembicaraan *X (universe of discourse)* adalah kelas atau kumpulan kejadian pasangan elemen *x* (*x* anggota dari *X*) dengan derajat keanggotaan *(grade of membership)* elemen tersebut yaitu fungsi keanggotaan $\mu_A(x)$ dengan nilai riil, interval (*0 , 1)* pada tiap *x* dalam *X*. Derajat kebenaran logika *fuzzy* didasarkan μ_A, dimana $\mu_A(x)=1$, berarti *x* sebagai anggota penuh himpunan *A*, tetapi bila $\mu_A(x)=0$, berarti *x* bukan anggota himpunan A [2, 5, 7, 15, 22].

$$\mu_A(x) = \begin{cases} 1 \rightarrow bila\ dan\ hanya\ bila\ x \in A \\ 0 \rightarrow bila\ dan\ hanya\ bila\ x \notin A \end{cases}$$

(3.1)

Bila pendukung sekumpulan *X* dalam himpunan *fuzzy A* maka dapat dinyatakan:

$$A = \{(x, \mu_A(x)) / x \in X\}$$

(3.2)

Untuk *X* diskrit dengan *n* elemen pendukung *(n =1, 2, 3, …, n)* dari *A*, maka :

$$A = \mu_1(x)/x_1 + \mu_2(x)/x_2) + ... + \mu_n(x)/x_n)\}$$

(3.3)

$$A = \sum_{i=1}^{n} \mu_A(x_i)/x_i \qquad [x:diskrit]$$

(3.4)

$$A = \int_x \mu_A(x)\ /\ x \qquad [x : kontinyu]$$

(3.5)

Bila elemen pendukung himpunan fuzzy *A,* $\mu_A(x) = 0.5$ merupakan titik silang *(cross-over)*, dan bila pendukung tunggal dengan $\mu_A(x) = 1.0$ berarti *fuzzy* tunggal *(singleton fuzzy*).

Tanda +, $\mathring{a}$, $\int_x$ pada rumus di atas merupakan operator gabungan *(union)*.

5.4.3. Fungsi Keanggotaan Fuzzy *(Fuzzy Membership Function)*

Pada dasarnya untuk menyatakan fungsi keanggotaan tergantung pada pendifinisian fungsi keanggotaan *fuzzy* dan bentuk permasalahan yang dibahas.

Berdasarkan hasil pengamatan studi literatur ada 2(dua) metode pendefinisian yaitu pendefinisian secara bentuk fungsi untuk himpunan dengan pendukung kontinyu dan bentuk numerik yang diterapkan untuk himpunan dengan pendukung diskrit [7].

5.4.3.1. Pendefinisian Bentuk Fungsi

Pendefinisian secara bentuk fungsi adalah suatu penyelesaian permasalahan untuk himpunan *fuzzy* dengan pendukung/penyokong kontinyu.

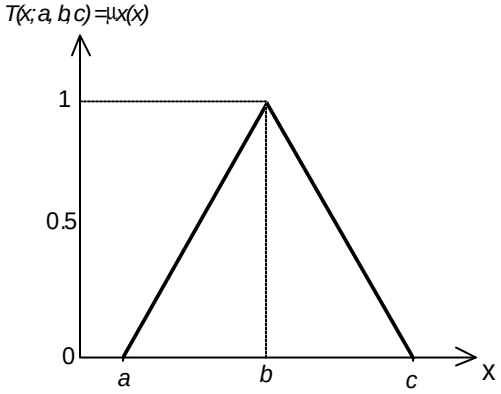
Tipe fungsi keanggotaan *fuzzy* berdasarkan pendifinisian bentuk fungsi adalah fungsi *segitiga , trapesium* , pi *(p), S, Bell-shaped, Gaussian* dan lain-lain. [2, 5, 7, 15, 22].

a. Fungsi Keanggotaan Triangular *(T-function)*

Fungsi keanggotaan berbentuk segitiga atau *Triangular function (T-function)* adalah paling banyak digunakan dalam proses *fuzzifikasi*, terutama dalam penerapan teori *fuzzy* pada sistem pengaturan maupun pada pengenalan pola.

Penggunaan fungsi keanggotaan dengan distribusi segitiga ini sangat beralasan karena disamping lebih sederhana bentuk formulasinya, lebih mudah pula dalam analisis perhitungan untuk menentukan algoritmanya, sehingga tidak banyak menyita waktu dalam melakukan proses perhitungannya [22].

Fungsi keanggotaan distribusi bentuk segitiga *(triangular)* seperti Gambar 3.1.



Gambar 5.25 Fungsi Keanggotaan bentuk Segitiga (Triangular) [5, 7, 22]

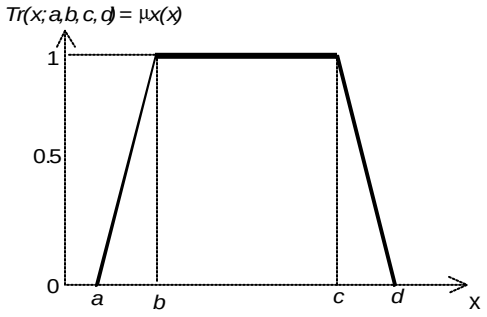
Persamaan secara matematik fungsi keanggotaan bentuk segitiga *(triangular)* adalah :

$$T(x;a,b,c) = m_x(x) = \begin{cases} 0 & x \le a \\ \dfrac{(x-a)}{(b-a)} & a \le x \le b \\ \dfrac{(c-x)}{(c-b)} & b \le x \le c \\ 0 & x \ge c \end{cases}$$

(3.6)

b. Fungsi Keanggotaan jenis Trapezoidal*(Tr-function)*

Fungsi keanggotaan bentuk trapesium *(Tr-function)* seperti Gambar 3.2 :



Gambar 5.26 Fungsi Keanggotaan bentuk Trapesium [5, 7, 22]

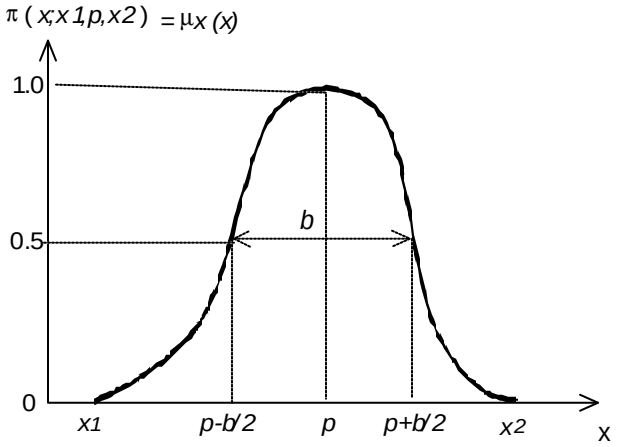
Persamaan secara matematik fungsi keanggotaan bentuk trapesium *(trapezoid)* adalah:

$$Tr(x;a,b,c,d) = \begin{cases} 0 & x \le a \\ \dfrac{(x-a)}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{(d-x)}{(d-c)} & c \le x \le d \\ 0 & x \ge d \end{cases}$$

(3.7)

c. Fungsi Keanggotaan jenis *pi* (π)

Fungsi keanggotaan bentuk π-*function* disebut juga sebagai fungsi keanggotaan jenis *Lonceng* yaitu fungsi keanggotaan yang terdiri dari 2(dua) parameter seperti Gambar 3.3 :



Gambar 5.27 Fungsi Keanggotaan bentuk π [5, 7, 15, 22]

Persamaan matematik fungsi keanggotaan bentuk *pi (p)* adalah [5, 7, 15, 22]:

$$p(x;b,p) = m_x(x) = \begin{cases} 2\dfrac{(x-p+b)^2}{b^2} & p-b \le x \le p-\dfrac{b}{2} \\ 1-2\dfrac{(x-p+b)^2}{b^2} & p-\dfrac{b}{2} \le x \le p+\dfrac{b}{2} \\ 2\dfrac{(x-p-b)^2}{b^2} & p+\dfrac{b}{2} \le x \le p+b \end{cases}$$

(3.8)

5.4.3.2. Pendefinisian Bentuk Numerik

Pendefinisian secara numerik hanya untuk fungsi keanggotaan dengan pendukung diskrit, yaitu mengambil nilai bentuk fungsi untuk tiap pendukung *x* yang berhingga jumlahnya [2, 5, 15, 22].

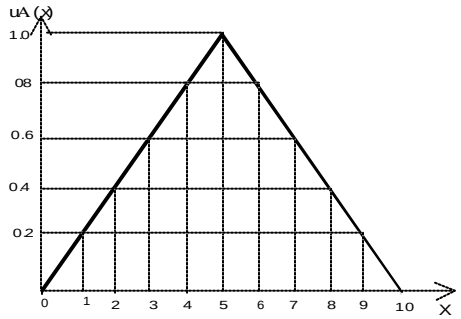
Sebagai contoh:

X={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; m_A*={0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 0.8, 0.6, 0.4, 0.2, 0.0 }*

Sehingga himpunan *fuzzy* yang didefinisikan untuk himpunan elemen *x* secara diskrit adalah seperti persamaan ini dan bila digambar secara grafik seperti Gambar 3.4.

A ={0.0/0, 0.2/1, 0.4/2, 0.6/3, 0.8/4, 1.0/5, 0.8/6, 0.6/7, 0.4/8, 0.2/9, 0.0/10}

A={0.0/0+0.2/1+0.4/2+0.6/3+0.8/4+1.0/5+0.8/6+0.6/7+0.4/8+0.2/9+0.0/10}



Gambar 5.28 Definisi Himpunan Fuzzy A secara Diagramatik [2, 5, 7, 15, 22]

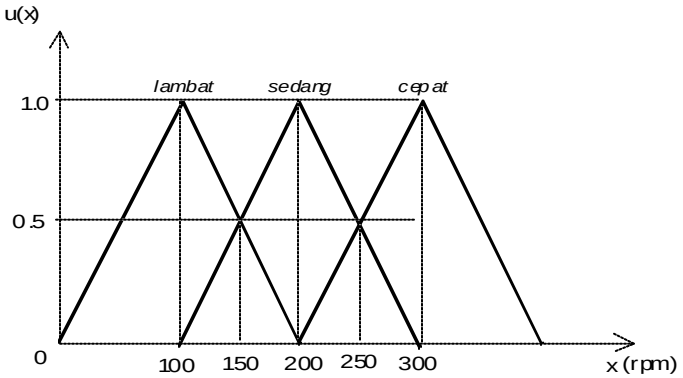
5.4.4. Variabel Linguistik

Sistem pengaturan dengan pendekatan logika *fuzzy* merupakan sistem pengaturan yang yang menirukan cara kerja manusia dalam melakukan proses pengambilan keputusan, kaidah atur melalui ungkapan-ungkapan kualitatip terhadap informasi yang diteranya, misalnya menyatakan kecepatan motor : *sangat cepat*, *cepat*, *sedang*, *rendah* dan *berhenti*. Semua kata-kata dalam bentuk kualitatip dengan variabel linguistik ini selanjutnya ditabelkan dan diformulasikan dalam bentuk aturan dasar *(rule base)* sesuai dengan pengalaman operator atau manusia ketika mengatur proses secara manual [22].

Variabel linguistik sebagai ganti dari variabel numerik yang biasa digunakan dalam pendekatan kuantitatip yang menyatakan

fungsi dari himpunan bagian semesta pembicaraan *(universe of discourse)* dari himpunan *fuzzy.* Variabel linguistik ini misalkan dinyatakan dalam *(u,T(u),X),* dimana *u* adalah nama variabel dan *T(u)* merupakan pernyataan himpunan *u* yaitu nilai linguistik himpunan *fuzzy* atas semesta *X*.

Misalkan variabel *level kecepatan* putaran motor dapat dinyatakan dengan istilah : *T(kecepatan) = {lambat, sedang, cepat}*. Variabel ini didefinisikan untuk semua *X = [0, 200, 300]*, dengan kata lain *lambat* disekitar *100 rpm*, *sedang* disekitar *200 rpm* dan *cepat* sekitar *300 rpm*. Maka fungsi segitiga mendefinisikan secara fungsional kedua himpunan ini dipilih sedemikian rupa sehingga penafsiran secara grafis dari pendefinisian terjadi di titik silang *(cross-over)* masing-masing terletak di titik *x = 150 rpm.*, *250 rpm* dan seterusnya dengan pendukung nilai keanggotaan *m(x) = 0.5* untuk himpunan *rendah, sedang* dan *cepat* seperti Gambar 3.5.



Gambar 5.29. Penafsiran Grafis Variabel Linguistik

Penafsiran grafis gambar di atas memberikan nilai-nilai keangotaan untuk yang besar bila titik tersebut berada di titik *100 rpm*.,*200 rpm*. dan *300 rpm.,* masing-masing mempunyai derajat kebenaran *(1.0)* berarti sebagai anggota himpunan penuh dari masing-masing himpunan *lambat*, *sedang,* dan *cepat*, biasanya disebut *fuzzy singleton*.

Untuk titik *0 £ x £ 100 rpm., 100 rpm.£ x £ 200 rpm., 200 rpm.£ x £ 300 rpm.* mempunyai nilai keanggotaan kurang dari *1.0* untuk himpunan *lambat*, *sedang* dan *cepat*. Ini berarti pada *x £ 100 rpm* mempunyai kebenaran yang kuat untuk menjadi anggota himpunan *lambat*, sebaliknya karena nilai keanggotaan yang kecil misalkan pada daerah *100 rpm.£ x £ 200 rpm* terletak pada himpunan. Titik *150 rpm*. mempunyai nilai kebenaran yang sama *(0.5)* untuk menjadi anggota himpunan *lambat* maupun *sedang,* dan titik *210 rpm.* lebih dominan

menjadi anggota himpunan *sedang* dari pada menjadi anggota himpunan yang lain, namun untuk *250 rpm* merupakan angota himpunan dari kedua kebenaran *level kecepatan* yaitu anggota himpunan *sedang* dan *cepat* dengan derajat kebenaran yang *sama (0.5),* yang disebut *cross-over* yang sama.

Dari penafsiran Gambar 3.5 dapat dituliskan *X = {0,50,100,150,200,250,300}*, maka secara definisi matematik himpunan *fuzzy* yaitu :

A º lambat = 0.0/0 + 0.5/50 + 1.0/100 + 0.5/150 + 0.0/200

B º sedang = 00./100 + 0.5/150 + 1.0/200 + 0.5/250 + 0.0/300

C º cepat = 00./200 + 0.5/250 + 1.0/300 + 0.5/..+ 0.0/ dan seterusnya.

3.1.5. Operasi Himpunan Fuzzy

Jika *A* dan *B* merupakan himpunan fuzzy dalam semesta pembicaraan *X (universe of discourse)* dengan fungsi keanggotaan $\mu_A(x)$ dan $\mu_B(x)$, maka operasi dasar dari himpunan fuzzy dapat diuraikan sebagai berikut [2, 5, 7, 12, 15, 22, 25]:

1). Himpunan yang sama (*Equality)*

$$\mu_A(x) = \mu_B(x), \text{untuk } x \in X \quad (3.9)$$

2). Gabungan (Union)

$$\mu_{(A \cup B)}(x) = \max\{\mu_A(x), \mu_B(x)\}, \quad x \in X \quad (3.10)$$

3). Irisan (Intersection)

$$\mu_{(A \cap B)}(x) = \min\{\mu_A(x), \mu_B(x)\}, \quad x \in X \quad (3.11)$$

4). Komplemen (Complement)

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x), \quad x \in X \quad (3.12)$$

Bila himpunan $\bar{A}$ merupakan komplemen dari *A,* maka untuk aturan yang lain berlaku:

$$\mu_{(A \cap \bar{A})}(x) = \min\{\mu_A(x)\mu_{\bar{A}}(x)\} \leq 0.5 \quad (3.13)$$

$$m_{(A \cup \bar{A})}(x) = \max \{m_A(x), m_{\bar{A}}(x)\} \geq 0.5$$

(3.14)

5). *Normalisasi (Normalization)*

$$m_{Normal(A)}(x) = m_A(x) / \max(m_A(x)), \quad x \in X$$

(3.15)

6). *Konsentrasi (Concentration)*

$$m_{CON(A)}(x) = (m_A(x))^2, \quad x \in X$$

(3.16)

7). *Dilasi (Dilation)*

$$m_{DL(A)}(x) = (m_A(x))^{0.5}, \quad x \in X$$

(3.17)

8). *Produk Aljabar (Algebraic Product)*

$$m_{(A.B)}(x) = m_A(x).m_B(x), \quad x \in X$$

(3.18)

9). *Gabungan Terikat (Bounded Sum)*

$$m_{(A \oplus B)}(x) = \min\{1, m_A(x) + m_B(x)\}, \quad x \in X$$

(3.19)

Gabungan Terikat atau Hasil Penjumlahan Terikat sering disebut *Bounded Sum*

10). *Produk Terikat (Bounded Product)*

$$m_{(A \ominus B)}(x) = \max\{0, m_A(x) - m_B(x)\}, \quad x \in X$$

(3.20)

Atau $= \max\{0, m_A(x) + m_B(x) - 1\}, \quad x \in X$

Produk Terikat atau Hasil Kali Terikat dalam bahasa lain disebut *Bounded Product* atau juga sering disebut *Bounded Difference*.

11).*Jumlah Probabilistik (Sum of Probability)*

$$m_{\wedge (A+B)}(x) = m_A(x) + m_B(x) - m_A(x).m_B(x), \quad x \in X$$

(3.21)

12). *Intensifikasi (Intensification)*

$$m_{INT(A)}(x) = \begin{cases} 2(m_A(x) & 0 \le m_A(x) \le 0.5 \\ 1-2(1-m_A(x)^2 & 0.5 \le m_A(x) \le 1 \end{cases}$$

(3.22)

13). *Produk Drastis (Drastic Product)*

$$m_{INT(A)}(x) = \begin{cases} m_A(x) & m_B(x) = 1 \\ m_B(x) & m_A(x) = 1 \\ 0 & m_A(x), m_B(x) < 1 \end{cases}$$

(3.23)

14). *Produk Cartesian (Cartesian Product)*

Jika $A_1, A_2, \ldots, A_n$ adalah himpunan fuzzy dalam produk $X_1, X_2, \ldots, X_n$ dengan fungsi keanggotaan, maka didefinisikan sebagai:

$$m_{A1xA2x\ldots xAn}(x_1, x_2, \ldots, x_n) = \min\{ m_{A1}(x_1), m_{A2}(x_2), \ldots, m_{An}(x_n)\}$$

(3.24)

15). *Relasi Fuzzy (Fuzzy Relation)*

Bila *n* relasi *fuzzy* adalah himpunan *fuzzy* dalam $X_1 \times X_2, x \ldots \times X_n$ dan fungsi keanggotaannya didefinisikan sebagai:

$$m_{X_1 \ldots . Xn} = \{((x_1, x_2, \ldots, x_n) m_R(x_1, x_2, \ldots, x_n)) / (x_1, x_2, \ldots, x_n) \in x_1 \times x_2 \times \ldots \times x_n\}$$

(3.25)

16). *Komposisi Sup-Star (Sup-Star Composition)*

Jika *R* dan *S* merupakan relasi *fuzzy* dalam $X \times Y$ dan $Y \times X$, maka komposisi *R* dan *S* adalah relasi *fuzzy* yang dinyatakan dengan *R o S* dan didefinisikan sebagai:

$$R \text{ o } S = \{[(x,z), \sup (m_R(x,y) * m_s(y,z))], x \in X, y \in Y, z \in Z\}$$

(3.26)

Notasi * : operator bentuk segitiga, minimum, produk aljabar, produk terbatas.

5.4.6. Metode Perancangan KLF

Metoda perancangan kontrol klasik seperti *Nyquist, Bode, Root Locus* dan *Nichols* umumnya didasarkan atas acuan asumsi bahwa proses yang dikontrol adalah linier dan stasioner [6, 8, 22]. Namun kenyataannya proses yang dikontrol yang ada sampai saat ini merupakan sistem yang komplek, non-linier dan mudah dipengaruhi faktor-faktor gangguan di sekelilingnya.

Oleh karena itu perancangan sistem kontrol otomatik untuk keperluan proses tersebut digunakan metoda yang memadai, dalam hal ini digunakan teori logika *fuzzy* sebagai basis kontrol pada perancangan kontroler.

Secara umum Kontroler Logika *Fuzzy* (KLF) mempunyai kemampuan[15]:

1. Beroperasi tanpa campur tangan manusia secara langsung, memiliki efektifitas yang sama dengan manusia.
2. Mampu memecahkan masalah pada sistem yang komplek, non-linier, tidak stasioner.
3. Memenuhi spesifikasi operasional dan kriteria kinerja.
4. Strukturnya sederhana, kuat dan beroperasi *real time*.

Perancangan Kontroler Logika *Fuzzy* (KLF) atau "*Fuzzy Logic Controller (FLC)"* adalah gabungan dari pendefinisian himpunan *fuzzy* dengan logika *fuzzy,* yang mana untuk memperoleh kontroler yang menyerupai cara kerja operator.

Menurut penelitian Harris, Moore, dan Brown (1993) mengungkapkan metodologi perancangan Kontroler Logika *Fuzzy* (KLF) secara umum, artinya tidak mempunyai prosedur yang baku.

3.1.6.1. Struktur Dasar KLF

Kontroler Logika *Fuzzy* (KLF) pada dasarnya mempunyai struktur dasar yang sederhana seperti Gambar 3.6 berikut ini.

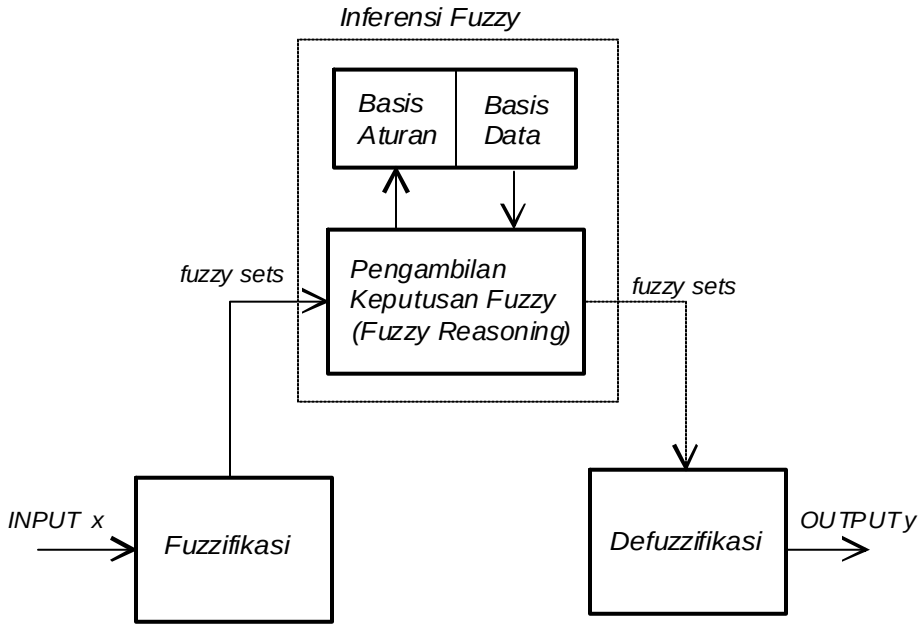
Namun dalam pengembangannya sebagai kontrol *closed loop*, maka banyak mengalami pengembangan dalam penyempurnaan rangkaiannya. Sehingga struktur KLF bila digabung dengan *plant* yang akan dikontol digambarkan seperti Gambar 3.7.

Setiap blok diagram struktur KLF berdasarkan fungsinya adalah sebagai berikut:

1. Fuzzifikasi *(fuzzifier)* berfungsi untuk mentransformasikan variabel masukan berupa sinyal masukan *(input)* yang bersifat bukan *fuzzy* (variabel numerik) untuk dikonversi atau diubah menjadi variabel *fuzzy* (variabel linguistik) dengan menggunakan operator *fuzzifikzsi (fuzzifier)*.
2. Basis Pengetahuan *(Knowledge Base)* terdiri dari basis data *(data base)* dan aturan dasar *(rule base)* yang mendefinisikan himpunan *fuzzy* atas daerah-daerah masukan dan keluaran yang disusun dalam perangkat aturan kontrol.
3. Inferensi *(Inference)* merupakan inti dari KLF karena di sini tempat logika pengambilan keputusan atau sering disebut juga "*reasoning"*, dimana harus mempunyai kemampuan seperti layaknya manusia dalam mengambil keputusan. Aksi dari

beberapa hasil pengambilan keputusan *fuzzy* (aksi atur) disimpulkan menggunakan relasi *fuzzy* (implikasi *fuzzy*) dan mekanisme inferensi *fuzzy*.

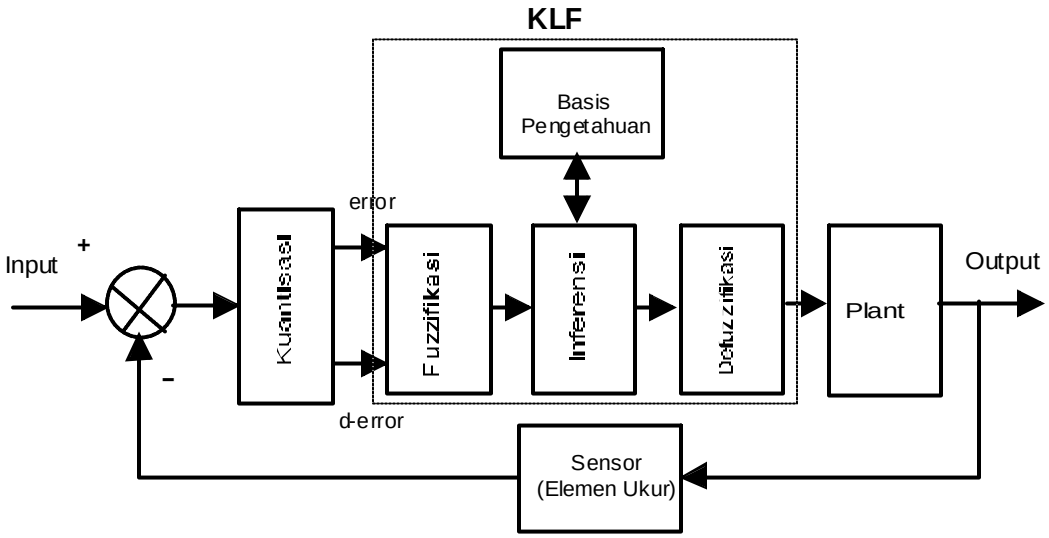
4. Defuzzifikasi *(defuzzier)* berfungsi untuk mentransformasikan kesimpulan tentang aksi dari pengambilan keputusan (aksi atur) yang bersifat *fuzzy* menjadi sinyal sebenarnya yang bersifat pasti *(crispy)* dengan menggunakan operator *defuzzier*.
5. Kuantisasi merupakan proses pengubahan sinyal masukan menjadi *error (e)* dan *delta error (de)* untuk diproses selanjutnya pada tingkat *fuzzifikasi*.
6. *Plant* adalah suatu sistem yang dikontrol berupa seperangkat peralatan yang bekerja bersama-sama untuk operasi KLF.



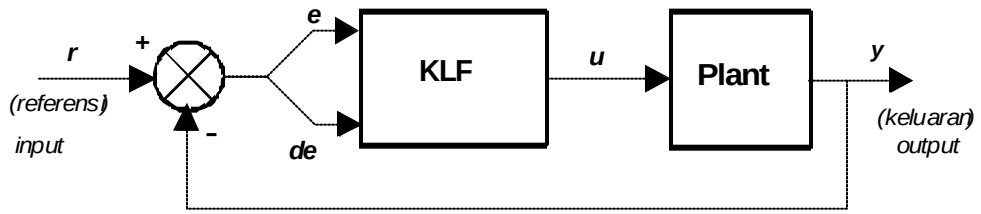
Gambar 5.30 Struktur Dasar Kontroler Logika Fuzzy [5, 7, 12]

5.4.6.2. Fuzzifikasi *(Fuzzifier)*

Untuk membahas *fuzzifikasi* sistem pengaturan *plant* dengan *input* tegangan *error(e)* dan *delta error(de)* dari kontroler logika *fuzzy*, berikut digambarkan diagram blok (Gambar 3.8) dan *step response* dari *plant* yang dikontrol (Gambar 3.9), sehingga proses *fuzzifikasi* mudah dipahami.



Gambar 5.31 Struktur Kontroler Logika Fuzzy dengan Plant [2, 5, 7, 12, 15]

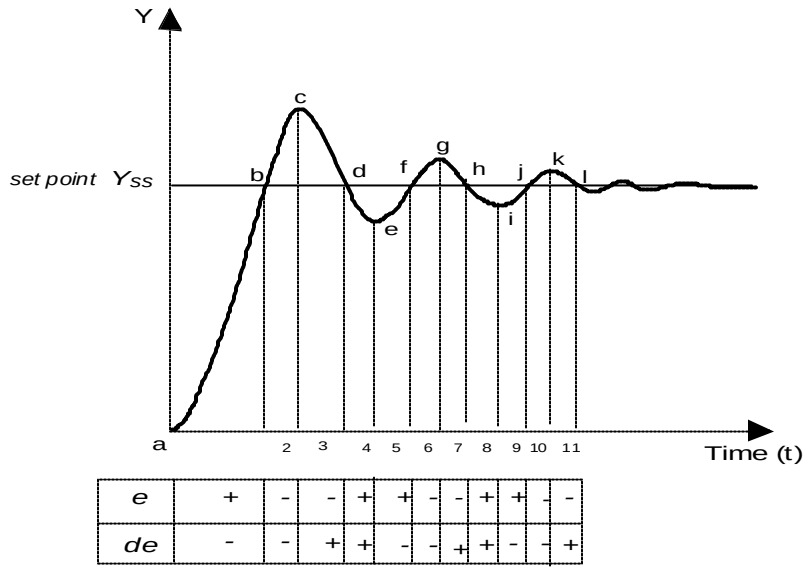


Gambar 5.32 Diagram Blok System Kontrol Logika Fuzzy dengan

Besaran *input* dari *plant* melalui umpan balik *(feedback)* lewat sensor bersifat pasti (*crisp)* dan kuantitatif, bisa juga berupa variabel numerik lalu dilakukan *fuzzifikasi* menjadi variabel linguistik *fuzzy*. *Fuzzifikasi* adalah pemetaan *input* kesemesta himpunan *fuzzy* dan secara simbolis pemetaan ini dinyatakan sebagai:

$$x = fuzzifier\ (x_o) \tag{3.27}$$

Dimana x_o : masukan *crisp*, x: himpunan *fuzzy* dan *fuzzifier* adalah operator *fuzzifikasi*.



Gambar 5.33 Tanggapan(step response) Sistem Loop Tertutup [5, 7, 22]

Jika variabel masukan yang diinginkan adalah 2(dua) variabel *error (e)* dan *delta error (de)*, maka dilakukan kuantisasi lalu diubah dahulu kedalam variabel *fuzzy.* Melalui fungsi keanggotaan *(membership function)* yang telah disusun nilai *error (e)* dan *delta error (de)* nilai kuantisasi akan didapatkan derajat keanggotaan bagi masing-masing *error (e)* dan *delta error (de)* tersebut.

Dari Gambar 3.9 adalah *step response* dari keluaran dengan masukan *unit step* delengkapi dengan uraian cara mem-*fuzzifikasi*, sehingga dapat dibuat bentuk penurunan pembagian ruang *input* misalkan dengan 3(tiga) ruang *input* dengan nilai linguistik *Positip(P), Negatip(N) dan Nol(NL)*.

Nilai kualitatip ditentukan pada setiap titik yang bertanda huruf *(a-l)* seperti contoh hasil analisis protipe aturan kontrol dengan 3(tiga) nilai linguitik pada Tabel 5.1.

Tabel 5.4. Protipe Aturan Kontrol Logika Fuzzy dengan 3(tiga) Nilai Linguistik [7]

Aturan No.	*Error (e)*	*d-Error (de)*	*Output (u)*	*Titik Referensi*	*Fungsi*
1	*P*	*NL*	*P*	*a,e,i*	*pemendekan rise time*
2	*NL*	*N*	*N*	*b,f,j*	*overshoot tereduksi*

3	*N*	*NL*	*N*	*c,g,k*	*overshoot tereduksi*
4	*NL*	*P*	*P*	*d,h,l*	*oscillasi tereduksi*
5	*NL*	*NL*	*NL*	*set point*	*sistem teredam*

Peninjauan secara kualitatif setiap daerah yang bertanda angka pada grafik Gambar 5.33 secara step respon menunjukkan daerah *1* mempunyai pengaruh pada pemendekan waktu naik *(shorten rise time),* dan daerah 2 berhubungan dengan pengurangan lewatan (*reduce overshoot)*, sehingga secara heuristik penalaran ini memberikan penambahan atau penyempurnaan aturan kontrol seperti pada Tabel 5.2.

Tabel 5.5 Penyempurnaan Aturan Kontrol Logika Fuzzy dengan 3(tiga) Nilai Linguistik [7]

Aturan No.	*Error (e)*	*d-Error (de)*	*Output (u)*	*Titik Referensi*	*Fungsi*
6	*P*	*N*	*P*	*Range: ab, ef*	*pemndekan rise time*
7	*N*	*N*	*N*	*Range:bc, fg, jk*	*overshoot tereduksi*
8	*N*	*P*	*N*	*Range: cd, gh*	*overshoot tereduksi*
9	*P*	*P*	*P*	*Range: de,hi*	*oscillasi tereduksi*
10	*P*	*N*	*NL*	*Range: ij*	*sistem teredam*
11	*N*	*P*	*NL*	*Range: kl*	*sistem teredam*

Beberapa teori aturan dasar kontrol dalam bentuk tabel fungsi keanggotaan dengan matrik 7x7 fungsi keanggotaan diantaranya oleh M. Braae dan D.A. Rutherford, 1979 : "*Theoritical and Linguistic Aspects of the Fuzzy Logic Controller"*.

Kemudian MacVicar-Whelan telah meneliti penyempurnaan aturan kontrol *fuzzy* ini dengan menemukan pola umum hubungan antara *error (e), delta error (de*) dan *output* kontrol *(u)* atau *delta output(du)*. Pendefinisian aturan dasar *(rule base)* pola ini berlaku bagi KLF dengan *input error(e)* dan *delta error(de)* seperti Tabel 5.3.

Tabel 5.6 Aturan Kontrol Fuzzy Mac Vicar-Whelan [22]

	NBe	*NMe*	NSe	ZEe	PSe	PMe	PBe
NBd	NBc	NBc	NBc	NBc	NMc	NSc	ZEc
NMd	NBc	NBc	NBc	NMc	NSc	ZEc	PSc
NSd	NBc	NBc	NMc	NSc	ZEc	PSc	PMc
ZEd	NBc	NMc	NSc	ZEc	PSc	PMc	PBc
PSd	NMc	NSc	ZEc	PSc	PMc	PBc	PBc
PMd	NSc	ZEc	PSc	PMc	PBc	PBc	PBc
PBd	ZEc	PSc	PMc	PBc	PBc	PBc	PBc

5.4.6.3. Penentuan Basis Data *(Data Base)*

Himpunan-himpunan *fuzzy* dari sinyal masukan dan sinyal keluaran supaya dapat digunakan oleh variabel linguistik dalam basis aturan maka didefinisikan lebih dahulu. Dalam perancangan basis data, ada hal-hal yang perlu diperhatikan yaitu:

a). Kuantisasi dan normalisasi

b). Pembagian ruang masukan dan keluaran

c). Penentuan fungsi keanggotan

a. Kuantisasi dan Normalisasi

Kuantisasi adalah mendiskritkan semesta pembicaran yang kontinu kedalam sejumlah segmen-segmen tertentu sebagai *level* kuantisasi. Di sini sebagai contoh proses pengambilan masukan yang berupa variabel numerik dari masukan *error (e)* dan *delta error (de)* Cara kuantisasi ini diperlukan bila pendefinisian fungsi keanggotaannya dinyatakan dalam bentuk numerik, dan biasanya

dengan cara membentuk tabulasi sebagai tabel pandang *(look-up table).*

Sebaiknya jumlah *level* kuantisasi tergantung permasalahan yang dikontrol dan sebaiknya dibuat sebanyak mungkin selama memori komputer yang digunakan untuk penghitungan mampu menyimpannya.

Level kuantisasi dapat ditentukan dengan menggunakan penghitungan berikut[7]:

$$L(Level) = \frac{X\max - X\min}{RES} \quad (3.28)$$

RES merupakan resolusi kontrol yang digunakan *(control resolution)*, *[Xmin, Xmax]* merupakan *range* semesta pengukuran *(universe of measurement)* [7].

Pemilihan jumlah *level* kuantisasi mempengaruhi kepekaan dari KLF terhadap masukan dan kehalusan *(smooth)* aksi pengaturan pada keluaran. Semakin banyak *level* kuantisasi dengan interval yang kecil yang diterapkan pada daerah masukan dan keluaran maka semakin sensitif dan terjadi deviasi yang kecil pada KLF tersebut, sehingga semakin *smooth* aksi kontrolnya [7].

Sedangkan normalisasi digunakan bila semesta pembicaraan dari himpunan tersebut terbatas dalam jangkauan tertentu, misalkan *-1.0* sampai dengan *+1.0*. Normalisasi merupakan pemetaan semesta pembicaraan masukan ke semesta pembicaraan baru yang terbatas. Skala pemetaan bisa seragam *(uniform)* atau bisa tidak seragam *(non-uniform)* tergantung kebutuhan perancangannya. Normalisasi ini diperlukan bila fungsi keanggotaan didefinisikan secara fungsional.

b. Pembagian Ruang Masukan dan Keluaran *(Input and Output)*

Pendefinisian himpunan *fuzzy* atas daerah masukan *(input)* dan keluaran *(output)* berarti membagi smesta pembicaraan atas nilai variabel linguistik himpunan *fuzzy*. Derajat nilai kebenarannya dinyatakan dengan linguistik seperti NB, NM, NS, ZE, PS, PM, PB dan sebagainya [5, 7, 12, 22].

Penentuan jumlah himpunan *fuzzy* dan aturan kontrol yang disusun sangat ditentukan oleh banyak sedikitnya pembagian ruang masukan dan keluarandalam bentuk variabel lingustik, yang dalam hal ini akan berakibat langsung pada kehalusan *(smooth)* dari pengaturan *fuzzy* tersebut.

c. Penentuan Fungsi Keanggotaan

Pendefinisian secara numerik dari tingkat keanggotaan pendukung/penyokong dalam himpunan *fuzzy* dinyatakan dalam bentuk tabulasi seperti Tabel 3.1 dan fungsi keanggotaan yang sering digunakan adalah fungsi segitiga *(triangular)*, trapesium *(trapezoid)* dan fungsi *pi* (π) seperti Gambar 3.1, 3.2, dan 3.3 halaman sebelumnya.

Pembagian ruang dibuat 7(tujuh) nilai linguistik himpunan *fuzzy* seperti Aturan Kontrol *Fuzzy Mac Vicar-Whelan* Tabel 3.3 yaitu: NB, NM, NS, ZE, PS, PM, PB.

Distribusi parameter dipilih seragam *(uniform)* atau bisa tidak seragam *(non uniform)* biasanya diletakkan pada bagian awal dan akhir dari fungsi keanggotaan dan ditentukan sedemikian rupa sehingga titik silang *(cross-over)* tepat ditengah-tengah dua rataan diantara himpunan yang bersebelahan seperti dicontohkan pada Gambar 3.10.

5.4.6.4. Basis Aturan *(Rule Base)*

Aturan dasar sistem *fuzzy* adalah sekelompok aturan *fuzzy* dalam hubungan antara keadaan sinyal masukan dan sinyal keluaran. Ini didasarkan atas istilah linguistik dari pengetahuan pakar yang merupakan dasar dari pengambilan keputusan atau inferensi untuk mendapatkan aksi keluaran sinyal kontrol dari suatu kondisi masukan yaitu *error (e)* dan *delta error (de)* dengan dasar aturan *(rule)* yang ditetapkan.

Dari proses inferensi ini menghasilkan sinyal keluaran yang masih dalam bentuk bilangan *fuzzy*, yaitu dengan derajat keanggotaan dari sinyal kontrol.

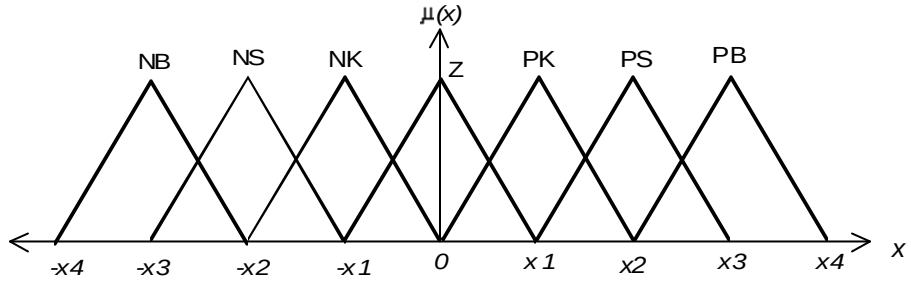
Untuk perancangan kontrol *fuzzy* ada beberapa hal dalam menentukan aturan kontrol yang penting diantaranya:

1). Pemilihan variabel masukan dan keluaran

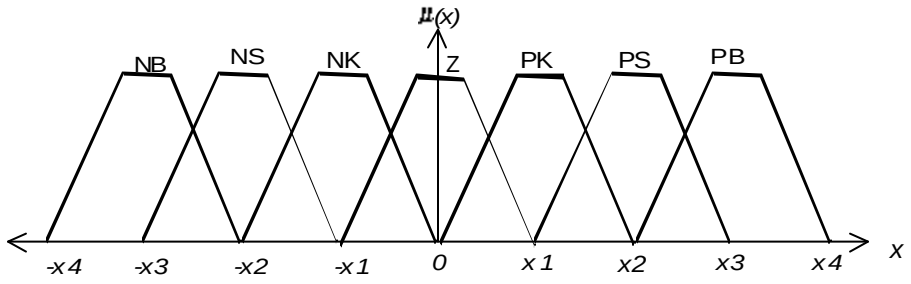
2). Penurunan aturan kontrol *fuzzy*

a. Pemilihan Variabel Masukan dan Keluaran

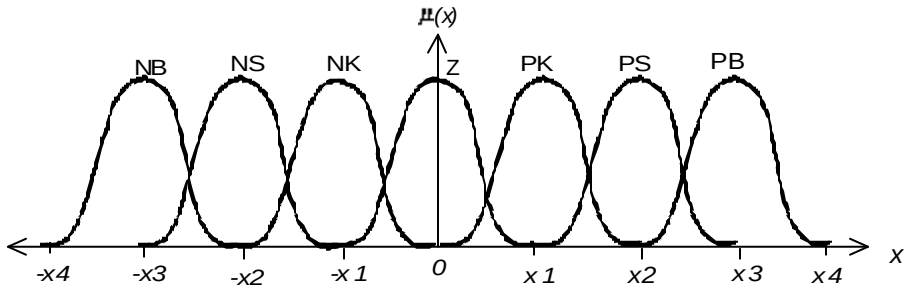
Variabel masukan dan keluaran yang dipilih memberikan pengaruh kuat pada karakteristik kontroler. Pemilihan variabel ini memerlukan pemahaman prilaku atau sifat-sifat dari *plant* dan prilaku pengaturannya. Pada KLF acuan aturan kontrolnya sama dengan kontrol konvensional yaitu *error (e)* dan perubahan atau *delta error (de)* dari plant. Sehingga *error (e)* dan *delta error (de)* digunakan sebagai variabel masukan dan keluarannya berupa sinyal atur.



a). Fungsi segitiga



b). Fungsi Trapesium



c). Fungsi Eksponensial

Gambar 5.34 Fungsi-fungsi Keanggotaan Uniform

b. Penurunan Aturan Kontrol Fuzzy

Untuk menurunkan aturan kontrol dari KLF dengan cara mengumpulkan aturan-aturan kontrol *fuzzy* yang dibentuk dari analisis prilaku obyek atur yang didapat dari koreksi simpangan keluaran *plant* terhadap kondisi yang diinginkan. Hasil yang diperoleh akan berupa prototipe aturan kontrol seperti Gambar 3.9 dan Tabel 3.1.

Ada beberapa model untuk menyempurnakan aturan kontrol yang berupa prototipe yaitu metoda Pemetaan skala *(Scale Mappings)* dengan model Mamdani (1977), model Takagi-Sugeno-Kang (TSK) (1985) dan model Kosko (1996) (yang biasa disebut : *Standard Additive Model :"SAM")* [5]. Prinsipnya adalah untuk menentukan sinyal atur sehingga sistem *loop* tertutup berakhir sesuai yang diinginkan.

5.4.6.5. Logika Pengambilan Keputusan

Sebagai inti dari pengaturan *fuzzy* terletak pada bagian logika pengambilan keputusan yang didasarkan pemikiran dan keputusan yang dilakukan manusia.

Ada 2(dua) tipe pengambilan keputusan logika *fuzzy (fuzzy logic reasoning)* yaitu *generalized modus ponens(GMP)* dan *generalized modus tollens(GMT)* [2].

GMP merupakan keputusan langsung *(direct reasoning)* dan *GMT* adalah keputusan tidak langsung *(indirect reasoning)*.

Dalam logika pengambilan keputusan yang penting adalah fungsi implikasi *fuzzy*, penafsiran kata hubung, operator komposional dan mekanisme inferensi.

a. Aturan Kontrol Fuzzy

Pada dasarnya seorang operator mesin dalam bekerja telah menggunakan aturan kotrol yang berupa hubungan antara *input* dan *output* menggunakan aturan *JIKA-MAKA (IF-THEN)* dalam mengambil keputusan. Aturan kontrol pada logika *fuzzy* menggunakan variabel linguistik dari himpunan *fuzzy* yang didasarkan atas pengetahuan dan pengalaman dari obyek yang diatur. Sehingga prilaku sistem pengaturan *fuzzy* didasari oleh pernyataan linguistik berbasis pengetahuan yang meniru kerja operator.

Pernyataan *JIKA-MAKA (IF-THEN)* digunakan untuk menentukan relasi *fuzzy* antara *input* dan *output* kontrol yang biasanya disebut implikasi *fuzzy* [2]. Dalam penerapannya misalkan *IF (input $_i$) THEN* (*output $_j$),berati misalkan IF i = 1,2, …,n , THEN j=1,2,…,n.* Dimana *i* adalah *antecedent* yaitu *input* yang telah difuzzifikzsi, sedangkan *j* adalah *cosequent* yaitu sebagai aksi kontrol *(output).*

Sebagai contoh misalkan himpunan *fuzzy* dinotasikan dengan *A, A',B, B'* dan variabel linguistik dinotasikan dengan *x, y,* maka *GMP* dan *GMT* dinyatakan [2, 5]:

Gneralized Modus Ponens (GMP):

Pernyataan 1 (aturan) : *IF x is A THEN y is B*

Pernyataan 2 (fakta) : *y is A'*

Penyelesaian : *y is B'*

(3.29)

Penyelesaian relasi *fuzzy* juga diekpresikan secara implikasi *fuzzy* seperti berikut:

$$R = A \rightarrow B \quad \textit{(relasi fuzzy)} \tag{3.30}$$

$$B' = A' \circ R = A' \circ (A \rightarrow B)$$

Dengan R adalah relasi *fuzzy* dari implikasi *fuzzy* " *if A then B'* ", tanda ° adalah *operator komposisi,* dan *A'* adalah himpunan *fuzzy* yang mempunyai bentuk fungsi keanggotaan : *A, sangat A lebih atau kurang A, tidak A* dan sebagainya.

Gneralized Modus Tollens (GML):

Pernyataan 1 (aturan) : *IFx is A THEN y is B*

Pernyataan 2 (fakta) : *y is B''*

Penyelesaian : *y is A'*

(3.31)

Dengan cara yang sama akan didapatkan:

$$R = A \rightarrow B \quad \textit{(relasi fuzzy)} \tag{3.32}$$

$$A' = B' \circ R = B' \circ (A \rightarrow B)$$

b. Fungsi Implikasi Fuzzy

Penggunaan fungsi implikasi *fuzzy* dengan aturan kontrol (sebab akibat) dapat ditulis secara formulasi sebagai berikut:

$$\mu_{Ri} \overset{\Delta}{=} \mu_{(Ai;\, dan\ Bi;\, \rightarrow Ci)}(x, y, z) = \left[\mu_{Ai}(x)\ dan\ \mu_{Bi}(y)\right] \rightarrow \mu_{C_i}(z) \tag{3.33}$$

Ai dan *Bi* adalah himpunan *fuzzy Ai x Bi* dalam *X x Y*, sedangkan Ri Δ = *(Ai dan Bi)® Ci* adalah implikasi *fuzzy* dalam *X x Y x Z*, dan notasi ® adalah implikasi *fuzzy.*

Fungsi implikasi *fuzzy* yang digunakan adalah salah satunya hasil penelitian Mamdani. Fungsi implikasi ini disebut aturan operasi-mini *(mini-operation rule)* yang dinotasikan dengan *Rc.*

$$Rc = A \times B = \int_{XxY} m_A(x) \wedge m_B(y) / (x,y) \quad (3.34)$$

Karena pada penelitian ini nanti termasuk kategori digunakannya sistem *MISO (Multi Input Single Output),* maka akan diperoleh persamaan :

$$Rc = (A \times B) \times C = \int_{XxYxZ} m_{AxB}(x,y) \wedge m_C(z) / (x,y,z) \quad (3.35)$$

Untuk bentuk *A x B* adalah merupakan produk *Cartesian input* dalam *X x Y*.

c. Penafsiran Kata Hubung

Setiap aturan kontrol *fuzzy* biasanya dinyatakan dengan suatu relasi, karena itu prilaku sistem *fuzzy* ditentukan oleh relasi-relasi *fuzzy* tersebut. Sedangkan kata hubung *DAN (AND)* digunakan pada produk *Cartesian* yang dikarenakan oleh derajat nilai keanggotaan yang berbeda.

Penggunaan operator komposisi sangat menentukan hasil kesimpulan, karena setiap operator komposisi akan diperlukan sesuai dengan kegunaannya. Secara umum ditunjukkan sebagai komposisi *sup-star* dengan *star (*)* sebagai operator dalam aplikasi tertentu. Operator *sup-min* dan *sup-product* adalah operator yang sering digunakan pada teknik pengaturan [2].

Tabel 5.7 Beberapa Aturan Implikasi Fuzzy [2]

Aturan	Formulasi Implikasi	Implikasi *Fuzzy*
R_c: *min*	*a® b = a Ù b*	$= m_A(u) \text{ Ù } m_B(v)$
R_p: *product*	*a® b = a . b*	$= m_A(u) . m_B(v)$
R_{bp}: *bounded*	*a® b = 0 Ú (a+b-1)*	$= 0 \text{ Ú } [m_A(u) + m_B(v)$-

R_{dp}: drastic product	$a \to b = \begin{cases} a, & b = 1 \\ b, & a = 1 \\ 0, & a, b < 1 \end{cases}$	$= \begin{cases} m_A(u), & m_B = 1 \\ m_B(v), & m_A = 1 \end{cases}$
R_a: arithmetic	$a \to b = (1 \wedge (1 - a + b))$	$= 1 \wedge (1 - m_A(u) + m_B(v))$
R_m: max-min	$a \to b = (a \wedge b) \vee (1 - a)$	$= ((m_A(u) \wedge m_B(v)) \vee (1 -$
Rs: standard sequence	$a \to b = \begin{cases} 1, & a \le b \\ 0, & a > b \end{cases}$	$= \begin{cases} 1, & m_A(u) \le m_B(v) \\ 0, & m_A(u) > m_B(v) \end{cases}$
R_b: Boolean	$a \to b = (1 - a) \vee b$	$= (1 - m_A(u)) \vee m_B(v)$
R_g: Gödelian logic	$a \to b = \begin{cases} 1, & a \le b \\ b, & a > b \end{cases}$	$= \begin{cases} 1, & m_A(v) \le m_B(v \\ m_B(v), & m_A(u) > m_B(v) \end{cases}$
R_D: Goguen's fuzzy implication	$a \to b = \begin{cases} 1, & a \le b \\ b/a, & a > b \end{cases}$	$= \begin{cases} 1, & m_A(v) \\ m_B(v)/m_A(u), & m_A(u \end{cases}$

d. Operator Komposisi

Fungsi implikasi sistem *SISO (Single Input Single Output)* diberikan persamaan dalam bentuk komposisi :

$$B' = A' \circ R \quad (3.36)$$

Pada penelitian ini digunakan sistem *MISO* (Multi Input Single Output), maka persamaan menjadi:

$$C' = (A', B') \circ R \quad (3.37)$$

Berbagai operator komposisi juga dirumuskan oleh peneliti sebelumnya untuk digunakan pada fungsi implikasi *fuzzy* diantaranya Operasi :

1. Sup-min (Max-min operation) [Zadeh, 1973]

2 Sup-product (Max product operation) [Kaufmann,1975]

3. Sup-bounded-product (Max. bounded product (max-⊙) operation) [Mizumoto, 1981]

4.Sup-drastic-product (Max drastic product (max-Ù) operation) [Mizumoto,1981].

e. Mekanisme Penalaran/Inferensi

Sebagaimana diketahui bahwa sistem pengaturan *fuzzy* terdiri dari seperangkat aturan kontrol dan untuk mengkombinasikan aturan digunakan kata hubung *JUGA.*

Pada penelitian ini aturan kontrol *fuzzy* yang digunakan termasuk secara *MISO (Multi Input Single Output),* dengan *x* dan *y* adalah *input* dan *z* adalah *output* dalam semesta pembicaraan *X, Y* dan *Z*, maka mekanisme inferensi dinyatakan sebagai :

Input	: *x is A' AND y is B'*		
R_1	: *IF x is A_1 AND y is B_1, THEN z is C_1*		
R_2	: *IF x is A_2 AND y is B_2, THEN z is C_2*		
…..	….	……	……
…..	….	…..	……
R_n	*: IF x is A_n AND y is B_n, THEN z is C_n*		
Penyelesaian	: *z is C'*		

(3.38)

Dari keseluruhan aturan kontrol tersebut harus dapat menghasilkan kesimpulan yang dinyatakan dengan yang diperoleh dari mekanisme inferensi dengan komposisional *sup-star* yang didasarkan dari fungsi implikasi *fuzzy* dan kata hubung *"DAN"(AND)* dan *"JUGA"(ALSO).* Ada beberapa persamaan matematik penting yang berhubungan dengan mekanisme inferensi *fuzzy* diantaranya :

$$(A', B') \, o \bigcup_{i=1}^{n} R_i = \bigcup_{i=1}^{n} (A', B') \, o \, \mathrm{R}$$

(3.39)

dengan R_i adalah relasi aturan kontrol ke-*i*.

Persamaan (3.40) *output* dari semua perangkat aturan kontrol merupakan gabungan semua *output* aturan kontrol. Sedangkan untuk fungsi implikasi operasi-mini Mamdani :

$$(A', B') \, o \, (A_i \text{ dan } B_i \rightarrow C_i = [A' o (A_i \rightarrow C_i)] \cap [B' o (B_i \rightarrow C_i)]$$

(3.40)

jika $m_{AixBi} = m_{Ai} \wedge m_{Bi}$.

$$(A', B') \, o \, (A_i \text{ dan } B_i \rightarrow C_i = [A' o (A_i \rightarrow C_i)] \, . \, [B' o (B_i \rightarrow C_i)]$$

(3.41)

jika $m_{AixBi} = m_{Ai} \, . \, m_{Bi}$.

Bilamana *input fuzzy* berupa *input* tunggal $A'=x_0$ dan $B'=y_0$ akan diperoleh persamaan :

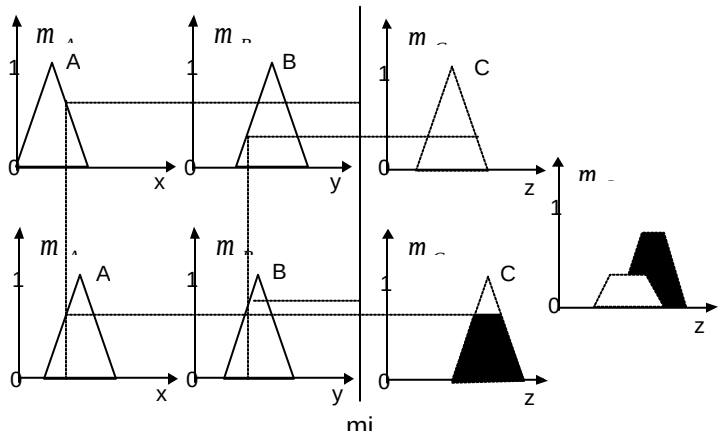
$$R_c : \hat{a}_i \wedge m_{c_i}(z), \; \hat{a}_i = m_{Ai}(x_0) \wedge m_{Bi}(x_0),$$

(3.42)

$$R_c : \overset{*}{a}_i \wedge m_{c_i}(z), \; \overset{*}{a}_i = m_{Ai}(x_0) \wedge m_{Bi}(x_0),$$

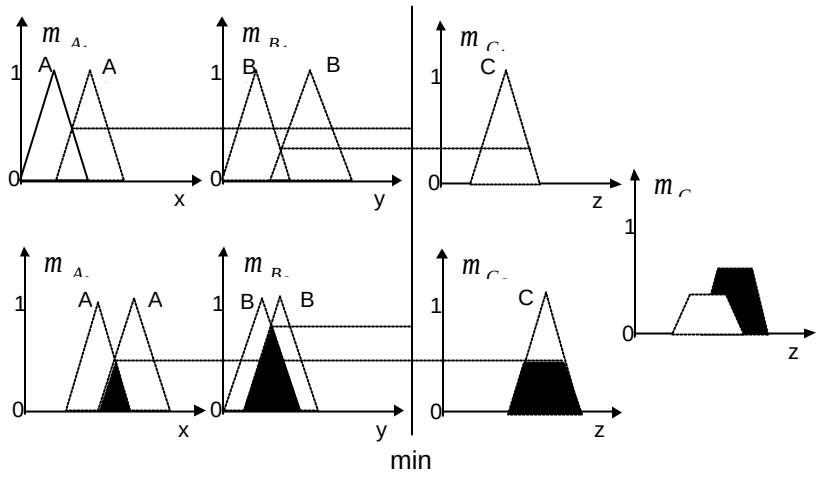
(3.43)

Dari persamaan (3.40), (3.41), (3.42), dan (3.43) menguraikan fungsi implikasi operasi



mini Mamdani *(R_c)* dengan menggunakan kedua jenis produk Cartesian.

Gambar 5.35 Penafsiran Grafis Persamaan (3.40; 3.41) dengan *a^* dan *Rc* [2, 5, 7]



Gambar 5.36 Penafsiran Grafis Persamaan (3.42) dan (3.43) [2, 5, 7]

Sehingga persamaan itu disederhanakan menjadi :

$$R_C' : m_C' = \bigcup_{i=1}^{n} a_i \wedge m_{C_i} \quad (3.44)$$

α_1 merupakan besarnya kontribusi aturan ke-*i* untuk aksi kontrol *fuzzy*, dan faktor pembobot tersebut ditentukan oleh dua pilihan yaitu dengan operasi minimum *(α_1^)* dan produk aljabar *(α_1*)* dalam produk Cartesian.Bila dilihat kembali persamaan (3.40) dan (3.41) mensyaratkan bahwa *iput* aturan kontrol berupa himpunan *fuzzy*, yaitu diperoleh aturan kontrol *fuzzy* secara grafis dengan asumsi

R_1 : IF x is *A_1 AND y* is *B_1 THEN z* is *C_1*

R_2 : IF x is *A_2 AND y* is *B_2 THEN z* is *C_2*

Bila asumsi di atas diberikan *input* himpunan *fuzzy A'* dan *B'*, maka penafsiran grafis dari kedua aturan kontrol *fuzzy* itu dengan mekanisme inferensi operasi minimum adalah seperti Gambar 3.11 untuk persamaan (3.40), (3.41) dan Gambar 3.12 untuk persamaan (3.42), (3.43).

Penafsiran grafis Gambar 3.11 dan 3.12 mempunyai pengertian bahwa input kontrol logika *fuzzy* (KLF) diperlakukan sebagai *fuzzy* tunggal, sehingga permasalahan dalam mekanisme inferensi menjadi sederhana.

Dari aturan kontrol *R_1* dan *R_2* dengan menerapkan fungsi implikasi operasi minimum Mamdani (*Rc*) persamaan (3.42), (3.43) produk Cartesian operasi minimum diperoleh faktor pembobot :

$$a_1 = m_{A1}(x_0) \wedge m_{Bi}(y_0)$$
(3.45)

$$a_2 = m_{A2}(x_0) \wedge m_{B2}(y_0)$$
(3.46)

Sehingga fungsi keanggotaan sinyal atur yang dihasilkan mempunyai persamaan :

$$m_C'(z) = \bigcup_{i=1}^{n} a_i \wedge m_{C_i}(z)$$
(3.47)

C_i merupakan himpunan *fuzzy* sinyal atur.

5.4.6.6. Defuzzifikasi

Defuzzifikasi pada dasarnya merupakan pemetaan ruang aksi kontrol *fuzzy* menjadi ruang aksi kontrol *non-fuzzy* (*crispy*). Prinsip dari strategi *defuzzifikasi* bertujuan untuk menghasilkan sinyal atur yang nyata yang dapat merepresentasikan distribusi dari aksi atur masing-masing aturan kontrol.

Bentuk persamaan umum proses *defuzzifikasi* seperti berikut :

z_0 = *defuzzier (z)*
(3.48)

Beberapa metode defuzzifikasi yang dapat digunakan secara umum yaitu metode titik *pusat (the Center of Area ="COA")*, metode titik tengah maksimum *(the Mean of Maximum ="MOM")*, dan metode kriteria *max (the Criterion Max)* [2, 5, 7, 12, 22].

a. Metode Titik Pusat (COA)

Metode titik pusat *(the Center of Area ="COA")* adalah metode *defuzzifikasi* yang sering digunakan yaitu dengan menentukan *output* aksi kontrol dari pusat berat *(the Center of Gravity ="COG").*

Formulasi umum dari metode ini dalam menentukan nilai *output* aksi kontrol *(z_o)* kasus diskrit sebagai berikut [2, 5,7, 12, 22]:

$$z_{o(COA)} = \frac{\sum_{j=1}^{n} m_C(z_j) z_j}{\sum_{j=1}^{n} m_C(z_j)}$$
(3.49)

n adalah jumlah *level* kuantisasi dari *output*, z_j adalah besarnya *output* pada *level* kuantisasi ke-*j*, dan $m_c(z_j)$ nilai fungsi keanggotaan dari *output* himpunan fuzzy.

Dan jika kasus semesta pembicaraan *(universe of discourse)* adalah kontinyu, maka formulasi kontinyu dapat dinyatakan [2, 5]:

$$z_{o(COA)} = \frac{\int_z m_c(z) z \, dz}{\int_z m_c(z) \, dz} \quad (3.50)$$

b. Metode Titik Tengah Maksimum (MOM)

Metode titik tengah maksimum *(the Mean of Maximum="MOM")* adalah metode defuzzifikasi untuk menghitung harga titik tengah *output* dari semua aksi kontrol yang mempunyai fungsi keanggotan *fuzzy* maksimum [2, 5, 7, 12].

Formulasi umum kasus diskrit, nilai *output* aksi kontrol (z_o) dari metode ini dapat diekpresikan sebagai berikut [2, 5, 7, 12, 22]:

$$z_{o(MOM)} = \sum_{j=1}^{m} \frac{z_j}{m} \quad (3.51)$$

z_j adalah nilai pendukung *output* dengan fungsi keanggotaan bernilai maksimum ke-*j* atau $m_c(z_j)$ dan *m* adalah banyaknya nilai pendukung.

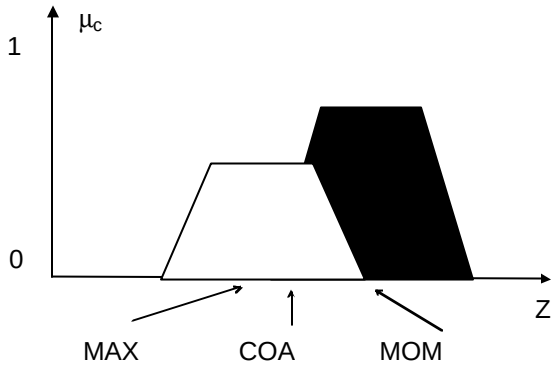
Dari kedua metode strategi defuzzifikasi ini saling mempunyai keunggulan. Untuk metode *COA* berdasarkan penelitian Braae and Rutherford (1978) mempunyai hasil yang sangat bagus terutama performansi keadaan tunak *(steady state)* lebih bagus karena rata-rata kesalahan kuadratnya kecil *(lower mean square error)*. Selanjutnya penelitian Lee, (1990) metode defuzzifikasi *MOM* performansi tanggapan peralihan *(transient respons)* hasilnya lebih baik [2].

c. Metode Kriteria Max (MAX)

Metode defuzzisifikasi *MAX* menghasilkan titik dimana distribusi yang mungkin pada aksi kontrol bernilai maksimum. Metode ini biasanya jarang digunakan karena ketelitiannya tidak begitu baik.

Dengan berdasarkan pada keunggulan masing-masing metode defuzzifikasi ini, maka dengan *MOM* performansi KLF cocok untuk

sistem *relay* multilevel *(multilevel relay system),* sedangkan strategi *COA* cocok untuk kontroler konvensional PI dan sebagainya.



Gambar 5.37 Interpretasi Grafik Strategi Defuzzifikasi [2, 5, 7]

5.5. Kontroler PID *(Proportional + Integral + Derivative Controller)*

Kontroler PID *(Proportional + Integral + Derivative Controller)* merupakan kontroler *feed-forward* yang berfungsi mengolah sinyal *error* menjadi sinyal kontrol. Hubungan sinyal kontrol terhadap sinyal *error* bisa proporsional, integral, diferensial atau gabungan diantaranya [22].

Sinyal *output U(t)* kontroler tipe-PID diberikan persamaan dalam model matematik domain waktu adalah :

$$U(t) = Kp\,[e(t) + \frac{1}{t_i}\int e(t)dt + t_D \frac{d}{dt} e(t)]$$

(3.52)

Dengan menggunakan *Laplace* didapatkan persamaan sinyal kontrol *U(s)*:

$$U(s) = Kp\,[E(s) + \frac{1}{t_i s} E(s) + t_D s E(s)]$$

(3.53)

Untuk *t* kontinyu *(continuous)* diberikan persamaan sinyal kontrol *U(t)* :

$$U(t) = Kp\ e(t) + K_i \int e(t)dt + K_D \frac{d}{dt} e(t)]$$

(3.54)

Untuk *t* diskrit *(discrete)* diberikan persamaan sinyal kontrol *U(k)* :

$$U(k) = Kp\ e(k) + K_i T_s \sum_{i=1}^{n} e(i) + \frac{K_D}{T_s} de(k)$$

(3.55)

Kp : merupakan *gain* proporsional, $(\frac{Kp}{t_i}) = K_I$ *: gain* kontrol integral, $Kp\,t_D = KD$: *gain* kontrol diferensial; t_i , t_D : konstanta waktu masing-masing integral , diferensial, *Ts*: periode *sampling*, *n* : jumlah *sample*.

Besarnya perubahan sinyal *error* (*delta error*) diberikan pendekatan *de(t):*

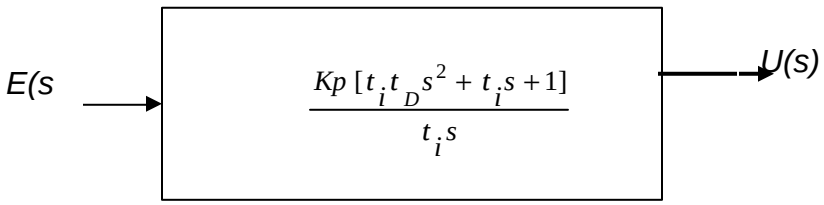
$$de(t) \overset{\Delta}{=} e(k) - e(k-1)$$

(3.56)

Dari persamaan (3.53) dapat dibuat persamaan fungsi alih dari kontroler PID :

$$\frac{U(s)}{E(s)} = Kp[1 + \frac{1}{t_i s} + t_D s] = \frac{Kp[t_i t_D s^2 + t_i s + 1]}{t_i s}$$

(3.58)



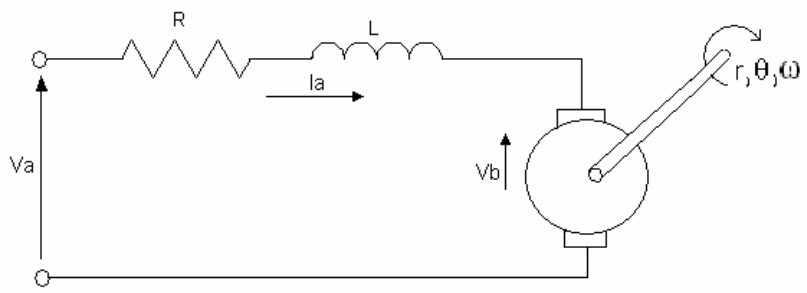
Gambar 5.38 Diagram Blok Fungsi Alih Kontroler PID standar [8, 22]

5.6 Aktuator

Aktuator berfungsi mengatur aliran energi kepada system yang dikontrol. Alat ini disebut sebagai elemen pengontrol akhir (final control element). Yang termasuk actuator misalnya motor listrik, pompa, pnematik, silinder hidraulik. Elemen kluaran ini harus mempunyai kemampuan untuk menggerekan beban ke suatu nilai yang diinginkan.

5.6.1 Motor DC Magnet Permanen

Motor direct current (DC) adalah peralatan elektromekanik dasar yang berfungsi untuk mengubah tenaga listrik menjadi tenaga mekanik yang dirancang dan diperkenalkan oleh
Michael Faraday. Rangkaian ekivalen dari sebuah motor DC magnet permanent dapat ditunjukan seperti dalam Gambar 5.1 berikut ini .



Gambar 5.39 : Rangkaian ekivalen motor DC magnet permanent

Notasi: V_a= Tegangan armature, Ia=Arus motor, R=resistensi armatur, L=Induktansi lilitan armatur,V_b=Tegangan induksi balik, r=Torsi motor, ?=Kecepatan putar motor, ?=Sudut putaran poros motor.

Persamaan tegangan Va adalah,

$$V_a = L\frac{dI_a}{dt} + RI_a + K_b w \quad (5.1)$$

Dengan K_b adalah konstanta yang diukur dari tegangan yang dihasilkan oleh motor ketika berputar setiap satuan keceptan (Volt.det/rad). Magnitud dan polaritas K_b adalah fungsi dari kecepatan angular, ? dan arah putar poros motor. Persamaan (5.1) dikenal sebagai persamaan DC motor secara umum. Dalam domain waktu ditulis :

V_a (t)= $L\frac{dI_a(t)}{dt} + RI_a(t) + V_b(t)$, dengan $V_b = K_b\dot{q}(t)$

(5.2)

Sesuai dengan hokum Kirchoff, V= I.R atau I= V/R, dan dengan menggunakan transformasi Laplace, persamaan arus motor dalam dapat ditulis,

$$I_a = \frac{V_a(s) - sK_b q(s)}{R + sL}$$

(5.3)

dengan mensubstitusikan persamaan (5.3) ke dalam persamaan umum torsi output motor, $r(t)=K_m I_a(t)$, dengan K_m adalah konstanta proporsional torsi motor, dalam transformasi Laplace didapat,

$$r(s) = K_m I_a(s) = K_m\left[\frac{V_a(s) - sK_b q(s)}{R + sL}\right]$$

(5.4)

dengan memperhatikan pesamaan tosi output motor ditinjau dari pembebanan,

$$r(t) = J_{eff}\ddot{q}(t) + f_{eff}\dot{q}(t),$$

(5.5)

$J_{eff}=J_m + J_L$, dimana J_m adalah momen inersia poros (rotor) motor, J_L adalah momen inersia beban pada poros motor, dan $f_{eff}=f_m + f_L$ adalah koefisien friksi viscous pada poros motor, f_L adalah koefisien friksi viscous pada beban di poros motor, maka transfer function tegangan armatur V_a terhadap pergerakan sudut motor ? dapat ditulis ,

$$\frac{q(s)}{V_a(s)} = \frac{K_m}{s\left[s^2 J_{eff}L + (Lf_{eff} + RJ_{eff} + K_m K_b)\right]}$$

(5.6)

Persamaan (5.6) dapat ditulis dengan singkat,

$$\frac{q(s)}{V_a(s)} = \frac{K_m}{s\left[sRJ_{eff} + Rf_{eff} + K_m K_b\right]} = \frac{K}{s(T_m s + 1)}$$

(5.7)

dengan $K = \frac{K_m}{Rf_{eff} + K_m K_b}$, konstanta penguatan motor (gain), dan

$T_m = \frac{RJ_{eff}}{Rf_{eff} + K_m}$, konstanta waktu motor.

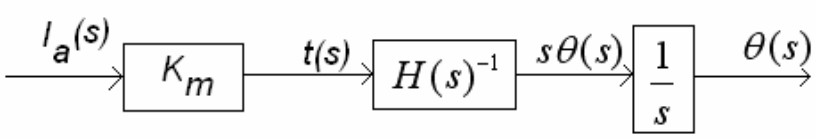
Jika motor menggunakan gearbox dengan rasio, $n = \frac{N_2}{N_1}$ maka persamaan (5.7) dapat ditulis,

$$\frac{q_L(s)}{V_a(s)} = \frac{nK_m}{s\left[sRJ_{eff} + Rf_{eff} + K_m K_b\right]} \quad (5.8)$$

dengan $?_L$ adalah sudut poros output gearbox.

Dalam aplikasi jarang dijumpai atau diperoleh data-data spesifikasi/ parameter motor secara lengkap. Produsen motor biasanya hanya memberikan informasi dalam bentuk grafik antara torsi dengan arus motor, torsi dengan tegangan, torsi dengan kecepatan (rpm). Sehingga hampir tidak mungkin melakukan pemodelan motor secara ideal dalam disain sistem kontrol otomasi industri. Akibatnya pada kebanyakan proses disain banyak dilakukan asumsi-asumsi.

Jika motor DC dianggap linier, yaitu torsi berbanding lurus dengan arus (motor ideal), maka model matematik dapat disederhanakan dengan memperhatikan konstanta proporsional motor (K_m) saja. Dengan asumsi bahwa motor DC adalah dari jenis torsi motor dan inputnya dipertimbangkan sebagai arus maka transfer function open loop dapat digambarkan seperti Gambar 5.40 berikut.



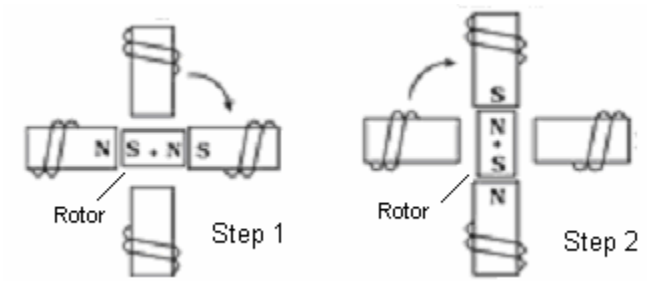
Gambar 5.40: Transfer function open loop Torsi Motor DC

5.6.2 Motor DC Stepper

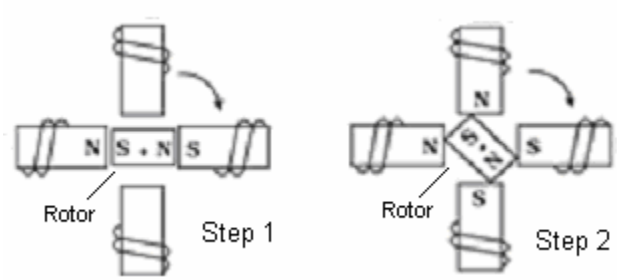
Prinsip kerja motor DC stepper sama dengan motor DC magnet permanent, yaitu pembangkitan medan magnet untuk memperoleh gaya tarik ataupun gaya lawan dengan menggunakan catu tegangan DC pada lilitan / kumparannya. Motor DC magnet permanent menggunakan gaya lawan untuk menolak atau mendorong fisik kutub magnet yang dihasilkan, sedangkan pada motor DC stepper menggunakan gaya tarik untuk menarik fisik kutub magnet yang berlawanan sedekat mungkin ke posisi kutub magnet yang dihasilkan oleh kumparan. Gerakan motor DC stepper terkendali, karena begitu kutub yang berlawanan tadi sudah tarik menarik dalam posisi yang paling dekat, gerakan akan terhenti dan di rem.

Lihat Gambar 5.41 dan 5.42 , jika kumparan mendapat tegangan dengan analogi mendapat logika “1”, maka akan dibangkitkan kutub magnet yang berlawanandengan kutub magnet tetap pada rotor. Sehingga posisi kutub magnet rotor akan ditarik mendekati lilitan yang menghasilkan kutub magnet berlawanan tadi. Jika langkah berikutnya, lilitan yang bersebelahan diberi tegangan, sedang catu tegangan pada lilitan sebelumnya dilepas, maka kutub magnet tetap pada rotor itu akan berpindah posisi menuju kutub magnet lilitan yang dihasilkan. Berarti telah terjadi gerakan 1 step. Jika langkah ini diulang terus- menerus, dengan memberikan tegangan secara bergantian ke lilitan-lilitan yang bersebelahan, maka rotor akan berputar.

Logika perputaran rotor tersebut dapat dianalogikan secara langsung dengan data 0 atau 1 yang diberikan secara serentak terhadap semua lilitan stator motor. Hal ini sangat memudahkan bagi system designer dalam hal merancang putaran-putaran motor DC stepper secara bebas dengan mengatur bit-bit pada data yang dikirimkan ke rangkaian interface motor DC stepper tersebut.



Gambar 5.41:Prinsip kerja motor DC stepper untuk gerakan full step



Gambar 5.42: Prinsip kerja motor DC stepper untuk gerakan half step

Untuk motor DC stepper 4 fasa pada prinsipnya ada dua macam cara kerja , yaitu full step dan half step. Lihat table 5.1. Penjabaran formasi logika dalam table ini adalah untuk mewakili putaran penuh 360^o relatif terhadap fasa dari motor.

Motor DC stepper yang ada di pasaran sebagian besar melipatgandakan jumlah kutub magnet kumparannya dengan memperbanyak kumparan stator sejenis melingkar berurutan dalam konfigurasi penuh 360^o riil terhadap poros rotor (dengan jumlah fasa tetap). Kondisi ini dilakukan untuk memperoleh efek riil putaran satu step yang lebih presisi, misalnya $3,6^o$ / step atau $1,8^o$ / step.

Untuk memperoleh efek cengkeraman yang lebih kuat, modus data yang diberikan pada mode full wave dapat dimanipulasi dengan memberikan double active bits pada setiap formasi (lihat table 5.2). Dengan cara ini torsi yang dihasilkan akan lebih besar. Namun demikian, penggunaan arus akan berlipat dua karena dalam satu saat yang bersamaan dua lilitanmendapatkan arus kemudi. Dalam aplikasinya, sumber daya yang tersedia perlu diperhatikan.

Tabel 5.8: Formasi tegangan / logika pada motor DC step

Step ke	Full Step				Half Step			
1	1	0	0	0	1	0	0	0
2	0	1	0	0	1	1	0	0
3	0	0	1	0	0	1	0	0
4	0	0	0	1	0	0	1	0
5	Berulang ke step 1				0	0	1	0
6					0	0	1	1
7					0	0	0	1
8					1		0	1
					Berulang ke step 1			

Tabel 5.9: Formasi double active bit untuk mode putaran full step

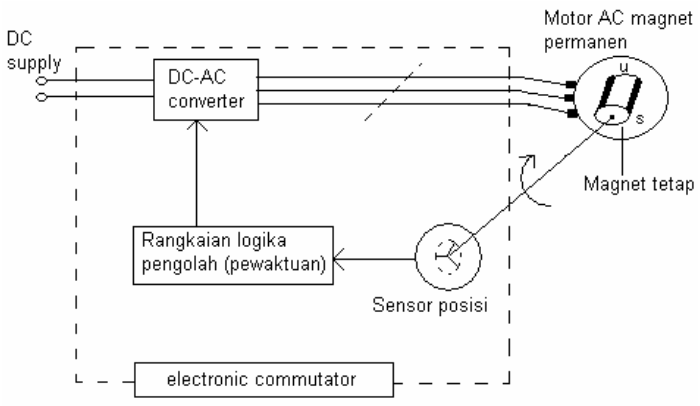
Step ke	Full Step (double active bits)			
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Pada full step, suatu titik pada sebuah kutub magnet di rotor akan kembali mendapat tarikan medan magnet stator pada lilitan yang sama setelah step ke 4. berikutnya dapat diberikan lagi mulai dari step ke 1. Untuk half step, setiap kutub magnet pada rotor akan kembali mendapatkan tarikan dari medan magnet lilitan yang sama setelah step ke 8. Berikutnya kembali mulai dari
step 1.

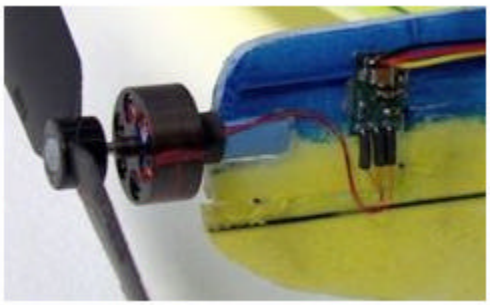
Dengan melihat bahwa pergerakan motor DC stepper adalah berdasarkan perubahan logika pada input lilitan-lilitannya maka menjadi mudah bagi programmer untuk mengubah-ubah arah gerakan dan kedudukan rotor pada posisi yang akurat. Hal ini salah satu keuntungan dari penggunaan motor DC stepper. Agar dapat membuat gerakan yang lebih presisi, biasanya jumlah batang magnet di rotor diperbanyak dan lilitan dibuat berpasang-pasangan sesuai dengan posisi kutub magnet rotor. Cara lain adalah dengan menggunakan system gear pada poros rotor tanpa mengubah karakteristik motor DC steppernya.

5.6.3 Motor DC brushless

Motor Dc brushless menggunakan pembangkitan medan magnet stator untuk mengontrol geraknya, sedang medan magnet tetap berada di rotor. Prinsip kerja motor Dc brushless mirip seperti motor AC asinkron. Putaran diperoleh dari perbedaan kutub medan magnet yang dihasilkan oleh fasa tegangan yang berbeda. Gambar 5.8 memperlihatkan diagram skema dan prinsip kerja motor DC brushless.



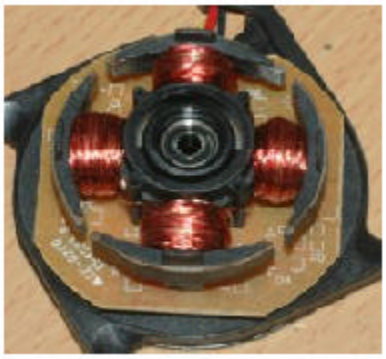
Gambar 5.43: Diagram skema Motor DC brushless



Gambar 5.44:Motor DC brushless menggerakan baling-baling pesawat

Sebuah motor DC brushless menggerakkan pesawat terbang dengan sebuah mikro remote control, diperlihatkan pada Gambar 5.7, motor DC brushless dihubungkan dengan sebuah mikroprosessor. Rotor berisikan magnet yang melingkar mengelilingi lilitan pada stator.

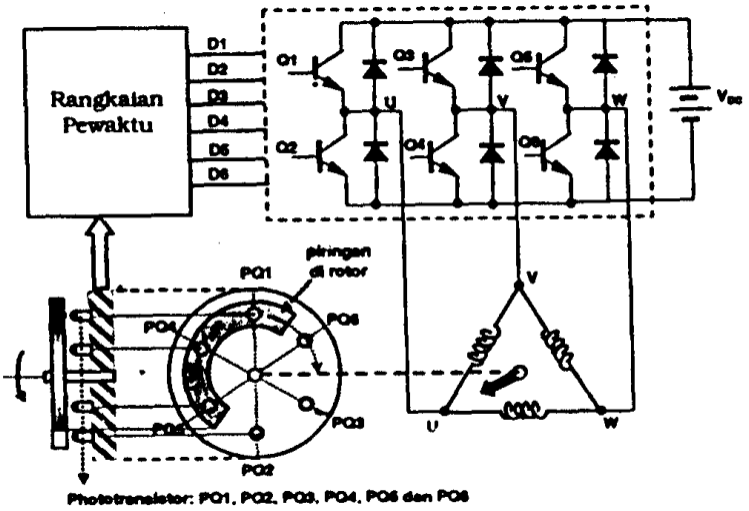
Pada motor DC brushless, electromagnet tidak bergerak, melainkan, magnet permanent yang berputar dan armature tetap diam. Kondisi ini menimbulkan problem, bagaimana mentransfer arus ke armatur yang bergerak. Untuk melaksanakan fungsi ini, system brush / commutator dilengkapi oleh sebuah kontrol elektronik cerdas.



Gambar 5.45 Kutub pada stator motor DC brushless 2 fasa

Gambar 5.45 memperlihatkan kutub-kutub pada stator dari motor DC brushless 2 fasa, rotornya telah dilepas. Konstruksi macam ini banyak dijumpai sebagai fan / kipas pendingin personal komputer.

Sebuah contoh motor DC brushless yang menggunakan rangkaian switching transistor untuk secara berurutan mengaktifkan pembangkitan medan magnet dililitan, diperlihatkan pada Gambar 5.7 di bawah ini.



Gambar 5.46:Rangkaian switching dalam sebuah motor DC brushless

Keterangan:

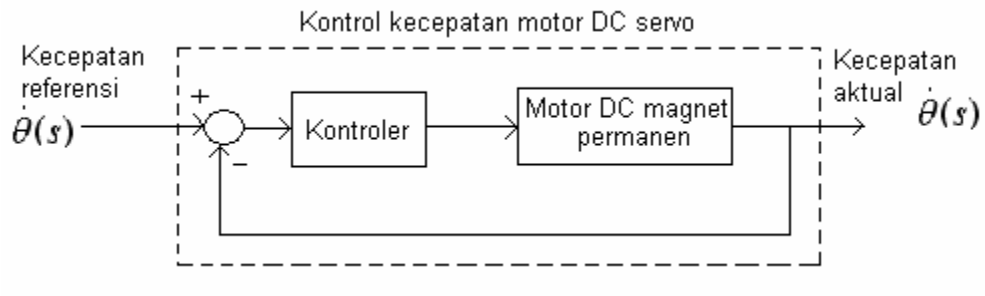
- Rangkaian pewaktu: terdiri dari rangkaian logika sekuensial yang berfungsi memberikan sinyal aktif secara berurutan dengan

konfigurasi tertentu kepada input rangkaian switching (basis transistor).

- Rangkaian switching:terdiri dari 6 buah rangkaian transistor bipolar atau komponen solid-state switching yang lain
- Konstruksi motor: terdiri dari 3 buah lilitan di stator dalam konfigurasi U, V dan W (membentuk sudut 120^o satu sama lain.
- Sensor dan piringan pengaktif: jika rotor berputar, piringan akan menutupi cahaya yang menuju ke photo transistor tertentu. Prinsip ini digunakan untuk memberikan umpan balik ke rangkaian pewaktu agar mengaktifkan transistor-transistor tertentu dalam urutan dan arah putar yang dikehendaki.

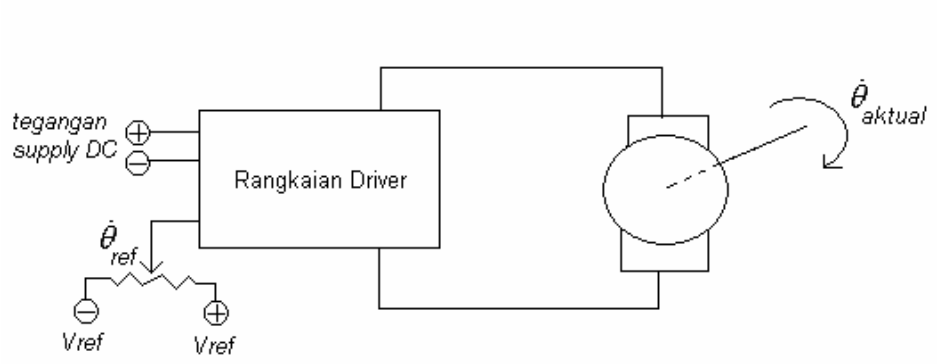
5.6.4 Motor DC Servo

Motor DC servo pada dasarnya adalah motor DC magnet permanen dengan kualifikasi khusus yang sesuai dengan aplikasi servoing di dalam teknik kontrol. Secara umum dapat didefenisikan bahwa motor DC servo harus memiliki kemampuan yang baik dalam mengatasi perubahan yang sangat cepat dalam hal posisi, kecepatan dan akselerasi. Beberapa tipe motor DC servo yang dijual bersama dengan paket rangkaian drivernya telah memiliki rangkaian control kecepatan yang menyatu di dalamnya. Putaran motor tidak lagi berdasarkan tegangan supply ke motor, namun berdasarkan tegangan input khusus yang berfungsi sebagai referensi kecepatan output. Dalam blok diagram dapat digambarkan sebagai berikut.



Gambar 5.47 Blok diagram kontrol kecepatan motor DC servo

Gambar 5.47 jika digambar dalam rangkaian, dapat dinyatakan sebagai berikut,

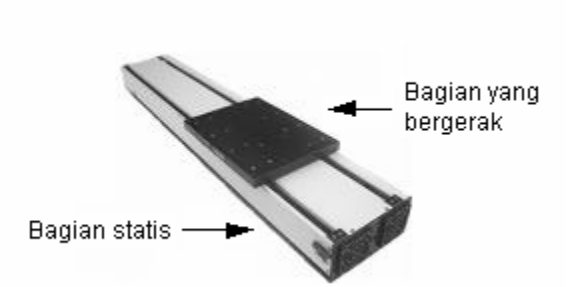


Gambar 5.48 Kontrol kecepatan motor DC servo

Dalam Gambar 5.9 nampak bahwa kecepatan putar motor tidak diatur dari tegangan supply DC, namun melalui tegangan referensi yang diartikan sebagai . Dalam beberapa tipe produk, nilai tegangan sebagai ini mempunyai karakteristik yang linier terhadap .

5.6.5 Motor linier

Motor linier adalah adalah motor DC yang rotornya bergerak secara translasi. Dengan demikian tidak ada bagian yang berputar pada motor linier ini. Motor linier dirancang khusus untuk keperluan permesinan atau manufacturing yang memiliki kepresisian sangat tinggi. Misalnya mesin CNC (Computer Numerical Control), EDM (Electric Discharge Machine), dan sebagainya. Dengan menggunakan motor linier tidak diperlukan lagi sistem gear dan perangkat transmisi daya lainnya.



Gambar 5.49 Motor linear buatan inteldrive

5.6.6 Pnematik

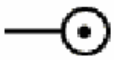
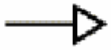
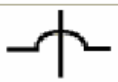
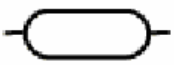
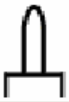
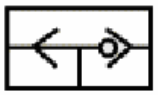
Istilah “pnema” berasal dari istilah yunani kuno, yang berarti nafas atau tiupan. Pnematik adalah ilmu yang mempelajari gerakan atau perpindahan udara dan gejala atau penomena udara.
Ciri-ciri perangkat system pnematik:

- Sistem pengempaan, udara dihisap dari atmosphere dan kemudian dikompresi.
- Udara hasil kempaan, suhunya harus didinginkan
- Ekspansi udara diperbolehkan, dan melakukan kerja ketika diperlukan.
- Udara hasil ekspansi kemudian dibuang lagi ke atmosphere.

5.6.6.1 Simbol-simbol

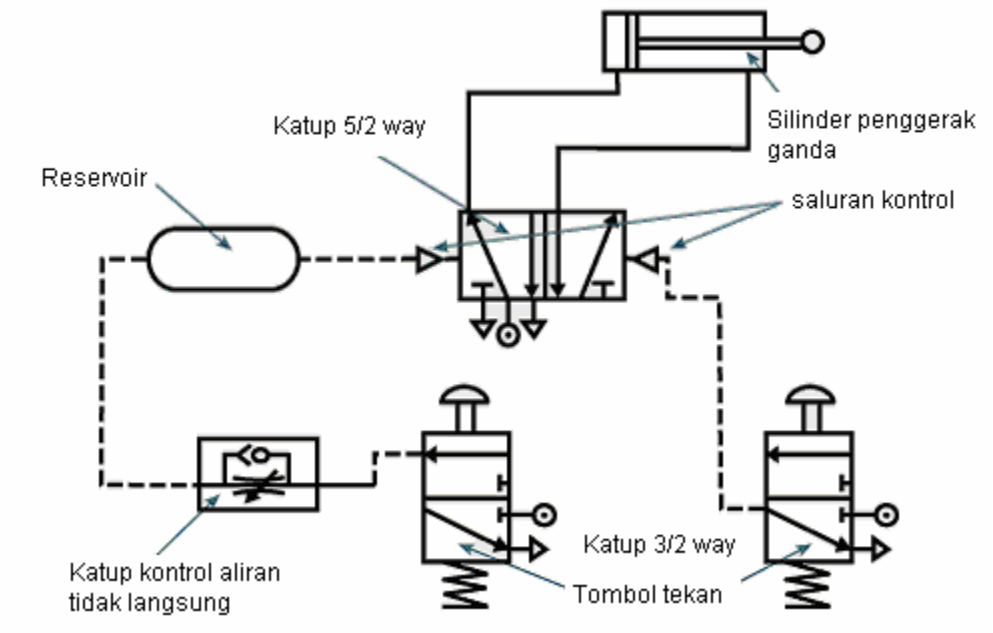
Biasanya pada suatu komponen pnematik selalu tertera symbol daripada komponen tersebut. Pada setiap alat pnematik selalu terdapat simbol disebelah kanan bawah daripada gambar bagian.

Tabel 5.10 Simbol-simbol pnematik

	Sumber tekanan		Tuas rol
	Pembuangan udara		Sensitive tekanan
	Saluran kontrol (saluran pemandu)		Operasi solenoid
	Saluran hubungan		Pegas
	Persimpangan saluran		Silinder penggerak tunggal
	Saluran kebocoran		Silinder penggerak ganda, dgn batang torak tunggal
	Katup 3/2 way posisi normal menutup		Katup 5/2 way
	Tombol tekan		Reservoir
	Plunyer		Regulator aliran/ katup kontrol aliran tdk langsung
	Tuas		Katup bola

5.6.6.2 Rangkaian Waktu Tunda

Waktu tunda dalam rangkaian pnematik yaitu sela waktu antara operasi katup dan gerakan piston. Waktu tunda dapat dilakukan dengan cara menghubungkan seri katup kontrol aliran tidak langsung dengan sebuah reservoir, seperti yang diperlihatkan pada Gambar 5.50.



Gambar 5.50 Rangkaian Waktu Tunda

Rangkaian waktu tunda akan bekerja :

- Jika tombol tekan di katup 3/2 way ditekan, aliran udara dibatasi oleh katup kontrol aliran tidak langsung dan perlahan-lahan masuk ke dalam reservoir.
- Tekanan terbentuk di dalam reservoir secara perlahan, menyebabkan waktu tunda.
- Ketika tekanan cukup tinggi, katup 5/2 way bekerja terjadi pelepasan udara dan menyebabkan piston memukul ke luar.

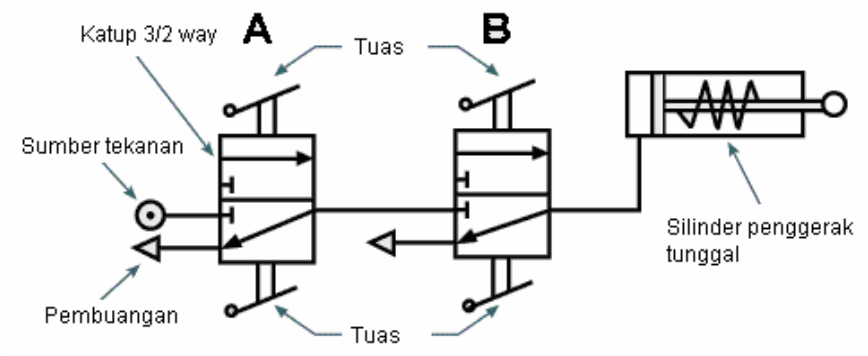
Ketika tombol ditekan pada katup kedua 3/2 way (yang sebelah kanan), akan menggerakan katup 5/2, terjadi pembuangan udara dan menyebabkan piston memukul ke dalam.

5.6.6.3 Rangkaian Logika

Rangkaian logika pnematik adalah rangkaian yang sangat kompleks, karena dikontrol lebih dari sebuah katup, dalam hubungan seri (rangkaian AND) atau dalam hubungan parallel (rangkaian OR).

5.6.6.3.1 Rangkaian AND

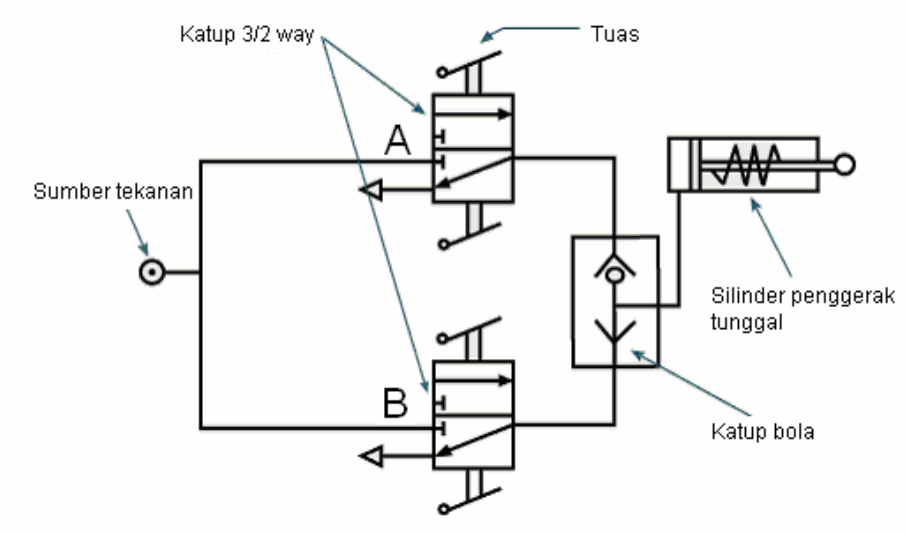
Sebuah rangkaian AND terdiri dari dua katup yang dihubungkan seri. Pada Gambar 5.51 memperlihatkan dua buah katup 3/2 way (A dan B) akan tertekan ketika piston silinder penggerak tunggal memukul ke luar. Katu B akan kehilangan tekanan ketika katup A terbuka. Katup B mengontrol tekanan silinder penggerak tunggal.



Gambar 5.51 Rangkaian AND-dua buah katup dihubungkan seri

5.6.6.4 Rangkaian OR

Sebuah rangkaian OR terdiri dari dua buah katup yang dihubungkan parallel. Pada Gambar 5.52 memperlihatkan dua buah katup 3/2 way (A dan B) mempunyai sebuah sumber tekanan, jika salah satu tertekan, udara akan mengalir ke dalam silinder penggerak tunggal, dan menyebabkan piston memukul keluar. Katup bola mengatur udara ke silinder dan mencegah aliran udara keluar langsung ke pembuangan, katup menjadi tidak aktif.



Gambar 5.52 Rangkaian OR-dua buah katup dihubungkan parallel

5.6.6.5 Kalkulasi

Gaya yang keluar dari silinder dapat dihitung dengan formula:

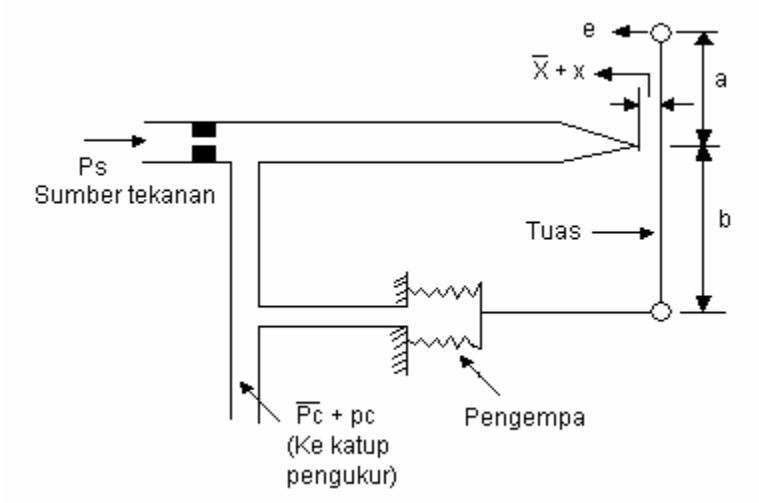
Gaya (F) = tekanan (p) x luas (A)

Gaya (F)	satuannya Newtons (N)
tekanan (p)	satuannya Newtons/mm² (N/mm²).
luas (A)	satuannya mm²

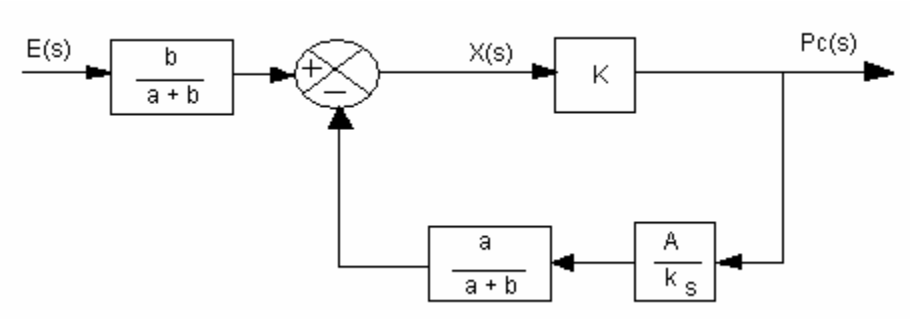
contoh, sebuah piston mempunyai radius 20mm, dan tekanan di dalam silinder 4 bar maka :
tekanan 4 bar = 4/10 = 0,4 N/mm²
luas permukaan piston adalah :
3.14 x 20 x 20 = 1256mm²
Sehingga gaya (F) yang keluar dari silinder =0.4 x 1256 = 502.4N

5.6.6.6 Kontroler Pnematik

Pada Gambar 5.53 memperlihatkan pnematik diimplementasikan sebagai kontroler jenis proporsional. Dari setiap perubahan sekecil apapun pada Gambar 5.53, dapat dibuatkan diagram blok kontrolernya seperti diperlihatkan pada Gambar 5.54.



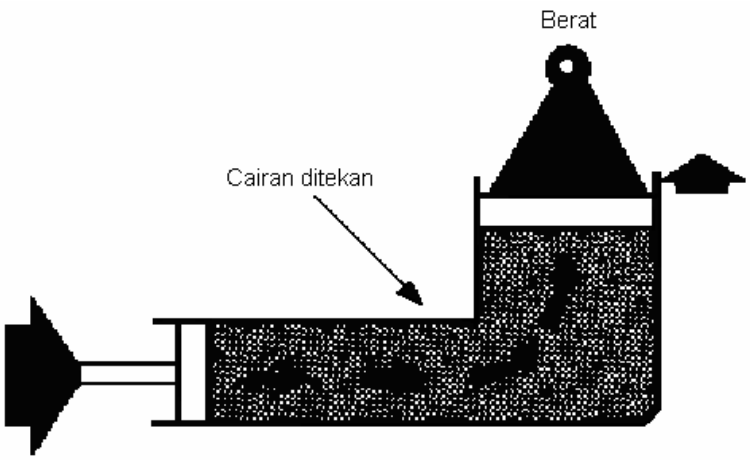
Gambar 5.53 Kontroler proporsional pnematik



Gambar 5.54 Diagram blok kontroler

5.6.7 Dasar-dasar Hidrolika

Hidrolika adalah ilmu pemindahan gaya dan / atau gerak melalui media cairan. Didalam perangkat hidrolika, tenaga dipindahkan oleh dorongan cairan. Pada Gambar 5.55 diperlihatkan perangkat hidrolika sederhana. Untuk mengoperasikan system tenaga-cairan, operator seharusnya memiliki pengetahuan dasar daripada zat cair.

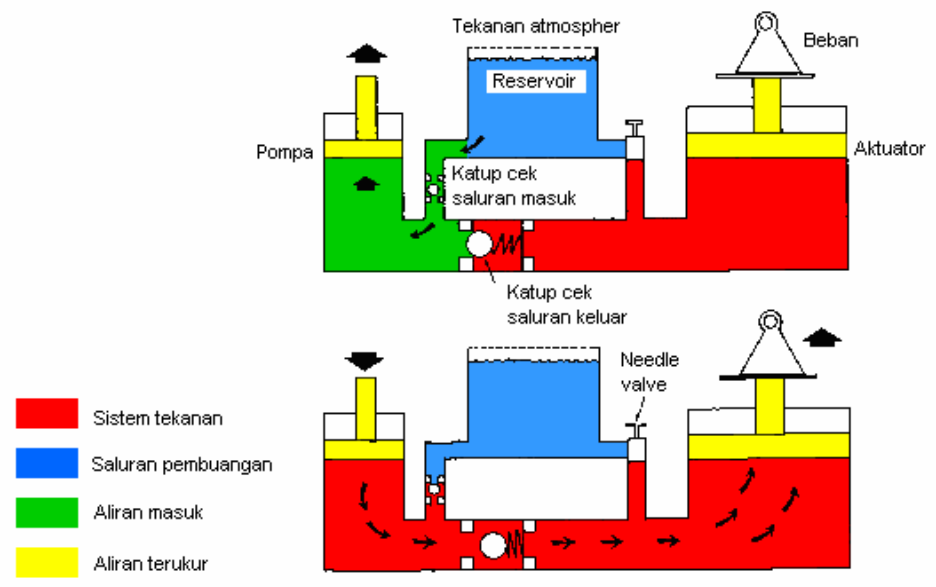


Gambar 5.55 Perangkat hidrolik sederhana

5.6.7.1 Sistem Dasar Hidrolika

5.6.7.1.1 Hidrolika Jack

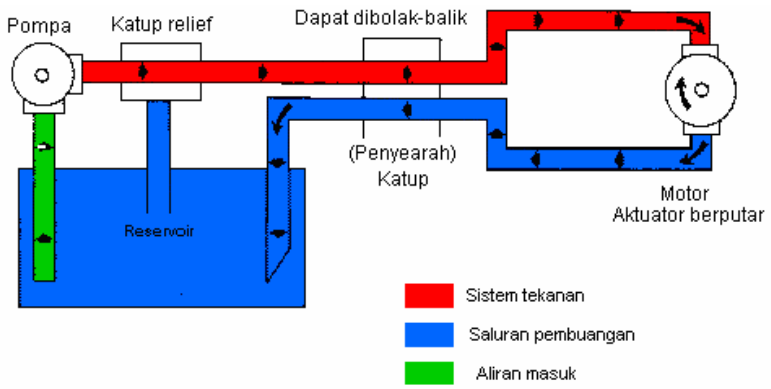
Pada Gambar 5.56 , sebuah reservoir dan sebuah system katup ditambahkan pada tuas hidrolika Pascal untuk mendesak silibder kecil atau pompa secara terus-menerus and mengangkat piston besar atau actuator pada setiap desakan. Diagram A memperlihatkan sebuah desakan masuk. Katup cek saluran keluar tertutup dibawah tekanan beban, dan katup cek saluran masuk terbuka, segingga cairan dari reservoir mengisi ruang pompa. Diagram B memperlihatkan pompa mendesak turun. Katup cek saluran masuk tertutup oleh tekanan dan katup saluran keluar terbuka. Banyak cairan dipompa dibawah piston besar hingga mengangkat piston tersebut. Pada beban yang rendah, katup ketiga (needle valve) terbuka, maka terjadi area terbuka di bawah piston besar hingga reservoir. Kemudian beban menekan piston turun dan mendesak cairan masuk ke dalam reservoir.



Gambar 5.56 Hidrolika Jack

5.6.7.1.2 Sistem Motor Bolak-balik

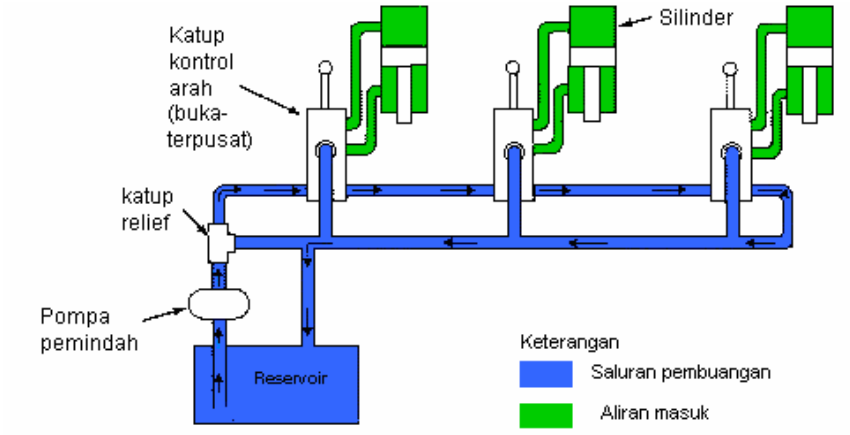
Pada Gambar 5.57 memperlihatkan operasi pompa pengendali-tenaga sebuah motor berputar bolak-balik arah. Sebuah katup bolak-balik mengarahkan cairan ke salah satu sisi daripada motor dan kembali ke reservoir. Sebuah katup relief (pembebas) melindungi sistem perlawanan kelebihan tekanan dan dapat mem-bypass pompa keluar menuju reservoir, jika tekanan naik terlalu tinggi.



Gambar 5.57 Sistem Motor Bolak-balik

5.6.7.1.3 Hubungan Seri

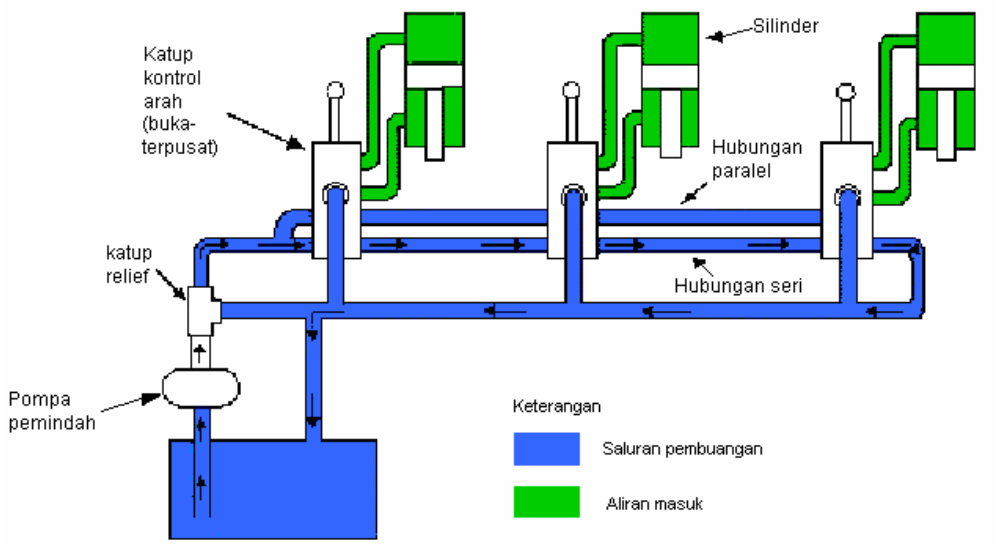
Gambar 5.58 memperlihatkan sebuah hubungan seri sistem buka-terpusat dengan. Oli dari sebuah pompa melewati tiga buah katup control yang dihubungkan seri. Selesai dari katup pertama masuk ke katup kedua, dan begitu seterusnya. Dalam keadaan netral, oli meninggalkan katup-katup dan kembali ke reservoir, seperti yang ditunjukan anak panah. Ketika katup kontrol dioperasikan, kedatangan oli dialihkan ke silinder. Kembalinya oli dari silinder diarahkan melalui jalan balik dan begitu pula pada katup berikutnya.



Gambar 5.58 Hubungan seri sistem buka-terpusat

5.6.7.1.4 Hubungan Seri / Paralel

Gambar 5.59 memperlihatkan variasi pada hubungan seri. Oli dari pompa melewati katup kontrol dalam hubungan seri, sebagaimana dalam hubungan parallel. Katup kadang-kadang disusun untuk memenuhi lintasan tambahan. Dalam kondisi netral, cairan langsung melewati katup-katup dalam hubungan seri,seperti yang ditunjukan anak panah. Ketika tak satupun katup beroperasi, pembalik tertutup dan oli yang tersedia menuju semua katup melalui hubungan parallel.

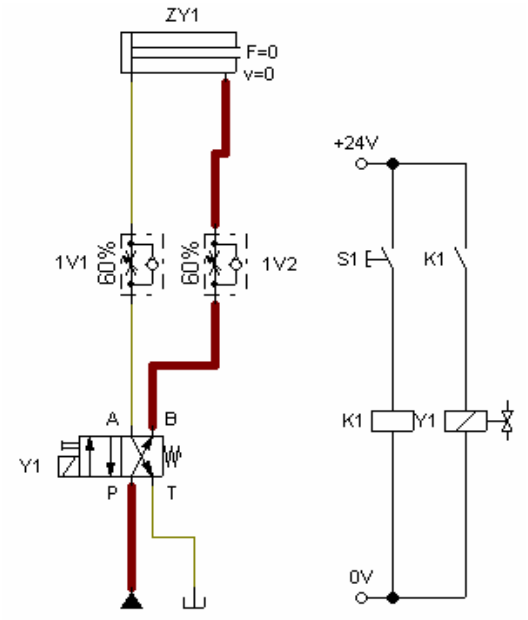


Gambar 5.59 Hubungan Seri / Paralel

Ketika dua katup atau lebih dioperasikan dengan serentak, silinder yang membutuhkan tekanan paling kecil akan beroperasi pertama, kemudian silinder dengan paling sedikit berikutnya, dan seterusnya. Kemampuan untuk beroperasi dua katup atau lebih secara terus-menerus adalah keuntungan daripada hubungan seri.

5.6.7.2 Sistem Elektro Hidrolika

Pada sistem ini kontrol yang dipakai adalah minyak tekan dan dikontrol oleh elektrik (elektro hidrolika). Gambar 5.60 memperlihatkan salah satu contoh rangkaian elektro hidrolika.

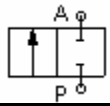
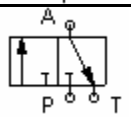
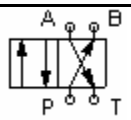
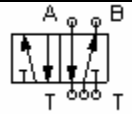


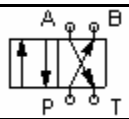
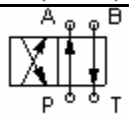
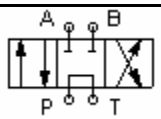
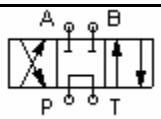
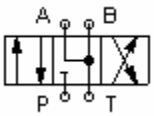
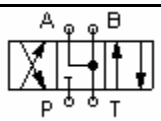
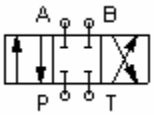
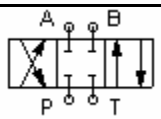
Gambar 5.60 Rangkaian elektro hidrolika

5.6.7.3 Simbol – simbol Hidrolika

Sebagaimana system elektrik, pnematik yang memiliki simbol-simbol komponen, demikian pula sistem hidrolika memiliki simbol-simbol komponen untuk operasional di industri yang telah distandarisasikan.

Tabel 5.11 Simbol – simbol Katup Hidrolika

SIMBOL	NAMA KATUP
	2/n Way Valve, N/C
	3/n Way Valve, N/C
	4/n Way Valve, N/O
	5/n Way Valve, N/C

	4/2 Way Valve, N/O
	4/2 Way Valve, N/O
	4/3 Way Valve with bypass position, N/C bersirkulasi
	4/3 Way Valve with bypass position, N/C bersirkulasi
	4/3 Way Valve with floating position, N/C
	4/3 Way Valve with floating position, N/C
	4/3 Way Valve with shutoff position, N/C
	4/3 Way Valve with shutoff position, N/C

Tabel 5.12 Simbol – simbol Pengaktifan Manual

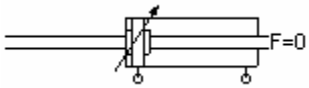
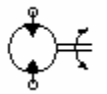
Jenis Pengaktifan Manual	Keterangan
	Operasi handle
	Operasi handle dengan pegas kembali
	Tombol
	Tombol dengan pegas kembali
	Operasi tuas
	Operasi tuas dengan pegas kembali

	Pedal kaki
	Pedal kaki dengan pegas kembali

Tabel 5.13: Simbol – simbol Pengaktifan Elektrik

Jenis Pengaktifan Elektrik	Keterangan
	Operasi degan solenoid
- - -	Operasi dengan tekanan hidrolik
	Operasi degan tekanan hidrolik dan solenoid

Tabel 5.14: Simbol – simbol Aktuator Hidrolika

SIMBOL	NAMA KOMPONEN
	Silinder kerja ganda dengan batang piston ganda dan memakai bantalan minyak ganda, dapat diatur pada kedua sisi.
	Silinder kerja ganda dengan bantalan minyak ganda, dapat diatur pada kedua sisi.
	Silinder kerja ganda.
	Motor hidrolik.
	Silinder kerja tunggal.

5.6.7.4 Kontroler Hidrolika

Servomotor hidrolika pada dasarnya adalah actuator dan penguat daya hidrolika yan dikontrol katup pandu (pilot). Katup pandu adalah katup berimbang, yaitu semua tekanan yang bekerja terhadapnya adalah berimbang. Keluaran daya yang besar dapat dikontrol oleh katup pandu, yang dapat diposisikan dengan daya yang kecil.

Perhatikan Gambar 5.61 jika masukan X menggerakkan katup pandu ke kanan, maka port I terbuka, dan oli tekanan tinggi akan memasuki sebelah sisi kanan torak daya. Karena port II dihubungkan dengan port pembuangan, maka oli di sebelah kiri torak daya dikembalikan ke pembuangan. Oli mengalir ke silinder daya pada tekanan tinggi, oli mengalir ke luar dari silinder daya ke dalam pembuangan pada tekanan rendah. Hasil dari perbedaan tekanan tersebut pada kedua sisi, akan menyebabkan torak daya bergerak ke kiri.

$$A\rho\, dy = q\, dt$$
(7-1)

A : luas permukaan torak

ρ : massa jenis oli

Di asumsikan laju arus oli q sebanding dengan perpindahan katup pandu x, maka

$$q = K_1 x$$
(7-2)

K_1: adalah konstanta

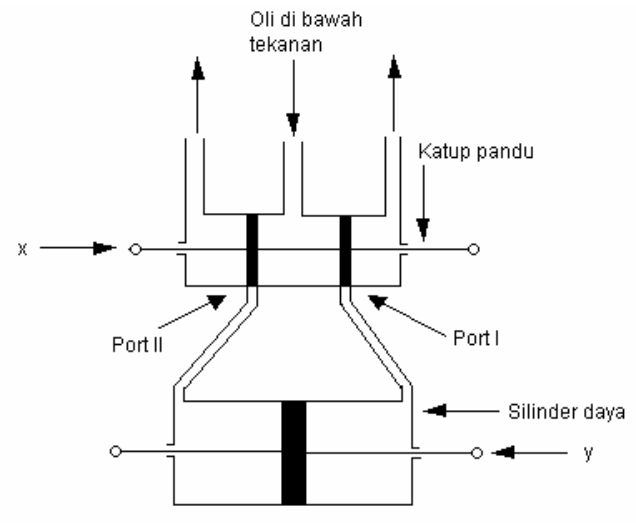
Dari persamaan (7-1) dan (7-2) diperoleh :

Dalam tranformasi Laplace ditulis:

$$A\rho\, Y(s) = K_1 X(s)$$

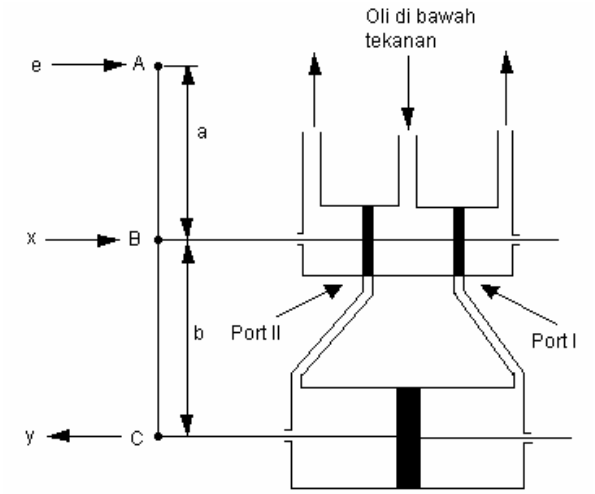
atau

dengan $K = K_1/(A\rho)$, sehingga aksi servomotor pada Gambar 7.7 adalah sebagai kontroler integrator.

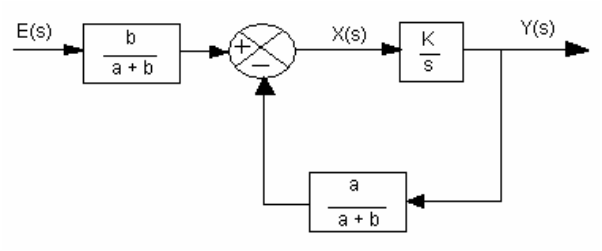


Gambar 5.61 Servomotor hidrolika berfungsi sebagai kontroler integrator

Diatas telah dibahas bahwa servomotor pada Gambar 5.61 dapat berfungsi sebagai kontroler integrator. Selain itu servomotor dapat dimodifikasi menjadi kontroler proporsional dengan menggunakan hubungan umpan balik, perhatikan Gambar 5.62, dan diagram bloknya diperlihatkan pada Gambar 5.63.



Gambar 5.62 Servomotor hidrolika berfungsi sebagai kontroler proporsional



Gambar 5.63 Diagram blok servomotor hidrolika sebagai kontroler proporsional

BAB VI Sistim Mikrokomputer

6.1 Aritmetika Komputer

6.1.1 Sistim Bilangan

Sistem bilangan desimal adalah sistem bilangan yang palng sering digunakan dan yang paling kita pahami. Sistem ini berbasiskan angka 10. 10 adalah dasar sistim bilangan desimal yang menggambarkan angka 0 sampai 9, yang jumlah keseluruhan symbol ada 10 buah.
Misalkan suatu angka sistem desimal adalah :

$2305{,}51 = \quad 2 \cdot 10^3 + 3.10^2 + 0.\ 10^1 + 5.\ 10^{-1} + 1 \cdot 10^{-2}$

Pada komputer digital, angka harus dapat di representasikan dengan berbagai macam nilai dari kuantitas fisik. Seperti tegangan, arus, medan magnet, dan lain sebagainya. Misalkan kita mengalokasikan angka 1 sebagai 1 Volt, 2 sebagai 2 Volt, dan seterusnya. Sehingga 0 sampai dengan 9 volt bisa direpresentasikan oleh angka 0 sampai dengan 9 pada sistem desimal. Penerapan sistim bilangan pada komputer secara teknik memungkinkan tetapi sangat rumit dan mahal. Cara yang paling sederhana adalah dengan menerapkam sistim bilangan duaan yang terdiri dari dua keadaan kunatitas phisik dan dinyatakan dalam dua buah angka misalnya 0 dan 1 dengan ketentuan sebagai berikut :

$0 \rightarrow$ yang berarti tidak ada arus, tidak ada tegangan , berlogika rendah
$1 \rightarrow$ yang berarti ada tegangan, ada arus , berlogika tinggi

Semua komputer digital modern menggunakan prinsip ini tanpa perkecualian. Sistim bilangan duaan tersebut hanya menggunakan dua simbol 0 dan 1, dan kita sebut sebagai sistim bilangan biner. Sisitim bilangan biner tetap menerapkan sistim yang sama seperti pada sistim bilangan desimal.
Suatu bilangan biner dapat dipahami sebagai berikut :

$1011{,}01 = \quad 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$

Pada prinsipnya, perhitungan biner dan desimal tidak berbeda satu sama lain. Ketika menghitung, jumlah angka akan bertambah satu angka ke depan apabila terjadi carry. Pada sistim biner terjadi penambahan angka apabila perhitungan melebihi angka 1 dan pada sistim desimal akan terjadi apabila perhitungan melebihi angka 9.

Tabel 6.1 Konversi Biner Ke Desimal

Urutan angka biner	Urutan angka desimal
0	0
1	1
1 0	2
1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	10
. . .	. . .

Ketika sistim bilangan yang berbeda digunakan pada waktu yang bersamaan, untuk membedakannya pada umumnya adalah dengan menuliskan sistim bilangan dasar sebagai indeks seperti berikut ini

1001_{10} mewakili bilangan desimal $1 . 10^3 + 0 . 10^2 + 0 . 10^1 + 1 . 10^0$

1001_2 mewakili bilangan desimal $1 . 2^3 + 0 . 2^2 + 0 . 2^1 + 1 . 2^0$

Suatu angka dalam sistim bilangan biner membutuhkan sekitar tiga kali jumlah digit untuk nilai angka yang sama dibandingkan dengan sistim bilangan desimal seprti dicontohkan berikut ini :

$2049_{10} = 100000000001_2$

Bilangan biner kadang menjadi mudah dimengerti dengan cepat karena berapa nilainya karena panjang atau banyaknya digit. Untuk penerapan bilangan biner pada komputer digunakan metode pengelompokan 3 atau 4 digit untuk mempermudah dan mempercepat mengetahui nilaiya. Sistim pengelompokan tersebut dipergunakan pada sistim bilangan oktal dan heksadesimal.
Bilangan oktal adalah suatu sistim bilangan delapanan yang memiliki simbol angka 0 sampai 7 seperti berikut ini

Tabel 6.2 Konversi Biner Ke Oktal

Urutan angka biner	Urutan angka oktal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7
1 0 0 0	10
1 0 0 1	11
1 0 1 0	12
1 0 1 1	13
. . .	. . .

Mengkonversi bilangan biner ke bilangan oktal sangat mudah dilakukan dengan cara mengelompokkan bilangan biner per tiga digit kemudian setiap kelompok tiga digit dikenversi sendiri-sendiri seperti contoh berikut

Biner 101 110 011 111 101 010

Oktal 5 6 3 7 5 2

Jadi $10111001111110101 0_2 = 563752_8$

Bilangan heksadesimal adalah suatu sistim bilangan enambelasan yang memiliki simbol angka 0 sampai 9 dan kemudian huruf A sampai F seperti berikut ini

Tabel 6.3 Konversi Biner Ke Heksadesimal

Urutan angka biner	Urutan angka heksadesimal
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

Mengkonversi bilangan biner ke bilangan heksadesimal sangat mudah dilakukan dengan cara mengelompokkan bilangan biner per empat digit kemudian setiap kelompok empat digit dikenversi sendiri-sendiri seperti contoh berikut :

Biner 1101 1111 0011 0101

Heksa D F 3 5

Jadi $1101111100110101_2 = DF35_{16}$

Dalam praktek pemrograman dan dalam buku-buku mikroprosesor, keempat sistim bilangan tersebut sangat sering dipergunakan. Sistim bilangan mana yang paling baik dipergunakan tergantung pada permasalahan yang dihadapi. Kita harus paham untuk menentukan sistim bilangan yang mana lebih lebih mudah dan lebih tepat untuk menyelesaikan suatu permasalahan.

6.1.2 Konversi Antar Sistim Bilangan

Ketika banyak sistim bilangan dipergunakan bersama-sama, perlu dilakukan konversi dari satu sistim bilangan ke sistim bilangan lainnya. Hal tersebut telah ditunjukkan di atas untuk konversi dari biner ke oktal dan dari biner ke heksadesimal. Terdapat banyak cara dan metode untuk mengkonversi sistim bilangan. Pada prakteknya yang paling penting

adalah mengkonversi dari sistim bilaingan desimal ke sistim bilangan lainnya dan sebaliknya. Pada pembahasan berikut ini akan dijelaskan secara detail macam macam metode konversi.

6.1.2.1 Metoda Pembagian

Metoda ini paling tepat dipergunakan untuk mengkonversi bilangan desimal ke sistimbilangan lainnya. Metode pembagian ini akan diilustrasikan dengan contoh mengkonversi bilangan desimal ke bilangan biner.

Contoh 457_{10} = ...$_2$

cara mengkonversi adalah dengan membagi 2 bilangan tersebut sampai diperoleh sampai habis seperti berikut ini :

				hasil	Sisa
457	÷	2	=	228	1
228	÷	2	=	114	0
114	÷	2	=	57	0
57	÷	2	=	28	1
28	÷	2	=	14	0
14	÷	2	=	7	0
7	÷	2	=	3	1
3	÷	2	=	1	1
1	÷	2	=	0	1

Jadi hasil konversi $457_{10} = 111001001_2$

6.1.2.2 Metoda Perkalian

Metoda perkalian dpergunakan untuk mengkonversi bilangan desimal pecahan dengan nilai kurang dari satu misalnya 0,xxx ke sistim bilangan lainnya.

Contoh $0{,}5625_{10}$ = ……$_2$

				Hasil	Angka bulat	Angka pecahan
0.5625	.	2	=	1.1250	(MSB) 1	0.1250
0.1250	.	2	=	0.2500	0	0.2500
0.2500	.	2	=	0.5000	0	0.5000
0.5000	.	2	=	1.0000	(LSB) 1	0.0000

Jadi $0{,}5625_{10} = 0{,}1001_2$

6.1.2.3 Konversi Bilangan Rasional

Untuk mengkonversi bilangan rasional yang terdiri dua bagian yaitu bagian bilangan bulat (angka didepan koma) dan bilangan pecahan (angka di belakang koma) maka proses konversi dilakukan dengan dua metoda yang berbeda. Bagaian bilangan bulat dikonversi menggunakan metoda pembagian dan bagianm pecahan dikonversi dengan metoda perkalian.

Contoh $21{,}375_{10}$ = ……$_{2}$

Konversi pertama untuk bagian bilangan bulat angka di depan koma (21)

				hasil	Sisa
21	÷	2	=	10	1
10	÷	2	=	5	0
5	÷	2	=	2	1
2	÷	2	=	1	0
1	÷	2	=	0	1

Konversi kedua untuk bagian bilangan pecahanangka di belakang koma (0.375)

				Hasil	Angka bulat	Angka pecahan
0.375	.	2	=	0.750	0	0.750
0.750	.	2	=	1.500	1	0.500
0.500	.	2	=	1.000	1	0.000

Jadi $21{,}375_{10} = 10101{,}011_{2}$

6.1.2.4 Konversi Ke Sistim Bilangan Desimal

Contoh bilangan 11001_{2} = ……$_{10}$

Penyelesaian :

$$11001_{2} = 1 . 2^{4} + 1 . 2^{3} + 0 . 2^{2} + 0 . 2^{1} + 1 . 2^{0}$$
$$= 1 . 16 + 1 . 8 + 0 . 4 + 0 . 2 + 1 . 1$$
$$= 25_{10}$$

6.1.3 Aritmetika Biner

Di dalam semua sistem bilangan juga diterapkan aturan yang sama dalam perhitungan.

Dalam bagian ini akan dijelaskan 4 operasi aritmatika dasar dalam sistem biner

6.1.3.1 Penjumlahan

Misalkan $100101_2 + 010111_2 = \ldots\ldots\ldots_2$

	1	0	0	1	0	1
+	0	1	0	1	1	1
Carry	0	0	1	1	1	0
Hasil	1	1	1	1	0	0

Jadi $100101_2 + 010111_2 = 111100_2$ carry 0

Carry = 0 diambil dari nilai carry yang paling kiri, bit yang terbesar (MSB)

Dalam penjumlahan biner carry untuk satu digit selanjutnya didapatkan bila terdapat penjumlahan dalam satu kolom yang bernilai lebih dari 2.

6.1.3.2 Pengurangan

Misalkan $10010_2 - 01101_2 = \ldots\ldots\ldots_2$

Bobot		2^4	2^3	2^2	2^1	2^0
		1	0	0	1	0
	-	0	1	1	0	1
Borrow		1	1	0	1	0
Hasil		0	0	1	0	1

Pada digit 2^0 angka 0 dikurang 1, artinya $0 \cdot 2^0 - 1 \cdot 2^0$ ini hanya mungkin bila dipinjami oleh digit 2^1, sehingga angka 1 pada digit 2^1 akan bergeser ke kanan menjadi $1 \cdot 2^1 - 1 \cdot 2^0 = 2 - 1 = 1$.
Angka 1 yang muncul ini akan keluar sebagai hasil pada kolom 2^0. Sedangkan peminjaman dari digit 2^1 ditunjukkan pada baris borrow. Pada kolom ini operasi yang terjadi adalah 1 - 0 - 1 = 0 . Proses ini diulangi pada digit 2^2. Pada digit 2^3 angka 0 akan mengurangi angka 1 dan meminjam pada digit 2^4, sehingga pada digit 2^4 muncul nilai 1 pada baris borrow.

Jadi $10010_2 - 01101_2 = 00101_2$ borrow 1

Borrow = 1 diambil dari nilai borrow yang paling kiri, bit yang terbesar (MSB)

Pengurangan pada angka di atas sesuai dengan pengurangan angka desimal :

$18_{10} – 13_{10} = 5_{10}$

6.1.3.3 Perkalian

Aturan perkalian bilangan biner juga memiliki aturan yang sama dengan perkalian bilangan desimal.

Contoh $1001_2 \cdot 1101_2 =$ .....$_2$

1	0	0	1	·	1	1	0	1
	1	0	0	1				
		1	0	0	1			
			0	0	0	0		
				1	0	0	1	
	1	1	1	0	1	0	1	

Hasil perkalian di atas sesuai dengan hasil perkalian pada bilangan desimal :

9_{10} X $13_{10} = 117_{10}$

Karena angka yanag dikalikan hanya bernilai 1 atau 0 maka dalam perkalian biner operasi yang dilakukan hanya penjumlahan dan pergeseran digit.

6.1.3.4 Pembagian

Aturan pembagian pada bilangan biner sama halnya dengan pembagian pada bilangan desimal.
Operasi pembagian dalam bilangan biner adalah pengurangan dan pergeseran.

Contoh $1110101_2 \div 1001_2 = ...{}_2$

```
  1  1  1  0  1  0  1  ÷  1  0  0  1  =  1  1  0  1
- 1  0  0  1
-------------
  0  1  0  1  1
  - 1  0  0  1
  ------------
     0  0  1  0  0
     - 0  0  0  0
     ------------
           1  0  0  1
        - 1  0  0  1
        ------------
           0  0  0  0
```

6.1.4 Operasi Logika Antara Dua Bilangan Biner A dan B

Dalam teknologi komputer, operasi logika dilakukan bit per bit sesuai dengan lokasi digitnya.

Contoh operasi AND
11011010_2 AND $10100110_2 = 10000010_2$

	A	1	1	0	1	1	0	1	0
	B	1	0	1	0	0	1	1	0
A AND B		1	0	0	0	0	0	1	0

Contoh operasi OR

11011010_2 OR $10100110_2 = .11111110._2$

	A	1	1	0	1	1	0	1	0
	B	1	0	1	0	0	1	1	0
A OR B		1	1	1	1	1	1	1	0

Contoh operasi EXOR

11011010_2 XOR $10100110_2 = 01111100_2$

	A	1	1	0	1	1	0	1	0
	B	1	0	1	0	0	1	1	0
A XOR B		0	1	1	1	1	1	0	0

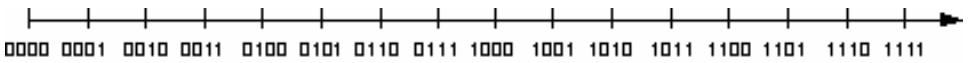
Contoh operasi NOT

NOT $11011010_2 = 00100101_2$

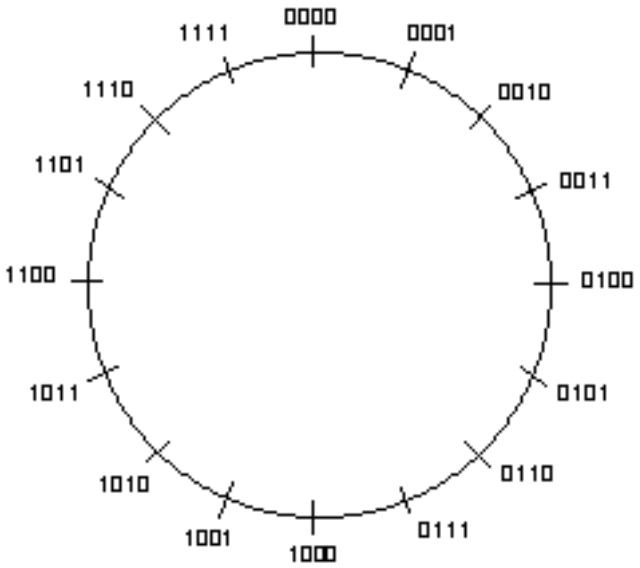
A	1	1	0	1	1	0	1	0
$\overline{A}$	0	0	1	0	0	1	0	1

6.1.5 Aritmetika Integer

Sampai saat ini kita masih berkonsentrasi pada bilangan positip saja dan juga tidak menentukan batas digit yang dapat diproses dan ditampilkan sebagai hasil suatu operasi.
Suatu mikroprosesor 8 bit dapat mengolah data biner mulai 00000000_2 sampai 11111111_2 atau dalam bilangan desimal mulai 0_{10} sampai 255_{10} dan tidak diijinkan suatu operasi melebihi batas angka tersebut.
Suatu mikroprosesor 4 bit dapat mengolah data biner mulai 0000_2 sampai 1111_2 atau dalam bilangan desimal mulai 0_{10} sampai 15_{10}. Batas angka yang dapat diproses apabila direpresentasikan dalam grafik skala arah ditunjukkan pada Gambar 4.1.



Gambar 6.1 Grafik skala arah sistim bilangan positip 4 bit



Gambar 6.2 Grafik lingkaran angka sistim bilangan integer positip 4 bit

Operasi suatu komputer 4 bit selain dengan grafik skala arah dapat juga direpresentasikan dalam grafik lingkaran angka seperti tampak pada Gambar 4.2.

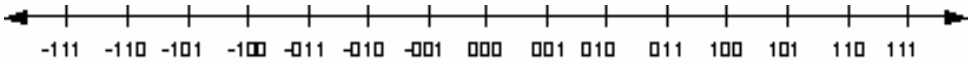
Penjumlahan dua bilangan hasilnya benar apabila hasil operasi penjumlahan masih dalam batas 0000_2 sampai 1111_2, tetapi apabila penjumlahan hasilnya melebihi batas tersebut maka angka yang tampil sebagai hasil akrhir operasi menjadi salah. Perhatikan contoh di bawah ini

$1111_2 + 0010_2 = 10001_2$

Dengan menggunakan grafik lingkaran angka, hasil penjumlahan tersebut adalah 0001_2 saja karena sistim mikroposesor adalah 4 bit, sedangkan digit 1 yang tidak muncul pada hasil akan masuk sebagai carry. Proses penjumlahan yang melebihi batas angka dalam lingkaran kita sebut overflow.
Terjadinya carry pada digit ke 5 tidak hilang tetapi disimpan dalam suatu flip-flop yang disebut Carry Flip-flop atau biasanya disebut Carry flag. Carry flag dapat dilihat setelah operasi aritmetika dilakukan oleh suatu program.
Kondisi di atas hanya untuk contoh mikroprosesor 4 bit. Untuk mikroprosesor dengan bit yang lebih tinggi dalam instruksi aritmetika suatu overflow adalah mengindikasikan adanya carry pada most significant binary digit (MSB) dan disimpan dalam suatu flag register.

6.1.6 Aritmetika Komplemen Dua

Bilangan posistip dan negatip ditunjukan dalam grafik skala bilangan sebagai berikut



Gambar 6.03 Grafik Skala Bilangan Positif Negatif

Komplemen satu secara umum diartikan membuat suatu angka menjadi bilangan terbesar dengan jumlah digit yang sama, misalnya komplemen dari angka 0001_2 adalah 1110_2.
Komplemen satu dalam sistim bilangan biner dapat diartikan opeasi membalik (inverting) angka biner.
Dalam sistim mikroprosesor bilangan negatif direpresentasikan sebagai komplemen dua.

Rumus komplemen dua adalah

$\overline{A} + 1 = -A$

Pada dasarnya penjumlahan A dengan komplemen duanya akan menghasilkan 0.

$A + (\overline{A} + 1) = 0$
$(\overline{A} + 1) = -A$

Contoh operasi komplemen dua :

A	0	1	0	1	1	0	1	0	
$\overline{A}$	1	0	1	0	0	1	0	1	(komplemen satu)
+1								1	
$-\overline{A} = A + 1$	1	0	1	0	0	1	1	0	(komplemen dua)

Contoh operasi A - A =

A		0	1	0	1	1	0	1	0
-A		1	0	1	0	0	1	1	0
A +(-A)	1	0	0	0	0	0	0	0	0

Jadi sekarang dalam aritmatika biner kita bisa mebmbuat pengurangan desimal
$9_{10} – 3_{10} = 6_{10}$ atau juga yang dituliskan dengan $9_{10} + (- 3_{10}) = 6_{10}$

Contoh operasi $9_{10} + (- 3_{10})$ untuk sistim bilangan biner 4 bit

9_{10}		1	0	0	1	
$+(- 3_{10})$		1	1	0	1	(komplemen dua dari 0011_2 atau 3_{10}
$9_{10}+(-3_{10})$	1	0	1	1	0	Hasil = 6_{10} carry 1

Contoh operasi $6_{10} + (- 9_{10}) =$

6_{10}		0	1	1	0	
$+(- 9_{10})$		0	1	1	1	(komplemen dua dari 1001_2 atau 9_{10})
$6_{10}+(-9_{10})$	0	1	1	0	1	Hasil = 13_{10} carry 0

Hasil = 13_{10} kelihatannya salah karena hasil yang benar seharusnya adalah $–3_{10}$.
Jangan tergesa-gesa mengambil kesimpulan salah, karena angka 13 dalam sistim mikroprosesor 4 bit, angka 13_{10} atau 1101_2 adalah komplemen dua dari angka 3_{10} atau 0011_2.

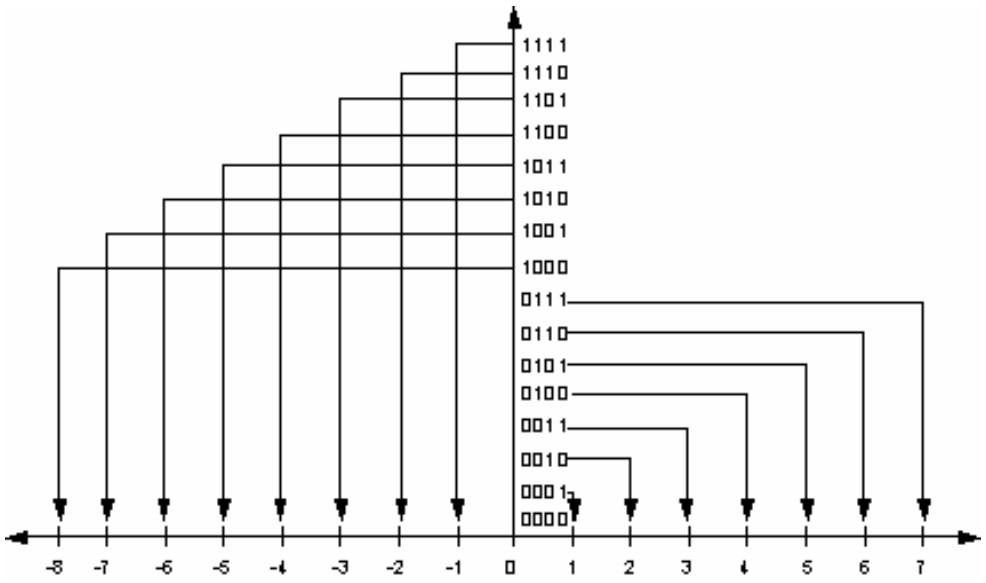
A	0	0	1	1	
$\overline{A}$	1	1	0	0	(komplemen satu dari 0011_2 atau 3_{10})
+1				1	
-A	1	1	0	1	(komplemen dua dari 0011_2 atau 3_{10})

Dengan demikian hasil operasi = 13_{10} carry 0 adalah benar.

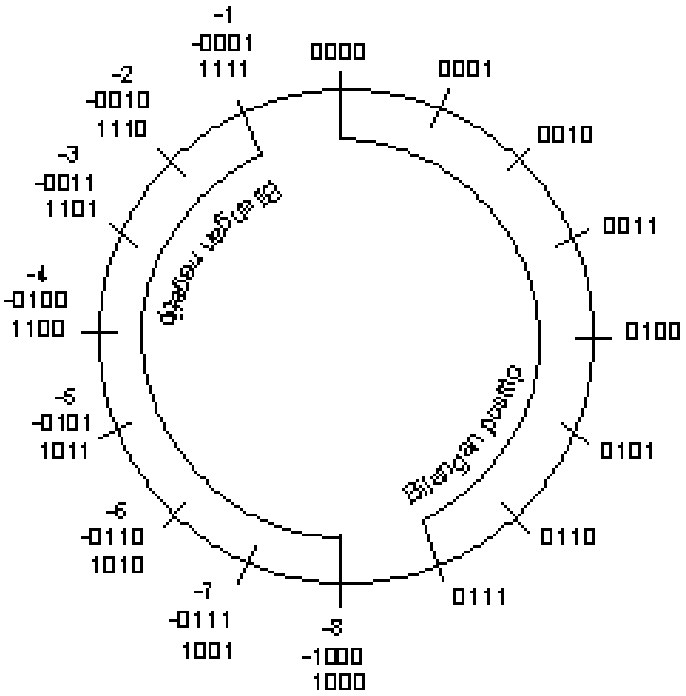
Dari dua contoh di atas, kita dapat melihat bahwa suatu bilangan itu positip atau negatip dapat diketahui dari bit yang paling tinggi. Apabila bit yang paling tinggi (MSB) berlogika 0 maka bilangan tersebut adalah bilangan positip dan apabila bit yang paling tinggi (MSB) berlogika 1 maka bilangan tersebut adalah bilangan negatip.

A	- A
0001	1111
0010	1110
0011	1101
0100	1100
0101	1011
0110	1010
0111	1001

Kolom sebelah kiri merupakan bilangan positif dari 0001 sampai dengan 0111, sedangkan kolom sebelah kanan merupakan kolom negatif dari – 0001 sampai dengan – 0111.



Gambar 6.4 Representasi Komplemen Dua



Gambar 6.5 Representasi komplemen dua untuk sistim bilangan 4 bit

Representasi dalam bentuk lingkaran memperlihatkan bahwa bilangan positip terletak pada sisi kanan dan bilangan negatip terletak pada bagian sisi kiri lingkaran. Tampak bahwa most significant bit menentukan tanda bilangan positip atau negatip. Angka negatip terbesar adalah 1000_2 untuk sistim bilangan 4 bit dan 10000000_2 untuk sistim bilangan 8 bit .
Format komplemen dua untuk sistim bilangan 8 bit adalah sama seprti sistim bilangan 4 bit, rumus dasar komplemen dua adalah $-A = A + 1$ dan berlaku untuk berapapun lebar bit data.
Contoh kasus komplemen dua untuk sistim 8 bit

A	1	0	0	0	0	0	0	0
$\overline{A}$	0	1	1	1	1	1	1	1
+1								1
-A	1	0	0	0	0	0	0	0

Hasil komplemen dua dari bilang tersebut adalah sama, tentunya hal ini adalah tidak benar maka kita harus menghindari komplemen dua dari suatu bilangan negatip.

6.1.7 Pengkodean Biner Dari Bilangan Desimal

Tabel 6.4 Konversi Desimal Ke Kode BCD

Desimal	Kode BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Penerapan komputer sebagai instrumen pengukuran bekerja dengan sistim bilangan desimal. Kita mengenal kode BCD (binary code desimal) yang memiliki sombol angka 0 sampai 9 yang mengkodekan bilangan biner 4 bit. Kode bilangan BCD ini disebut pula kode 8421 yang terdiri dari sepuluh kombinasi , sedangkan Sisa 6 kombinasi bit dari 1010 sampai 1111 tidak dipergunakan.
Contoh kode BCD untuk angka 19378 adalah 0001 1001 0011 0111 1000

6.2 Mode Operasi Komputer

Mode operasi suatu komputer pada dasarnya dapat diketahui dengan melihat kemampuannya dalam memproses data yang diinstruksikan oleh suatu program. Artinya dengan program yang berbeda suatu komputer dapat digunakan untuk menyelesaikan masalah yang berbeda.
Beberapa tahun yang lalu komputer hanya dipergunakan untuk instalasi yang sangat terbatas, sekarang dengan teknologi semikonduktor komputer dapat dibuat menjadi lebih kecil dan lebih murah sehingga penggunaannya menjadi sangat luas tak terbatas. Kompter kecil ini disaebut komputer mini.

Langkah berikutnya dalam pengembangan teknologi pembuatan komputer adalah penggunaan komponen LSI (Large Scale Integration) yang diterapkan pada CPU (Central Processing Unit) suatu mikrokonmputer yang didalamnya terdapat ribuan komponen semikonduktor hanya dalam satu chip CPU. CPU suatu mikrokomputer disebut mikroprocesor. Salah satu jenis mikroprosesor yang dipergunakan dalam eksperimen ini adalah Intel 8080 didalamnya terdapat lebih dari 4500 transistor MOS dalam sattu chip yang berukuran 23 mm^2. Dengan adanya lebih dari 4500 transistor memungkinkan untuk mengimplementasikan suatu kontrol dan aritmetic logic unit yang lengkap

untuk mikrokomputer. Dalam aritmetic logic unit, operasi aritmetika dan logika dapat dilakukan karena adanya control unit yang mengontrol prosesor internal dalam komputer. Agar mikroprosesor dapat bekerja disuatu mikrokomputer maka diperlukan perangkat tambahan seprti memory program, memory data, input output port, tambahan rangkaian digital dal lain sebagainya tergantung dari aplikasi individual. Pada prakteknya tidak ada suatu masalah yang dapat diselesaikan oleh satu mikroprosesor saja, perangkat tambahan pasti dibutuhkan. Tetapi bahwa mikroprosesor pasti selalu diperlukan.
Dalam pembahasan berikut ini akan dijelaskan prinsip kerja mode operasi suatu mikrokomputer secara langkah demi langkah.

6.2 Adder

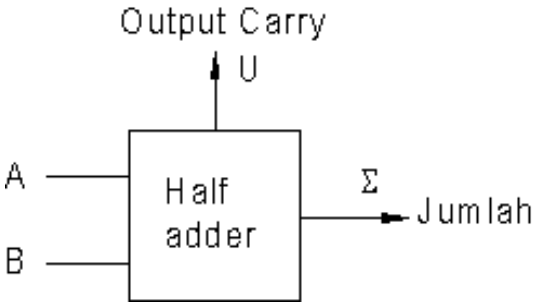
Adder adalah rangkaan digital yang memiliki fungsi sebagai penjumlah biner.

A + B	Penjumlah	Carry
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

Rangkaian digital yang dapat melakukan operasi aritmetika yang hasil keluarnnya seperti tabel di atas disebut Half adder.

Tabel kebenaran

Masukan		Keluaran	
A	B	Σ	U
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Gambar 6.6 Half adder.

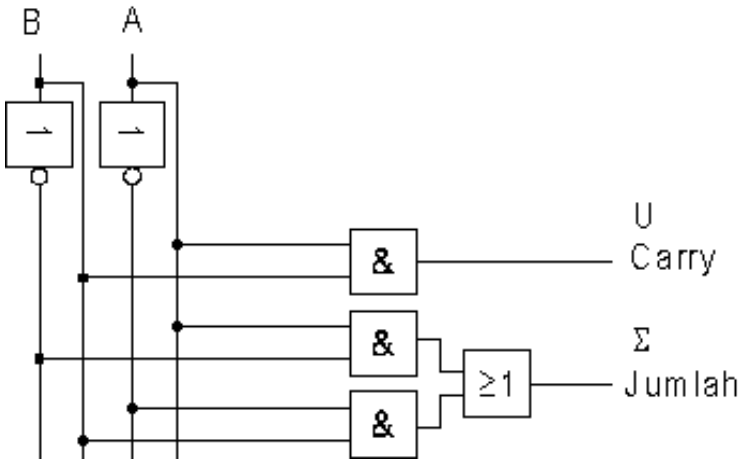
Dari tabel kebenaran dapat diperoleh persamaan fungsi logika untuk keluaran penjumlah Σ dan carry U sebagai beikut :

Penjumlah

$\Sigma = \left(A \wedge \overline{B}\right) \vee \left(\overline{A} \wedge B\right) = A \vee B$ (EXCLUSIVE-OR)

Output Carry $U = \left(A \wedge B\right)$ (AND)

Dari persamaan di atas dapat ditemukan gambar rangkaian digital seperti diperlihatkan pada berikut :



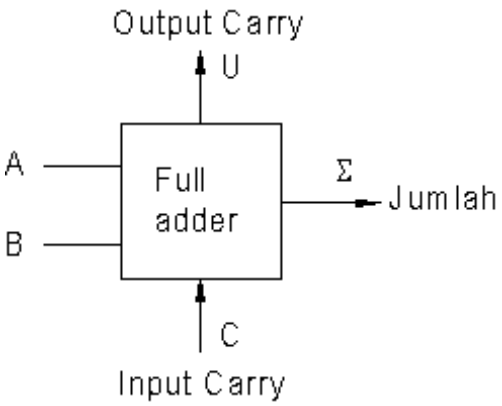
Gambar 6.7 Rangkaian digital half adder

Jika jumlah bit lebih banyak kombinasi (disebut word), penggunaan half adder tidak dapat dilakukan karena tidak ada masukan carry. Untuk itu kita harus menambahkan masukan carry pada half adder. Rangkaian penjumlah yang dengan tambahan masuak carry ini disebut full adder.

Tabel kebenaran

Masukan			Keluaran	
A	B	C	Σ	U
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Diagram blok



Gambar 6.08 *Full Adder*

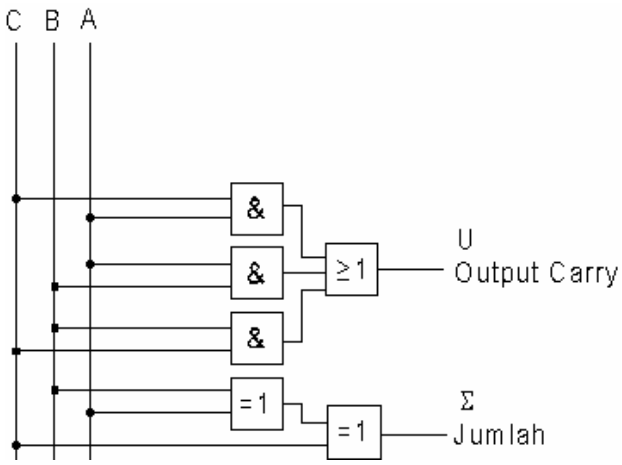
Persamaan fungsi logika untuk keluaran jumlah full adder adalah

Penjumlah $\Sigma = (\overline{A} \wedge B \wedge \overline{C}) \vee (A \wedge \overline{B} \wedge \overline{C}) \vee (\overline{A} \wedge \overline{B} \wedge C) \vee (A \wedge B \wedge C)$

$= (A \forall B) \forall A$

Output Carry $U = (A \wedge B \wedge \overline{C}) \vee (\overline{A} \wedge B \wedge C) \vee (A \wedge \overline{B} \wedge C) \vee (A \wedge B \wedge C)$

$= (A \wedge B) \vee (B \wedge C) \vee (A \wedge C)$

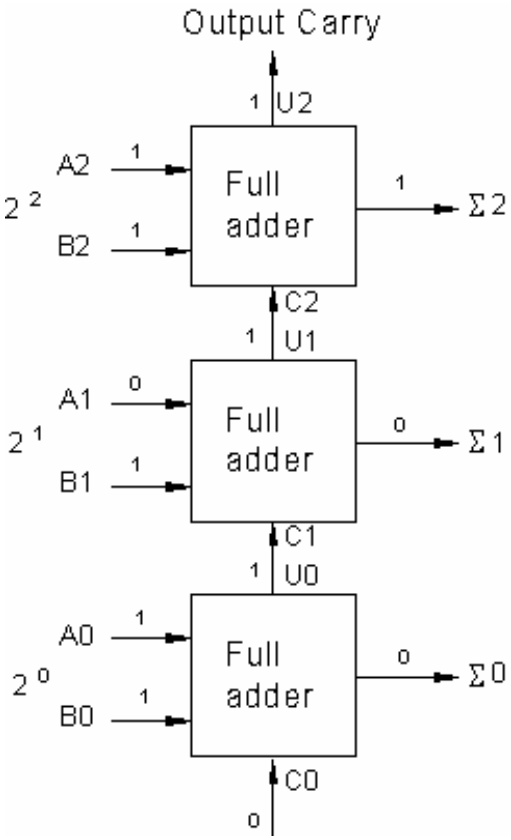


Gambar 6.09 Rangkaian digital *full adder*

Jika dua n-bit word dijumlahkan diperlukan beberapa full adder yang saling disambungkan, sebagai contoh sederhana penjumlahan 3 bit word diperlihatkan di bawah ini

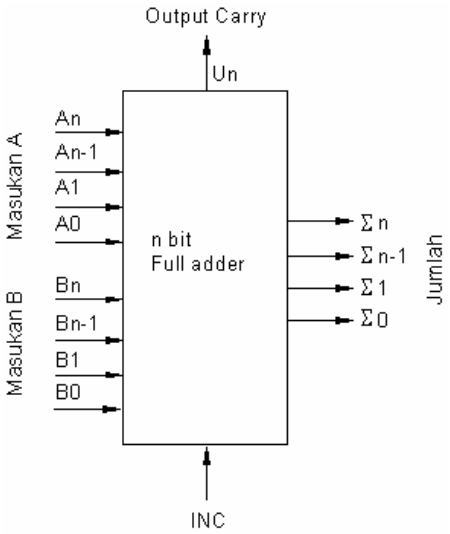
		2^2	2^1	2^0
A		1	0	1
B		1	1	1
U	1	1	1	
Σ		1	0	0

Pada prakteknya penjumlahan 3 bit word memerlukan tiga buah full adder yang disambungkan secara berurutan dan masukan carry C0 diset = 0 seperti tampak pada berikut :

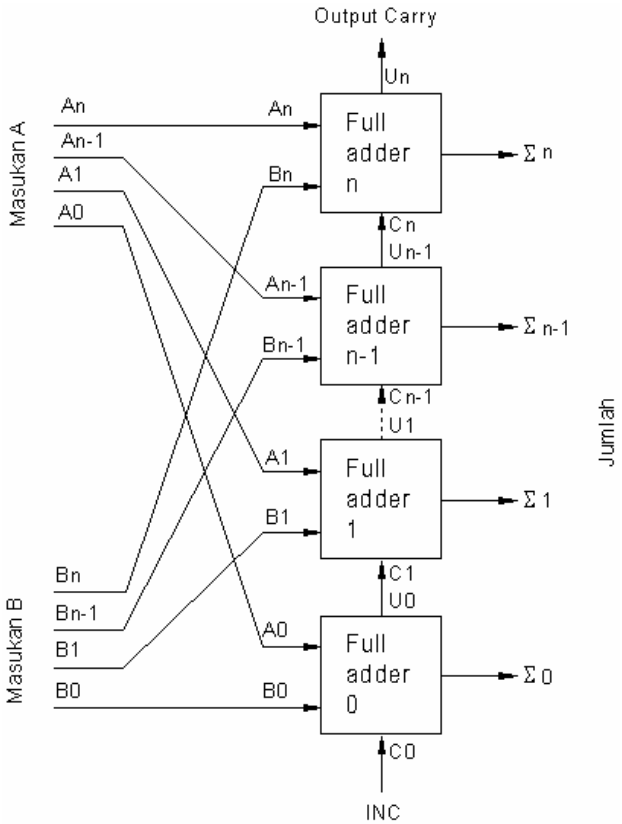


Gambar 6.10 Penjumlah biner 3-bit

Apabila masukan C0 kita set = 1 maka hasil penjumlahan akan bertambah 1. Dalam praktek penambahan satu ini sangat penting untuk bererapa aplikasi. Masukan carry C0 digunakan untuk masukan increment (INC) dan suatu rangkaian penjumlah digital yang dilengkapi dengan sebuah masukan incremen disebut ripple-carry adder.



Gambar 6.11 Diagram Blok Ripple-Carry Adder

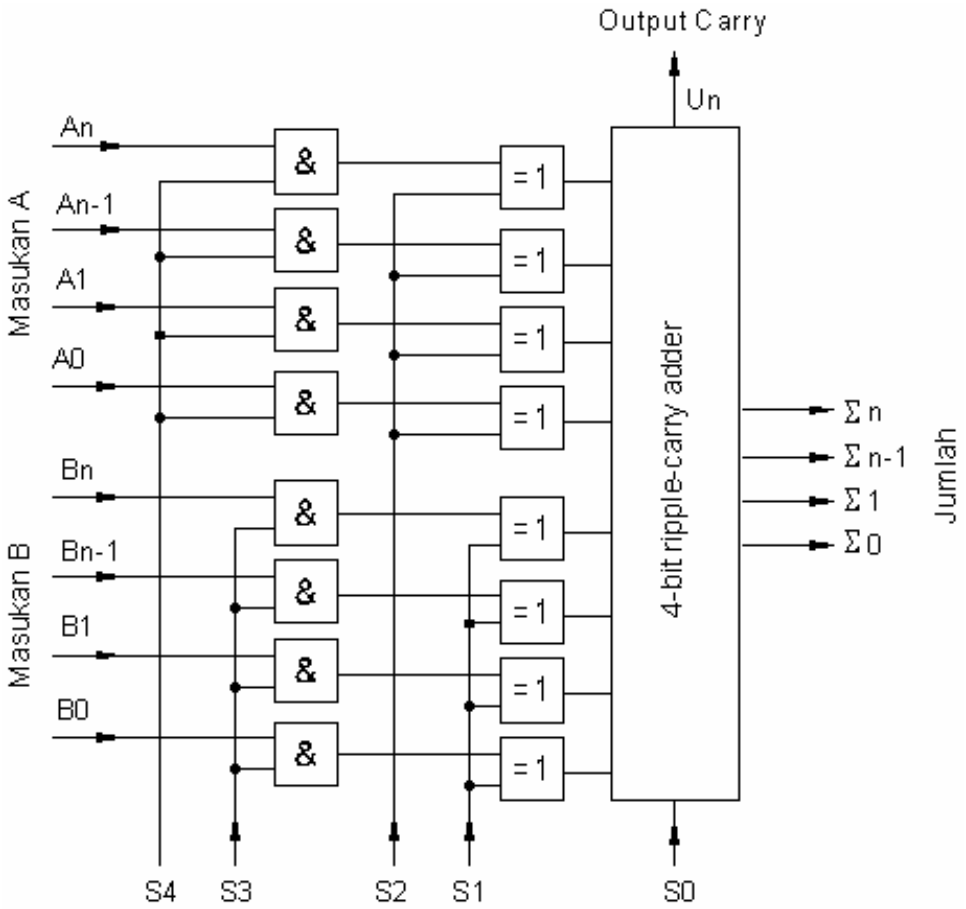


Gambar 6.12 Rangkaian Digital Ripple-Carry adder

6.2.2 Adder/Subtracter

Rangkaian penjumlah Gambar 3.6. dapat dikembangkan dengan menambahkan gerbang AND dan EXOR pada masukannya sehingga beberapa fungsi rangkaian berkembang tidak hanya sebagai penjumlah melainkan berfungsi pula sebagai rangkaian pengurang, sehingga rangkaian ini disebut adder/substractor.

Rangkaian adder/subtractor di atas memiliki 5 masukan kontrol S4 samapi S0 yang dipergunakan untuk memilih operasi. Tabel 3.1. memperlihatkan variasi masukan kontrol S4 sampai S0 dan fungsi keluaran.



Gambar 6.13 Rangkaian adder/subtractor 4-bit

Table 6.4 Tabel Fungsi *Adder/Subtracter*

S4	S3	S2	S1	S0	Fungsi Keluaran
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	-1
0	0	0	1	1	0
0	0	1	0	0	-1
0	0	1	0	1	0
0	0	1	1	0	-2
0	0	1	1	1	-1
0	1	0	0	0	B
0	1	0	0	1	B + 1
0	1	0	1	0	- B - 1 = $\overline{B}$
0	1	0	1	1	- B
0	1	1	0	0	B - 1
0	1	1	0	1	B
0	1	1	1	0	- B – 2 = $\overline{B}$ - 1
0	1	1	1	1	- B - 1 = $\overline{B}$
1	0	0	0	0	A
1	0	0	0	1	A + 1
1	0	0	1	0	A - 1
1	0	0	1	1	A
1	0	1	0	0	- A – 1 = $\overline{A}$
1	0	1	0	1	- A
1	0	1	1	0	- A – 2
1	0	1	1	1	- A – 1 = $\overline{A}$
1	1	0	0	0	A + B
1	1	0	0	1	A + B + 1
1	1	0	1	0	A – B – 1
1	1	0	1	1	A – B
1	1	1	0	0	B – A – 1
1	1	1	0	1	B – A
1	1	1	1	0	- A – B –2
1	1	1	1	1	- A – B - 1

Dari tabel di atas terdapat 32 kemungkinan fungsi yang dapat dioperasikan. Kita ambil salah satu contoh dari kemungkinan yang

diperlihtakan pada tabel diatas misalnya fungsi kontrol S4 sampai dengan S0 = 1 1 0 0 0 menghasilkan fungsi A + B.

Jika S3 dan S4 diset pada 0 maka masukan A dan B akan mati karena setiap gerbang AND tersambung pada setiap masukan A dan B sedangkan amsukan gerbang lainnya tersambung ke saklar masukan kontrol S3 untuk masukan A dan S2 untuk masukan B, sehingga ketika S3 = 0 maka apapun masukan A akan diset = o dana ketika S2 = 0 maka masukan B akan diset = 0. Dengan demikian fungsi kontrol S3 dan S2 adalah untuk meloloskan data masukan ke tahap berikutnya untuk diproses atau tidak oleng blok penjumlah.

Setelah keluar dari gerbang AND sebagai pelolos data, setiap keluaran AND tersambung pada gerbang XOR yang mana setiap gerbang XOR tersebut salah satu masukan lainnya tersambung pada saklar masukan kontrol S2 dan S1. S2 dipergunakan untukmengontrol masukan A dan S1 dipergunakan untuk mengontrol masukan B.

Jika S2 = 0 maka keluaran gerbang EXOR adalah A XOR 0 = A dan jika S2 = 1 maka keluaran gerbang EXOR adalah A XOR 1 = A. dengan demikian ketika S2 = 1 masukan A akan dibalik (komplemen satu).

Hal yang sama berlaku juga untuk masukan S1 yang mengontrol masukan B untuk fungsi komplemen satu.

Saklar S0 merupakan masukan carry untuk rangkaian penjumlah yang berfungsi sebagai masukan incremen (INC). S0 sangat diperlukan pada operasi pengurangan untuk mendapatkan komplemen dua.

Untuk fungsi kontrol S4 sampai dengan S0 = 0 0 0 1 0 , S1 =1 dan masukan kontrol yang lainnya adalah 0, ini berarti semua gerbang keluaran pada keempat gerbang XOR bagian atas adalah 0. Sementara 4 gerbang XOR bagian bawah menghasilkan 1 dan pada bagian adder, penjumlahan akan dlakukan seperti yang dibawah ini :

A		0 0 0 0
B	+	1 1 1 1
Keluaran		1 1 1 1

Hasil tersebut berarti sama dengan –1 pada aritmatika komplemen dua.
Ketika fungsi kontrol S4 sampai dengan S0 diset = 11011, fungsi keluaran adalah pengurangan A – B.

Penjelasan per scalar kontrol sebagai beikut :

S4 = 1 Data masukan A diloloskan
S3 = 1 Data masukan B diloloskan
S2 = 0 Data masukan A tidak dibalik (tidak di komplemen satu)
S1 = 1 Data masukan B dibalik (komplemen satu)
S0 = 1 Incremen 1 (+1)

Sehingga fungsi keluaran adalah

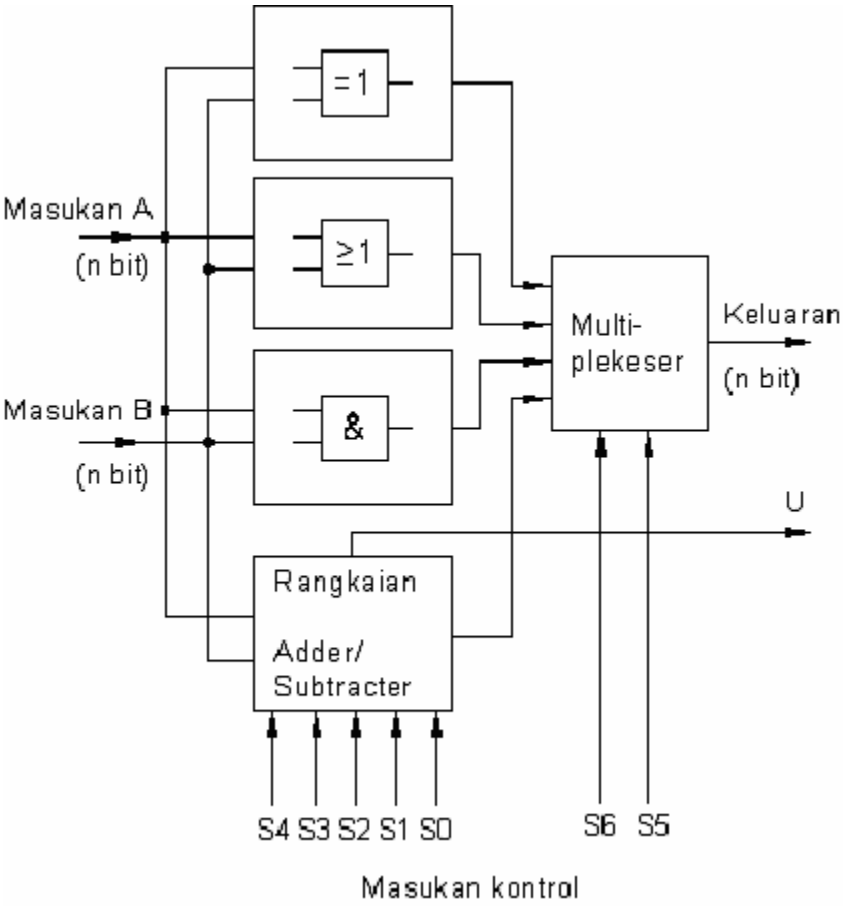
A + B + 1 = A - B

6.2.3 Arithmetic Logic Unit (ALU)

Agar mikroprosesor tidak hanya dapat melakukan operasi aritmatika tetapi juga dapat juga melakukan operasi fungsi logia, maka kita harus mengembangkan rangkaian adder/subtractor dengan menambahkan gerbang logika EXOR, OR dan AND serta sebuah multipekser.
Dengan adanya tambahan tiga gerbang logika tersebut sekarang operasi logika XOR, OR dan AND dapat dilakukan misalnya

A AND B
A OR B
A XOR B

Masukan kontrol S6 dan S5 adalah kontrol multiplekser yang dipergunakan untuk memilih operasi aritemetika atau logika. Jika S6 = 0 dan S5 = 0 operasi adalah fungsi aritmetika. Pada saat S6 dan S5 pada kondisi yang lain maka operasi adalah fungsi logika dan selama fungsi logika maka kontrol S4 sampai dengan S0 tidak berpengaruh karena kontrol S4 sampai dengan S0 adalah kontrol untuk operasi aritmetika.
Pada prinsipnya dengan kontrol sebanyak 7 bit (S6 sampai dengan S0) sehaarusnya terdapat 2^7 = 128 variasi fungsi tetapi tidak semua variasi tersebut diperlukan.



Gambar 6.14 Rangkaian ALU

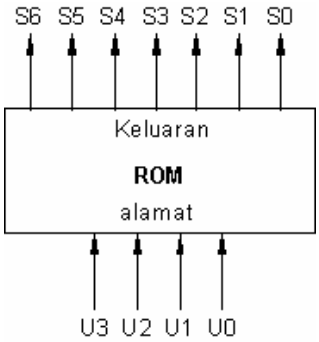
Perhatikan tabel fungsi adder/subtracter terdapat 32 fungsi dan terjadi pengulangan fungsi yang sama dan sebagian besar tidak begitu penting. Untuk itu kita harus membatasi fungsi yang penting saja dengan cara menggunakan ROM.

Didalam ROM dsimpan data-data kontrol untuk S6 sampai S0 pada alamat alamat tertentu. Pada rancangan ALU ini kita batasi fungsi yang disediakan adalah 13 fungsi dan dikodekan dalam 4 masukan kontrol saja yaitu U3 sampai U0.

Sesungguhnya dalam ROM U3 smapai U0 ini adalah jalur alamat sedangkan kode operasi adalah data pada suatu lokasi memory. Contoh untuk instruksi aritmetika A + B kode instruksi dalam table fungsi ALU adalah U3 = 0, U2 = 1, U1 = 1 dan U0 = 0, kalau kita cermati maka kode tersebut adalah alamat pada ROM 0101_2 sedangkan untuk operasi A + B,

masukan kontrol untuk C6 sampai C0 adalah 0011000 (S6=S5=0 dan S4 sampai dengan S0 lihat table fungsi adder/subtracter operasi A+B).
Dengan demikian kita dapat mengetahuii bahwa isi ROM pada alamat 0101_2 adalah 0011000_2.

Sekarang kita mengenal kode instruksi yang yang disimpan pada ROM dan tidak penting lagi untuk mengetahu kontro yang harus diberikan ke rangkaian yang sebenarnya.

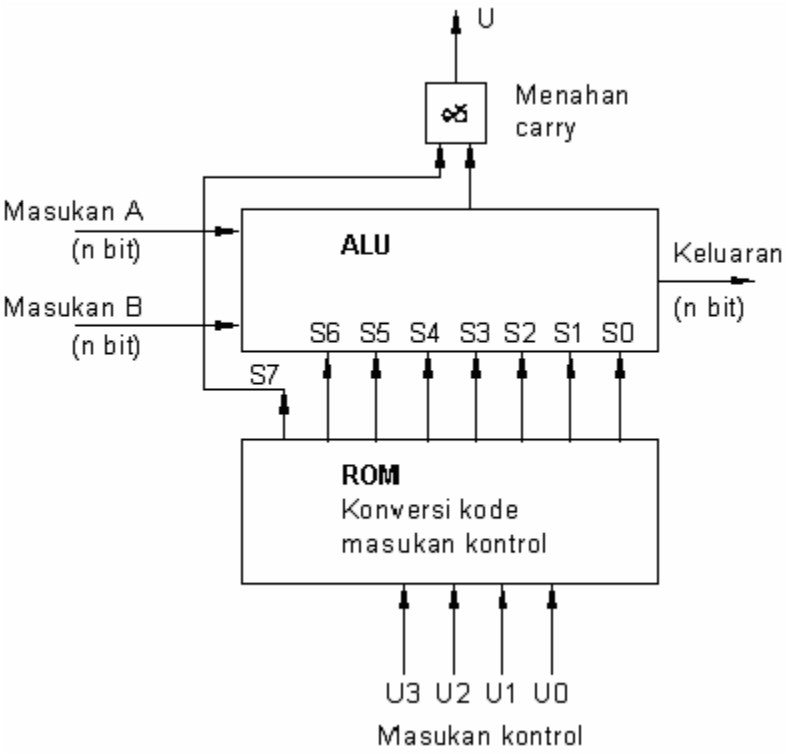


Gambar 6.15. ROM untuk mengkonversi kode instruksi ke masukan kontrol

Table 6.5 Tabel fungsi ALU

U3	U2	U1	U0	Fungsi keluaran
0	0	0	0	A
0	0	0	1	1
0	0	1	0	$\overline{A}$
0	0	1	1	B
0	1	0	0	0
0	1	0	1	A + 1
0	1	1	0	A - 1
0	1	1	1	A + B
1	0	0	0	A - B
1	0	0	1	A AND B
1	0	1	0	A OR B
1	0	1	1	A XOR B
1	1	0	0	- 1
1	1	0	1	
1	1	1	0	
1	1	1	1	

Dengan jumlah saluran kontrol 4 bit terdapat 16 kemungkinan fungsi yang termasuk 3 fungsi untuk pengembangan sistem nantinya.
Rangkaian ALU yang dilengkapi dengan konversi kode untuk masukan kontrol ditunjukkan dalam Gambar 6.11. berikut ini.

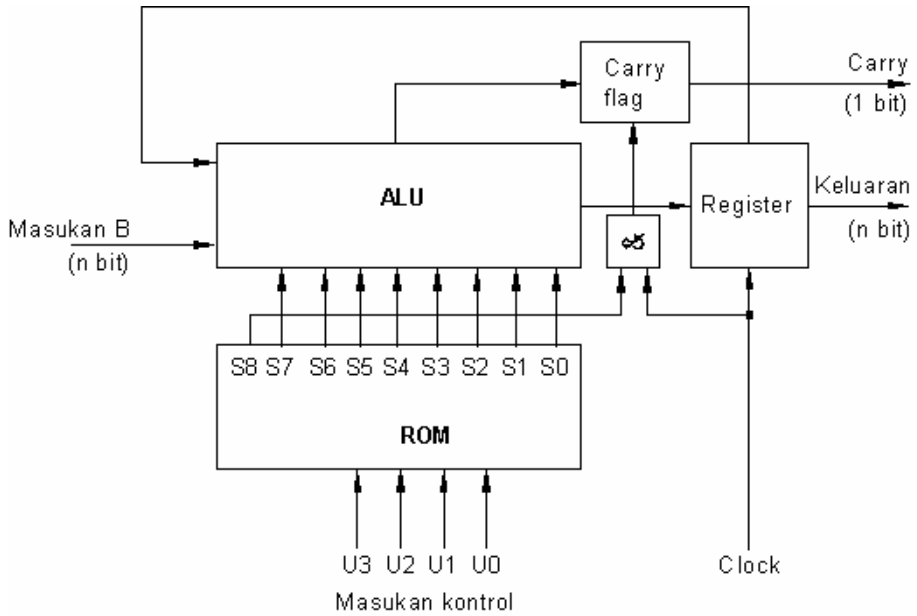


Gambar 6.16 ALU Dengan Konversi Kode

6.2.4. Accumulator (ACCU)

Accumulator (yang disingkat menjadi accu) adalah tingkatan selanjutnya dari ALU. Berikut adalah gambar accumulator dengan carry flag.
Terpisah dari fungsi yang telah ditunjukkan pada gambar3.11 rangkaian ini terdiri atas register dan carry flag sebagai komponen tambahan yang penting. Register menyediakan memory penyangga untuk hasil yang dikeluarkan nanti. Untuk mencapai tujuan ini salah satu masukan pada ALU disambungkan ke keluaran register yang juga merupakan keluaran hasil operasi (feedback). Informasi yang ada di masukan B sekarang dikombinasikan dengan keluaran yang dihasilkan register. Hasil yang dikeluarkan oleh ALU di simpan pada register oleh pulsa clock.
Pada tahap ini data yang ada pada register sebelumnya akan hilang. Untuk menghindari kemungkinan munculnya carry yang hanya sesaat saja maka keluaran carry ini disimpan pada suatu flag. Clock pada flag ini

berdasarkan bit tambahan S8 dalam ROM (fungsi gerbang AND pada Gambar 3.12). Hal ini hanya perlu bila carry ini dikeluarkan oleh fungsi ALU. Fungsi yang akan diproses oleh accumulator ini ditampilkan pada tabel 3.3. Untuk kombinasi U3 sampai dengan U0 = 0101 fungsi operasi keluaran adalah A+1. Karena adanya umpan balik untuk masukan A, maka besarnya A bergantung pada isi yang ada di register saat itu. Dengan instruksi A + 1 ini, isi accumulator ini akan bertambah 1 tiap satu clock.
Dengan kombinasi kontrol seperti ini accumulator disini akan berfungsi sebagai counter.
Jika counter dikehendaki sebagai counter down, maka isntruksi yang harus diberikan adalah dengan mengataur kombinasi U3 sampai dengan U0 = 0110 dan fungsi A -1 akan dilakukan.



Gambar 6.17 Accumulator dengan carry flag

Tabel 6.6 Tabel fungsi accumulator

U3	U2	U1	U0	Singkatan	Fungsi	Carry Flag
0	0	0	0	NOP	No operation	ya
0	0	0	1	SP1	Set accu = 1	ya
0	0	1	0	CMA	Complement accu	tidak
0	0	1	1	LDA	Load B into accu	tidak
0	1	0	0	CLA	Clear accu	tidak

0	1	0	1	INC	Increment accu	ya
0	1	1	0	DEC	Decrement accu	ya
0	1	1	1	ADD	Add B into accu	ya
1	0	0	0	SUB	Substract B from accu	ya
1	0	0	1	AND	Accu AND B into accu	ya
1	0	1	0	IOR	Accu OR B into accu	ya
1	0	1	1	XOR	Accu XOR B into accu	ya
1	1	0	0	SM1	Set accu = -1	ya
1	1	0	1			
1	1	1	0			
1	1	1	1			

Pada kolom carry flag dapat dilihat kapan carry flag ini keluar pada saat di clock. Pada banyak mikroprosesor flag ini juga di clock untuk operasi logika. Selama tidak ada carry yang dihasilkan oleh ALU, flag ini akan di-clear.

Jika suatu fungsi sangat rumit akan diproses oleh accumulator, maka fungsi ini haraus dibagi menjadi operasi-operasi yang lebih sederhana. Untuk operasi ini diperlukan beberapa siklus.

Sebagai contoh masalah perkalian untuk menghitung $3 \cdot B$ yang mana B adalah data masukan dari saklar. Selama tidak ada fungsi perkalian maka akan diperlukan berepa fungsi yang dasar yang sederhana.

Oleh karena itu kita harus membuat langkah (program) yang harus dilakukan untuk memecahkan masalah perkalian tersebut seperti ditunjukkan beikut ini :

Urutan langkah	Instruksi	Keterangan
1	LDA	Isi accumulator dengan data dari masukan B
2	ADD	Tambahkan isi accumulator dengan data dari masukan B
3	ADD	Tambahkan isi accumulator dengan data dari masukan B

Dalam prakteknya implementasi program di atas adalah sebagai berikut :

1. Siapkan data masukan B
2. Siapkan data kontrol LDA dengan mengatur U3 = 0, U2 = 0, U1 = 1 dan U0 = 1
3. Beri pulsa clock
4. Siapkan data kontrol ADD dengan mengatur U3 = 0, U2 = 1, U1 = 1 dan U0 = 1
5. Beri pulsa clock
6. Siapkan data kontrol ADD dengan mengatur U3 = 0, U2 = 1, U1 = 1 dan U0 = 1
7. Beri pulsa clock

Instruksi pertama LDA menyebabkan data masukan B masuk ke akumulator dengan pulsa clock. Instruksi kedua ADD setelah pulsa clock diberikan akan menghasilkan isi accu sebelumnya ditambah masukan B.

Sekarang B sudah ditambahkan pada isi Accu. B + B akan terbentuk. Dengan fungsi kontrol yang sama, pemberian pulsa clock selanjutnya dibutuhkan untuk menambah B sekali lagi untuk hasil B+B. Sehingga Accumulator akan menghasilkan nilai $3 \cdot B$

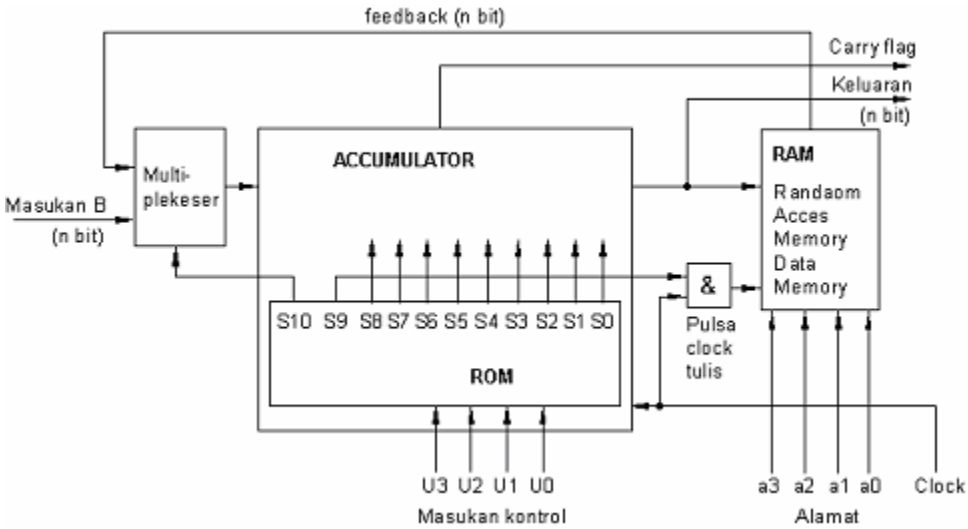
Dengan memilih kombinasi instruksi kontrol yang tepat, maka ekspresi yang sulit akan dapat dihitung.

6.2.5. Accumulator Dengan Memory Data

Agar supaya accumulator dapat diterapkan pada komputer, diperlukan kemampuan untuk menyimpan hasil operasi kemudian mengambil kembali untuk pemrosesan data selanjutnya. Kemampuan menyimpan data dan mengambil simpanan data sebelumnya dapat dilakukan oleh sebuah komponen tambahan yaitimemory data atau disebut RAM (Radom Acces Memory) yaitu suatu jenis memory yang dapat ditulisi maupun dibaca.
Input B dan output RAM dikontrol oleh multiplekser. Accumulator memiliki keluaran tidak hanya untuk keperluan diluar sistim tetapi juga memiliki keluaran yang digunakan sebagai masukan dari RAM. Dengan demikian memungkinkan untuk mengeluarkan isi akumulator ke memory data atau memasukkan data dari memory melalui multiplekser menuju ke accumulator.
Untuk itu perlu penambahan unit kontrol 2 bit yaitu S9 dan S10 untuk melaksanakan instruksi baca tulis memory data. Selanjutnya kode instruksi pada ROM juga berubah dengan tambahan fungsi seperti yang ditunjukkan oleh tabel 3.4.

Keluaran ROM tambahan S10 disediakan sebagai switching multiplekser. Pada kondisi normal multiplekser menghubungkan data memory dengan masukan accumulator.

Dengan kombinasi bit kontrol U3 sampai dengan U0 = 1101, multiplekser membuat masukan B terhubung secara langsung ke accumulator, dimana data akan disimpan sementara oleh register.
Pada saat kombinasi bit kontrol U3 sampai dengan U0 = 0011, multiplekser membuat masukan accumulator terhubung dengan data memory melalui multiplekser.
Bit tambahan S9 pada keluaran ROM digunakan menyiapkan pulsa clock untuk data memory pada saat instruksi menyimpan data ke memory dengan kombinasi bit kontrol U3 ampai dengan U0 = 1110. Untuk fungsi kontrol ini, isi akumuluator akan ditulis ke data memory.



Gambar 6.18 Accumulator dengan memory data

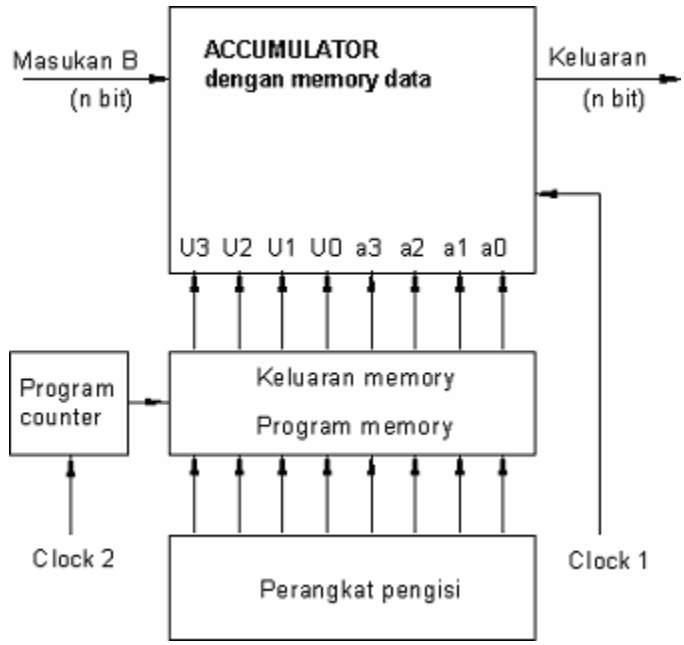
Tabel 6.7 Tabel fungsi accumulator dengan memory data

U3	U2	U1	U0	a3	a2	a1	a0	Singkatan	Fungsi	Carry Flag
0	0	0	0	x	x	x	x	NOP	No operation	ya
0	0	0	1	x	x	x	x	SP!	Set accu = 1	ya
0	0	1	0	x	x	x	x	CMA	Complement accu	tidak
0	0	1	1	a	a	a	a	LDA	Load contents address a a a a into accu	tidak

0	1	0	0	x	x	x	x	CLA	Clear accu	tidak
0	1	0	1	x	x	x	x	INC	Increment accu	ya
0	1	1	0	x	x	x	x	DEC	Decrement accu	ya
0	1	1	1	a	a	a	a	ADD	Add contents address a a a a into accu	ya
1	0	0	0	a	a	a	a	SUB	Substract contents address a a a a from accu	ya
1	0	0	1	a	a	a	a	AND	Accu AND contents address a a a a	ya
1	0	1	0	a	a	a	a	IOR	Accu OR contents address a a a a	ya
1	0	1	1	a	a	a	a	XOR	Accu XOR contents address a a a a	ya
1	1	0	0	x	x	x	x	SM!	Set accu = -1	ya
1	1	0	1	x	x	x	x	INP	Load B inputs into accu	tidak
1	1	1	0	a	a	a	a	STA	Store accu into address a a a a	tidak
1	1	1	1	x	x	x	x			

Keterangan : a a a a = alamat memory data
x x x x = tidak dipedulikan

6.2.6 Komputer Sederhana



Gambar 6.19 Aplikasi program counter dan program memory

Langkah berikutnya untuk membangun sebuah komputer lengkap adalah dengan menempatkan pola urutan kontrol ke dalam sebuah memory program yang akhirnya memungkinkan operasi secara otomatis dapat dilakukan.

Urutan kontrol word atau instruksi (program) yang akan diproses disimpan pada memory program. Urutan program atau instruksi di sini sangat penting. Instruksi yang tersimpan dalam memory harus disimpan secara berurutan sesuai dengan kenaikan alamat yang cesara otomatis akan dilakukan oleh program counter.

Pada intinya bahwa setiap program yang akan dijalankan oleh suatu komputer maka program tersebut harus diisikan ke dalam memory. Ada dua kemungkinan memasukkan program ke dalam memori. Kemungkinan pertama jika program tersebut bersifat tetap dan tidak ada perubahan lagi maka program sejenis ini sebaiknnya disimpan pada ROM (Read Only Memory).

Kemungkinan kedua apabila tersebut bersifat sementara atau masih memungkinkan adanya perubahan perubahan maka sebaiknya program tersebut disimpan pada RAM (Random Acces Memory). Untuk keperluan ini diperlukan perangkat tambahan untuk mengisikan program ke dalam memory.

Meskipun beberapa mikroprosesor memisahkan memory tempat menyimpan data dengan memory tempat menyimpan program instruksi, tetapi pada dasrnya suatu memory dapat menyimpan keduanya dalam satu jenis memory bersama. Alternatif lain dalam penerapannya adalah banyak dilakukan dengan menggabungkan ROM dan RAM.

Penggunaan memory bersama (untuk program dan data sekaligus) memiliki keuntungan sebagaia berikut :

Lebih fleksibel, tergantung dari masalahnya, besarnya memory yang diperlukan untuk data atau untuk program dapat disesuaikan oleh si pemakai apakah program membutuhkan memori lebih banyak atau sebaliknya data memerlukan memory yang lebih besar dari pada program.
Lebih sedikit sambungan, tidak perlu menyediakan sambungan banyak jenis memory.

Langkah-langkah program dapat juga diproses sebagai data.

Sedangkan kekurangan dari single memory ini adalah dibutuhkannya rangkaian yang lebih dalam mikroprosesor yang memungkinkan

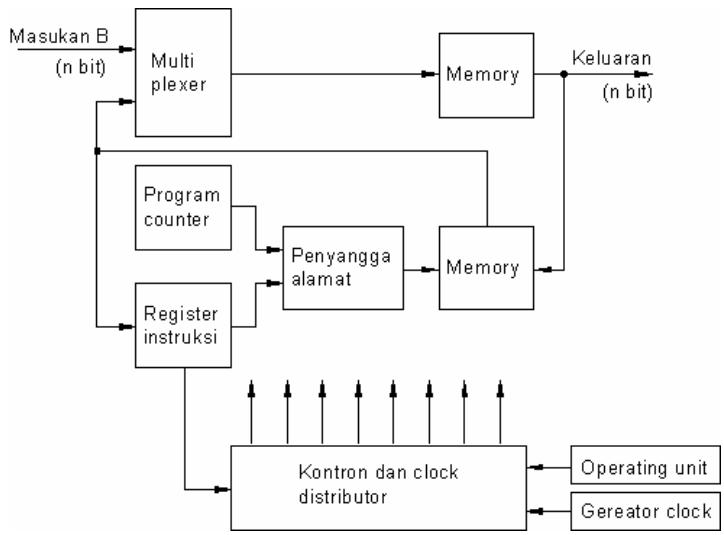
mengambil dan menyimpan data dari program counter atau dari saluran alamat dari suatu instruksi yang semuanya itu harus memlaui suatu multiplekser yang terkontrol. Implementasi yang mungkin dari sistim komputer tersebut ditujukkan pada Gambar 3.15.

Karena program dalam komputer dapat berjalan secara otomatis, maka diperlukan instruksi HALT untuk menghentikan eksekusi pada saat akhir program. Tanpa instruksi ini maka program akan berjalan tanpa henti. Pola kontrol instruksi HALT adalah dengan mengatur masukan kontrol U3 sampai U0 = 1111.

Control unit dan clock generator diperlukan untuk mengatur langkah-langkah pengoperasian komputer yang bekerja untuk menghasilkan urutan langkah – langkah yang harus dikerjakan oleh tiap – tiap blok bagian komputer.

Program counter memberikan alamat awal program yang akan dijalankan dan sekaligus menaikan satu alamat berikutnya setelah satu instrukasi dijalankan. Isi program counter dikirim ke alamat memory. Instruksi yang ada dimemory dijalankan dan disangga oleh register instruksi.

Instruksi umumnya terdiri dari kode operasi (opcode) dan alamat, kontrol unit akan mengerti apakah suatu instruksi memerlukan mengakses alamat memory atau tidak selanjutnya sinyal kontrol akan didistribusi ke blok-blok yang berkepentingan.



Gambar 6.20 Komputer sederhana dengan common data dan program memory

6.2.7 Komputer Lengkap

Setiap mikroprosesor selalu dilengkapi dengan instruksi yang dipergunakan untuk mengontrol aliran program. Instruksi penting dalam komputer adalah instruksi loncat (JUMP). Instruksi JUMP adalah memberikan nilai terntu ke program counter. Pada kondisi normal program counter akan menghitung naik satu demi satu, tetapi dengan adanya masukan nilai tertentu ke dalam program counter dapat melakukan perhitungan tidak lagi naik satu demi satu tetapi dapat langsung loncat ke angka tertentu baik pada arah naik maupun mundur. Dengan demikian memungkinkan suatu program diulang-ulang. Instruksi JUMP ini banyak macamnya, lompat dengan syarat atau loncat tanpa syarat ke alamat tertentu pada RAM.

Loncat dengan syarat biasanya syaratnya adalah flag hasil operasi sebelumnya. Beberapa flag dalam sistim mikroprosesor adalah :

Carry flag atau C flag
Flag ini mengindikasikan terjadinya overflow hasil operasi melebihi batas bilangan aritmetika integer

Arithmetic arry flag atau V flag
Flag ini mengindikasikan terjadinya overflow hasil operasi melebihi batas bilangan dalam artimetika komplemen dua

Zero flag atau Z flag
Flag ini mengindikasikan hasil operasi sama dengan nol

Negative flag atau N flag
Flag ini mengindikasikan hasil operasi sama dengan negatip

Parity flag atau P flag
Flag ini mengindikasikan hasil operasi merupakan bilangan ganjil atau genap.

VII. MIKROPROSESOR Z-80

Perencanaan Jalannya Program

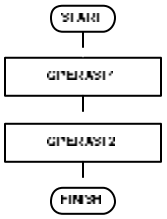
Untuk memberikan kejelasan, setelah menganalisa masalah yang terjadi dengan tepat, maka dibuat sedikit atau banyak detail dari jalannya program yang telah direncanakan. Dengan ini kita dapat mengurangi kesukaran pemrograman, walaupun demikian pada saat kita membuat kesalahan logika pada suatu jalannya program, sering memberikan pelajaran yang baik untuk mendapatkan pemahaman dan jalan keluar lebih baik.
Sebuah perencanaan jalannya program menggambarkan urut-urutan proses atau jalannya fungsi sebuah program, perencanaan ini juga menunjukkan urutan yang mana atau syarat yang mana dari pelaksanaan kejadian sebelumnya, yang akan diulang.
Pada perencanaan jalannya program dapat dituangkan dalam bentuk aliran program (Flow Chart). Hal yang penting untuk menghindari banyak kesalahan pada flow chart, bahwa masing-masing blok mempunyai sebuah masukan dan sebuah keluaran.

Benar Salah

Gambar 7.01 Blok masukan-keluaran diagram alir

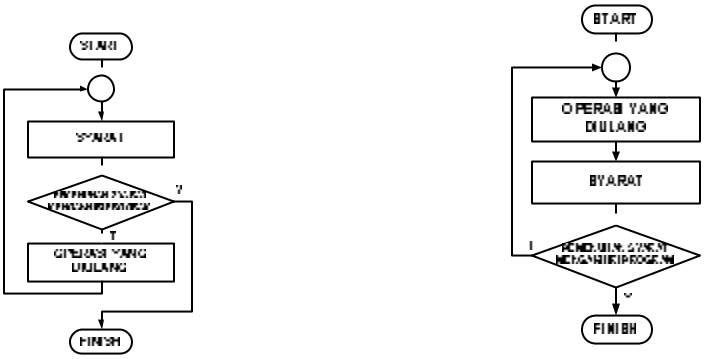
Pada struktur program ada 3 bentuk yang boleh digunakan sebagai bangun program.

1. Struktur linier (berurutan/sequens)



2. Struktur Pengulangan (Loop)

Pengulangan program bagian yang dapat dijalankan berulang-ulang disebut sebagai tubuh dari pengulangan. Pada masing-masing pengulangan, minimal ada satu syarat loncat dan pada setiap pelaksanaan pengulangan, syarat loncat tersebut harus diuji.



Gambar 7.03 *Flowchart* Struktur Pengulangan

Hasil pengujian akan mempengaruhi jalannya program bagian / tubuh pengulangan atau jalannya perintah berikutnya.

Pengulangan pada umumnya terdiri dari beberapa bagian berikut ini :

Inisialisasi

- Pengisian register atau lokasi memori yang dipakai pada pengulangan. Contoh : alamat awal penyimpan data, nilai awal dari suatu penghitung.
- Pengosongan register atau flag. Bagian ini hanya dilaksanakan satu kali

Tubuh pengulangan

- Hal ini tediri dari program yang harus berulang-ulang dilaksanakan. Contoh : Perhitungan, memeriksa kondisi masukan keluaran.
- Rangka pengulangan ini dapat terdiri dari struktur pengulangan atau alternatip.

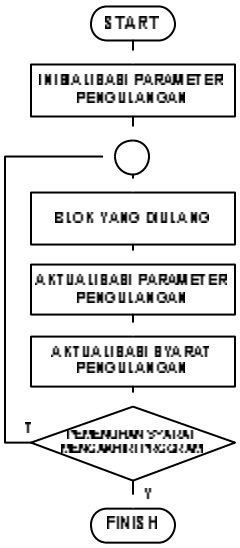
Aktualisasi dari Parameter Pengulangan

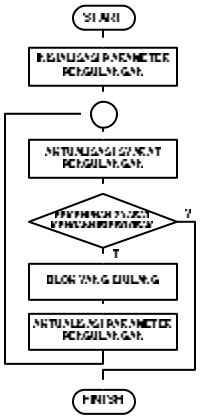
- Pada pemprosesan blok data, contoh penunjuk alamat pada setiap pelaksanaan pengulangan harus dinaikkan 1 atau diturunkan 1.
- Aktualisasi syarat untuk mengakhiri pengulangan untuk digunakan :
  - Penghitung, yaitu setiap pelaksanaan pengulangan isi register atau lokasi memori dinaikan atau diturunkan. Pada pencapaian suatu nilai tertentu maka pengulangan akan berakhir.
  - Juga pada pengujian sebuah hasil (perhitungan bila sesuatu nilai tercapai, lebih besar atau lebih kecil maka pengulangan berakhir.
  - Sebuah kemungkinan lain adalah pengujian bit kontrol dari isi register atau mencari, pengulangan dapat dipengaruhi melalui hasil dari jalannya program atau melalui pembacaan blok masukan keluaran.

Keputusan untuk mengakhiri pengulangan

Kriteria keputusan selalu melihat pada kondisi bit flag.

a. Pengujian Pengulangan pada akhir struktur pengulangan (Repeat Until). Kekurangan pada instruksi ini, bahwa bagian yang diulang akan langsung berjalan pada saat masuk ke dalam pengulangan dan pengulangan berikutnya tergantung dari pemenuhan syarat.





Penutup Pengulangan

Di sini hasil yang didapat dari perhitungan pada saat pengulangan disimpan atau isi asli/awal dari register yang dipakai pada saat pengulangan diisi kembali pada register yang bersangkutan

Contoh 1

Sebuah Mikroprosessor harus menambah 12 bilangan biner (panjang 8 bit) yang berada pada blok data, alamat awal blok data diberi simbol DATA. Hasil akhir penjumlahan diletakkan pada register HL. Alamat awal Mikroprosessor : 0900H, alamat DATA ; 0A00H.

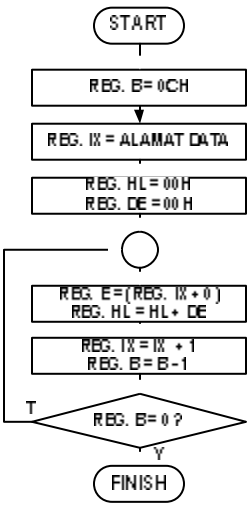
Pemecahan :

- Penjumlahan harus terjadi pada tubuh pengulangan, jumlah pengulangan harus dihitung pada register B.
- Karena penghitung, untuk memulai pengulangan tidak pernah diisi dengan 0, maka dipergunakan struktur repeat - until.
- Untuk blok data, diperlukan 1 register alamat awal data, register IX.
- Karena hasil dapat lebih besar dari 8 bit, maka Register HL dipakai sebagai tempat penghitung dan hasil.
- Operan tunggal / data 8 bit dapat diisi ke register E dan untuk kemudian ditambahkan dengan isi register HL untuk mendapatkan hasil sementara pada setiap pengulangan.

Parameter Masukan : alamat awal dari blok data (DATA).

Parameter Keluaran : HL berisikan hasil dari program

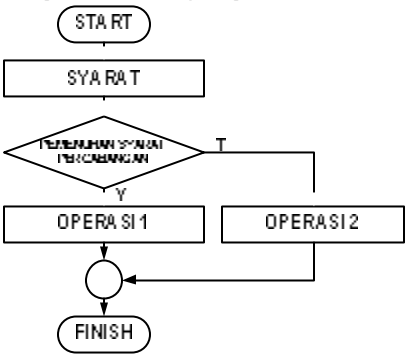
Register yang berubah : B, DE, HL, IX



Program :

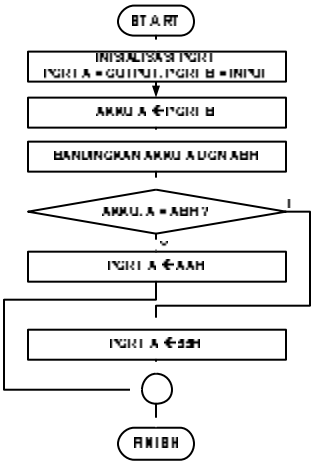
Tanda	Alamat (HEX)	Kode Operasi (HEX)	Mnemonik
	0900	06 0C	LD B, 0CH
	0902	DD 2A 00 0A	LD IX,DATA
	0906	21 00 00	LD HL, 0000H
	0909	11 00 00	LD DE, 0000H
LOOP	090C	DD 5E 00	LD E,(IX+0)
	090F	19	ADD HL, DE
	0910	DD 23	INC IX
	0912	05	DEC B
	0913	C2 0C 09	JP NZ, LOOP
	0916	FF	HALT

3. Struktur Keputusan (Percabangan)



Gambar 7.07 Flowchart Percabangan

Struktur ini terdiri dari sebuah blok pengontrol, yang telah ditentukan dan akan menjalankan alternatip bila syarat tertentu terpenuhi.
Contoh :



Ikhtisar Arsitektur Mikroprosessor

Pada dasarnya mikroprosesor adalah terdiri tiga bagian pokok yang saling bekerja sama antara yang satu bagian pokok yang saling bekerja sama antara yang satu dengan yang lainnya.

1. PENGONTROL

1.1. Register Perintah

Register perintah diisi langsung dari bus data sistem melalui bus data internal. Pada informasi 8 bit yang dibawah ke register ini adalah selalu menunjukkan suatu kode operasi dari sebuah perintah.

1.2. Pendekoder Perintah

Masing-masing bit dalam register perintah di uji / di periksa keadaan tegangannya (H atau L) oleh pendekoder perintah . Dengan demikian hal tersebut dapat dipastikan bahwa informasi yang disimpan dalam register perintah adalah merupakan suatu kode operasi tertentu.

1.3. Pengontrol Waktu dan Aliran (Pengontrol Waktu dan Aliran)

Unit ini berfungsi mengkoordinasikan antara jalannya sinyal di dalam dan di luar mikroprossesor dengan waktu. Unit pengontrol ini menyimpan informasi internal mikroprossesor yang berasal dari pendekoder perintah dan dari luar unit sistem. Sinyal yang di terima dari luar adalah sinyal detak (clock), sinyal control (WR,RD) dan sinyal penawaran (Riset, int) pengontrol waktu dan logik memberikan informasi balik pada unit sistem seperti sinyal tulis diberikan ke unit sistem menunjukkan bahwa pada unit ini akan di tulis sebuah data.

Keseluruhan dari sinyal masuk dan keluar pada unit pengontrol waktu dan logika ini disebut bus kontrol.

2. PENYIMPAN

Prinsip dari Mekanisme Penyimpan dari sebuah Mikroprossesor

Multiplexer		
	A	F
	B	C
	D	E
	H	L
	Penghitung perintah (PC)	
	Penunjuk Stack (Stack Printer)	
	Penyimpan sinyal alamat	

Mekanisme Penyimpan dari Z 80 dibagi dalam 6 kelompok fungsi

2.1.Multi plexer / Pemilih Register

Melalui multiplexer 1 pemilih register, lokasi memori dalam blok register yang dipilih dapat di tulis atau di baca.

2.2.Register Sementara A - F

Hal ini mengenai dua register 8 bit , yang dapat dipakai sebagai register tunggal (8 bit) atau dipakai sebagai register pasangan (16 bit) untuk proses internal Mikroprossesor.

Register A - F adalah sama dengan penghitung data dari penghitung sederhana.

Dengan kata lain, dalam register A - F , sebagai contoh : bagian alamat 16 bit dari sebuah perintah disimpan untuk sementara.

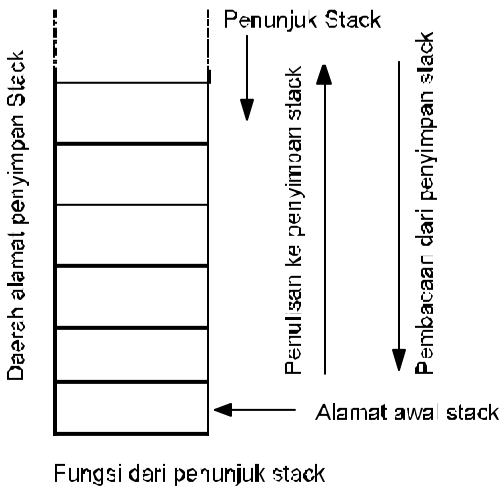
2.3.Register pasangan BC, DE, HL

Register ini dalam program dipakai sebagai register tunggal atau sebagai register pasangan. Bila dipakai sebagai register tunggal maka dia dapat dipakai sebagai penyimpan 8 bit . Bila dipakai sebagai register pasangan, dia dapat menyimpan 16 bit , sebagai contoh alamat lokasi memori 16 bit. Dalam mikroprosessor tersedia perintah khusus untuk register 16 bit ini.

2.4.Penunjuk Strack (Strack Printer)

Melalui program adalah mungkin untuk menjelaskan proses penulisan/pembacaan data ke/dari alamat stack yang telah di tentukan.

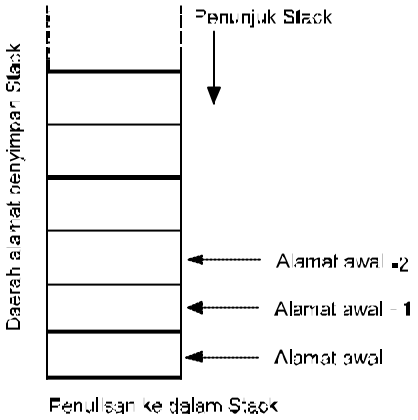
Alamat awal dari stack diisi ke penunjuk stack melalui sebuah perintah khusus.



Gambar 7.9 Stack Pointer

Bekerja dengan Stack.
Bila sebuah data dari mikroprosessor ditulis ke dalam stack, maka pertama adalah isi dari penunjuk stack dikurangi 1 dan data tersebut di tulis pada alamat ini (alamat awal stack -1), kemudian penunjuk stack dikurangi 1, sehingga data berikutnya ditulis pada alamat awal stack -2.

Proses ini terus berlangsung pada setiap penulisan data ke dalam stack. Penunjuk ini terus berlangsung pada setiap penulisan data ke dalam stack, penunjuk stack selalu menunjuk pada alamat lokasi stack yang ditulis terakhir.



Gambar 7.10 Penunjukan Alamat Stack

Pada pembacaan sebuah data dari stack, pertama ini dari alamat stack yang aktif saat itu (alamat awal stack -2) di baca dan kemudian penunjuk printer di tambah 1.
Kemudian di penunjuk stack terisi alamat awal stack -1 dibaca. Jadi data yang terakhir ditulis pada stack akan di baca pertama pada saat pembacaannya.
Jadi proses pembacaan pada saat stack digambarkan sebagai LIFO (Last In First Out)

2.5. Penghitung Perintah

Dalam penghitung perintah terdiri dari alamat masing - masing data yang dibaca sebagai alamat penyimpan program berikutnya. Data yang disimpan dalam penyimpan program selalu adalah kode operasi (up - code) , perintah dan data (sebagai contoh bagian alamatnya) Penghitung perintah mempunyai tugas untuk selalu meletakkan mikroprosessor pada posisinya yang benar pada jalannya program.

2.6. Penyimpan Sinyal Alamat (Adress Catch)

Bila data dari blok register dihubungkan ke bus alamat, maka selanjutnya data ini disimpan sementara dalam penyimpan sinyal alamat.

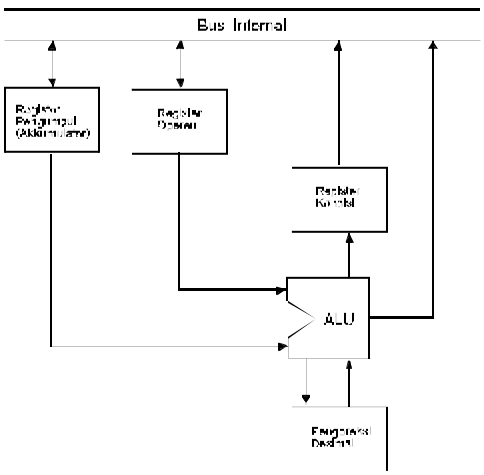
Sebagai contoh mikroprosessor mengakses stack, maka isi dari penunjuk stack di isi dalam penyimpan sinyal alamat. Pengurangan isi dari penunjuk stack pada proses penulisan dalam stack atau penambahan isi penunjuk stack pada proses pembacaan dari stack terjadi melalui penghitung naik/turun.

Bila mikroprosessor mengakses penyimpan program , maka isi dari penghitung perintah diisi ke dalam penyimpan sinyal alamat. Pembentukan alamat dari instruksi yang akan dilaksanakan berikutnya (penambahan isi penghitung perintah) terjadi melalui penghitung naik.

Bila alamat yang dibentuk dengan register pasangan HL, DE, BC, W2, penyimpanan sementara dalam penyimpan sinyal alamat terjadi dalam cara yang serupa.

3. OPERASI

Prinsip Mekanisme Operasi Ssebuah Mikroprosessor



Gambar 7.11 Mekanisme Operasi Mikroprosessor

Mekanisme Operasi Z 80 dibagi dalam 5 klompok fungsi :

3.1.Unit Aritmatik Logika

ALU melaksanakan semua operasi aritmatik dan logika

3.2.Register Sementara (Register Operan)

dan

3.3.Akkumulator

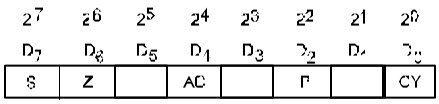
Operasi Aritmatik dan Logik selalu dijalankan dengan operan-operan pertama disimpan sementara dalam akkumulator operan ke dua disimpan sementara dalam penyimpan sementara (register sementara)

Kedua operan dijalankan pada operasi yang ada di akkumulator. ALU mengisi hasil operasi ke akkumulator.

3.4. Register kondisi (PSW = Program Stakes Word)

Dalam register kondisi 8 bit terdiri dari 5 flip-flop syarat, yang diset atau di reset tergantung dari hasil operasi aritmatik atau logik dari ALU.

Flag :



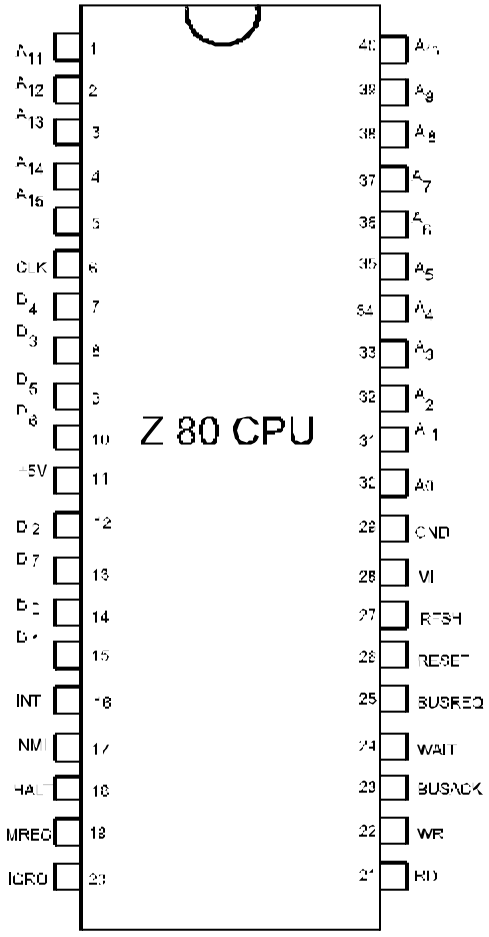
5 flag dalam unit sentral dari Z 80 adalah :

1. Bit DQ (posisi 2^1) adalah Flag carry
2. Bit D2 (posisi 2^2) adalah Flag parity
3. Bit D4 (posisi 2^4) adalah Flag carry pembantu
4. Bit D6 (posisi 2^6) adalah Flag zero
5. Bit D7 (posisi 2^7) adalah Flag tanda

Dalam bit D1, D3, dan D5 tidak terdapat informasi mereka di abaikan.

3.5. Pengontrol Desimal

Dengan cara ini untuk merubah hasil biner dari perintah penjumlahan ke dalam bilangan BCD (Bilangan desimal yang dikodekan secara binner)



Gambar 7.12 Konfigurasi Mikroprosessor Z 80

Ao ... A15	1 ... 5	Out	Tristate output, address bus dapat

	30 ... 40		menentukan alamat memori 64 KByte dan 8 bit terendah untuk menentukan alamat I/O (lebih dari 256 peralatan I/O dalam proses penukaran data). Untuk kebutuhan pengalamatan masukan dan keluaran (I/O) dibutuhkan 8 bit rendah dari CPU (A0 ... A7). Sedangkan untuk pengalamatan isi akumulator dibutuhkan 8 bit tinggi (A8 ... A15). Pada pengalamatan port juga menggunakan sinyal dari alamat A8 ... A15.
Do ... D7	7,8,9, 10,12,13 ,14,15	Inp /Out	Tristate input/output, merupakan 8 bit data bus dua arah dan berfungsi untuk melayani proses transfer data.
INT	16	Inp	Input aktip berlogika 0, interup ini dihasilkan oleh peralatan I/O. Jika CPU menerima interup INT maka signal IORQ selama waktu MI akan dikeluarkan CPU pada awal siklus instruksi berikutnya.
NMI	17	Inp	Input triger /negatip, mempunyai prioritas lebih tinggi dari INT dan signal ini akan menempatkan PC pada alamat 0066 H dan secara otomatis menyimpan isi PC pada stack sehingga setelah terjadi interupsi ini pemrogram dapat mengalihkan ke proses program sebelum diinterup.
HALT	18	Out	- Signal LOW pada HALT memberi tahukan bahwa CPU telah melaksanakan instruksi HALT dan sekarang menunggu Interupt. Selama keadaan HALT, CPU menyelesaikan instruksi NOP untuk mempertahankan refresh. **Nop = No Operation** **HALT = Penghentian**
MREQ	19	Out	Tristate output aktip dengan logika 0, untuk melayani permintaan proses transfer data yang menggunakan memori.
IORQ	20	Out	Tristate output aktip dengan logika 0, untuk melayani permintaan proses transfer data yang menggunakan I / O
RD	21	Out	Tristate output aktip dengan logika 0, merupakan signal yang dikeluarkan oleh CPU jika ingin membaca data baik dari memori maupun dari I / O

WR	22	Out	Tristate output aktip dengan logika 0, merupakan signal yang dikeluarkan oleh CPU jika ingin menulis data baik dari memori maupun dari I / O
BUSAK	23	Out	Output aktip berlogika 0, signal ini memberikan informasi kepada peralatan luar CPU bahwa Adress Bus, data bus dan tristate output signal kendali pada keadaan impedansi tinggi serta siap untuk dikendalikan oleh peralatan luar
WAIT	24	Out	Input aktip berlogika 0, memberikan signal bahwa address memori atau I / O tidak siap untuk proses data transfer dan CPU akan aktip kembali jika signal wait aktip.
BUSRQ	25	Inp	Input aktip berlogika 0, signal ini meminta CPU agar address bus, data bus dan tristate output signal kendali pada keadaan impedansi tinggi sehingga memungkinkan peralatan lain dapat mengendalikan bus - bus tersebut.
RESET	26	Inp	Input aktip berlogika 0, signal ini menempatkan isi PC = 00H, register I = 00 H, register R = 00 H dan Interupt Mode = 0. Selama waktu reset address bus dan data bus mempunyai impedansi tinggi dan output signal kendali pada keadaan tidak aktip.
M1	27	Out	Output aktip dengan logika 0, memberikan signal indikasi pelaksanaan op code instruksi selama satu siklus mesin, untuk 2 byte op code akan dihasilkan signal setiap satu siklus
RFSH	28	Out	Output aktip berlogika 0, menunjukkan bahwa 7 bit terendah dari address bus berisi refresh addres memori dinamis dan bersama signal MREQ untuk membaca memori dinamis.

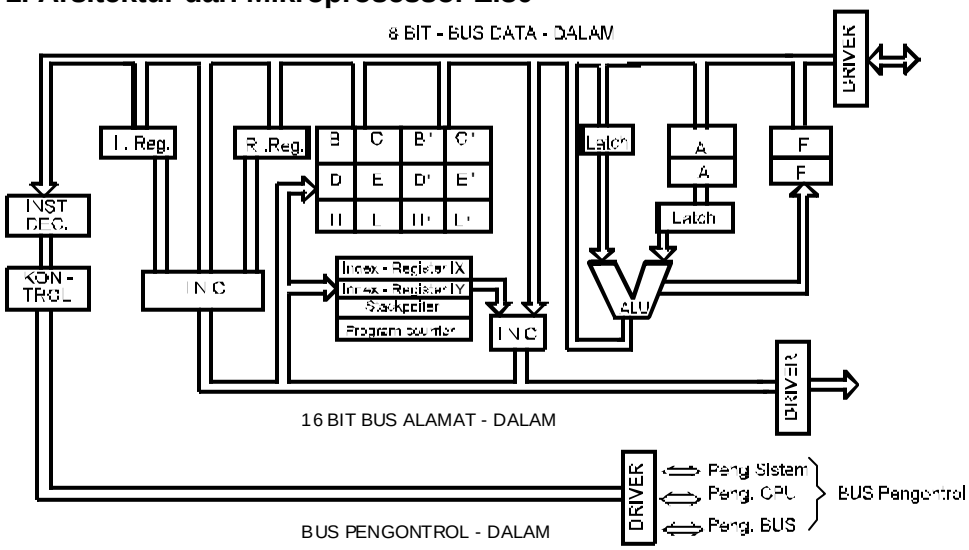
CPU - Struktur Bus

Supaya memori, peranti I/0 dan bagian lain dari suatu komputer dapat dibuat saling hubungan , maka biasanya dipakai suatu BUS. BUS banyak dipakai dalam mini komputer dan mikrokomputer, bahkan dalam sistim komputer besar, untuk modul - modul dengan aliran data yang berlebihan.
Seringkali perangkat - perangkat komputer yang menghubungkan dengan bus harus memakai saluran - saluran data secara bersama - sama, untuk itu dipergunakan penggerak tiga keadaan (3 STATE) , yang merupakan komponen dasar dari BUS (lihat gambar 1)
Bus dapat dibagi menjadi tiga bagian dalam menstransfer data / instruktusi yaitu :

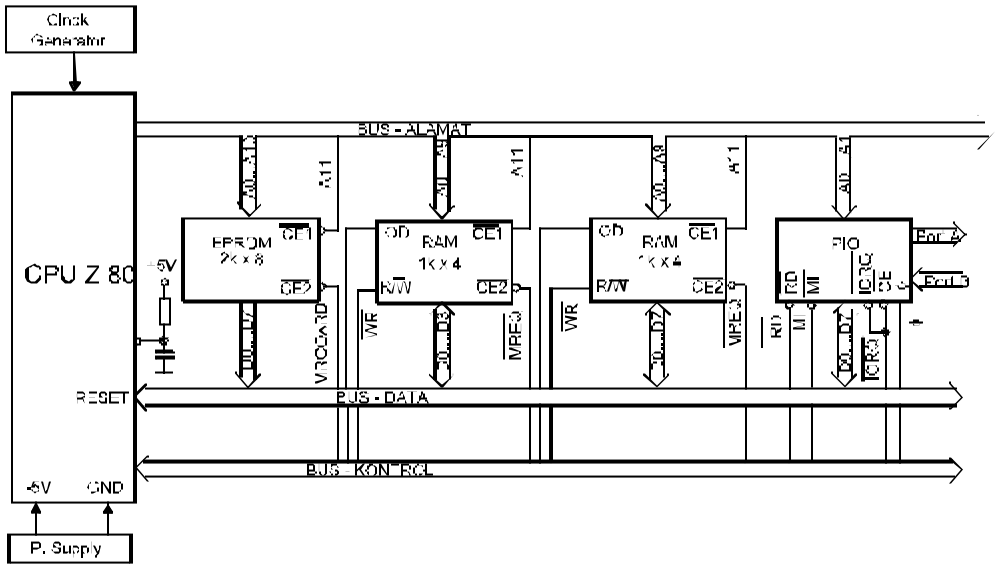
a. BUS - Data (DATA - BUS)
b. BUS - Alamat (ADDRESS - BUS)
c. BUS - Kontrol (CONTROL - BUS)

MIKROPROSESSOR Z.80 DAN 8080

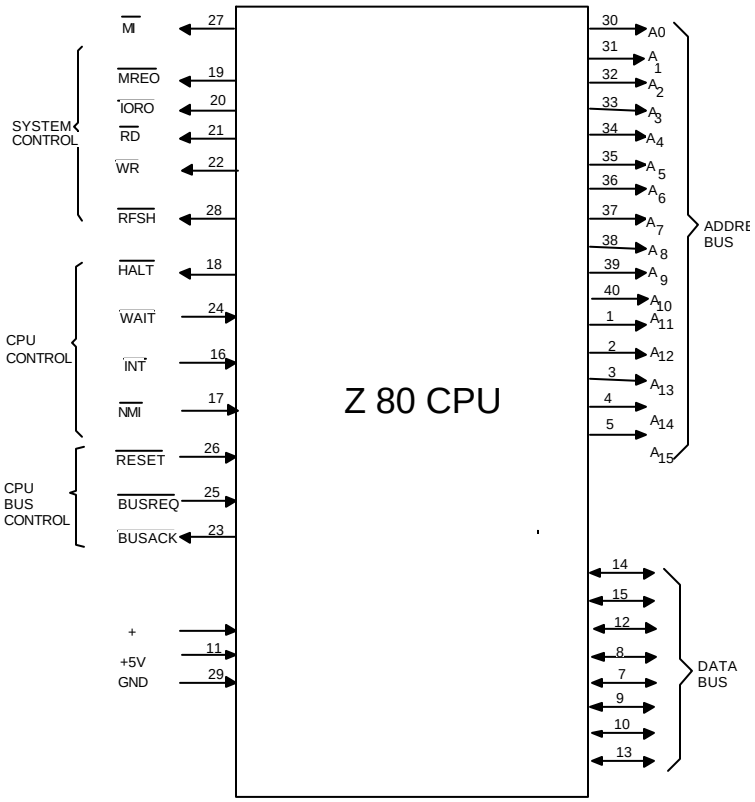
1. Arsitektur dari Mikroprosessor Z.80



Gambar 7.15 Arsitektur Mikroprosessor Z-80



Gambar 7.15 Arsitektur Mikroprosessor Z-80



Gambar 7.16 Bus Sistem CPU Z-80

BUS - DATA

Menggambarkan sejumlah penghantar paralel yang menghubungkan satuan fungsi dari sistem mikroprosessor .
Data yang bekerja didalam mikroprosessor ditransfer melalui data bus
Transfer data berjalan dalam dua arah (Bi Directional)
Mikroprosessor Z. 80 dan 8085 mempunyai penghantar data - bus 8 bit.
Untuk mengirim data secara bidireksional (lintasan dua arah) antar berbagai chip yang terdapat dalam suatu sistem mis : dari perantara I/O menuju mikroprosessor dan dari mikroprosessor menuju memori.

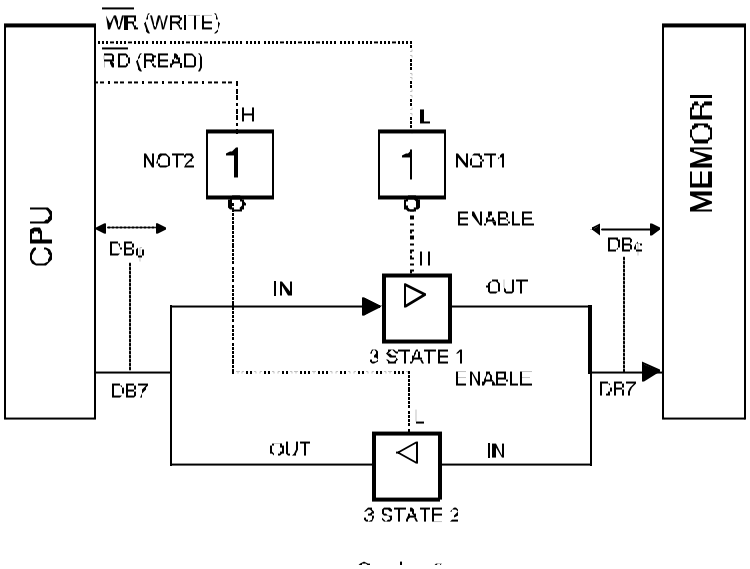
BUS ALAMAT

Kombinasi sinyal pada penghantar Bus Alamat dari blok fungsi mikroprosessor.
Misalnya : Alamat memori untuk program / data , yang mana pada lokasi memori ini, - data akan ditulis atau dibaca.
Bus alamat 8085 / Z. 80 terdiri dari 16 buah penghantar yang paralel dan membentuk alamat dengan lebar sebanyak 16 bit, yaitu $2.^{16}$ (2 pangkat 16) atau 65536 Byte (64 KB).
Kombinasi sinyal pada penghantar Bus - alamat dikirim dari mikroprosessor, dengan demikian Bus - alamat bekerja secara Uni Directorial (satu arah).
Bus ini berpangkal dari mikroprosessor dan digunakan untuk menghubungkan alamat-alamat CPU dengan semua chip yang mempunyai alamat. Bus ini digunakan dalam hubungannya dengan bus data untuk menentukan sumber atau tujuan data yang dikirim pada bus data.

BUS KONTROL

Sistem penghantar bekerja dengan kombinasi secara tepat dan logis untuk mengontrol proses jalannya sinyal diluar dari pada mikroprosessor (mengsinkronkan kerja CPU dengan blok - blok lainnya)
Penghantar Bus - kontrol bersifat uni directional dengan arah yang berbeda – beda.
Untuk membawa sinyal - sinyal penyerempak antara mikroprosessor dan semua alat/chip yang dihubungkan dengan Bus - bus
Mis : sinyal baca , tulis, interupsi wait dan .

PENULISAN (WRITE) DATA PADA MEMORI

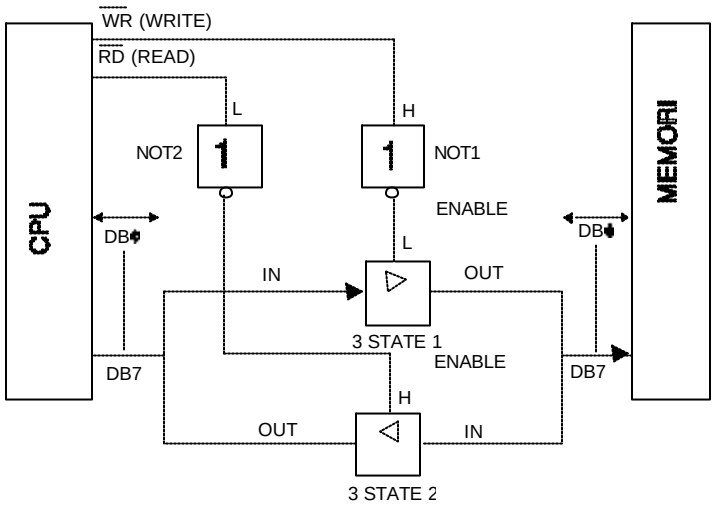


Gambar 7.17 Penulisan Data pada Memory

Proses Kerja

⇒ Bila PIN WR (WRITE) pada CPU/mikroprosessor menghasilkan logika L (" 0 ") maka output gerbang NOT 1 berlogika H (" 1 ") dan gerbang 3 STATE mendapat logika H (" 1 ") Hal ini menyebabkan 3 STATE 1 bekerja sehingga seluruh data dari CPU dimasukkan ke memori. DATA DBO - DB7 dari CPU masuk ke DBO - DB7 MEMORI.

⇒ Pada saat bersamaan pin RD (Read) berlogika High (" 1 ") dan output NOT2 berlogika L (" 0 ") menyebabkan 3 STATE2 tidak bekerja.



Gambar 7.18 Pembacaan Data pada Memory

Proses Kerja

⇒ PIN $\overline{RD}$ (READ) berlogika L (" 0 ") menyebabkan 3 STATE2 bekerja sehingga data DBO - DB7 pada dipindahkan (dibaca) ke DBO - DB7 pada CPU melalui Bus - Data 3 STATE2.

⇒ Pada saat yang bersamaan pin $\overline{WR}$ (WRITE) pada CPU berlogika H menyebabkan output NOT 2 berlogika1 (" 0 ") sehingga 3 STATE tidak bekerja.

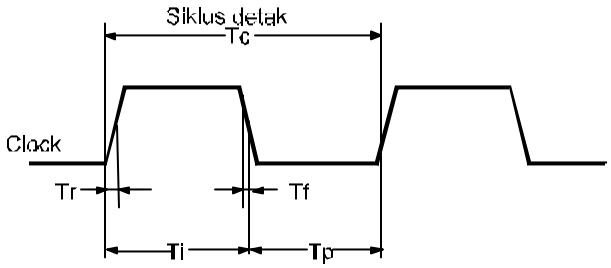
Signal Kontrol

* Basis operasi dari Z. 80 terdiri dari :
  - Pembacaan (Read) memori atau penulisan (Write) di memori
  - Pembacaan dari pheripherial atau penulisan ke pheripherial
  - Pelaksanaan interrupt.

1. Siklus Detak, Siklus Mesin, Siklus Instruksi

Setiap basis operasi dapat terdiri dari 3 sampai 6 siklus detak (satu siklus detak adalah satu periode dari frekuensi yang terpasang).

Siklus detak membentuk siklus mesin, dari beberapa siklus mesin membentuk siklus instruksi.



Gambar 1.a.

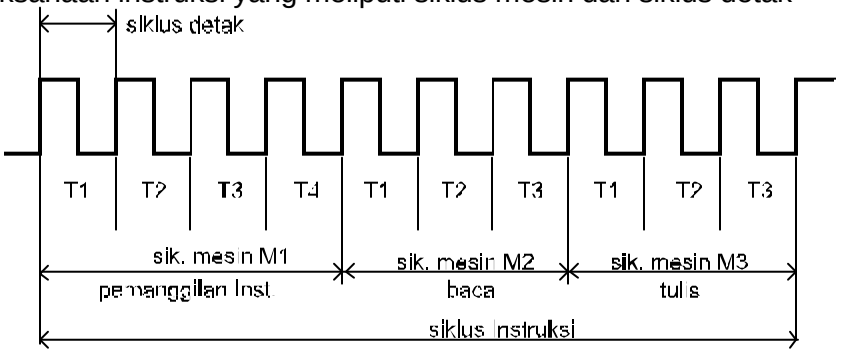
Gambar 7.19 Siklus Detak

Biasanya Ti ≈ Tp

Tr = Tf ≤ 30 ns

Type	Tcmin	Fmax
Z80	400 ns	2,5 MHz
Z80 A	250 ns	4,0 MHz
Z80 B	165 ns	6,0 MHz

Pelaksanaan instruksi yang meliputi siklus mesin dan siklus detak



Gambar 1.b.

Gambar 7.20 Siklus Instruksi

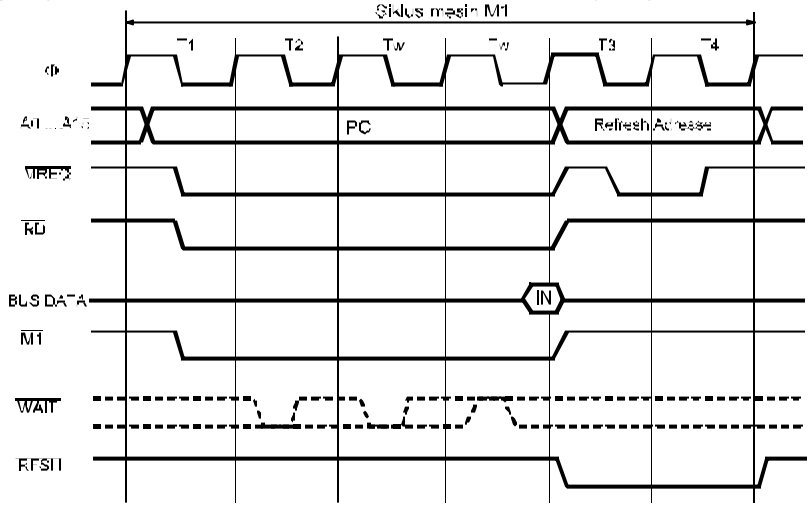
Gambar Ib. adalah contoh 1 instruksi dengan tiga siklus mesin dan 10 siklus detak.

Siklus mesin pertama suatu Instruksi (disebut op-code fetch) adalah pemanggilan/pengambilan operation code. Siklus mesin yang lain (3 sampai 5 siklus detak) merupakan siklus instruksi pembacaan data dari memori atau pheripherial dan penulisan data di memori.

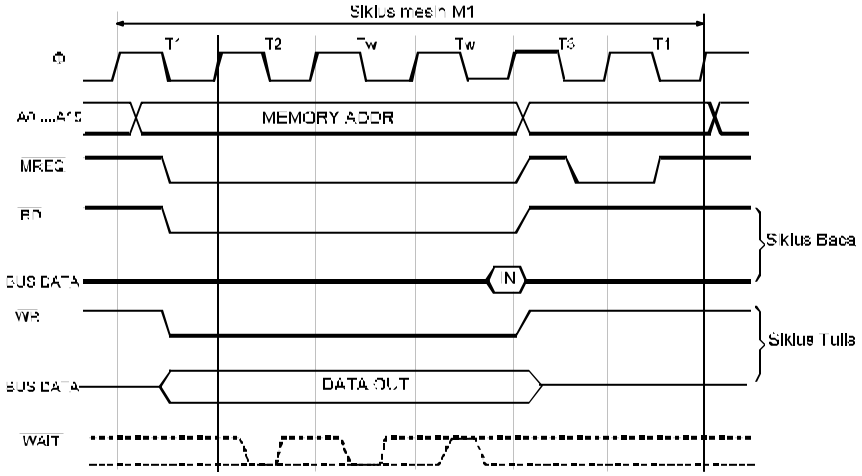
2. Diagram Waktu Dari Pemanggilan Instruksi

Gambar 2. Menjelaskan diagram waktu dari pemanggilan instruksi. Isi dari program counter (PC) berada pada bus alamat (Adress bus) setelah awal dari siklus mesin dimulai (lihat detak naik pada T1).

Dengan detak turun pada T1, MREQ dan RD menjadi aktip (keadaan aktip = “0”) CPU membaca sinyal pada data bus dan dengan naiknya siklus detak T3 (pada saat detak T3 naik), sedangkan MREQ dan RD kembali tidak aktip (sinyal “1”). Siklus detak ke 3 dan ke 4 dipergunakan oleh CPU untuk Refrech memori yang dinamis.



Gambar. 7.21. Timing Diagram Pemanggilan Instruksi



Gambar. 7.22 Timing Diagram Read/Write Memory

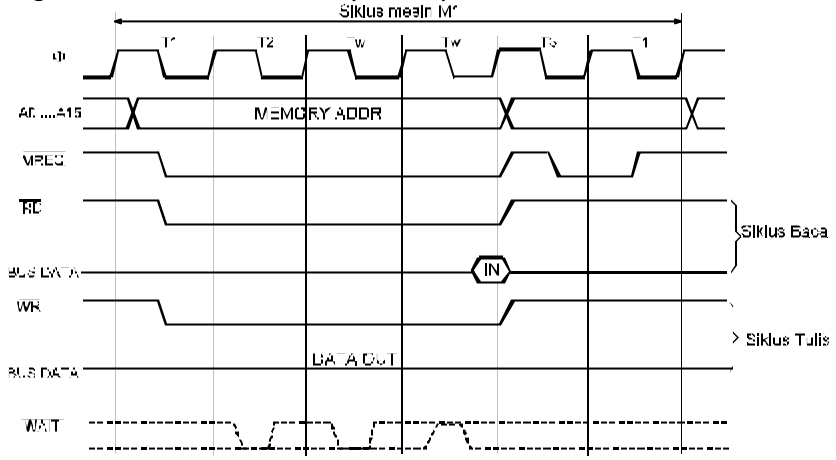
3. Diagram Waktu Pembacaan atau penulisan data pada Memori ke

Sinyal kontrol MREQ akan aktip “0” selama alamat memori dikirim adalah sah.

Pada proses baca, sinyal kontrol RD akan aktip “0”, selama CPU menerima data dan memori melalui Bus data .

Pada proses tulis, sinyal kontrol WR akan aktip “0”, selama CPU mengirim data ke memori melalui Bus data.

4. Diagram waktu dari siklus input/output



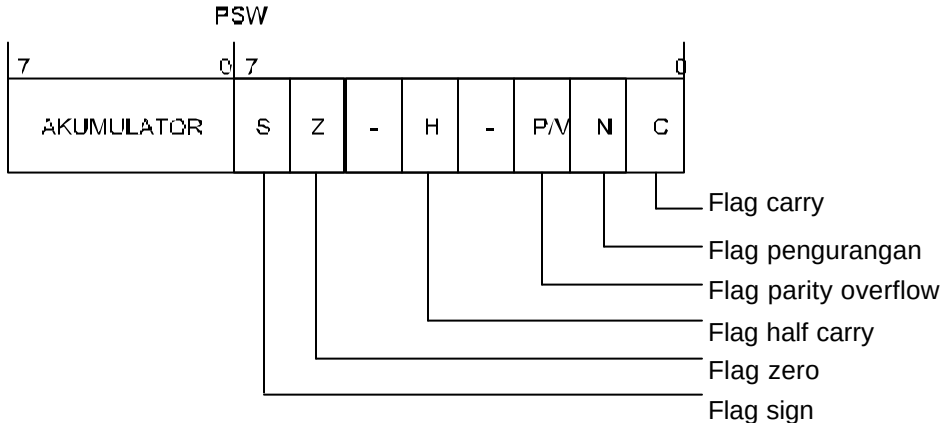
Sinyal kontrol IOREQ akan aktip “0”, selama alamat I/O yang dikirim adalah sah .

Pada proses baca, sinyal kontrol RD akan aktip “0”, selama CPU menerima data dari PIO melalui Bus data.

Pada proses tulis, sinyal kontrol NR akan aktip “0”, selama CPU mengirim data ke PIO melalui Bus data.

Flag

Flag adalah sebuah flip-flop di dalam blok penghitung dari CPU dan disebut sebagai register Flag.
Keadaan flag ini setelah pelaksanaan sebuah instruksi (yang mempengaruhi flag) akan menghasilkan sifat dari hasil sebuah operasi.Pada Z. 80, flag dipasangkan dengan akumulator dan dikenal dengan Program Status Wort (PSW)



Gambar 7.23 Register Flag Mikroprosessor Z-80

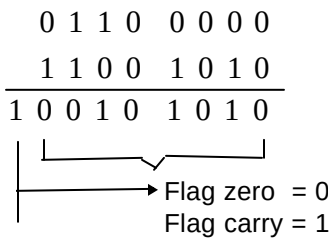
Jenis Flag

1. Flag Zero (Z)

Jenis flag ini menunjukkan apakah pada pelaksanaan terakhir dari operasi ,hasil pada semua bit adalah = 0

- Flag Zero = 1, bila pada semua bit register hasil = 0
- Flag Zero = 0, bila tidak semua bit pada register hasil = 0

Contoh :



2. Flag Carry (C)

Flag carry menunjukkan bahwa apakah pada proses operasi sebuah bit carry dipindahkan dari bit tertinggi MSB pada register hasil.
Kondisi ini dapat terjadi pada operasi :

⇒ Penjumlahan, bila hasil proses data lebih dari 8 bit, atau 16 bit.
⇒ Pengurangan a - b , bila b > a, hasilnya juga negatip
⇒ Pergeseran , bila nilai 1 bit pada bit tertinggi atau terendah digeserkan ke carry.
- Flag carry = 1, bila terjadi carry (lebihan/bawaan)
- Flag carry = 0, bila tidak terjadi carry.

Flag carry dapat diset melalui perintah SCF dan disalin melalui perintah CSF.

3. Flag Sign (S)

Pada operasi yang mempengaruhi flag, flag sign menyimpan kondisi bit tertinggi dari register hasil.
- Flag sign = 1 bila bit tertinggi dari register hasil = 1
- Flag sign = 0 bila bit tertinggi dari register hasil = 0

4. Flag Parity/Overflow (r /v)

Bit kedua dari register flag mempunyai 4 (empat) arti yang berbeda, tergantung dari hasil akhir pelaksanaan operasi.

a. Flag Overflow

Pengertian ini berlaku setelah pelaksanaan dari perintah aritmatik;
- ADD, ADC, SUB, SBC
- INC, DEC

Flag overflow diset 1 pada proses perpindahan dari bit ke 7 ke bit 8, yaitu yang mempengaruhi tanda bilangan positip atau negatip pada perhitungan bilangan.

b.Flag parity

Pengertian ini berlaku setelah pelaksanaan dari perintah berikut ini :
- Perintah logika → AND, OR, XOR
- Perintah geser → RL, RR, RLC, SLA, SRA, SRL, RLD, RRD
- Aritmatik BCD → DAA
- Perintah input dengan pengalamatan tidak langsung IN r, (c)
  - Flag parity = 1, bila jumlah bit dari hasil akhir operasi adalah genap
  - Flag parity = 0, bila jumlah bit dari hasil akhir operasi adalah ganjil.

c. Penjumlahan Nol pada perintah Blok

Pada perintah berikut untuk transfer Blok dan pembanding Blok. Flag P/V menunjukkan keadaan register BC, yang pada operasi ini dipakai sebagai register penghubung.
- Transfer Blok → LDI, LDIR, LDD, LDDR
- Pengamatan Blok → CPI, CPIR, CPD, CPDR.

Hal tersebut diatas berlaku jika :
- Flag P/V = 0, bila register penghitung BC = 0000 H
- Flag P/V = 1, bila register penghitung BC ≠ 0000 H

5. Flag Pengurangan (N)
   Urutan perhitungan untuk persamaan desimal (DAA) pada operasi penjumlahan berbeda dengan operasi pengurangan, hal ini tergantung pada kondisi bit flag N. Pada operasi pengurangan nilai flag N di set, sedangkan untuk operasi penjumlahan flag N di reset.
6. Flag Halt Carry (H)
   Bila pada penjumlahan terdapat perpindahan Carry dari Bit 3 ke bit 4 maka Flag Half Carry (H) diset, bila tidak ada carry, flag half carry (H) di reset.
   Pada pengurangan flag half carry (H) di set bila terjadi perpindahan pada bit ke 4 ke bit 3.

Pengalamatan Memori (Penyimpan Program/data)

Kapasitas Pengalamatan memori

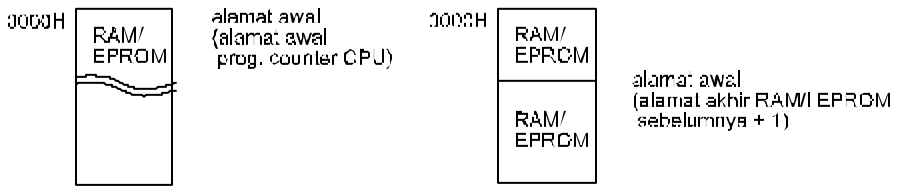
Kapasitas penyimpan pada RAM atau EPROM tergantung pada jumlah PIN alamat (Ao - An) dari RAM/EPROM tersebut, dan dihitung dengan rumus :

$$\text{Kapasitas Penyimpan} = 2^{n+1}$$

Sebagai contoh :
- Jumlah pin sebuah RAM 6116 = sebanyak 11 buah (Ao ~ A_{10})
- Maka kapasitas RAM ini adalah :

$2^{(10+1)} = 2^{11}$ = 2048 lokasi

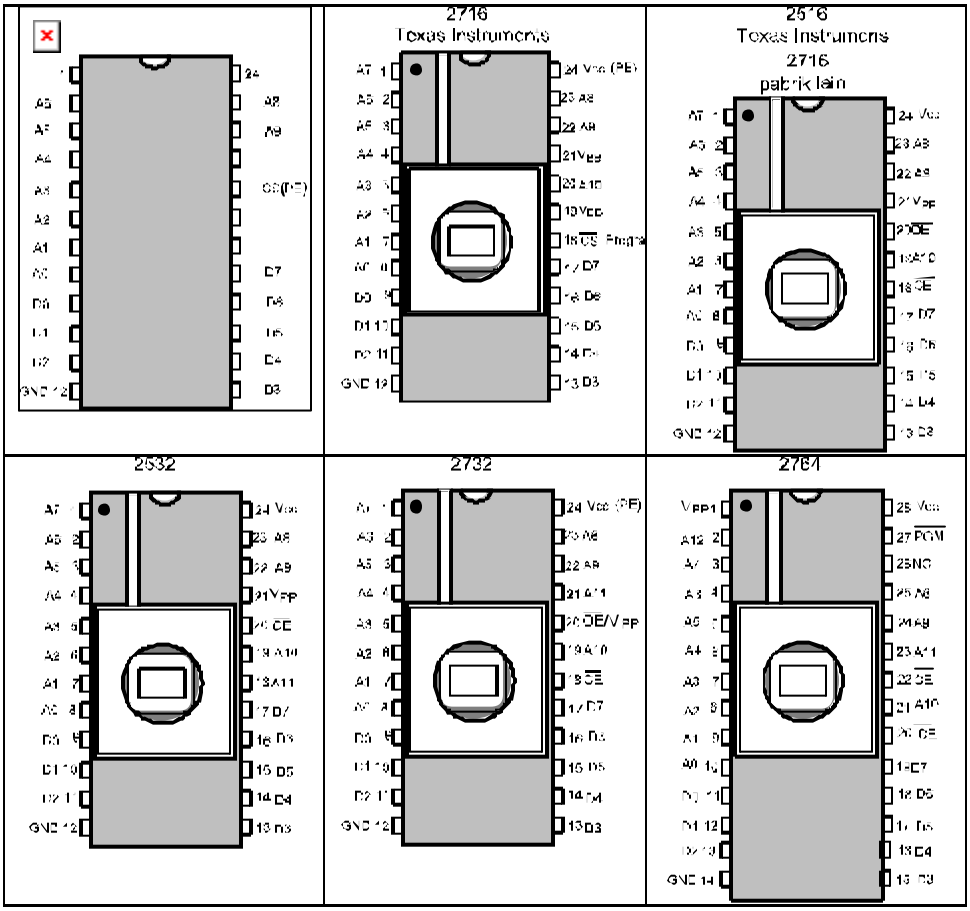
Untuk menentukan alamat awal dan akhir dari penyimpan/memori di atas dapat ditentukan sebagai berikut :
- Alamat awal dapat ditentukan 0000 H yaitu alamat awal program counter dari CPU atau alamat akhir RAM atau EPROM sebelumnya ditambah 1.



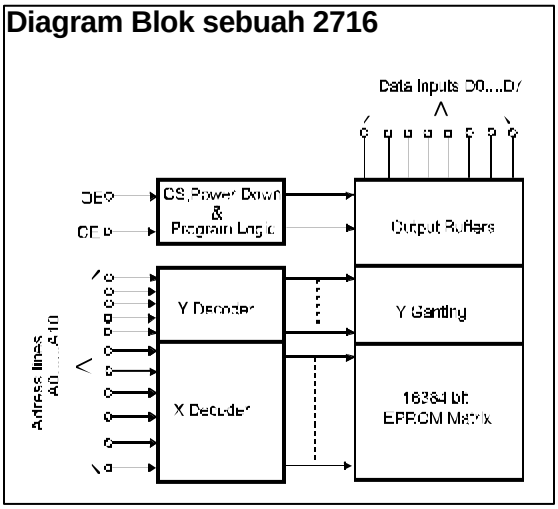
Gambar 7.24 Peta Memory RAM/EPROM Mikroprosessor Z-80

- Alamat akhir dapat ditentukan sesuai dengan jumlah kapasitas RAM/EPROM tersebut ditambah dengan alamat awalnya.

Susunan Pin EPROM



Gambar 7.25 Konfigurasi Pin EPROM

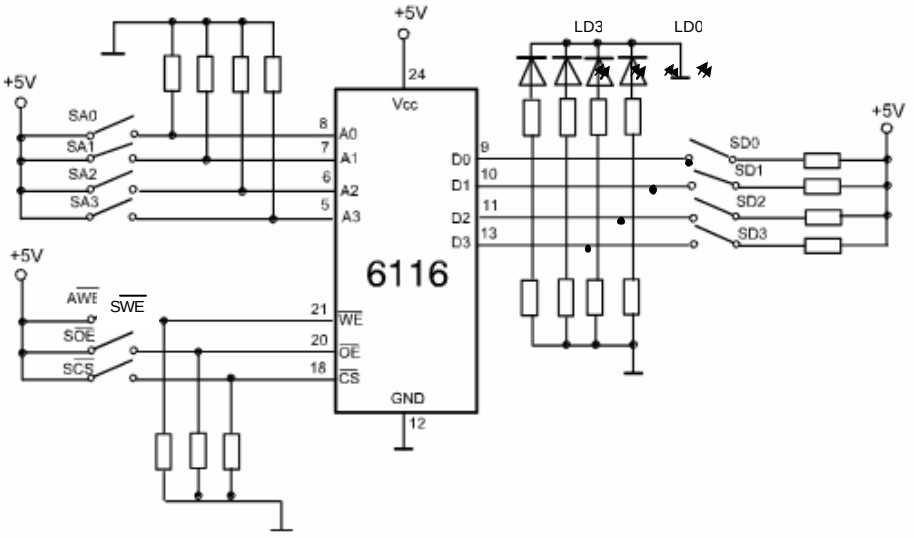


Gambar 7.26 RAM 6116 dan EPROM 2716

Organisasi EPROM

Pengorganisasian tiap tipe	**Susunan pin - pin**
2708 1024 x 8 bit 2716 2048 x 8 bit 2732 4096 x 8 bit 2764 8192 x 8 bit 2516 2048 x 8 bit 2532 4096 x 8 bit	Vcc = + 5 V VBB = - 5 V VDD = + 12 V Vpp = + 5 V dalam ragam siap + 25 V dalam ragam pengacaraan A0... An = Jalan masuk alamat D0... D7 = Masuk dan keluar data CS = Chip Select CE = Chip Enable OE = Output Enable PD = Power Down

3. Pengalamatan RAM 6116 dalam operasi dasar



Gambar 7.27 Rangkaian RAM 6116

Operasi dasar yang dilaksanakan pada RAM : adalah operasi penulisan data atau pembacaan data ke / dari RAM oleh CPU. Data yang tersimpan sifatnya sementara, tergantung pada catu daya pada RAM.

4. Proses jalannya operasi dasar RAM 6116

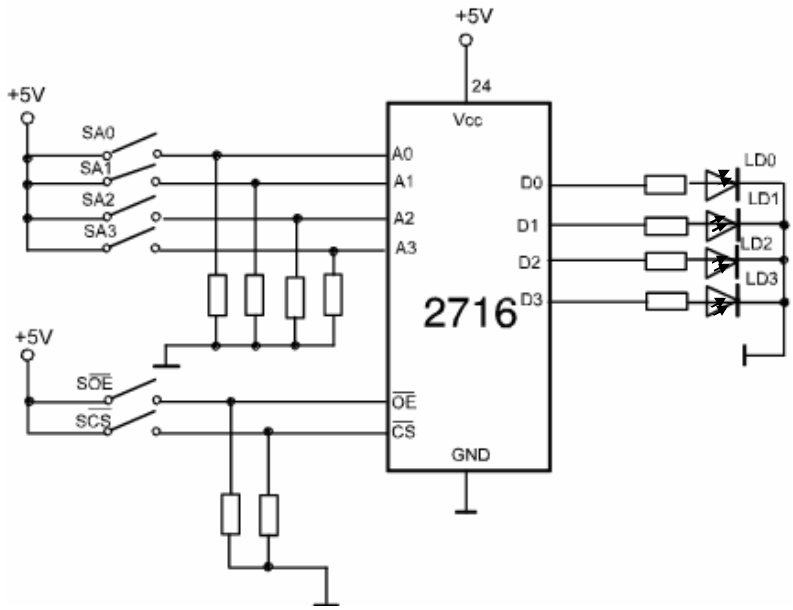
a. Proses Penulisan Data.

- Tentukan data pada Bus Data (SD3 - SD0), contoh : 6 H
- Tentukan Alamat Penyimpan (SA3 - SA0), contoh : OH
- S WE dibuka $\rightarrow$ operasi menulis
- S OE ditutup
- S CS ditutup - dibuka
- Ulangi proses penulisan diatas (langkah 1 - 5) untuk mengisi alamat lainnya yaitu 4 H dengan data EH (catu jangan diputuskan pada proses ini)

b. Proses Pembacaan Data

- Posisi sakelar SD3 - SD0 pada posisi terbuka semua
- Tentukan Alamat Penyimpan (SA3 - SA0) yang akan dibaca datanya, contoh : OH
- S WE $\rightarrow$ ditutup
- S OE $\rightarrow$ di buka $\rightarrow$operasi membaca
- S CS $\rightarrow$ ditutup - dibuka
- Pada LED LD3 - LD0 akan menunjukkan data 6 H
- Ulangi proses pembacaan diatas (langkah 1 - 6) untuk membaca isi alamat penyimpan lainnya, yaitu : 4 H.
- Data yang akan ditunjukkan pada LED LD3 - LD0 adalah EH.

5. Pengalamatan EPROM 2716 dalam operasi dasar



Gambar 7.28 Rangkaian EPROM 2716

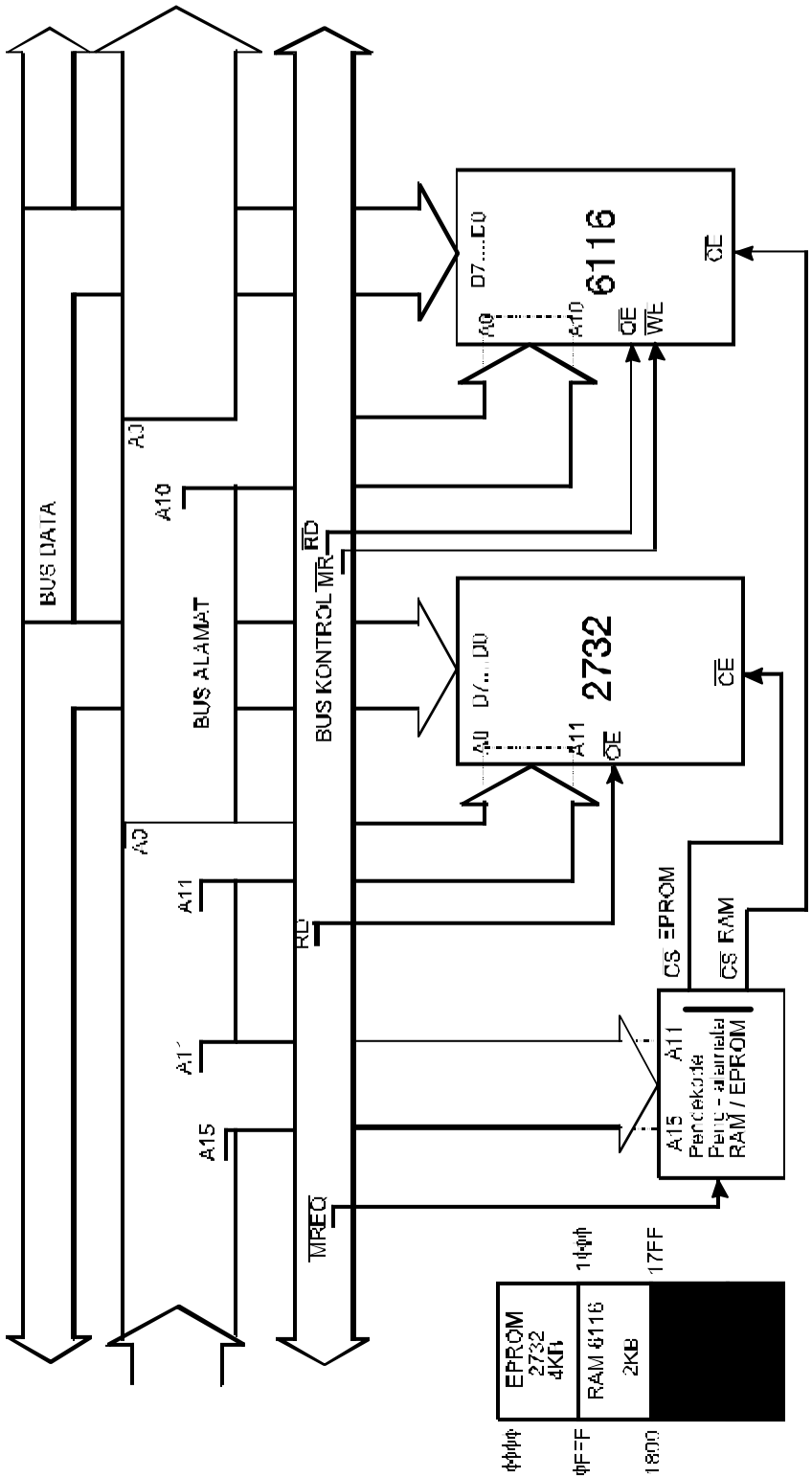
Operasi dasar yang dapat dilakukan pada EPROM adalah hanya operasi pembacaan data dari EPROM oleh CPU.
Data tersimpan tetap paten pada EPROM dan tidak tergantung pada catu daya . Pengisian data pada EPROM dilakukan dengan mempergunakan EPROM Writer/Programer.

6. Jalannya Operasi Dasar (Proses Pembacaan) EPROM 2716

Tentukan alamat penyimpan (SA3 - SA0) yang isinya akan dibaca
S OE $\rightarrow$ di buka
S CS $\rightarrow$ ditutup - dibuka
Pada LD3 - LD0 akan menunjukkan isi alamat yang dipilih
Ulangi langkah 1 - 4 untuk membaca data pada alamat lain
Putuskan catu daya chip 2716
Ulangi langkah 1 - 5 untuk membaca data alamat - alamat yang sama pada langkah 1 - 5
Hasil pada LD3 - LD0 menunjukkan data yang sama walaupun catu daya telah diputuskan.

7. Pengalamatan RAM 6116 dan EPROM 2716 pada Sistem Minimal Z - 80

Dalam pengalamatan ini , beberapa pin masukan dari CPU Z - 80, juga dipergunakan dalam pengalamatan RAM dan EPROM ini.
Selain pin - pin kontrol $\overline{WR}/\overline{WE}$, $\overline{RD}/OE$ dan Bus Data, dari CPU, juga digunakan pin - pin alamat
A15 - A0 dan $\overline{MREQ}$
A15 - A0 dipergunakan untuk memberikan data alamat RAM/EPROM.
$\overline{MREQ}$ digunakan bersama sinyal - sinyal alamat A15 - A0 untuk mengaktifkan RAM/EPROM.
Rangkaian pendekode pengalamatan RAM/EPROM berfungsi untuk mengaktifkan RAM/EPROM pada daerah pengalamatannya, yaitu mulai dari alamat awal sampai alamat akhir dari RAM/EPROM, sesuai dengan peta pengalamatannya.
Pin - pin alamat CPU yang tidak termasuk dalam daerah pengalamatan RAM/EPROM, harus diperhatikan dan diikutkan dalam pengalamatan RAM/EPROM.
Untuk menghindari adanya beberapa alamat RAM atau EPROM yang menunjuk pada data lokasi RAM/EPROM yang sama, oleh sebab itu pin - pin alamat CPU ini bersama - sama dengan sinyal
$\overline{MREQ}$ dipergunakan sebagai masukan dari pendekode pengalamatan RAM/EPROM.
Hasil pendekodean alamat (keluaran pendekode pengalamatan), dihubungkan ke pemilih Chip ($\overline{CS}$ / $\overline{CE}$) dari masing - masing Chip.


BUS DATA
BUS ADRESSE
BUS KONTROL
A0
A10
A11
A15
RD
WR
MREQ
D7....D0
6116
CE
OE
WE
2732
CS EPROM
CS RAM
Decodeur
Partie Adresse
RAM / EPROM
EPROM
2732
4KB
RAM 6116
2KB
0000
07FF
1000
17FF

8. Perencanaan Pendekode Pengalamatan RAM/EPROM

a. Pemetaan Lokasi Pengalamatan

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Daerah Memori
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000 H alamat awal EPROM
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFFH alamat akhir EPROM
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000H alamat awal RAM
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	17FFH alamat akhir RAM

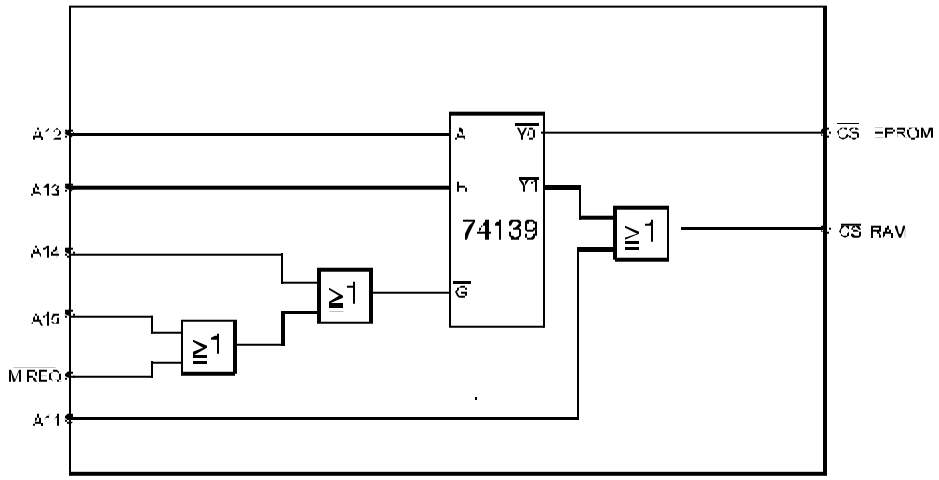
b. Persamaan Boole

$$\overline{CS}\,EPROM = ((\overline{MREQ} \vee \overline{A15}) \vee \overline{A14} \vee \overline{A13} \vee \overline{A12})$$

$$\overline{CS}\,RAM = (((\overline{MREQ} \vee \overline{A15}) \vee \overline{A14}) \vee \overline{A13} \vee A12) \vee \overline{A11})$$

$\rightarrow$ Bekerja dengan negatif $\rightarrow$ yang dicari

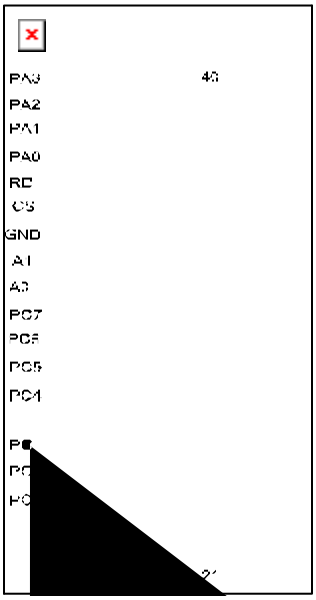
c. Rangkaian Pendekode Pengalamatan RAM /EPROM



Gambar 7.29 Rangkaian Dekoder RAM/EPROM

Pengalamatan PPI 8255

PPI 8255 terdiri dari 4 register port yang menampung data 8 bit dan berhubungan dengan bus data sistem melalui bus data internal. Dalam register ini ditempatkan data masukan, keluaran atau data kata kendala. Masing-masing register mempunyai alamat sendiri yang dapat dipilih melalui pengkodean pengalamatan PPI 8255.
Gambar 1 :
Menunjukkan Pin - Pin dari PPI 8255 dengan fungsinya masing - masing



D7 - D0	Data Bus (Bi-Directional)
RESET	Reset Input
CS	Chip Select
RD	Read Input
WR	Write Input
A0,A1	Port Adress
PA7-PA0	Port A (BIT)
PB7-PB0	Port B (BIT)
PC7-PC0	Port C (BIT)
Vcc	+5 Volts
GND	0 Volt

8255 OPERATIONAL DESCRIPTION

Gambar 7.30 Konfigurasi Pin PPI 8255

Pin - Pin Saluran Data :
⇒ Bus Data : D7 - D0
⇒ Bus Port A: PA7 - PA0
⇒ Bus Port B: PB7 - PB0
⇒ Bus Port C: PC7 - PC0

Pin - Pin Saluran Pengontrol :
⇒ Baca (Read) : RD
⇒ Tulis (Write) : WR
⇒ Reset : Reset

Pin-pin Pendekode Alamat :
Pin - pin yang sangat penting untuk mengkode alamat register PPI 8255 adalah :
Pin alamat : A1 dan A0 serta pemilih chip (Chip select) : CS

keterangan fungsi masing-masing pin dan penggunaannya dalam rangkaian

D7 - D0 : Dihubungkan ke sakelar dan LED
Sakelar dan LED menggantikan fungsi Bus Data sebagai jalannya data 2 arah (membaca dan menulis).
Untuk rangkaian ini pada saat operasi membaca data, posisi sakelar SD7 - SD0 harus terbuka.

PA7 - PA0 : Dihubungkan ke LED
Konfigurasi PPI 8255 mengatur port A sebagai terminal keluaran data dan LED dipakai untuk menampilkan data keluaran pada terminal port A.

PB7 - PB0 : Dihubungkan ke sakelar
Sakelar dipakai untuk memasukkan data ke terminal port B.

RD : Dihubungkan ke sakelar
Sakelar dipakai untuk memasukkan sinyal baca pada terminal RD

WR : Dihubungkan ke sakelar
Sakelar dipakai untuk memasukkan sinyal tulis pada terminal WR
Sakelar Write (S WR) terbuka : operasi menulis

RESET : Dihubungkan ke sakelar
Sakelar dipakai untuk memasukkan sinyal Reset pada terminal Reset
Sakelar Reset (S RS) terbuka : PPI tidak terreset

CS : Dihubungkan ke sakelar
Sakelar dipakai untuk memasukkan sinyal Pemilihan Chip pada terminal (CS) Chip Select
Sakelar CS terbuka : PPI aktif

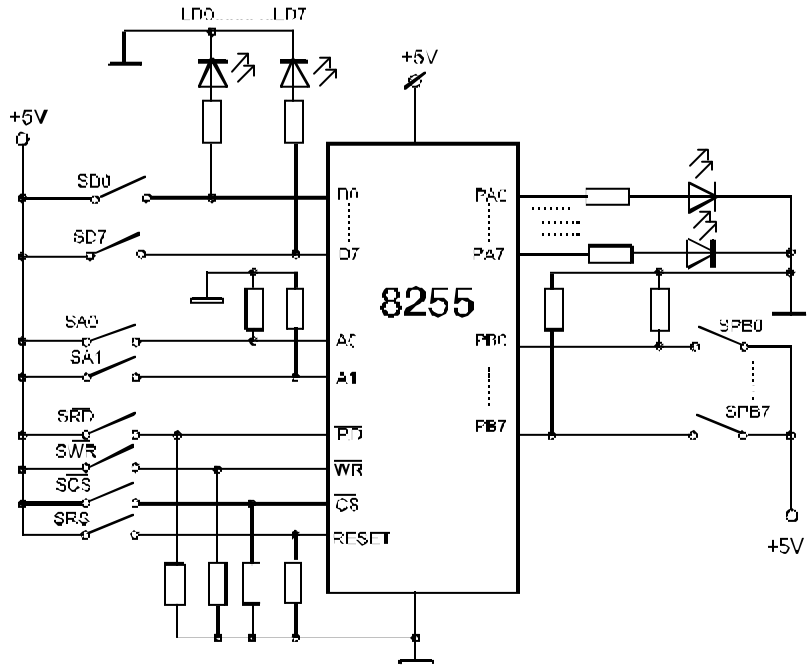
A1-A0 : Dihubungkan ke sakelar
Sakelar dipakai untuk memasukkan data alamat Port

1. Operasi Dasar PPI 8255

Reset	CS	RD	WR	A1	A0	Operasi reset
1	X	X	X	X	X	Port A, B dan C sebagai Masukan
Operasi Membaca (Read)						
0	0	0	1	0	0	Port A → Bus Data
0	0	0	1	0	1	Port B → Bus Data
0	0	0	1	1	0	Port C → Bus Data
Operasi Menulis (Write)						
0	0	1	0	0	0	Bus Data → Port A
0	0	1	0	0	1	Bus Data → Port B
0	0	1	0	1	0	Bus Data → Port C
0	0	1	0	1	1	Bus Data → Register Kontrol
Fungsi yang tidak diperbolehkan						
X	1	X	X	X	X	Bus Data → Berimpedansi tinggi
0	0	0	1	1	1	Kondisi tidak syah
0	0	1	1	X	X	Bus Data → Berimpedansi tinggi

Menunjukkan kepada kita bagaimana untuk hubungan masing-masing Pin dan penggunaannya dalam menstransfer data

.



Gambar 7.31 Rangkaian PPI 8255

2. Jalan Operasi Dasar PPI 8255 :

a. Proses Inisialisasi PPI 8255
- Tentukan data kata kendala pada Bus Data (S D7 - S D0)
- Contoh : 82H (Port A=Keluaran, Port B=Masukan).
- RS $\rightarrow$ dibuka
- SA0 dan SA1 $\rightarrow$ ditutup (Alamat Register Kontrol)
- S RD $\rightarrow$ ditutup
- S CS $\rightarrow$ dibuka
- S WR $\rightarrow$ ditutup - dibuka (Operasi Menulis).
- Hasil LED PA7 - PA0 = Padam

b. Proses Menulis Data dari Bus Data ke Port A
- Tentukan data (yang akan dikeluarkan ke Port A) pada Bus Data SD7 - SD0
- SR $\rightarrow$ dibuka
- S A0 dan S A1 $\rightarrow$ dibuka (alamat Port A)
- S RD $\rightarrow$ ditutup
- A CS $\rightarrow$ dibuka
- S WR $\rightarrow$ ditutup- dibuka (operasi Menulis)
- Hasil LED PA7- PA0 menyala sesuai data saklar SD7-SD0 berarti telah terjadi pemindahan data dari bus data ke port A (penulisan data dari Bus Data ke Port A)

c. Proses Membaca Data dari Port B ke Bus Data

S R $\rightarrow$ dibuka
S A0 $\rightarrow$ ditutup dan S A1 $\rightarrow$ dibuka (alamat Port B)
S $\overline{WR}$ $\rightarrow$ ditutup
S $\overline{CS}$ $\rightarrow$ dibuka
S $\overline{RD}$ $\rightarrow$ ditutup- dibuka (operasi Membaca)

3. Peng-alamatan PPI 8255 pada sistem minimal Z 80

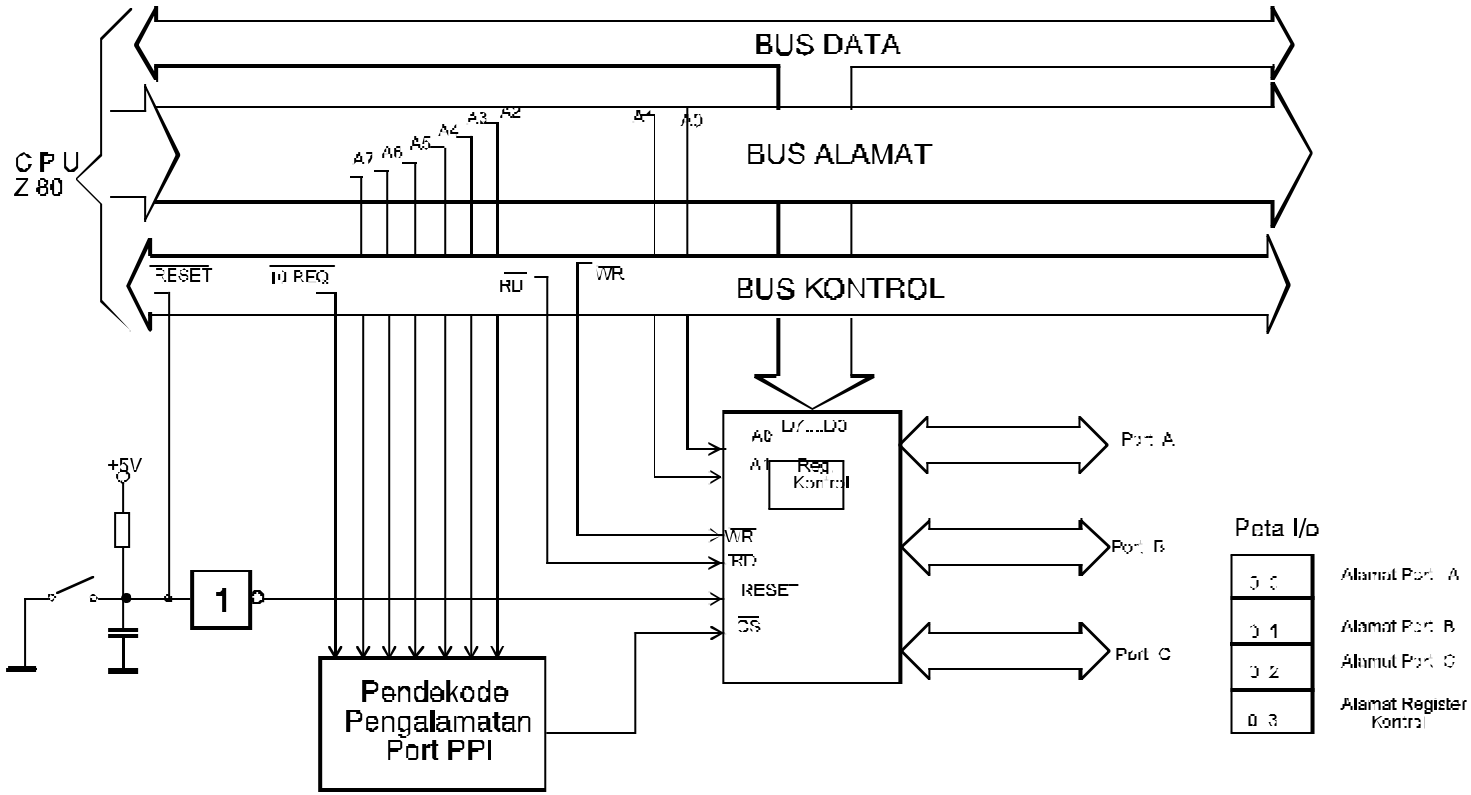
Beberapa pin masukan dan keluaran dari CPU Z 80 dipergunakan dalam peng-alamatan PPI ini. Selain Pin-Pin Kontrol : WR,RD,RESET dan Bus Data dari CPU, juga dipergunakan Pin alamat A7-A0 dan Pin IOREQ.

A7-A0 dipergunakan untuk memberikan data alamat port.

IOREQdipergunakan bersama sinyal-sinyal alamat A7-A2 untuk mengaktifkan PPI 8255.

A7-A2 dan IOREQ merupakan masukan dari Blok Pendekode Peng-alamatan Port PPI, yang mana dalam Blok Pengalamatan ini dibangun Rangkaian Pendekode. Rangkaian Pendekode ini berfungsi untuk mengaktifkan PPI 8255 pada daerah peng-alamatannya.

Kondisi data A7-A2 (yang bersama-sama IOREQ, dapat mengaktifkan PPI 8255 melalui CS) dan kondisi data A1-A0 dapat menentukan/menunjukkan alamat-alamat Port dan Register Kontrol PPI.



Gambar 7.35 Pengalamatan PPI 8255 pada Minimal Sistem Z-80

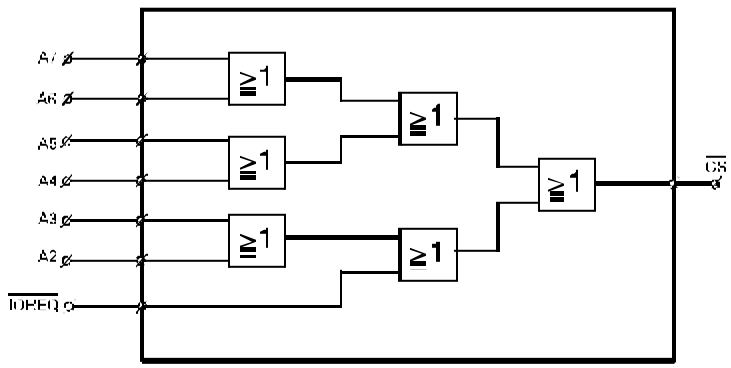
Sebelum rangkaian Pendekode/Pengalamatan Port PPI 8255 dibuat, maka kita harus menentukan peta alamat Port masukan keluaran.
Sebagai contoh :

Port A - - - - - - - - - - - - 00H
Port B - - - - - - - - - - - - 01H
Port C - - - - - - - - - - - - 02H
Register Kontrol - - - - - 03H

IOREQ	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1

Dari bantuan tabel diatas, kita dapat menganalisa, bahwa untuk mengaktifkan PPI 8255, kondisi A7-A2 dan IOREQ dari CPU harus berkondisi “LOW” (“0”). dan untuk menentukan alamat Port A,B,C dan Register Kontrol ditentukan oleh A1 dan A0.
Untuk membangun rangkaian Pendekode Pengalamatan Port sesuai data hasil analisa diatas,dapat dibangun dengan mempergunakan gerbang TTL dasar atau dengan Dekoder TTL 74138/74139.

Rangkaian Pendekode Pengalamatan Port PPI



Gambar 7.36 Rangkaian Pendekode Pengalamatan Port PPI

Pada sistem Mikroprosessor dengan PPI lebih dari 1, hubungan - hubungan PIN dari PPI dengan CPU, seperti pada sistem Mikroprosessor dengan 1 PPI yang jelas berbeda adalah pada Blok Pendekode Pengalamatan Port PPI.

Programmable Pheriperal Interface (PPI) 8255 (Perantara Pheriperal Terprogram)

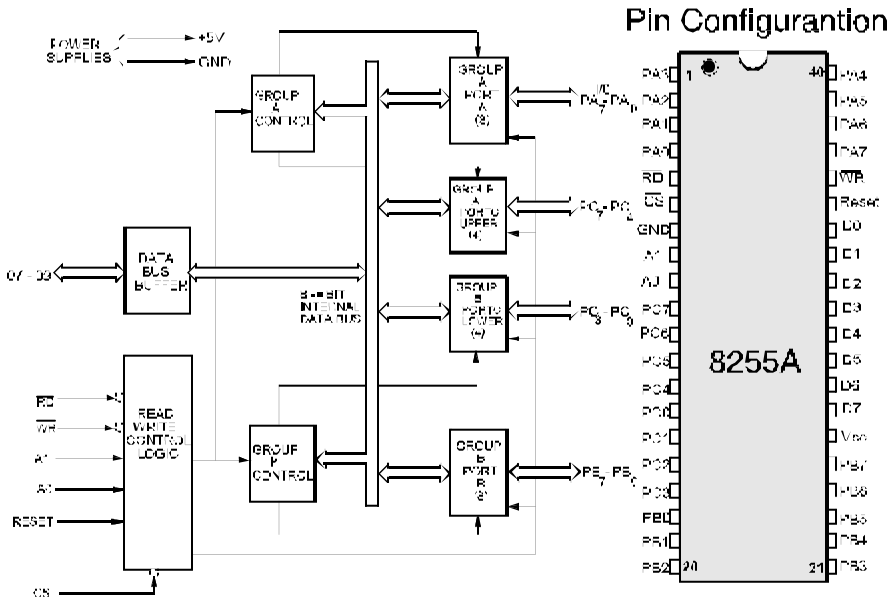
PERANTARA PHERIPHERAL TERPROGRAM 8255

Penjelasan Fungsi PPI 8255

IC PPI 8255 adalah peranti perantara pheriperal terprogram yang di desain untuk kegunaan dalam sistem Mikrokomputer. Fungsinya adalah sebagai komponen Multiguna masukan ataupun keluaran. Untuk perantara antara peralatan pheriperal luar dengan sistem mikrokomputer.

Konfigurasi Fungsi 8255 diprogram oleh sistem software tertentu.

Lihat gambar 1 dan 2 dibawah ini :



Gambar 7.37 Konfigurasi Rangkaian PPI 8255

Buffer Bus Data :

Buffer 8 bit dua (2) arah tiga (3) state ini, dipergunakan sebagai perantara 8255 dengan bus data sistem.

Data diterima atau dikirim oleh buffer tergantung perintah masukan atau keluaran oleh CPU.

Informasi kata kendala dan status dikirim melalui buffer.

Baca Tulis dan Logik Kontrol

Fungsi dari blok ini adalah untuk mengatur semua pengiriman internal dan external dari data dan kata kendala atau kata status.

8255 menerima masukan dari bus alamat dan bus kontrol CPU dan memfungsikannya untuk pelaksanaan tugas masing-masing kelompok kontrol 8255.

$\overline{CS}$
Chip Select (pemilih chip) kondisi “LOW” pada pin input ini, mengijinkan terjadinya komunikasi antara 8255 dengan CPU.

$\overline{RD}$
Read (pembacaan) kondisi “LOW” pada pin input ini, mengijinkan 8255 untuk mengirimkan informasi data ke CPU melalui Bus data.
Pada prinsipnya mengijinkan CPU membaca informasi data dari 8255.

$\overline{WR}$
Write (penulisan) kondisi “LOW” pada pin input ini, memungkinkan CPU untuk menulis informasi data ke 8255.

A_0 - A_1
Pemilih Port. Signal input-input ini, mengontrol pemilihan satu (1) dari empat (4) Port : Port A,B,C dan Register Kontrol. Ini biasanya dihubungkan dengan Bit - bit LSB dari bus alamat (A0 dan A1).
Lihat gambar 3 (tabel) dibawah ini :

A1	A0	RD	WR	CS	Input Operation (READ)
0	0	0	1	0	
0	1	0	1	0	
1	0	0	1	0	
					Output Operation (WRITE)
0	0	1	0	0	
0	1	1	0	0	
1	0	1	0	0	
1	1	1	0	0	
					Disable Function
X	X	X	X	1	
1	1	0	1	0	
X	X	1	1	0	

Reset
Reset kondisi “HIGH” pada input ini, akan menghapus isi Register Kontrol dan semua Port (A,B, dan C) dan semua Port di “SET” sebagai masukan.

B. Sistem Pengontrolan Port

Konfigurasi fungsi dari tiap-tiap "PORT" diprogram oleh software sistem, yang pada prinsipnya CPU mengirimkan data kata kendala ke register kontrol 8255. Kata kendala berisikan informasi seperti "MODE", "SET BIT","RESET BIT" dan seterusnya, yang akan meng-inisialisasi konfigurasi fungsi dari port 8255.

Setiap kelompok kontrol (untuk kelompok A dan B) menerima perintah dari "Logik Kontrol Read/Write", menerima kata kendala dari bus data internal dan mengijinkan untuk pembentukan hubungan/pengelompokan port dan fungsinya.

Kelompok kontrol A - Port A dan Port C atas (C7 - C4).

Kelompok kontrol B - Port B dan Port C bawah (C3 - C0).

Register kata kendala hanya dapat ditulis.

Operasi pembacaan pada register kata kendala tidak diijinkan.

Port A, B dan C

8255 terdiri dari 3 port 8 bit (A,B dan C), semua dapat dikonfigurasikan (dalam bermacam-macam fungsi) oleh Soft Ware Sistem.

Port A

Sebuah Buffer/Penyimpan keluaran data 8 bit dan sebuah penyimpan masukan data 8 bit.

Port B

Sebuah Buffer/Penyimpan keluaran data 8 bit dan sebuah penyimpan masukan data 8 bit.

Port C

Sebuah Buffer/Penyimpan keluaran data 8 bit dan sebuah Buffer masukan data 8 bit. Port ini dapat dibagi menjadi 2 port 4 bit melalui pengaturan Mode Kontrol. Setiap Port 4 bit terdiri dari sebuah penyimpan 4 bit dan itu digunakan untuk keluaran sinyal kontrol dan masukan sinyal status.

C. Penjelasan Proses Operasi 8255

1. Pemilihan Mode

Ada 3 mode, dasar operasi yang dapat dipilih oleh Software sistem :

- Mode 0 - Dasar masukan dan keluaran
- Mode 1 - Masukan/Keluaran sesuai sinyal isyarat
- Mode 2 - Bus dua arah

Bila masukan RESET menuju "High", semua port akan di set menjadi Mode masukan (keadaan berimpedansi tinggi). Mode untuk port A dan port B dapat ditentukan secara terpisah saat port C dibagi dalam 2 bagian sesuai yang ditentukan oleh pendefinisian port A dan port B.

Semua register keluaran termasuk flip-flop status akan direset bila mode diganti.

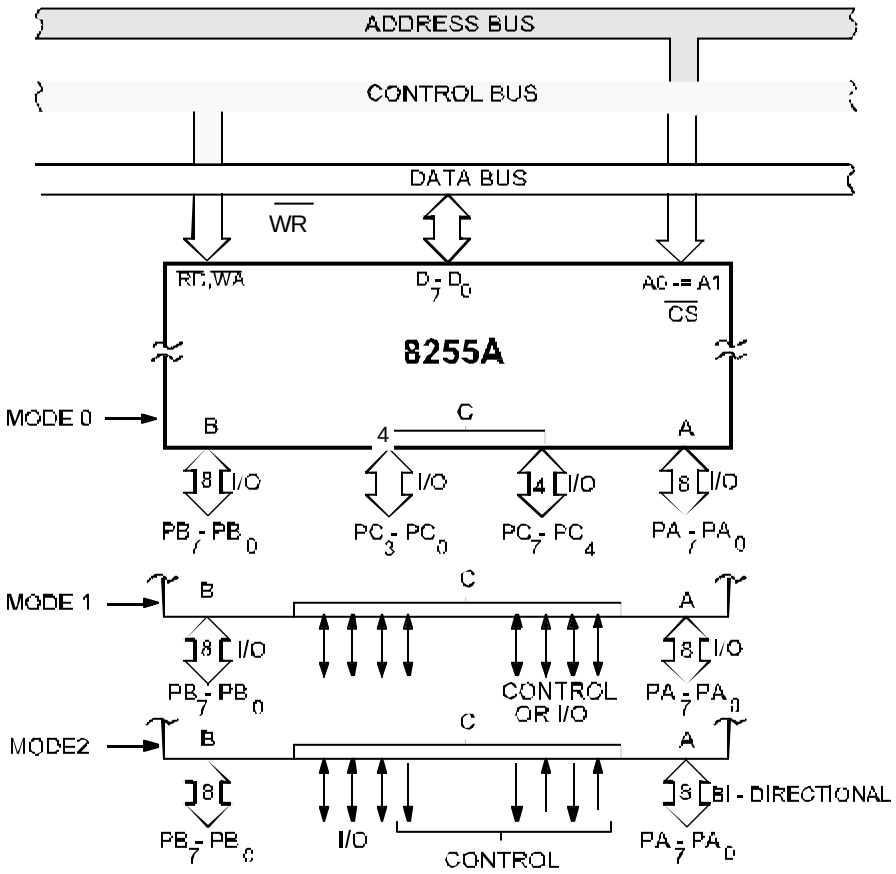
2. Bit Tunggal untuk Set - Reset

Beberapa bit dari 8 bit pada port C dapat di-Set atau di-Reset dengan menggunakan perintah Out. keistimewaan ini mengurangi Soft Ware, yang digunakan dalam aplikasi yang berdasarkan kontrol.

Bila port C digunakan sebagai status/control untuk Port A atau Port B, Bit ini dapat di-Set/Reset oleh penggunaan operasi Set/Reset bit. Sebagaimana jika digunakan sebagai port data keluaran.

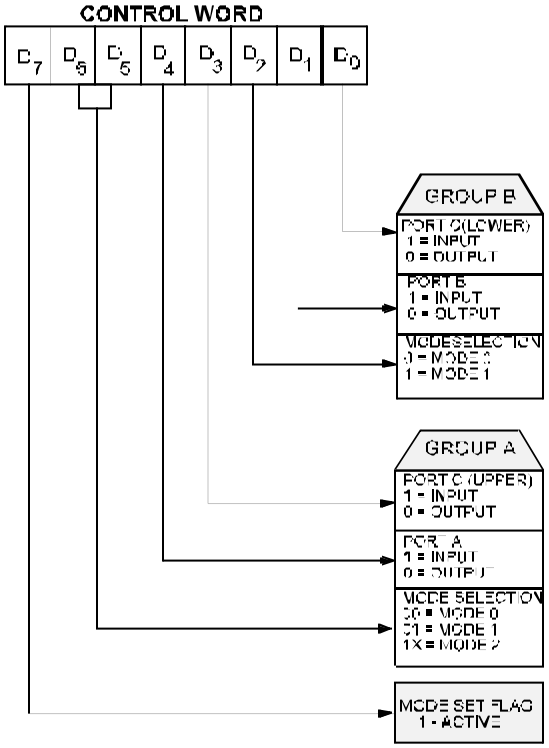
Perhatikan gambar 4 dibawah ini :

Definisi Mode Dasar dan Bus Perantara



Gambar 7.38 Mode Operasi PPI 8255

Definisi Format Mode



Gambar 7.39 Format Mode PPI 8255

Mode Operasi

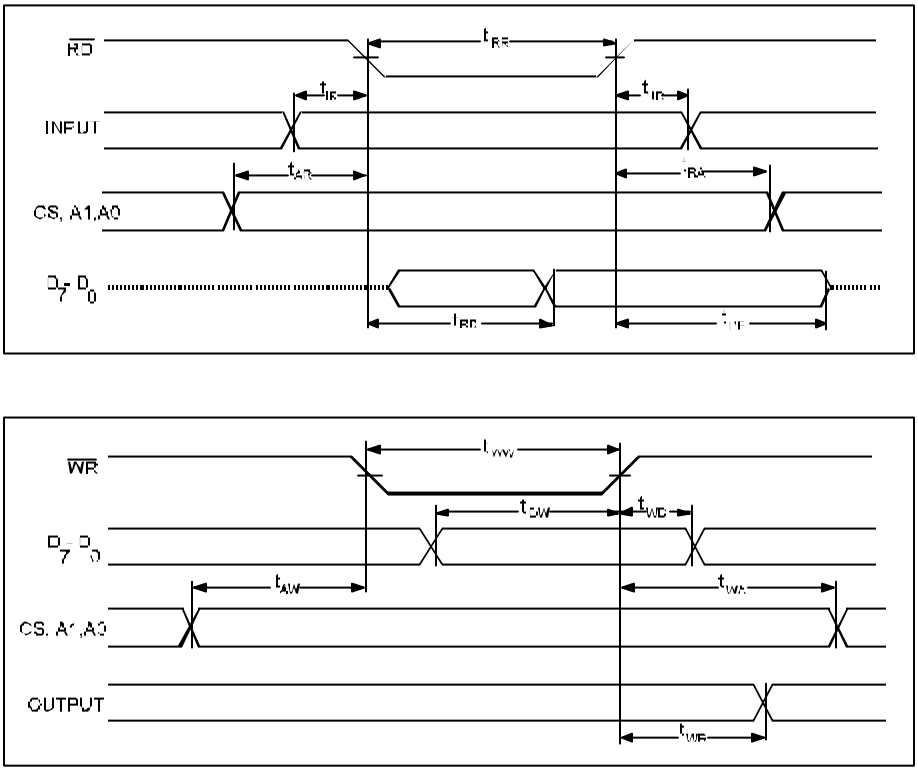
Mode 0 :

Konfigurasi fungsi menyediakan operasi masukan/keluaran yang sederhana untuk masing-masing port. Tidak mengharuskan ada "HandShaking" (pertukaran isyarat dari dua peranti yang saling berhubungan), data secara sederhana ditulis ke atau dibaca dari port tertentu.

Definisi dari fungsi dasar Mode 0 :

Dua (2) port 8 bit dan dua (2) port 4 bit
Setiap port dapat sebagai masukan atau keluaran
Keluaran di simpan
Masukan tidak disimpan
Memungkinkan 16 jenis konfigurasi masukan/keluaran

MODE 0 (BASIC INPUT) → Proses Sinyal Baca pada Mode 0

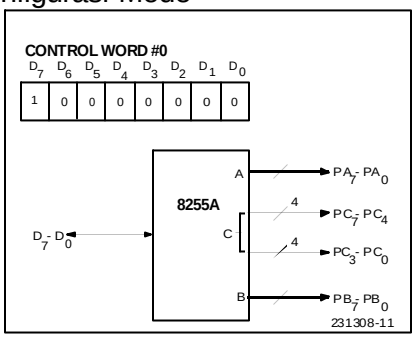


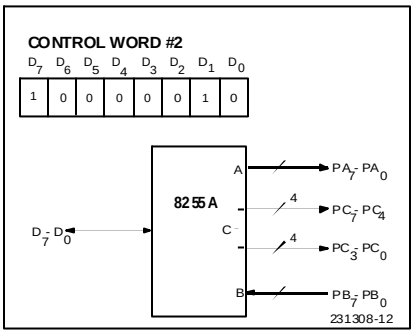
Gambar 7.40 Proses Sinyal Baca pada Mode Operasi 0

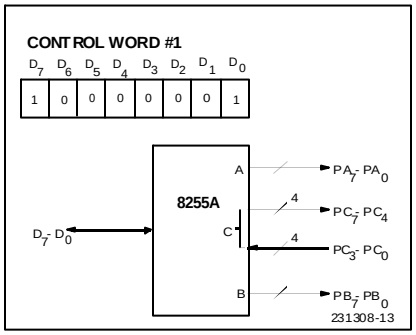
DEFINISI MODE 0 PORT

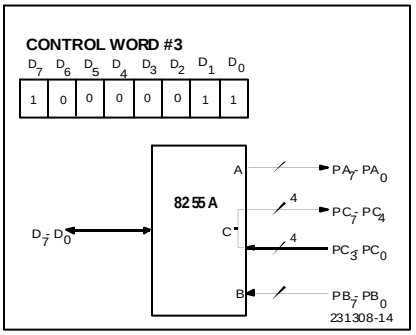
A		B		GROUP A		#	GROUP B	
D4	D3	D1	D0	Port A	Port C (Upper)		Port B	Port C (Lower)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

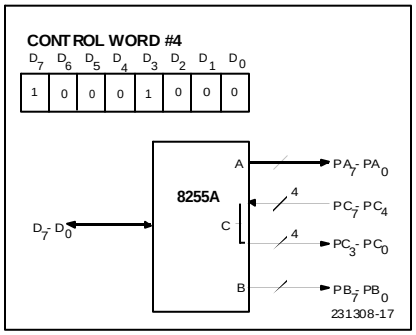
Konfigurasi Mode

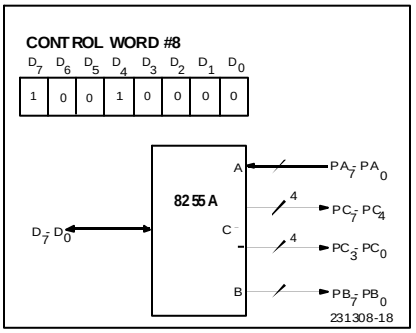


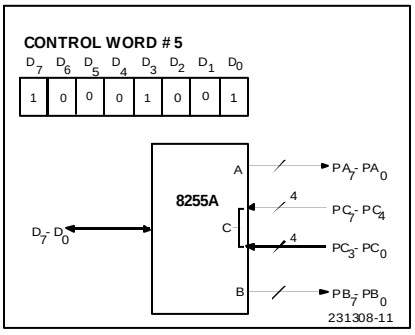


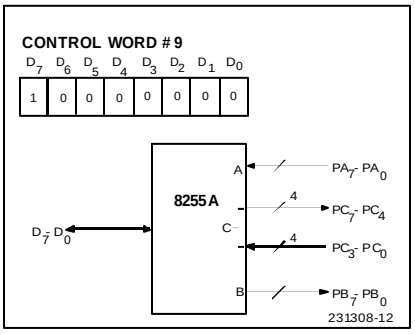




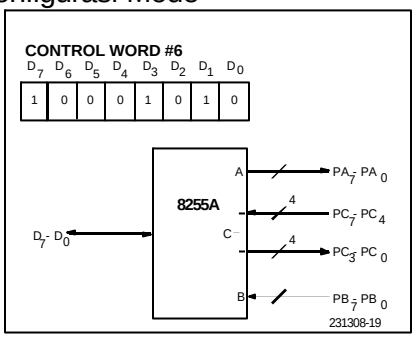


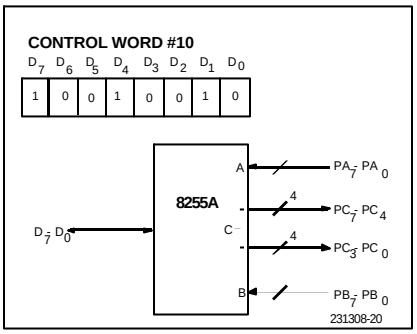


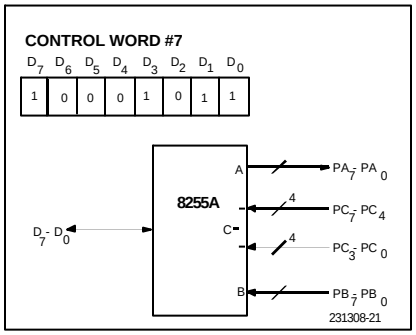


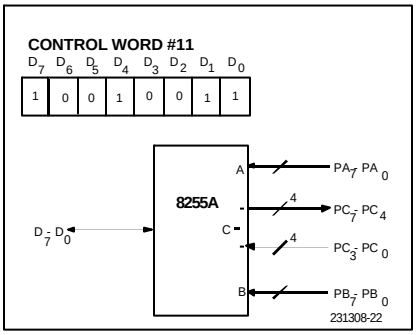


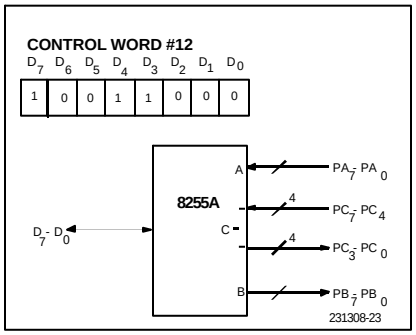
Konfigurasi Mode

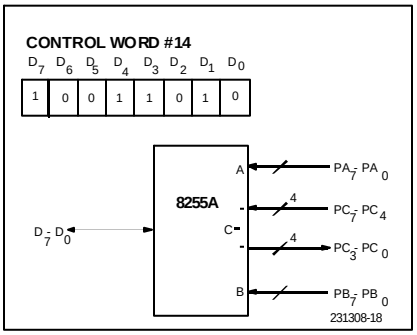


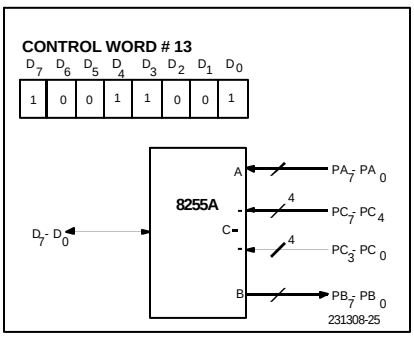


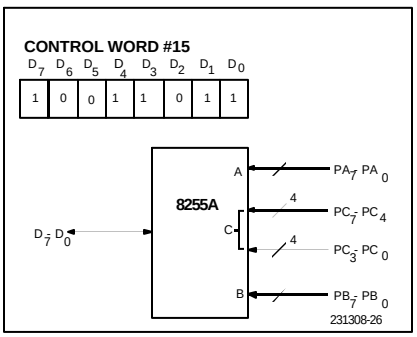












Perencanaan Minimal Sistem Z - 80

Rangkaian minimal sistem yang kita rencanakan dibawah ini meliputi :
A - Blok CPU Z - 80
B - Blok penyimpan
C - Blok masukan keluaran (input - Output).

Blok penyimpan data program dan blok masukan keluaran telah kita bahas pada pengalamatan penyimpan RAM - EPROM pengalamatan PPI 8255.

A. Blok CPU Z - 80 terdiri dari :
1. Bus data
2. Bus alamat
3. Bus kontrol

Bus kontrol terdiri dari :
- Kontrol sistem
- Kontrol CPU
- Kontrol Bus CPU

Dalam rangkaian minimal sistim Z - 80 ini, tidak semua pin dari CPU Z - 80 kita pergunakan hanya beberapa pin/fungsi yang kita pergunakan antara lain.

a. Bus data (D1 - D0) , Input Output.

Bus ini dipakai sebagai penghantar data 8 bit baik yang dari CPU ke penyimpan data (memori) atau masukan keluaran (Input - Output), atau sebaliknya.

Arah jalannya sinyal pada bus ini, masuk ke CPU atau keluar dari CPU.

b. Bus alamat (A15 - A0), output.

Dengan bantuan bus alamat ini, CPU dapat memilih lokasi penyimpan data/memori atau lokasi register dari masukan/keluaran (Input Output) yang berbeda-beda. Arah jalannya sinyal pada bus ini, keluar dari CPU.

c. Bus control.

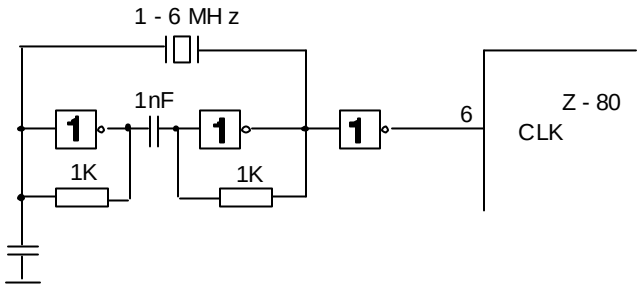
Terdiri dari 13 saluran, yang arahnya masuk atau keluar ke / dari CPU. Masing sinyal kontrol ini dijelaskan sebagai berikut :

Saluran detak (CLK), Input :

Sinyal detak diumpankan ke CPU melalui saluran CLK. Dengan demikian CPU dan juga bus sistim akan bekerja mengikuti sinyal detak.

CPU Z - 80 dapat bekerja baik dengan frekuensi detak 1 MHz - 6 MHz.

Rangkaian pembangkit detak :



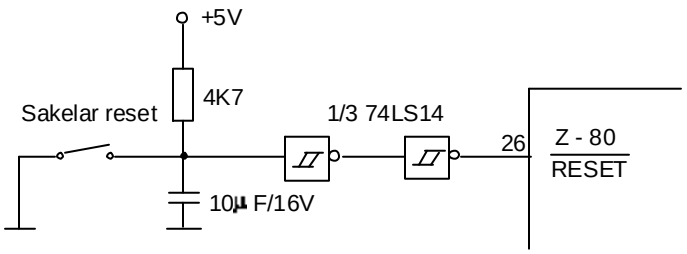
Gambar 7.41 Rangkaian pembangkit detak

Saluran Reset (RESET) , Input :

Sinyal " LOW " diberikan pada saluran ini, maka penghitung Program (PC) akan diset dengan nilai 0000H dan Interupt Enable (Pengaktif Interupt) akan direset.

Bila kemudian sinyal " High " diberikan pada saluran ini, maka program akan berjalan mulai dari alamat penyimpan program 0000H (Start awal)

Rangkaian pembangkit Input Reset



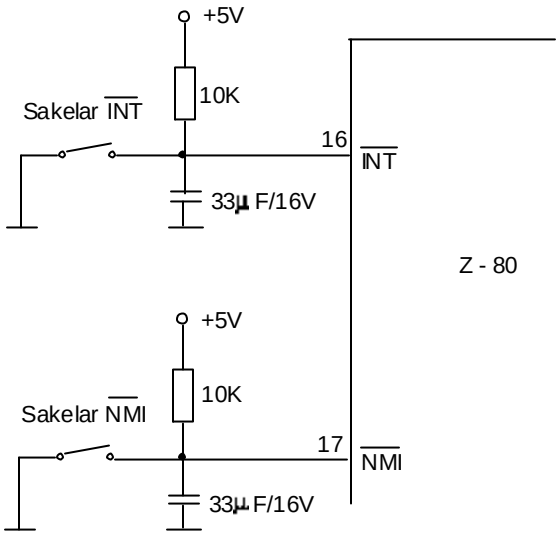
Gambar 7.42 Rangkaian pembangkit Input Reset

Saluran Penyela ($\overline{\text{INT}}$ dan $\overline{\text{NMI}}$), Input :

Sinyal " Low " yang diberikan pada saluran $\overline{\text{INT}}$, memungkinkan untuk melaksanakan salah satu dari 3 penyelaan yang tersebunyi yaitu dalam modus 0, 1 dan 2.

Sedang sinyal " Low " pada saluran $\overline{\text{NMI}}$, CPU akan menjalankan program bagian yang berada pada alamat penyimpan program (RAM EPROM) 0066H.

Rangkaian Pembangkit Sinyal $\overline{\text{INT}}$ dan $\overline{\text{NMI}}$



Gambar 7.43 Rangkaian Pembangkit Sinyal $\overline{INT}$ dan $\overline{NMI}$

Saluran $\overline{MREQ}$, Output

Apabila CPU membaca op-code (kode operasi) yang berhubungan dengan alamat penyimpanan data/program (memori), maka CPU akan mengaktifkan saluran $\overline{MREQ}$, $\overline{MREQ}$ = "Low".

Sinyal $\overline{MREQ}$ ini dipakai bersama dengan data alamat (A_{15}-A_0) untuk menunjuk alamat memori yang diinginkan.

Saluran $\overline{IOREQ}$, Output

Apabila CPU membaca op code yang berhubungan dengan unit masukan keluaran (PI0),

maka CPU akan mengaktifkan sinyal pengontrol masukan keluaran (Input - Output) $\overline{IOREQ}$, $\overline{IOREQ}$ = " Low ".Sinyal $\overline{IOREQ}$ dipakai bersama dengan data alamat (A_7-A_0) untuk menunjuk alamat Port Input Output yang diinginkan.

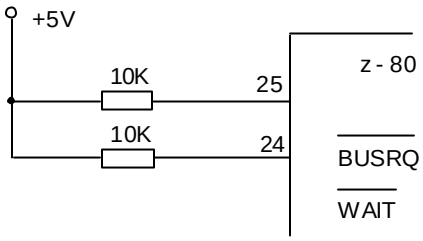
Saluran $\overline{RD}$, Output

CPU akan mengaktifkan saluran $\overline{RD}$, $\overline{RD}$ = " Low ", selama CPU melaksanakan perintah untuk membaca data dari lokasi penyimpanan data/program (memori) atau register Input Output (port).

Saluran $\overline{WR}$, Output

CPU akan mengaktifkan saluran $\overline{WR}$, $\overline{WR}$ = " Low " , selama CPU melaksanakan perintah untuk menulis data ke lokasi penyimpanan data/program (memori) atau register Input output (port).

Untuk saluran - saluran lain pada CPU Z - 80, yang jalannya sinyal sebagai Input, harus diaktifkan sesuai fungsinya dalam sistim.



Gambar 7.44 Rangkaian Pembangkit Sinyal

Untuk saluran - saluran lain yang arah jalannya sinyal sebagai output, biarkan tak terhubung kemana - mana.

Sistim Pengalamatan

Kejelasan yang sistimatik tentang cara pengalamatan sangat penting pada pengolahan data dalam jenis Prosesor, sebab program yang disusun tanpa penggunaan pengalamatan yang pasti, maka program tersebut menjadi kurang efektif dalam penganalisaannya.
Semakin panjang kode operasi sebuah perintah, maka dapat juga dikombinasikan banyak cara pengalamatannya.
Pada dasarnya cara pengalamatannya dapat dibagi menjadi 4 cara yang berbeda.

- **IMMEDIATE (SEGERA)**

Kode mesin mengandung konstanta untuk segera /langsung di akses

- **DIRECT (LANGSUNG)**

Kode mesin mengandung Register, alamat penyimpan atau alamat masukan / keluaran dari operan untuk diakses

- **INDIRECT (TIDAK LANGSUNG)**

Kode mesin mengandung hanya satu petunjuk, dimana untuk mendapatkan alamat dari operan yang akan diakses

- **TERINDEKS**

Alamat-alamat dari operan yang akan di akses dibentuk dalam beberapa bagian.

I. PENGALAMATAN IMMEDIATE

Disini operan yang akan diakses langsung terkandung pada kode mesin, ini adalah cara yang sangat sederhana, untuk mengisi konstanta ke Register atau lokasi penyimpanan . Tentu saja operan tidak dapat diubah lagi, maka kode mesin yang demikian kebanyakan disimpan di ROM . Kode operasi hanya dapat mengandung satu petunjuk tentang panjang dari operan yang mengikutinya . Selain itu bagian alamat masih harus mengandung sebuah keterangan tentang tujuan dimana konstanta harus dihubungkan kepadanya.

II. PENGALAMATAN LANGSUNG (DIRECT)

Disini kode mesin mengandung sebuah atau lebih alamat-alamat yang kemudian isi dari alamat-alamat ini akan diakses lebih lanjut. Panjang alamat-alamat ini dapat berbeda menurut keadaan apakah itu mengenai sebuah Register, alamat penyimpan atau alamat masukan/keluaran, perintah dapat mengandung sebuah petunjuk, apakah bagian pertama diberikan sebagai alamat tujuan atau sumber.

Contoh : **LD A, (1234H)**

adr	0	0	1	1	1	0	1	0	
adr + 1	0	0	1	1	0	1	0	0	Alamat sumber penyimpan 16
adr + 2	0	0	0	1	0	0	1	0	BIT

Contoh : **INC L**

adr	0	0	1	0	1	1	0	0	Op-kode yang mengandung alamat Register

III. INDIRECT (TIDAK LANGSUNG)

Kode mesin hanya mengandung sebuah petunjuk, dimana cara untuk mendapatkan alamat dari operan yang akan diakses . Petunjuk dapat terjadi pada sebuah register CPU atau pada sebuah lokasi penyimpan dan disana alamat efektif akan didapatkan untuk semua pengalamatan secara tidak langsung dengan Register , memberikan kode mesin yang pendek . Karena sebagai pengganti alamat 16 BIT diberikan hanya 3 BIT alamat Register. Perintah semacam itu, pada Z 80 :

Contoh : **PUSH** dan **POP** menaikkan atau menurunkan Register alamat yang dipakai secara otomatis dan terjadi apakah pada sebelum atau sesudah pelaksanaan perintahnya.

Pengalamatan tidak langsung, dapat di bagi :

Contoh : **RRC (HL)**

adr	1	1	0	0	1	0	1	1	Petunjuk pada HL Register
adr + 1	0	0	0	0	1	1	1	0	Alamat

Contoh : **PUSH DE**

adr	1	1	0	1	0	1	0	1	Pengalamatan langsung dari Register DE, isi dari Register SP dipakai sebagai alamat

III. TERINDEKS

Disini alamat efektif disusun dari beberapa bagian yang mana bagian-bagian ini dapat berasal dari Register-Register CPU, Register masukan/keluaran atau dari lokasi penyimpan.

Contoh : **LD E, (IX + 12)**

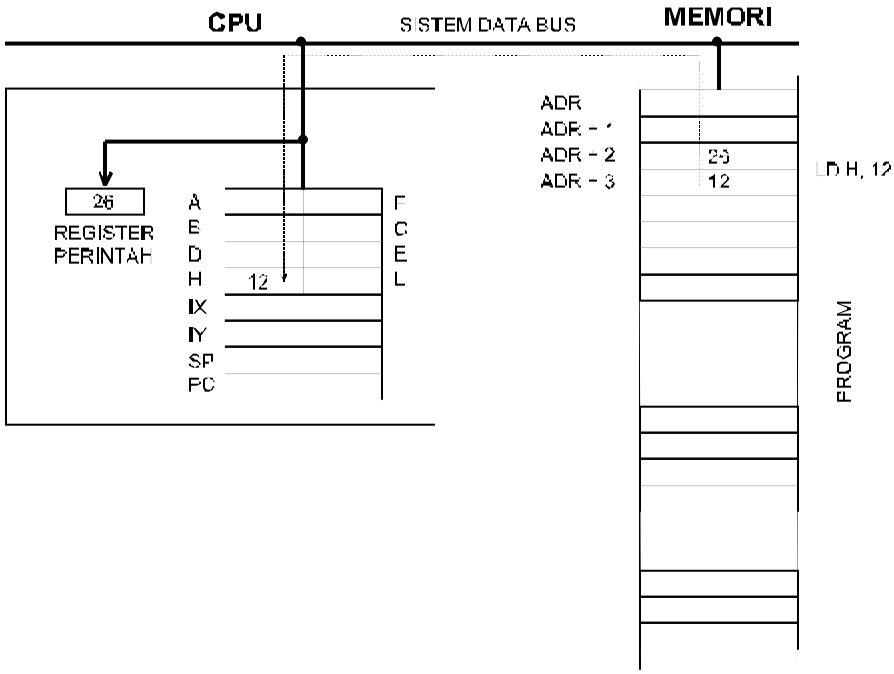
adr	1	1	0	1	1	1	0	1	Konstanta 12H ditambah
adr + 1	0	1	0	1	1	1	1	0	dengan isi Register IX
adr + 2	0	0	0	1	0	0	1	0	

Perintah Transfer

Untuk tranport data hanya dapat dilaksanakan kemungkinan-kemungkinan dibawah ini :
Register CPU ⇔ Register CPU
Register CPU ⇔ Memori
Register CPU ⇔ Register Input – Output

PERINTAH TRANSFER DENGAN PENGALAMATAN IMMEDIATE

Perintah ini melakukan kemungkinan sederhana untuk mengisi Register CPU 8 bit atau 16 bit dengan sebuah konstanta.



Gambar 7.45 Peta Memory dan Register pada Pengalamatan Immediate

Sebuah Register CPU diisi dengan konstanta yang mengikuti op code pada Memori

MNEMONIK

LD r,n

r = Register CPU (8 bit) A,B,C,D,E,F,H,L
n = Data (8 bit)

LD rr, nn

rr = Register CPU (16 bit) BC,DE,HL,IX,IY,SP
nn = Data (16 bit)

OPERASI

r ← n

rr ← nn

Register CPU r atau rr diisi dengan konstanta n atau nn yang mengikuti kode mesin

FORMAT

LD r, n

adr	0	0		r		1	1	0
adr + 1	**KONSTANTA**							

LD rr, nn

adr	0	0	rr		0	0	0	1
adr + 1	**KONSTANTA LSB**							
adr + 2	**KONSTANTA MSB**							

adr = Alamat Memori

r dan rr = dapat berupa :

A = 111	D = 010	L = 101	BC = 00	SP = 11
B = 000	E = 011		DE = 01	
C = 001	H = 100		HL = 10	

FLAG

Tidak berpengaruh

B. PERINTAH TRANSFER DENGAN PENGALAMATAN LANGSUNG

Pada pengalamatan langsung, alamat sumber dan tujuan, berada pada kode mesin. Alamat register atau memori dapat ditulis langsung. Perintah transfer pengalamatan langsung dapat dibagi menjadi :

- Pengalamatan Register - Register
- Pengalamatan Register – Memori

1. PENGALAMATAN REGISTER - REGISTER

Perintah ini mengakibatkan transfer data dari sebuah Register CPU ke Register CPU yang lain.

MNEMONIK

LD r,r'

OPERASI

r ← r'

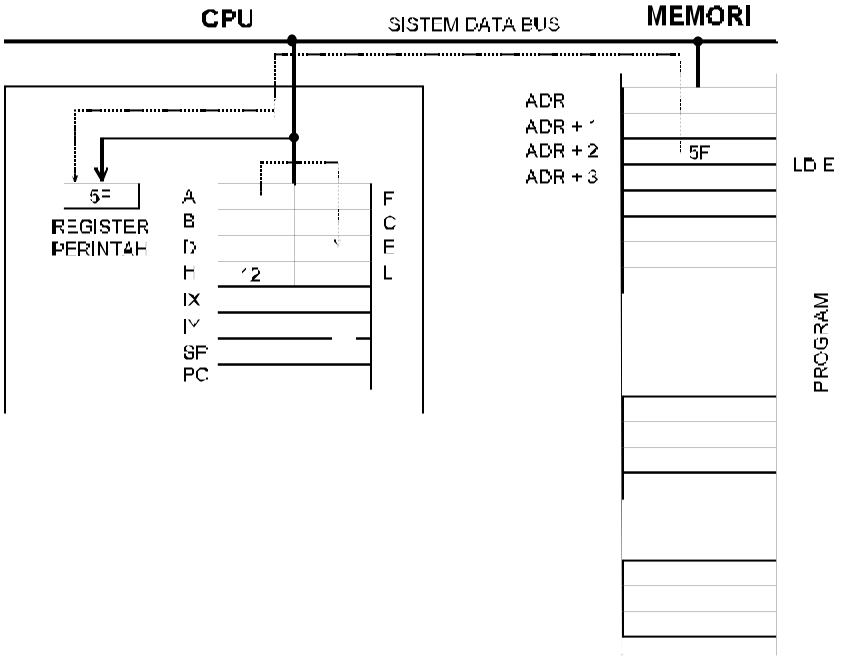
Register CPU tujuan r diisi dengan isi Register CPU sumber r'

FORMAT

0	1		r			r'	

FLAG

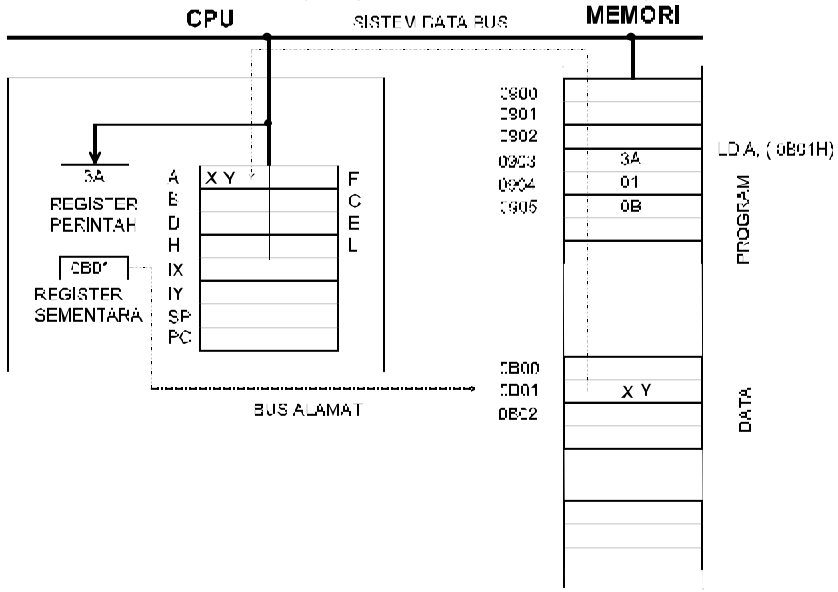
Tidak terpengaruh



Gambar 7.46 Peta Memory dan Register pada Pengalamatan Langsung Register-Register

2. PENGALAMATAN REGISTER - MEMORI

Jenis pengalamatan ini, melaksanakan transfer antara Register CPU dan lokasi Memori yang dituliskan di belakang kode mesin.



Gambar 7.48 Peta Memory dan Register pada Pengalamatan Langsung Register-Memori

Sebuah Register CPU diisi dengan isi dari lokasi memori yang dihasilkan di belakang mesin.

MNEMONIK

Register CPU ← Memori, Memori ← Register CPU

LD A, (nn) **LD (nn), A**

LD rr, (nn) **LD (nn), rr**

rr = Register CPU 16 bit (BC, DE, HL, SP)

(nn) = Isi dari tanda ini selalu diisi oleh alamat Memori dan tanda ini berarti isi dari alamat Memori yang ditunjuk oleh nn.

OPERASI

Register A CPU ← Memori

Register CPU diisi oleh isi dari lokasi Memori yang alamatnya di tunjuk oleh alamat dalam tanda kurung.

Memori ← Register A CPU

Memori yang alamatnya ditunjuk oleh alamat dalam tanda kurung diisi oleh isi Register A CPU.

Register CPU ← Memori

Register CPU 16 bit diisi oleh isi dari memori yang alamatnya ditunjuk oleh alamat dalam tanda kurung.

Memori ← Register CPU 16 Bit

Lokasi Memori yang alamatnya ditunjuk oleh alamat dalam tanda kurung diisi oleh isi dari Register CPU 16 Bit.

FORMAT

LD A, (nn)

adr	0	0	1	1	1	0	1	0
adr + 1	**Alamat LSB**							
adr + 2	**Alamat MSB**							

LD (nn), A

0	0	1	1	0	0	1	0	
Alamat LSB								
Alamat MSB								

LD rr, (nn)

adr	1	1	1	0	1	1	0	1
adr + 1	0	1	rr		1	0	1	1
adr + 2	**Alamat LSB**							
adr + 3	**Alamat MSB**							

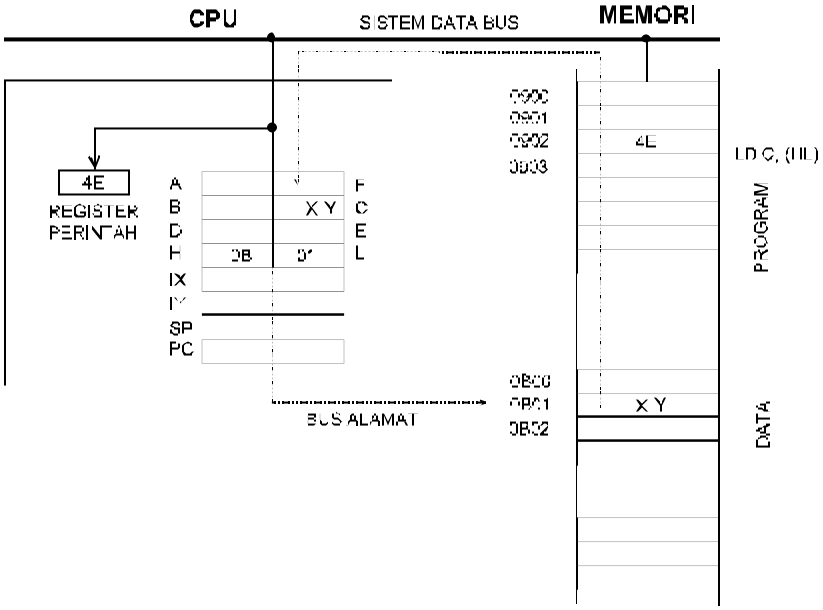
LD (nn), rr

1	1	1	0	1	1	0	1	
0	1	rr		0	0	1	1	
Alamat LSB								
Alamat MSB								

FLAG

Tidak terpengaruh.

PERINTAH TRANSFER DENGAN PENGALAMATAN TIDAK LANGSUNG MELALUI PAS ANGAN REGISTER



Gambar 7.49 Peta Memory dan Register pada Pengalamatan Tidak Langsung Melalui Register Pasangan

Sebuah Register CPU diisi oleh isi sebuah lokasi Memori yang ditunjuk oleh pasangan Register CPU.
Keuntungan dari jenis pengalamatan ini adalah :

- Kode mesin lebih pendek seperti pada pengalamatan langsung sehingga kebutuhan memori program sedikit dan pelaksanaan perintah lebih cepat.
- Data alamat dapat mudah dimanipulasi.

MNEMONIK

LD r, (rr) Register CPU ← Memori
LD (rr), r Memori ← Register CPU
r = Register CPU 8 bit (A,B,C,D,E,H,L)
rr = Regiser CPU 16 bit
HL : dapat dipergunakan untuk semua Register CPU
BC,DE : Hanya dapat dipergunakan untuk akkumulator

OPERASI

r ← (rr)
Register CPU r diisi oleh isi dari lokasi Memori yang ditunjuk oleh pasangan Register CPU rr

(rr) ← r
Lokasi Memori yang ditunjuk oleh pasangan Register CPU rr diisi oleh isi Register CPU r

FORMAT

LD (HL), r

0	1	1	1	0		r	

LD r, (HL)

0	1		r		1	1	0

LD (BC),A

0	0	0	0	0	0	1	0

LD A, (BC)

0	0	0	0	1	0	1	0

LD (DE), A

0	0	0	1	0	0	1	0

LD A, (DE)

0	0	0	1	1	0	1	0

LD (HL), n

0	0	1	1	0	1	1	0
KONSTANTA							

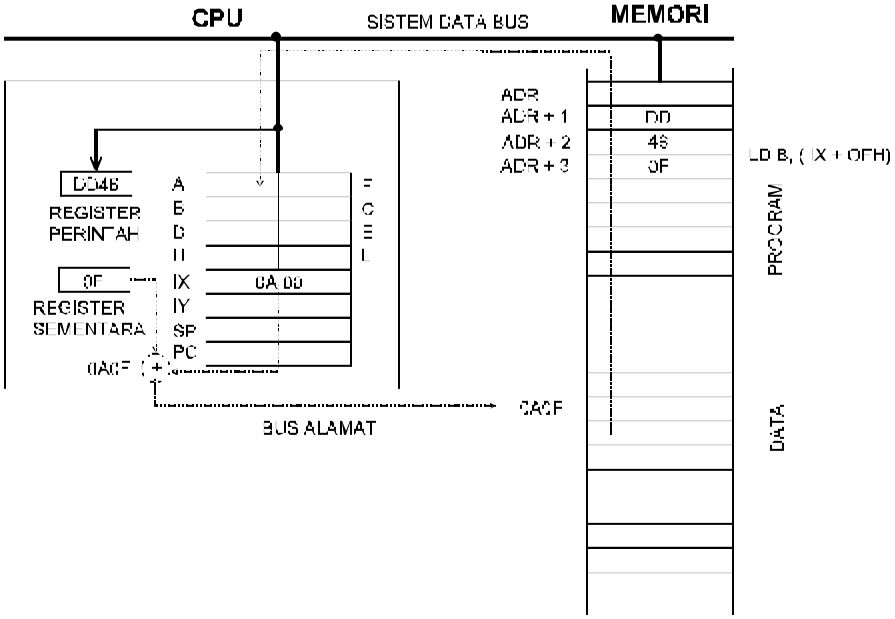
FLAG

Tidak terpengaruh.

D. PERINTAH TRANSFER DENGAN PENGALAMATAN TIDAK LANGSUNG MELALUI REGISTER + OFFSET

Dipergunakan untuk transfer data 8 bit antara Register, CPU dan Memori.

Penunjukkan yang tepat sebenarnya adalah Register tidak langsung + offset. Sebagai Register alamatnya biasa dipakai Register index IX dan IY. Untuk itu Register index ini harus diisi terlebih dahulu dengan alamat basis yang diinginkan.



Gambar 7.50 Peta Memory dan Register pada Pengalamatan Tidak Langsung Melalui Register + Offset

MNEMONIK :

LD r, (IR + e) Register CPU ← Memori

LD (IR + e), r Memori ← Register CPU

LD (IR + e), n

r = Register CPU 8 bit (A,B,C,D,E,HL)
IR = Register Index IX dan IY
e = Jarak,offset,konstanta 8 bit
n = Konstanta 8 bit

OPERASI

Register CPU ← Memori
Register CPU r diisi oleh isi dari lokasi Memori yang ditunjuk oleh isi Register Index + offset
Memori ← Register CPU
Lokasi memori yang ditunjuk oleh Register Index + offset diisi oleh register CPU r

FORMAT

LD r, (IX + e)

adr	1	1	0	1	1	1	0	1
adr + 1	0	1		r		1	1	0
adr + 2	KONSTANTA e							

LD (IX + e), r

	1	1	0	1	1	1	0	1
	0	1	1	1	0		r	
	KONSTANTA e							

LD r, (IY + e)

adr	1	1	1	1	1	1	0	1
adr + 1	0	1		r		1	1	0
adr + 2	KONSTANTA e							

LD (IY + e), r

	1	1	1	1	1	1	0	1
	0	1	1	1	0		r	
	KONSTANTA e							

LD (IX + e), n

1	1	0	1	1	1	0	1
0	0	1	1	0	1	1	0
KONSTANTA e							

LD (IY + e)

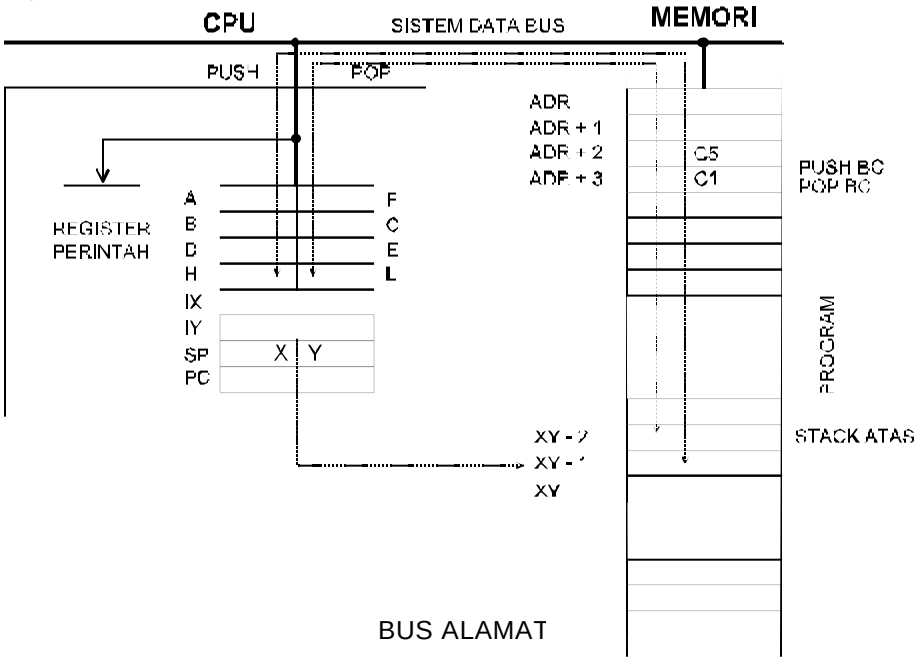
1	1	1	1	1	1	0	1
0	0	1	1	0	1	1	0
KONSTANTA e							
KONSTANTA n							

PERINTAH TRANSFER DENGAN PENGALAMATAN STACK

Perintah ini memungkinkan pemrograman untuk menyimpan isi Register CPU pada Memori sementara dan untuk pemrosesan secara sederhana pada blok data.

Pengalamatan Stack adalah prinsip pada pengalamatan Register CPU 16 bit.

Contoh : Stack pointer (SP, Stack pointer). SP secara otomatis akan dinaikkan atau akan diturunkan 2, setelah pembacaan atau penulisan pada Stack.



Gambar 7.51 Peta Memory dan Register pada Pengalamatan Stack

MNEMONIK

POP rr Register CPU ← Memori

PUSH rr Memori ← Register CPU

rr = Register CPU 16 bit (AF,BC,DE,HL,IX,IY)

OPERASI :

Register alamat SP ini mempunyai sifat yang sangat praktis yaitu sebelum penyimpanan sebuah byte oleh perintah push, isi dikurangi 1 dan setelah pembacaan sebuah byte oleh perintah POP, isi SP ditambah 1.

Proses penambahan dan pengurangan isi SP dilakukan secara otomatis oleh block pemroses perintah.

FORMAT :

POP rr

1	1	rr	0	1	0	1

PUSH rr

1	1	rr	0	0	0	1

PUSH

SP: = SP -1
(SP) = Register 16 bit tinggi
SP : = SP -1
(SP) Register 16 bit lebih rendah

POP

Register 16 bit rendah : = (SP)
SP : = SP + 1
Register 16 bit tinggi : = (SP)
SP : = SP + 1

SP selalu berisikan alamat Memori terakhir yang sedang aktiv setelah pelaksanaan perintah PUSH atau POP.

Contoh :

Register CPU BC berisikan konstanta bbccH

PUSH BC isi BC di simpan pada Stack

POP BC isi Stack di ambil dari stack

PROGRAM

0800	C	5
0807	C	1

0FFC		
0FFD	c	c
0FFE	b	b
0FFF		

POP PUSH

F. PERINTAH TRANSFER DENGAN PERTUKARAN DATA

Dengan kelompok perintah ini, tidak sama seperti register tujuan di isi dengan Register sumber, (isi pada register sumber tidak berubah), tetapi pada perintah ini, isi kedua Register saling bertukar.

MNEMONIK :

EX DE, HL
EX (SP), HL

FORMAT :

EX DE, HL

1	1	1	0	1	0	1	1

EX (SP), HL

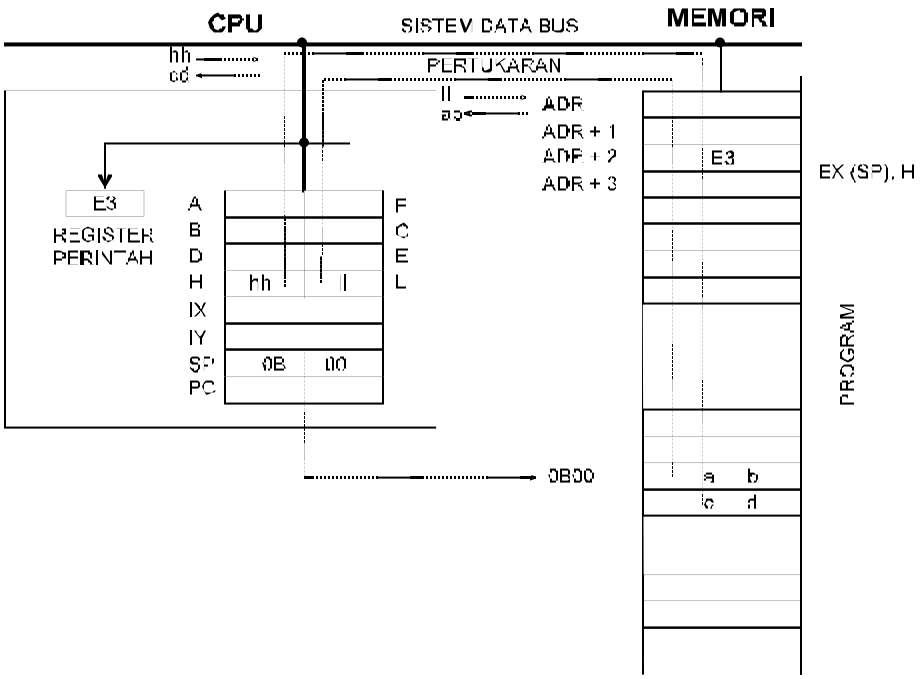
1	1	1	0	0	0	1	1

OPERASI :

Isi pasangan Register saling dipertukarkan
D = H
E = L
SP = H
SP + 1 = L

FLAG

Tidak terpengaruh



Gambar 7.52 Peta Memory dan Register pada Pertukaran Data Register Pasangan

G. PERINTAH UNTUK INPUT OUTPUT DENGAN PENGATURAN LANGSUNG

Perintah ini mengontrol lalu lintas bus data antara CPU dan piranti input output.

Pada sebagian sistim mikroprosessor sering dilengkapi dengan banyak blok input output dengan sebuah atau lebih register, yang melalui fungsi blok input output ini dipakai sebagai pelayanan penyangga data pada Register sementara atau pengaturan informasi kontrol.

Register pada piranti ini,yang dipakai sebagi penghubung sistim dengan dunia luar (peripheral) disebut sebagai “port”, dan alamatnya disebut “alamat port”. Jumlah alamat yang dipakai oleh perintah input output ini hanya 8 bit dan dalam pelaksanaannya diberikan melalui jalur penghantar A7 - A0.

Dengan demikian dapat dibentuk 256 alamat port yang berbeda.

MNEMONIK :

IN A, (n)	CPU ← Register I/O
OUT (n), A	Register I/O ← CPU

n : konstanta 8 bit

FORMAT :

	IN A, (n)							
adr	1	1	0	1	1	0	1	1
adr + 1	**Konstanta**							

	OUT (n), A							
	1	1	0	1	0	0	1	1
	Konstanta							

OPERASI :

A, (n)

Akku CPU diisi dengan isi register I/O yang beralamat n

(n), A

Register I/O yang beralamat n diisi oleh isi dari Akku CPU

FLAG

Tidak terpengaruh

CONTOH :

IN A,(20H)	0900	DB
	0901	20

H. PERINTAH UNTUK INPUT OUTPUT DENGAN PENGALAMATAN TIDAK LANGSUNG

MNEMONIK :

IN r, (C) CPU Register I/O

OUT (C), r Piranti I/O CPU

r = Register 8 bit A,B,C,D,E,H,L

C = Register C yang isinya diberikan sebagai penunjuk penghantar alamat A7 - A0

FORMAT

IN r, (C)

adr	1	1	1	0	1	1	0	1
adr + 1	0	1		r		0	0	0

OUT (C), r

	1	1	1	0	1	1	0	1
	0	1		r		0	0	1

OPERASI

Register CPU tujuan r diisi dengan isi dari Register C yang merupakan pengalamatan dari port I/O

Register C yang merupakan pengalamatan dari port I/O diisi dengan isi dari Register CPU r.

FLAG :

Pada perintah input (IN)

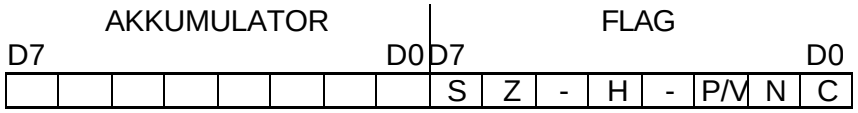
Flag S = 1, bila bit tertinggi = 1

Flag Z = 1, bila data yang dibaca = 0

Flag P = 1, pada parity genap dari data yang di baca

Register Flag

Flag adalah sebuah flip-flop di dalam blok penghitung dari CPU dan disebut sebagai Register Flag.
Keadaan Flag ini setelah pelaksanaan sebuah perintah (yang mempengaruhi Flag) akan menghasilkan sifat dari hasil sebuah operasi.
Pada Z-80 , Flag di pasangkan dengan Akkumulator dan dikenal dengan program status wort (PSW).



A. FLAG ZERO

Itu menunjukkan, apakah pada pelaksanaan terakhir ini operasi hasil pada semua bit adalah = 0.

Kondisi Flag

Flag Zero **= 1,** bila pada semua bit pada Register hasil = 0.
Flag Zero **= 0,** bila semua bit pada Register hasil $\neq$ 0.
Contoh :

```
  0000
  0000 +
 1 0000
```

Flag Zero = 1
Flag Carry = 1

B. FLAG CARRY

Itu menunjukkan apakah pada proses operasi sebuah bit carry dipindahkan dari bit tertinggi MSB pada Register hasil , itu dapat terjadi pada operasi :

Penjumlahan, bila hasil dari 8 bit atau 16 bit.
Pengurangan a - b, bila b>a, hasil juga negatif.
Pergeseran, bila nilai 1 pada bit tertinggi atau terendah di geserkan ke carry.

Kondisi Flag

- **Flag Carry** **= 1**, bila terjadi Carry.
- **Flag Carry** **= 0**, bila tidak terjadi Carry.

Flag Carry dapat set melalui perintah SCF dan dibalik melalui perintah SCF.

C. FLAG SIGN

Pada operasi yang mempengaruhi Flag, Flag sign menyimpan kondisi bit tertinggi dari Register hasil, jadi :

Kondisi Flag

- **Flag Sign** **= 1**, bila bit tertinggi dari Register hasil = 1
- **Flag Sign** **= 0**, bila bit tertinggi dari Register hasil = 0

D. FLAG PARITY/OVERFLOW

Bit ke 2 dari Register Flag mempunyai 4 arti yang berbeda, tergantung dari hasil akhir pelaksanaan operasi .

1. FLAG OVERFLOW

Pengertian ini berlaku setelah pelaksanaan perintah berikut :

a). **ADD, ADC, SUB, SBC.**
b). **INC, DEC**

Flag overflow diset 1 pada proses perpindahan dari bit ke 7 ke bit ke 8, yaitu yang
mempengaruhi tanda bilangan positif atau negatif pada perhitungan bilangan.

2. FLAG PARITY

Pengertian ini berlaku setelah pelaksanaan perintah berikut ini :

a). Perintah logika **AND, OR, XOR**
b). Perintah geser **RL, RR, RLC, RRC**
SLA, SRA, SRL
RLD, RRD
c). Aritmatik BCD **DAA**

Perintah input dengan pengalamatan tidak langsung **IN r, (C)**.

Kondisi Flag

- **Flag Parity** **= 1**, bila jumlah 1 dan hasil akhir operasi adalah genap
- **Flag Parity** **= 0**, bila jumlah 1 dan hasil akhir operasi adalah ganjil

3. PENUNJUKAN NOL PADA PERINTAH BLOK

Pada perintah berikut untuk transfer blok dan pengamatan blok, Flag P?V menunjukan keadaan
Register BC, yang pada operasi ini dipakai sebagai Register penghitung.

a). Transfer blok **LDI, LDIR, LDD, LDDR**
b). Pengamatan blok **CPI, CPIR, CPD, CPDR**

Itu berlaku :

- **Flag P/V** **= 0,** bila Register penghitung BC = 0000H
- **Flag Parity** **= 1,** bila Register penghitung BC ≠ 0000H

4. PENUNJUKAN DARI PENGAKTIF FLIP-FLOP INTRUPSI (IFF2)
Ini adalah kemungkinan satu-satunya, IFF2, yang menunjukkan keadaan yang sah untuk proses interupt , untuk membaca. Penggunaan dari Flag P/V ini , berlaku untuk perintah **LD A,I** dan **LD A,R**

Perintah Aritmatika

1. Operasi Aritmatika dengan operasi 8 bit

Operasi ini mempunyai format seperti dibawah ini :
ADD : hasil = operan 1 + operan 2
SUB : hasil = operan 1 - operan 2
ADC : hasil = operan 1 + operan 2 + carry
SBC : hasil operan 1 - operan 2 - carry
Kedua operasi disepakati sebagai bilangan biner dan operasinya berlangsung didalam ALU dan pengalih bilangan setiap instruksi berlaku 2 operan saja.

Mnemonik :

ADD A, r	SUB r	ADC A, r	SBC A, r

Operasi :

Isi dari akku dan register CPU disepakati sebagai bilangan biner 8bit dan saling ditambahkan.
Hasil berada pada akkumulator.

A = A+r	A = A-r	A = A+r+C	A = A-r-C

Isi akku yang baru adalah isi akku yang lama + atau isi register CPU r

Pada perintah ini, hasil masih ditambahkan pula isi dari flag carry
Contoh :

Operan 1	80 H
Operan 2	+ 20 H
Hasil sementara	A0 H
Carry flag	+ 01 H
Hasil akhir	A1 H

FORMAT

Dalam penjelasan format perintah bentuk ini, kita kelompokkan atas jenis pengalamatan

a. Register dengan regisater CPU A, B, C, D, E, H, L.

1	0	1	0	0	r	r	r	ADD A, r

1	0	0	0	1	r	r	r

SUB r

1	0	0	0	1	r	r	r

ADC A, r

1	0	0	1	1	r	r	r

SBC A,r

r = Pengalamatan register CPU A = 111 E = 011
B = 000 H = 100
C = 011 L = 101
D = 010

Konstanta 8 bit

Sebagai operan 2 digunakan konstanta yang penulisanya mengikuti Op - Code

1	1	0	0	0	1	1	0
Konstanta 8 bit							

ADD A, n

1	1	0	1	0	1	1	0
Konstanta 8 bit							

SUB n

1	1	0	0	1	1	1	0
Konstanta 8 bit							

ADC A, n

1	1	0	1	1	1	1	0
Konstanta 8 bit							

SBC A, n

c. Register tidak langsung

Sebagai operan 2 digunakan isi dari lokasi memori yang ditunjukkan melalui register CPU 16 bit HL

1	0	0	0	0	1	1	0

ADD A, (HL)

1	0	0	1	0	1	1	0

SUB (HL)

1	0	0	0	1	1	1	0

ADC A, (HL)

1	0	0	1	1	1	1	0

SBC A, (HL)

d. Indeks offset

Sebagai operan 2 digunakan isi dari lokasi memori yang ditunjukkan melalui register index + offset e

1	1	0	1	1	1	0	1
1	0	0	0	0	1	1	0

ADD A, (IX+e)

1	1	1	1	1	1	0	1
1	0	0	0	0	1	1	0

ADD A, (IY+e)

Konstanta offset e

Konstanta offset e

1	1	0	1	1	1	0	1	SUB
1	0	0	1	0	1	1	0	(IX+e)
Konstanta offset e								

1	1	1	1	1	1	0	1	SUB
1	0	0	1	0	1	1	0	(IY+e)
Konstanta offset e								

1	1	0	1	1	1	0	1	ADC A,
1	0	0	0	1	1	1	0	(IX+e)
Konstanta offset e								

1	1	1	1	1	1	0	1	ADC A,
1	0	0	0	1	1	1	0	(IY+e)
Konstanta offset e								

1	1	0	1	1	1	0	1	SBC A,
1	0	0	1	1	1	1	0	(IX+e)
Konstanta offset e								

1	1	1	1	1	1	0	1	
1	0	0	1	1	1	1	0	SBC A, (IY+e)
Konstanta offset e								

FLAG

Semua Flag terpengaruh

2. Operasi Aritmatika dengan Operan 16 Bit

Mnemonik

ADD HL, rr rr = BC, DE, HL, SP	ADD IX, rr rr = BC, DE, SP, IX	ADD IY, rr RR = BC,DE,SP,IY

Operasi

HL = HL + rr IX = IX + rr IY = IY + rr

Isi dari register CPU yang baru (HL, IX atau IY) terdiri dari penjumlahan isi yang lama dengan isi register CPU yang lain.

Format

0	0	r	r	1	0	0	1

1	1	0	1	1	1	0	1
0	0	r	r	1	0	0	1

1	1	1	1	1	1	0	1
0	0	r	r	1	0	0	1

rr = Register 16 bit
BC = 00
DE = 01
HL = 10
SP = 11

BC = 00
DE = 01
HL = 10
IX = 11

BC = 00
DE = 01
HL = 10
IY = 11

Flag

Bit flag H, N dan C terpengaruh.

Juga pada aritmatika dengan operan 16 bit terdapat perintah yang melibatkan bit carry dalam perhitungan.

Mnemonik

ADC HL, rr
rr = register 16 bit CPU

SBC HL, rr
BC, DE, HL, SP

Operasi

HL = HL + rr + C

Isi yang baru dari HL terdiri dari hasil penjumlahan yang lama dari HL ditambah/dikurangi isi dari register 16 bit CPU dan isi bit carry.

Format

1	1	1	0	1	1	0	1
0	1	r	r	1	0	1	0

1	1	1	0	1	1	0	1
0	1	r	r	0	0	1	0

rr = Register 16 bit
BC = 00
DE = 01
HL = 10
SP = 11

Flag :

Semua bit flag terpengaruh.

Perintah Biner AND, OR, EX-OR dan CP

Kebanyakan mikroprosessor dapat melaksanakan biner seperti AND, OR, EX-OR dan CP.
Seperti pada operasi aritmatika operasi 8 bit berlaku bahwa pada awalnya sebuah operasi harus berada pada akku dan hasil operasi kembali berada pada akkumulator CPU, dengan demikian penulisan mnemonik dapat disingkat sebagai berikut :

Mnemonik :

AND r | OR r | XOR r | CP r

r = data 8 bit

Operasi :

A = A AND r | A = A OR r | A = A XOR r | A - r

Isi yang baru dari akku terdiri dari isi akku yang lama dihubungkan secara AND/OR/EX-OR dengan isi dari r.
Pada perintah compare sesuai operasi aritmatik pengurangan, dimana hasil tidak kembali berada pada akku.

Format :

Format penulisan bahasa mesinnya ditentukan berdasarkan :

a. Pengalamatan register (r) : A, B, C, D, E, F, H, L
Register

1	0	1	0	0	r	r	r	AND r
1	0	1	0	1	r	r	r	XOR r r
1	0	1	1	0	r	r	r	OR r
1	0	1	1	1	r	r	r	CP r

A = 111 B = 000 C = 001 D = 010
E = 011 H = 100 L = 101

b. Pengalamatan langsung dengan konstanta

1	1	1	0	0	1	1	0	AND n
Konstanta n								

1	1	1	0	1	1	1	0	XOR n
Konstanta n								

1	1	1	1	0	1	1	0	OR n
Konstanta n								

1	1	1	1	1	1	1	0	CP n
Konstanta n								

c. Pengalamatan tidak langsung dengan register CPU 16 bit (melalui pasangan register (HL))

1	0	1	0	0	1	1	0	AND (HL)

1	0	1	0	1	1	1	0	XOR (HL)

1	0	1	1	0	1	1	0	OR (HL)

1	0	1	1	1	1	1	0	CP (HL)

d. Pengalamatan langsung dengan pengalamatan terindex

1	1	0	1	1	1	0	1	AND
1	0	1	0	0	1	1	0	(IX+e)
Konstanta offset e								

1	1	1	1	1	1	0	1	AND
1	0	1	0	0	1	1	0	(IY+e)
Konstanta offset e								

1	1	0	1	1	1	0	1	XOR
1	0	1	0	1	1	1	0	(IX+e)
Konstanta offset e								

1	1	1	1	1	1	0	1	XOR
1	0	1	0	1	1	1	0	(IY+e)
Konstanta offset e								

1	1	0	1	1	1	0	1	OR
1	0	1	1	0	1	1	0	(IX+e)
Konstanta offset e								

1	1	1	1	1	1	0	1	OR
1	0	1	1	0	1	1	0	(IY+e)
Konstanta offset e								

1	1	0	1	1	1	0	1	CP
1	0	1	1	1	1	1	0	(IX+e)
Konstanta offset e								

1	1	1	1	1	1	0	1	CP
1	0	1	1	1	1	1	0	(IY+e)
Konstanta offset e								

FLAG :

Untuk operasi AND, OR dan EX-OR, semua bit flag terpengaruhi C = 0, H = 1, N = 0
Untuk operasi CP seperti pada operasi pengurangan SUB.

PENGGUNAAN PERINTAH BINER

- Perintah AND digunakan untuk menghapus bit yang diinginkan dari sebuah data 8 bit.
  Contoh :

LD B, C1H	B = 1 1 0 0 0 0 0 1	C1H
IN A, (20H)	A = X X X X X X X X	X = 0 atau 1
AND B	$\wedge$ =	
	A = X X 0 0 0 0 0 X	Hasil

  Bit 5 .....1 direset 0, yang lainnya tetap tidak berubah

- Perintah OR digunakan untuk mengeset bit yang dinginkan dari sebuah data 8 bit
  Contoh :

IN A, (20H)		A = X X X X X X X X	X = 0 atau 1
OR 3CH	konstanta	= 0 0 1 1 1 1 0 0	CH
		A = X X 1 1 1 1 X X	Hasil

- Perintah EX-OR digunakan untuk membalik bit yang diinginkan dari sebuah data 8 bit.
  Dengan perintah ini juga dapat dipakai untuk menyamakan 2 byte, bit-bit yang tidak sama pada akku akan ditunjukkan sebagai 1
  Contoh :

IN A, (20H)		A = 1 0 0 0 1 1 1 1	8FH
XOR 3CH	konstanta	= 1 1 0 0 0 0 1 1	C3H
		$\forall$ =	
A = 0 1 0 0 1 1 0 0		Hasil	

Perintah Putar dan Geser

Dengan perintah ini register 8 bit dan lokasi memori dapat digunakan sebagai sebuah register geser.Operasinya sendiri tentu dilaksanakan dalam ALU dari CPU pada register-register geser, isi masing-masing bit dapat digeser ketetangga Flip-flop kiri atau kanannya.

A. Perintah Putar

1. Putar ke kiri atau kanan dalam Register atau memori

Mnemonik
⇒ R L r

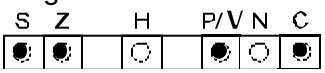
Mnemonik
⇒ R R r

r = Register CPU 8 bit atau isi memori 8 bit

Operasi :

Isi bit 7 dipindahkan ke flag carry dan isi bit Flag Carry dipindahkan ke bit 0 dan bit yang lain digeser 1 ke kiri.

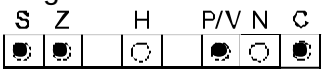
Flag :



Operasi :

Isi Flag Carry dipindahkan ke bit 7 dan isi bit 0 ke Flag Carry dan bit yang lain digeser 1 kekanan.

Flag :



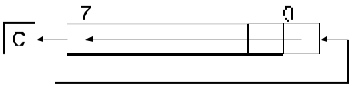
2. Putar ke kiri / kanan dalam register atau memori dan menuju flag carry.

Mnemonik :

⇒ RLC r

r = Register CPU 8 bit atau isi memori.

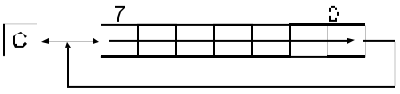
Operasi :



Mnemonik :

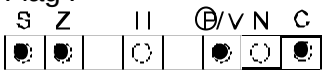
⇒ RRC r

Operasi :

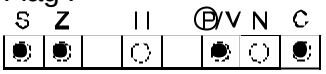


Informasi :
Bit akan bergeser 1 kekiri atau kekanan , tetapi isi flag carry tidak dalam lingkaran perputaran. Bit 7 berpindah ke bit 0 pada RLC, dan bit 0 berpindah ke bit 7 pada RRC.
Bit yang lain bergeser satu ke kiri → RLC.
Bit yang lain bergeser satu ke kanan → RRC.

Flag :



Flag :



3. Putar secara digit ke kiri / kanan

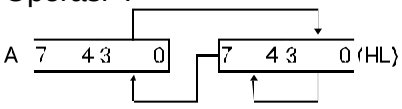
Mnemonik :

RLD (HL)

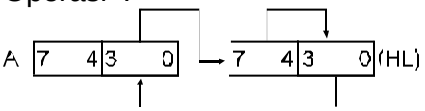
Mnemonik :

RRD (HL)

Operasi :

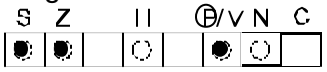


Operasi :

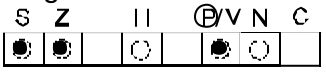


Ini adalah perintah untuk memutar 12 digit , dan 4 digit untuk pelaksanaan perintah.
Posisi perputaran selalu dibagi menjadi separuh dari akkumulator dan sebuah isi memori pengalamatannya melalui (HL).
Separuh bagian kiri dari Akku tidak berubah isinya.

Flag :



Flag :



B. Perintah Geser

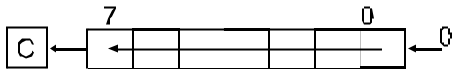
1. Geser ke kiri secara aritmetik dalam register atau memori

Mnemonik :

SLA r

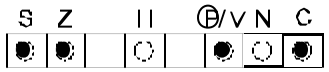
r = Register 8 bit atau isi memori

Operasi :



Informasi bit masing-masing digeser 1 (satu) ke kiri, isi bit 7 digeser ke flag carry, pada bit 0 diberikan 0 pada setiap pelaksanaan perintah ini. Bila isi dari register / memori adalah bilangan biner, maka masing-masing bit digeser 1 kekiri.
Perintah ini digunakan untuk operasi pengalian.

Flag :



2. Geser kekanan secara aritmetik dalam register / memori :

Mnemonik :

SRA r ; r = Register 8 bit atau isi memori

Operasi :

Informasi bit masing-masing , digeser 1 ke kanan , isi bit 0 digeser ke flag carry , bit 7 selain digeser 1 kekanan , isi bit 7 juga tidak berubah . Perintah ini digunakan untuk operasi pembagian untuk bilangan negatif atau pun bilangan positif.

3. Geser ke kanan secara logika dalam register / memori.

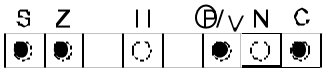
Mnemonik :

SRL r
r = Register 8 bit atau isi memori.

Operasi :

Setiap bit digeser 1 (satu) kekanan, isi bit 0 digeser ke flag carry pada bit 7 diberikan 0 pada setiap pelaksanaan perintah ini.

Flag



Format :

adr	1	1	0	0	1	0	1	1
adr + 1	0	0	p	p	p	r	r	r

Operasi ppp

RLC = 000
RRC = 001
RL = 010
RR = 011
SLA = 100
SRA = 101
SRL = 111

Kode Register rrr

B = 000
C = 001
D = 010
E = 011
H = 100
L = 101
(HL) = 110
A = 111

Perintah Percabangan

Perintah percabangan ini memungkinkan :

- Pengulangan sebuah bagian program tunggal (penundaan)
- Pemilihan program bagian yang berbeda (keputusan)
- Pengelompokan tugas-tugas yang besar dalam beberapa program-program bagian yang kecil

Pengelompokan perintah percabangan :

- Perintah - perintah loncat (JUMP)
- Pemanggilan program bagian (CALL),
- Loncatan kembali dari program bagian ke program pemanggil (RETURN),
- Perintah start ulang (RST).

Pada pelaksanaan sebuah perintah-perintah percabangan, alamat loncat yang mengikuti code operasinya, akan diisi pada Program Counter (PC). Pengolah kontrol kemudian akan menjalankan program yang berada mulai dari alamat tersebut.
Alamat tujuan loncat yang lebih besar (loncat ke depan) atau lebih kecil (loncat ke belakang) adalah alamat lokasi memori yang ada pada perintah percabangan.

Perintah percabangan bersyarat dan tidak bersyarat

Pada sebuah loncatan tidak bersyarat, pelaksanaan program pada setiap saat dimulai pada alamat tujuan loncat yang telah ditentukan pada perintah percabangan.
Pada sebuah loncatan bersyarat, pelaksanaan program pada alamat tujuan loncat yang telah ditentukan tdilaksanakan bila syarat loncat terpenuhi, bila syarat loncat tidak terpenuhi maka pelaksanaan program dimulai pada perintah berikutnya dibawah perintah percabangan.

Pada perintah percabangan bersyarat, syarat loncat yang dapat dipakai hanya kondisi masing-masing bit pada register flag.

Penggunaan perintah percabangan bersyarat

Dalam penggunaan perintah percabangan bersyarat ini, ada 2 hal yang harus jelas yaitu :

- Bagaimana syarat loncat dapat diberikan pada loncatan bersyarat (syarat loncat)
- Kemungkinan apa yang ada, untuk memberikan alamat tujuan loncat (pengalamatan).
-

Syarat loncat	Tanpa Syarat	Flag zero	Flag Carry	Flag P/V	Flag S
Langsung	JP adr	JP Z, adr JP NZ, adr	JP C, adr JPNC, adr	JP PE, adr JP PO, adr	JP M, adr JP P, adr
Relatif	JR e	JR Z, e JR NZ, e	JR C, e JR NC, e	-	-
tidak langsung (melalui register)	JP (HL) JP (IX) JP (IY)	-	-	-	-

1. Perintah loncat dengan pengalamatan langsung

Mnemonik :

JP adr JP cc, adr

adr = alamat tujuan loncat
cc = syarat loncat (kode kondisi flag)

Operasi

PC CPU diisi oleh alamat tujuan yaitu konstanta 16 bit yang mengikuti Opcode. CPU akan menjalankan perintah yang ada pada alamat yang ditunjuk oleh PC.

Bila syarat loncat yang ditunjukkan oleh ccc terpenuhi, maka PC akan diisi oleh alamat tujuan yaitu konstanta 16 bit yang mengikuti Op-code, CPU akan menjalankan perintah yang berada pada alamat yang sesuai dengan PC. Bila syarat tidak terpenuhi maka PC = PC + 1, sehingga CPU akan menjalankan perintah yang berada pada alamat sesuai dengan PC.

Format :

adr	C	3	Op-code	1	1	c	c	c	0	1	0	Op-code
adr + 1	alamat tujuan LSB			alamat tujuan LSB								
adr + 2	alamat tujuan MSB			alamat tujuan MSB								

Sebagai syarat loncat hanya dapat dipakai kondisi flag tertentu, seperti ditunjukkan oleh tabel dibawah ini.

Flag	ccc	Mnemonik	Arti
Zero	000	JP NZ, adr	loncat ke alamat tujuan bila hasil <> 0 (Z=0)
	001	JP Z, adr	loncat ke alamat tujuan bila hasil = 0 (Z =1)
Carry	010	JP NC, adr	loncat ke alamat tujuan bila carry = 0 (C=0)
	011	JP C, adr	loncat ke alamat tujuan bila carry = 1 (C=1
P/V	100	JP PO, adr	loncat ke alamat tujuan bila parity ganjil (P/V=0)
	101	JP PE, adr	loncat ke alamat tujuan bila parity genap (P/V=1)
Sign	110	JP P, adr	loncat ke alamat tujuan bila hasil positip (S=0)
	111	JP M, adr	loncat ke alamat tujuan bila hasil negatip (S=1)

Setiap perintah loncat selalu hanya dapat menguji sebuah kondisi flag tertentu yang dihasilkan melalui perintah sebelumnya. Flag Half Carry (H) dan Flag Subtract (N) tidak dapat dipakai sebagai syarat loncat.

Flag : tidak terpengaruh

2. Perintah loncat dengan pengalamatan relatif

Jenis pengalamatan ini, pada Z-80 hanya dipakai untuk perintah loncat.

Mnemonik

JR e JR cc, e

cc = syarat loncat hanya untuk flag carry dan zero
e = offset, jarak loncat

Format :

adr	1	8	Op-code	0	0	1	c	c	0	0	0	Op-code
adr + 1	Konstanta e			Konstanta e								

Pengertian masing-masing bit pada konstanta e, jarak adalah sebagai berikut :

7	6	5	4	3	2	1	0
V	X	X	X	X	X	X	X

V = 0 : loncat ke depan 0 0 0 0 0 0 0 0 → 0 sampai + 127 desimal (0 1 1 1 1 1 1 1)

V = 1 : loncat ke belakang 1 1 1 1 1 1 1 1 → -1 sampai -128 desimal (1 0 0 0 0 0 0 0)

Operasi

PC CPU diisi oleh alamat tujuan yaitu hasil dari PC saat itu ditambah/dikurangi konstanta e. CPU akan menjalankan perintah pada alamat yang ditunjukkan oleh PC.

Bila syarat loncat yang ditunjukkan oleh cc terpenuhi, maka PC akan diisi oleh hasil penjumlahan/pengurangan PC saat itu dengan konstanta e, bila tidak terpenuhi PC = PC + 1.

Sub Routine (Program Bagian)

A. Program Bagian (Sub Routine)

Dalam program yang mempergunakan sebuah kelompok program yang sering dipakai, maka kelompok program ini dapat ditulis sekali saja, dan dapat dipanggil dimana saja dalam program utama bila kelompok program ini diinginkan.

Kelompok program ini disebut Program Bagian atau Sub Routine

Pemanggilan program bagian dengan mempergunakan perintah CALL (. . . .) atau RST (Restart), selalu menyimpan alamat perintah berikutnya pada Stack, alamat yang disimpan ini menjadi tujuan saat kembali setelah pelaksanaan program bagian.

Perintah terakhir sebuah program bagian adalah selalu sebuah perintah return/kembali (RET), yang fungsinya mengisi penghitung program / program counter (PC), dengan alamat tujuan kembali yang disimpan di stack. Sehingga pelaksanaan perintah berikutnya pada program utama setelah kembali dari program bagian adalah perintah yang alamatnya tersimpan pada .Stack

A.1.Proses Pemanggilan Sebuah Program Bagian

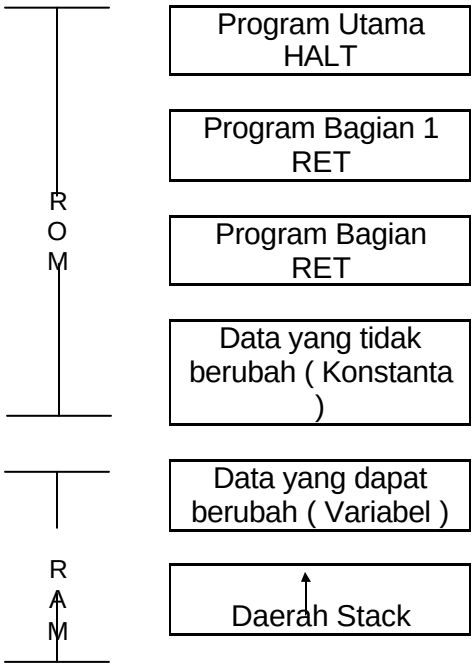
Proses berjalan sebagai berikut :

- Pada setiap pemanggilan program bagian dengan CALL, pertama-tama. menyimpan alamat tujuan lompat pada Stack.

- Kemudian terjadi sebuah lompatan untuk menjalankan perintah pada alamat pertama dari program bagian (PC = Alamat awal program bagian).
- Masing - masing perintah pada program bagian dijalankan secara berurutan.
- Perintah RET menggunakan alamat loncat balik (dalam Stack), sebagai alamat tujuan loncat balik ke program utama.(PC = alamat loncat balik).
- CPU akan melanjutkan pelaksanaan perintah berikutnya pada program utama mulai dari alamat loncat balik ini.

Urutan program bagian adalah bebas dan juga harus tidak saling mengikat dengan program bagian yang lain.
Perintah awal dari program bagian tidak ditentukan secara khusus, hanya perintah akhir dari program bagian harus selalu perintah RETURN (RET).

Peletakan program didalam penyimpan program dan data (memori) lebih banyak seperti pada gambar berikut ini :



A.2. Penyimpan Alamat Loncat Balik pada STACK

Alamat loncat pada dasarnya diletakkan pada RAM. Penyimpanan di RAM adalah lebih cepat, walaupun demikian kapasitas penyimpannya terbatas.

Untuk penanganan daerah stack, penunjukkan ini dipergunakan fasilitas perintah PUSH dan POP serta register penunjuk STACK (SP).

Pada setiap pemanggilan program bagian , pertama-tama isi Stack Pointer (SP) dikurangi satu (SP - 1) kemudian (SP - 1) diisi dengan MSB (bit tertinggi) dari alamat loncat balik, kemudian isi (SP - 2) diisi dengan LSB (bit terendah) dari alamat loncat balik.

Pada setiap loncat balik dari sebuah program bagian, bagian pertama LSB dari PC akan diisi oleh isi SP (alamat penyimpan stack saat itu) dan bagian MSB dari PC diisi oleh isi (SP + 1) dan kemudian (SP + 2) menunjukkan alamat awal dari pemakaian stack berikutnya.

Pemanggilan program bagian dengan mempergunakan penunjuk stack (SP), memiliki 2 keuntungan, yaitu :

- Pada pemanggilan program bagian yang tidak rumit, dibutuhkan 2 lokasi memori sebagai penyimpan sementara dari alamat loncat balik.
- Program bagian dapat dibuat bercabang secara bebas, yang berarti di dalam sebuah program bagian dapat dibuat program bagian yang lain. Pada setiap pemanggilan program bagian, (SP) akan berubah dan alamat stack yang terakhir akan naik 2 ke atas.

Oleh sebab itu program harus memperhatikan hal itu, yaitu tidak boleh ada tumpang - tindih pada program bagian yang lain, dan tidak melupakan perintah kembali (RET) pada akhir dari masing - masing program bagian.

A.3. Perintah Pemanggilan Program Bagian

A.3.1. Perintah Pemanggilan dengan Pengalamatan langsung

Mnemonik :

CALL adr CALL cc, adr

adr = Konstanta 16 bit alamat tujuan loncat (alamat pertama dari program bagian)

cc = Syarat loncat/kondisi flag.

Operasi :

pemanggilan program bagian selalu dilaksanakan.

pemanggilan program bagian hanya dijalankan bila syarat CC terpenuhi.

Operasi yang akan dilaksanakan :
- alamat loncat balik akan disimpan sementara pada stack.
- (SP) diturunkan dua.
- alamat pertama dari program bagian akan disimpan pada PC.

Format :

adr

C	D
alamat LSB dari prog. bagian	
alamat MSB dari prog. bagian	

1	1	C	C	C	1	0	0	
alamat LSB dari program bagian								
alamat MSB dari program bagian								

CCC : 000 ⇒ NZ
001 ⇒ Z
010 ⇒ NC
011 ⇒ C
100 ⇒ PO
101 ⇒ PE
110 ⇒ P
111 ⇒ M

Contoh :
SP : 1FF3H

⇒ Program Utama :

0900	CDH	CALL 0A10H	memanggil program bagian yang ada
0901	10H	-	pada alamat 0A10H
0902	0AH	-	
0903	FFH	HALT	
			menutup program utama

⇒ Program Bagian :

0A10	3EH	LD A, FFH	mengisi AKKU dengan data FFH
0A11	FFH	-	
0A12	C6H	ADD A, 01H	menambah isi AKKU dengan
0A13	01H	-	konstanta 01H
0A14	C9H	RET	menutup program bagian

⇒ Setelah pelaksanaan program, penunjuk stack (SP) = 1FF3H

1FF1	03H	alamat stack yang akhir
1FF2	09H	
1FF3	XXH	

A.3.2. Perintah Pemanggilan dengan Perintah Restart/start ulang

Mnemonik :

RST a

RST : Restart/start ulang pada alamat yang telah ditentukan.
a : 0, 1, 2, 3, 4, 5, 6, 7 atau 00, 08, 10, 18, 20, 28, 30, 38.

Operasi :

Perintah restart adalah pemanggilan program bagian tak bersyarat, yang selalu dilakukan seperti perintah CALL.
Operasi - operasi berikut ini akan dilaksanakan :

= Alamat loncat balik akan disimpan sementara pada stack.
= Stack pointer (SP) diturunkan 2.
= Alamat pertama dari program bagian akan disimpan pada PC.

Format :

adr

1	1	a	a	a	1	1	1

aaa : kependekan dari alamat awal program bagian.

Mnemonik Perintah	Op - Code	aaa	Alamat Awal Program Bagian
RST 0 atau RST 00	C7	000	0000H
RST 1 atau RST 08	CF	001	0008H
RST 2 atau RST 10	D7	010	0010H
RST 3 atau RST 18	DF	011	0018H
RST 4 atau RST 20	E7	100	0020H
RST 5 atau RST 28	EF	101	0028H
RST 6 atau RST 30	F7	110	0030H
RST 7 atau RST 38	FF	111	0038H

Perintah RST kebanyakan dipergunakan pada proses pelaksanaan program Interrupt, tetapi dapat juga dipakai pada program normal, yaitu memanfaatkan keuntungan dari formatnya yang rendah, karena kebutuhan penyimpan sementara (RAM) yang sedikit dan kecepatan pelaksanaannya yang cepat bila dibandingkan dengan perintah CALL.
Flag : Flag terpengaruh

A.4. Perintah Loncat Balik dari Program Bagian
A.4.1. Perintah Loncat Balik dari Program Bagian Normal

Mnemonik :

RET	RET CC
RET = Return, loncat dari program bagian	CC = Syarat loncat balik kondisi flag

Operasi :

loncat balik tanpa syarat	loncat balik akan dilaksanakan
	bila CC terpenuhi.

Setelah pelaksanaan perintah ini, mekanisme pengontrolan akan melaksanakan kegiatan berikut ini :

- PC diisi oleh isi dari 2 alamat teratas dari stack.
- SP (Stack Pointer) dinaikkan 2.
- Proses program berikutnya mulai dari alamat yang ada dalam PC.

adr		alamat stack sebelum pelaksanaan perintah RET
adr + 1		
adr + 2		alamat stack setelah pelaksanaan perintah RET

Flag :

Flag terpengaruh

A.4.2. Perintah Loncat Balik dari Program Bagian Normal

Perintah ini dipakai untuk loncat balik dari program bagian (yang pemanggilannya melalui pelaksanaan perintah Interupt).
Disebut juga sebagai Interupt service routine.
Mnemonik :

RET I	RET N
kembali dari maskable Interupt	kembali dari non maskable interupt

Operasi :

Berjalan dengan logika yang sama seperti perintah RET, khusus untuk keluarga Z 80 yang memiliki blok input - output (I/0). Kode

Operasi ini dapat diidentifikasikan (bila kode operasi ini muncul pada BUS data), maka dengan itu CPU dapat menentukan tingkat prioritas pelaksanaan perintah diatas.

Pemberian Parameter pada Program Bagian

Nilai masukan, alamat, hasil dan sebagainya, yang dipakai selama pelaksanaan program bagian disebut sebagai PARAMETER. Pemograman harus mempertimbangkan dengan baik, dimana nilai parameter ini disimpan sementara. Program bagian harus ditulis seumum mungkin dan tidak dipakai hanya untuk satu nilai parameter saja.

Juga hasil yang dihasailkan oleh program bagian harus juga diletakkan sedemikian rupa sehingga program utama dapat memberikan nilai yang lain, dengan kata lain, sebuah program bagian selama pelaksanaannya, harus dapat mengalamatkan parameter - parameter yang digunakan.

A.5.1. Pemberian Parameter dalam Register CPU

Parameter yang diperlukan dalam program bagian (yang harus ada sebelum pemanggilan program bagian) di simpan di Register CPU. Ini adalah metode yang sederhana dan cepat, namun hanya dapat digunakan bila terdapat hanya sedikit parameter yang digunakan dan di dalam program utama register tidak dipakai untuk tujuan lain.

Contoh :

Disini sebuah parameter dalam AKKU diberikan ke program bagian .

⇒ Program Utama :

```
0800   IN A, (20H)      | membaca sebuah byte dari port
....   .....            | beralamat 20H
081F   CALL 0A00H       | ....
       HALT             | memanggil program bagian
                        | penutup program utama
```

⇒ Program Bagian :

```
0A00   BIT 2,A          | program bagian untuk mengelola
....   ....             | data pada AKKU
0A5F   RET              | penutup program bagian
```

A.5.2. Pemberian Parameter dalam Penyimpan Data (RAM)

Pada metode ini, penyimpan data tertentu telah disediakan untuk parameter. Program utama memberikan data yang telah disimpan pada lokasi penyimpan data yang telah ditentukan, yang mana program bagian dapat mengambil data dari sana.

Contoh :

Alamat 0900H adalah parameter alamat awal blok data yang akan dikeluarkan ke port A yang alamatnya 20H. Alamat 30H adalah alamat awal blok data.

```
0000    .....
0002    CALL 0700H          ->  Program Utama
.....   .....
00FF    HALT

0700    LD HL, (0900H)
0703    LD A, (HL)
0704    AND A
0705    JP Z, 070EH         ->  Program Bagian
0708    OUT (20H), A
070A    INC HL
070B    JP 0703H
070E    RET
```

```
0900    1C
0901    30                  ->  Penunjukkan alamat awal blok data

301C    12
301D    57                  ->  Blok data
301E    3A
301F    00
```

DAFTAR PUSTAKA

Beuth, Klaus,"*Elektronik 4 Digitaltechnik*", Vogel-Buchverlag, Wuerzburg, 1982ac

"*Elektronika Daya*", Gunadarma

ELWE, "*Lehrsysteme Leistungelektronik*".

Europalehrmittel, "*Fachkunde Information Elektronik*", Verlag Stuttgart hal 13/14.

Horn / Nur Lesson plan 51520104 PPPGT Malang 1988.

Horn / Rizal, Lesson Plan 51510102.

Horn / Sutrisno Lesson plan 52520203 PPPGT Malang 1988.

ITB, Polyteknik Mekanik Swiss, "Teknik Listrik Terpakai", hal 39 – 47

Kamajaya, "*Fisika 1*", Ganeqa Exact, Bandung, 1994.

MC68HC11F1 Technical Data, Motorola Inc., Arizona, 1990

MC68HC11F1 Programming Reference Guide, Motorola Inc., Arizona

M68HC11 Reference Manual, Motorola Inc., Arizona, 2002

MC68HC11F1 Technical Summary 8-Bit Microcontroller,Motorola Inc., Arizona, 1997

M. Affandi Agus Ponijo, "*Pengetahuan Dasar Teknik Listrik*"

M. Affandi Agus Ponijo, "*Pengetahuan Dasar Teknik Listrik*"

"*Microprocessor and Microcomputer*", ITT Fachlergänge, Pforzheim, 1979

Nur / Supr , Lesson plan 51520101, PPPG Teknologi Malang, 1988.

Ogata, K(1997). "*Teknik Kontrol Automatik*". Jilid 1. Erlangga: Jakarta

PEDC, *"Ilmu Listrik"*, Bandung, PEDC, 1981, hal 127-129,PT Gunung Agung, 1981.

Pflaum, Richard. "*Elektronik IVA Leistungelektronik (lehrbuch) Werner Dzieia*", Verlag
KG Munchen
Pitowarno, E.(2006). "*Robotika Disain, Kontrol, Dan Kecerdasan Buatan*". Andi: Yogyakarta
Schmidt, Walf Deiter (1997). "*Sensor Schaltungs Technik*". Vogel (Wurzburg). Germany
Sugihartono, Drs.(1996). "*Dasar-dasar Kontrol Pneumatik*". Tarsito: Bandung
Suma'mur P.K, Msc, Dr ; "*Keselamatan kerja dan Pencegahan kecelakaan*" ;
Stielew, Roth, Prof, Dr, Ing. "*Institutfur Leistungselektronik und Elektrische Antriebe*"
W. Ernest, "*Elektrotecnik*", FranKfruft, Sauerlaender 1982, hal 13,14, 15-17.
Wil Helm Benz, "*Tabellen Buch Elektronik*", Kohl & Noltemeter & 10, Frankfurt, 1989
Uma'mur P.K, Msc, Dr ; "*Keselamatan kerja dan Pencegahan Kecelakaan*"
http://www.bbc.co.uk/schools/gcsebitesize/design/systemscontrol/pneumaticsrev1.shtml. (12.01.2008)
http://64.78.42.182/sweethaven/MechTech/hydraulics01/modulemain.asp? whichMod=0100. (12.01.2008)
en.wikipedia.org/wiki/**Brushless_DC**_electric_**motor** - 40k. (12.01.2008)

DAFTAR ISTILAH

AC Choper
ADC
adder
aktuator
akumulator
alkali
alternatif current
ALU
amper
amplifier
amplitudo
AND
anoda
aritmatika
arus cerat
arus listrik
arus pusar
asam nitrat
assembler
atom
band pas filter
basis
beda potensial
bias
biner
biner
biner
bit
BJT
boolean
break down
bus
caption
carry
celcius.
chip
clock
coil
common
coulomb
counter
CPU
crowbaring
cut-off
daya listrik
DC Choper
dekoder
demodulasi
density
depletion Layer
depolarisator
desimal
destilasi
DIAC
diagram bode
diamater
dielektrikum
difusi
digital
dimmer
dioda
dioda Schottky
dioda varactor
dioda zener
dipole
direct current
diskrit
double
download
drain
efisiensi
ekivalen

LAMPIRAN B.2

ekonomis
elekrolisa
elektro statis
elektroda
elektrolit
elektron
elektronika daya
emitor
EXOR
Fahrenheit.
farad
fase
ferro magnetik
filter
flag
flip-flop
flow chart
fluks
form
forward bias
foto cell
frekuensi
frekuensi modulasi
fungsi
fuzzy
fuzzylemps
galvanis
gate
gaya gerak listrik
gaya gerak magnet
generator
HandShaking
henry
hertz
hidrolika
high pass filter
histerisis
hole
horse power
hukum Kirchhoff
hukum ohm
IGBT
Impendansi
indeks
induksi
induktansi
induktor
input
input pembalik
instrumentasi
integer
ionisasi
ISA
isolasi
jembatan Wheatstone
jendela
joule
Joule
junction
kapasitansi
kapasitas panas
katoda
katup
kelvin.
kimiawi
kode program
koefisien
kolektor
kondensator
konduksi
konfigurasi
konstanta
kontrol
kontroler
konveksi
konversi
konverter
korona

korosi
kursor
LDR
loop
loop
low pass filter
LPT
LSI
medan magnit
mekanik
memori
mikrokomputer
mikrokontroller
mneumonic
modulasi
MOS
MOSFET
motor stepper
muatan listrik
multiplekser
Neutron
NOT
offset
ohm
oksidasi
op-amp
op-code
OR
orde dua
orde satu
oscilator
osilasi
osilator
osiloskop
output
overflow
parallel
parity
pengalamatan
penghantar
penyearah
penyulutan
penyulutan
permitivitas listrik
pewaktu
phasa
piston
plant
plasma
pneumatik
pointer
pointer
polarisasi
polaritas
pop
port
port
potensial barier
potensiometer
potensiometer
power supply
PPI
PROM
prosedur
Proton
pulsa
push
PWM
radiasi
radiator
radioaktif
RAM
Reamur.
register
register
reluktansi
RePROM
resonansi

LAMPIRAN B.4

reverse bias
ROM
root locus
RS232
satu fase
sekuensial
semi penghantar
semikonduktor
semikonduktor
sensor
seri
servo
siemens
silicon
silikon
sinyal
sinyal
Source
stack
statement
stator
string
subrutin
tabel kebenaran
tahanan
tahanan jenis
tegangan
Tegangan Knee
tegangan listrik
temperatur
tesla
thermocouple
thyristor
tiga fase
timer
Titik Q
Toleransi
Transistor
transistor
transistor bipolar,
Transkonduktansi
transportasi
triac
Unijunction Transistor
unipolar
usaha listrik
USB
valensi
variabel
variant
visual basic
volt
watt
weber

ISBN 978-979-060-089-8
ISBN 978-979-060-091-1

Buku ini telah dinilai oleh Badan Standar Nasional Pendidikan (BSNP) dan telah dinyatakan layak sebagai buku teks pelajaran berdasarkan Peraturan Menteri Pendidikan Nasional Nomor 45 Tahun 2008 tanggal 15 Agustus 2008 tentang Penetapan Buku Teks Pelajaran yang Memenuhi Syarat Kelayakan untuk digunakan dalam Proses Pembelajaran.

HET (Harga Eceran Tertinggi) Rp. 22,594.00