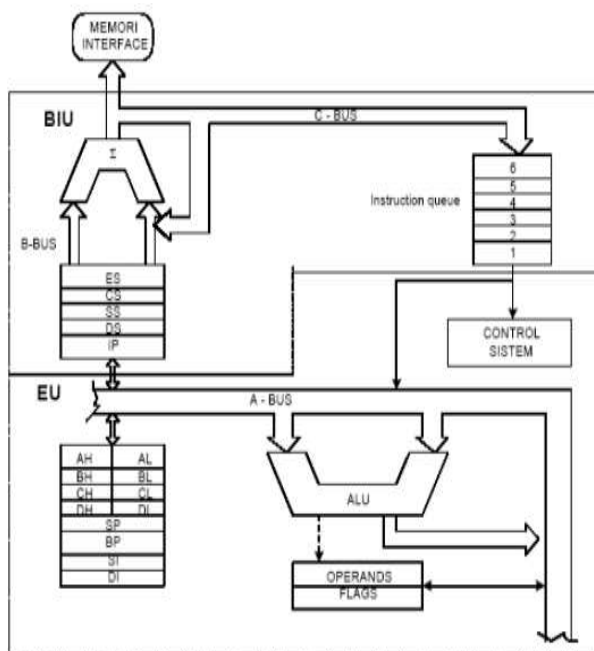


S1

μPCⁱ
Laboratory

Modul Praktikum Mikroprosesor dan Antarmuka



**LABORATORIUM MIKROPROSESOR DAN ANTARMUKA
FAKULTAS TEKNIK ELEKTRO
UNIVERSITAS TELKOM
BANDUNG
2014**

STRUKTUR ORGANISASI

LABORATORIUM MIKROPROSESOR DAN ANTARMUKA

Dekan Fakultas Elektro dan Komunikasi	: Dr.Ir. Rina Pudji Astuti M.T.
Pembina Lab Mikroprosesor dan Antarmuka	: Burhanudin Dirgantoro ST., MSc
Dosen Mata praktikum	: Junartha Halomoan,ST.,MT
Koordinator Asisten	: Muhammad Iqbal Imaduddin
Administrasi	: Luluk Listyani Ayunigtyas
Bendahara	: Aditya Bram Wiratma
Divisi Praktikum	: I Nyoman Adhi Trianajaya Dewa cahya junior Jorjiana Aminatus sa'adah IP
Divisi Hardware and Software	: Ariando Saragih Tedy Gumilang Sejati Iyodha Amanda Ecky Erlangga
Divisi Riset	: Fadli Pradityo Lucky Oktavianto Ganang Wahyu W. Achmed Farizal Fadli Kurnia Khafidahatur Rafiah Philip Daely
Tim Pendamping	: Muhammad Fakry Mentari Eka Putri Putri Amanda W. Dila Cahya Wardhani Muhammad Iqbal Ari Septayuda Denish Novenda Nur Shabrina

TATA TERTIB

PRAKTIKUM MIKROPROSESOR DAN ANTARMUKA

1. Kelengkapan Praktikum

- a. Praktikan wajib membawa kartu praktikum yang telah ditempel foto dan dicap saat pengumpulan TP pertama pada saat pelaksanaan praktikum.
- b. Praktikan wajib menggunakan seragam resmi mahasiswa Telkom University selama praktikum berlangsung (tidak memakai jeans).

2. Pelaksanaan Praktikum

- a. Pelaksanaan praktikum meliputi Tugas Pendahuluan, Tes Awal, Praktikum, Jurnal dan Tes Akhir.

b. Kedatangan

- Praktikan datang 10 menit sebelum praktikum dimulai.
- Praktikum dimulai pada waktu yang telah ditentukan.
- Praktikan yang terlambat kurang dari 20 menit diijinkan mengikuti praktikum.
- Praktikan yang terlambat lebih dari 20 menit tidak diijinkan mengikuti praktikum.

c. Pelaksanaan Praktikum

- Apabila tidak membawa kelengkapan praktikum (point 1) maka praktikan dipersilahkan melengkapinya sampai waktu pengerjaan Tes Awal selesai.
- Selama praktikum berlangsung praktikan dilarang melakukan hal-hal yang mengganggu pelaksanaan praktikum seperti : makan, minum, membuat kegaduhan,dll.
- Praktikan wajib mengikuti semua modul praktikum.

3. Tugas Pendahuluan

- a. Tugas pendahuluan bersifat tidak wajib.
- b. Bagi yang mengerjakan Tugas Pendahuluan wajib mengerjakan semua soal sesuai ketentuan yang ada.
- c. Tugas Pendahuluan ditulis tangan dalam buku TP dan dikumpulkan sesuai dengan waktu yang telah ditentukan
- d. Tugas Pendahuluan diumumkan setiap hari kamis malam sebelum minggu pertama praktikum di masing gedung N dan blog laboratoium mikroprosesor dan antarmuka.
- e. Pengumpulan Tugas Pendahuluan dilakukan pada hari senin mulai pukul 06.30 sampai 9.00 di minggu pertama praktikum di ruang N310.

4. Jurnal Praktikum

- a. Jurnal diberikan sebelum praktikum berlangsung.
- b. Jurnal harus dikerjakan semua dan dikumpulkan pada waktu yang telah ditentukan.

- c. Praktikan dianggap gugur pada modul yang bersangkutan apabila tidak mengerjakan jurnal praktikum.

5. Tes Akhir

Tes Akhir diberikan oleh asisten setelah praktikum selesai, bisa berupa tes lisan ataupun tertulis.

6. Tugas Tambahan

- a. Asisten bisa memberikan tugas tambahan jika dipandang perlu.
- b. Tugas tambahan bisa diberikan secara perorangan atau kelompok.
- c. Waktu pengumpulan tugas tambahan ditentukan asisten masing – masing.

7. Penilaian

Tugas Pendahuluan	: 20%
Tes Awal	: 10%
Praktikum	: 35%
Jurnal	: 20%
Tes Akhir	: 15%

8. Pertukaran Jadwal

- a. Pertukaran jadwal maksimal satu hari sebelum praktikum yang bersangkutan berlangsung.
- b. Pertukaran jadwal bisa dilakukan antar sesama praktikan dengan mengisi form tukar jadwal yang disetujui oleh asisten.

9. Praktikum Susulan

Praktikum susulan dilakukan jika dipandang perlu.

10. Kelulusan

- a. Praktikan dinyatakan lulus jika nilai akhir setiap modul $\geq 50\%$ dan nilai akhir praktikum mempunyai rata – rata $\geq 60\%$.
- b. Jika praktikan gugur dalam satu modul atau lebih maka wajib mengulang seluruh modul pada praktikum tahun berikutnya.

Bandung, Februari 2014

Mengetahui

Koordinator Asisten

Pembina Lab Mikroprosesor dan Antarmuka

M Fakry Abdul Ghani.

Burhanudin Dirgantoro ST., MSc.

MODUL I

Pengenalan Sistem Mikroprosesor dan Pemrograman Internal Register

A. TUJUAN

1. Dapat memahami sistem mikroprosesor, mikroprosesor, dan perbedaan mikroprosesor 8088 dengan 8086 secara umum.
2. Memahami langkah-langkah mikroprosesor dalam menjalankan suatu instruksi.
3. Dapat membuat instruksi sederhana pada BGC 8088.
4. Mengerti dan memahami pola pengalamatan pada mikroprosesor.
5. Memahami langkah dan mampu mengaplikasikan pembuatan *opcode* dalam bahasa *assembly* mikroprosesor.

B. PERALATAN

1. BGC 8088

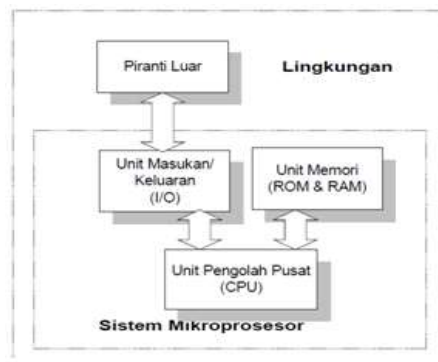
C. TEORI

1. PENGENALAN SISTEM MIKROPROSESOR SECARA UMUM

A. SISTEM MIKROPROSESOR

Sistem mikroprosesor adalah sistem mikroelektronika yang menggunakan mikroprosesor sebagai unit pemroses sentralnya.

Sistem mikroprosesor terbagi menjadi dua bagian perangkat, perangkat keras dan perangkat lunak. Perubahan fungsi sistem mikroprosesor tergantung dari program pada sistem perangkat lunak yang mendukung kerja sistem mikroprosesor.



Gambar 1. Sistem mikroprosesor secara umum

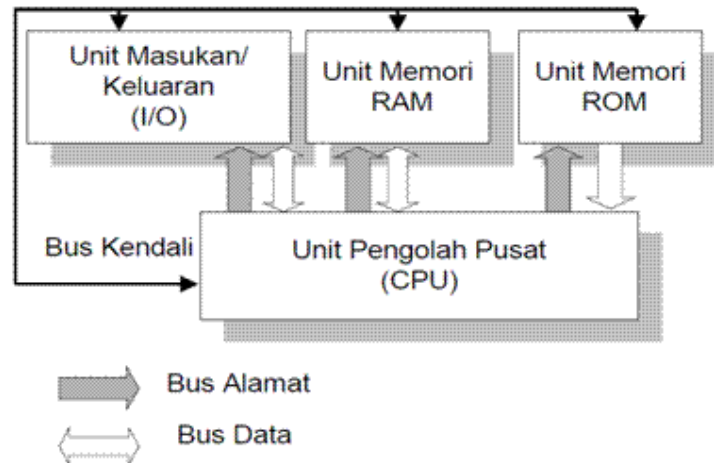
Gambar 1 memperlihatkan blok diagram hubungan antara sistem mikroprosesor dengan lingkungan luar. Dalam sistem mikroprosesor terdiri atas unit pengolah pusat (CPU), unit media penyimpan (memori), dan unit masukan dan keluaran. Unit masukan dan keluaran sebagai perantara antara sistem mikroprosesor dengan lingkungan luar sistemnya.

Prinsip kerja sistem mikroprosesor adalah mengolah suatu data masukan, yang kemudian hasil olahan tersebut akan menghasilkan keluaran yang dikehendaki. Proses pengolahan datanya dapat difungsikan sesuai dengan instruksi yang diprogramkan. Apabila sistem mikroprosesor tanpa unit masukan dan keluaran maka tidak ada masukan ataupun keluaran dari dan ke sistem, maka sistem hanya bekerja tanpa ada keluaran yang dapat diperoleh oleh lingkungan luarnya.

B. KONFIGURASI DASAR SISTEM MIKROPROSESOR

Masing – masing mikroprosesor memiliki bahasa pemrograman yang berbeda-beda. Namun secara prinsip, dasar dari tiap mikroprosesor adalah sama. Tiap Mikroprosesor memiliki satu bus data, satu bus alamat dan satu bus kendali. Dalam mikroprosesor terdapat suatu unit untuk mengerjakan fungsi – fungsi logika dan aritmatika, register – register untuk menyimpan data sementara dan unit pengendalian.

Bus data terdiri biasanya 4, 8, 16 atau 32 jalur (bit), 64 bit, tergantung dari jenis mikroprosesornya. Bus data berfungsi memuat data dari dan ke mikroprosesor. Arah panah menunjukkan arah data dikirim / diterima. Bus alamat merupakan bus yang berisi alamat – alamat yang datanya akan dikirim / diterima oleh mikroprosesor. Bus kendali digunakan untuk mensinkronkan kerja antara mikroprosesor dengan dunia luar sistem. Pada beberapa aplikasi ada yang disebut dengan istilah jabat tangan, seperti misalnya pada penerapan hubungan dengan pencetak (*printer*).



Gambar 2. Konfigurasi sistem mikroprosesor

C. CPU (CENTRAL PROCESSING UNIT) = PROCESSOR

Merupakan unit yang berfungsi untuk menjalankan perintah dalam melakukan sebagian besar pemrosesan data, serta untuk berkomunikasi dan mengendalikan operasi subsistem lainnya dengan meraih instruksi terkode biner (bahasa mesin) dari memori, lalu mengkodekan instruksi menjadi sederetan aksi untuk mengerjakannya.

Langkah langkah processor dalam menjalankan suatu instruksi :

1. Penjemputan Instruksi (IF = Instruction Fetch)

Merupakan proses pengambilan instruksi dari suatu lokasi dengan alamat tertentu pada memori utama. Instruksi ini dapat terdiri dari hanya kode operasi saja atau juga dapat terdiri dari kode operasi dan data yang menyertainya, tergantung dari format instruksi yang digunakan. Data yang diminta oleh instruksi disebut sebagai *operand*. Panjang word = 16 bit, dibagi menjadi dua bagian, yaitu 4 bit *opcode* dan 12 bit alamat.



Gambar 3. Format instruksi

Langkah – langkah pengambilan instruksi antara lain :

- $IR (Instruction Register) \leftarrow [CS + IP]$.
- Proses kerjanya dimulai dengan penjemputan instruksi baru dari memori ke IR
- CU (*Control Unit*) menerjemahkan isi register CS (*Code Segment*) dan IP (*Instruction Pointer*) untuk menentukan letak dari instruksi baru tersebut ke memori.
- Hasil terjemahan isi CS dan IP ini dikirim ke memori melalui bus alamat (*Address bus*).
- CU mengirim sinyal MemREAD untuk memberitahukan memori bahwa CU ingin membaca data memori.
- Setelah mendapatkan sinyal MemREAD, akan melihat isi dari bus alamat.
- Kemudian isi dari *cell* memori yang sesuai dengan alamat tersebut diletakkan di bus data (selebar 1 byte).
- Beberapa saat setelah mengirim sinyal MemREAD, CU membaca isi dari bus data dan meletakkannya di IR.

2. Dekode Instruksi (ID = Instruction Decode)

Yaitu proses penerjemahan kode-kode oleh tabel instruksi yang ada untuk mengetahui maksud dari suatu instruksi. Isi dari IR setelah proses pengambilan instruksi tersebut kemudian diterjemahkan oleh CU untuk mengetahui apa saja yang diinginkan oleh instruksi baru tersebut. Untuk tugas penerjemahan ini CU menggunakan bantuan tabel instruksi yang ada di ID (Instruction Decoder) untuk dapat memahami maksud dari instruksi tersebut.

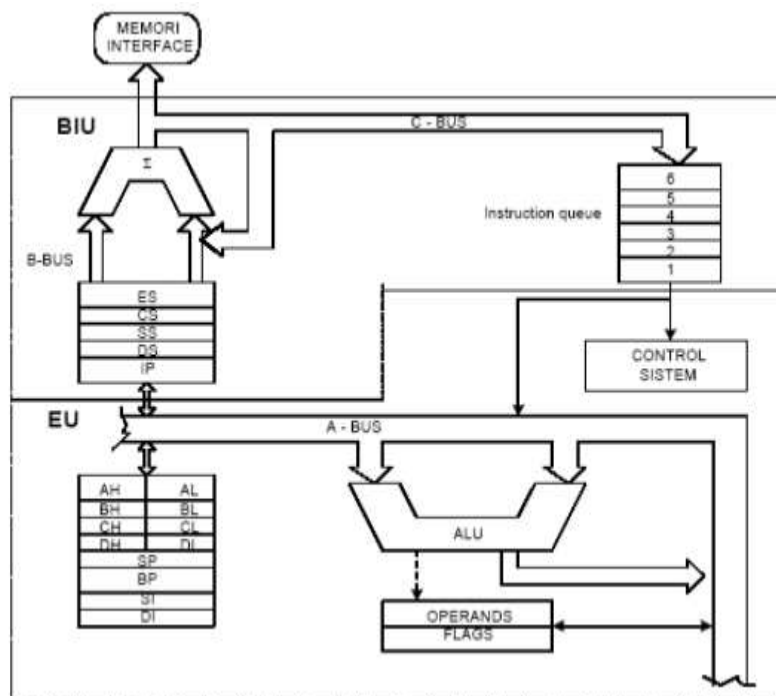
3. Eksekusi instruksi (EX= Execution)

Dari hasil dekoding akan didapatkan maksud dari instruksi, sehingga prosesor akan melaksanakan fungsi sesuai instruksi. Instruksi-instruksi tersebut antara lain:

- Operasi Aritmatika dan logika
- Data Transfer
- Control

Komponen Dasar CPU Terdiri dari :

- Arithmetic Logic Unit (ALU)
- Control Unit
- Clock
- Sistem bus internal
- Beberapa register khusus



Gambar 4. Komponen Dasar CPU mikroprosesor 8088

- **(Arithmetic Logic Unit = ALU)**

ALU merupakan mesin penghitung (kalkulator) dari CPU. *Control Unit* akan menggunakan ALU jika instruksi yang dikerjakan membutuhkan perhitungan aritmatika, logika, dan perhitungan ADD, SUB, MUL, DIV. Unit ini mempunyai beberapa status register atau flag, dimana keadannya dipengaruhi oleh hasil operasi yang dilakukan oleh ALU. Setiap flag akan diset atau direset tergantung dari hasil operasi yang dilakukan.

- **Control Unit**

Mempunyai fungsi utama untuk mengambil, mendekode (menerjemahkan kode / sandi) dan melaksanakan urutan instruksi sebuah program yang tersimpan dalam memori. CU mengatur urutan operasi seluruh sistem.

Prosesnya adalah setiap instruksi yang diterima oleh CPU akan disimpan oleh *register instruksi* (IR) dan akan dikenali oleh *Control Unit* untuk membangkitkan rutin-rutin mikroprogram yang tersimpan pada ROM. Setiap instruksi mempunyai rutin mikroprogram tersendiri yang khas, setiap rutin mikroprogram ini terdiri dari beberapa mikroinstruksi. Control Unit akan menghasilkan sekumpulan bit yang terhubung ke setiap bagian CPU dan komponen lain dari sistem mikroprosesor, di mana variasi bit-bit ini akan mengaktifkan komponen – komponen yang dibutuhkan untuk melaksanakan setiap mikroinstruksi, kumpulan bit-bit kendali yang dihasilkan oleh CU ini biasa disebut sebagai *Control Word* (CW).

- **Clock**

Bagian pewaktuan dan pengendalian memiliki fungsi utama untuk mengambil dan mendekodekan instruksi dari memori program dan membangkitkan sinyal kendali yang diperlukan oleh bagian lain dari mikroprosesor untuk melaksanakan instruksi tersebut. Unit pewaktu dalam hal ini akan mengatur dan mensinkronkan aktivitas *Control Word* (CW) dan semua

komponen komputer yang diaturnya sehingga setiap mikroinstruksi dapat dieksekusi setiap pulsa satuan waktu.

- **Sistem Bus Internal**

Merupakan sekumpulan saluran penghantar listrik yang menghubungkan setiap komponen secara paralel, memberikan suatu mekanisme dalam mengendalikan akses data, serta mengawasi setiap perubahan sinyal untuk berkomunikasi pada setiap bagian sistem.

- **Bus unidirectional** hanya dapat mengirimkan data dalam satu arah saja dan khususnya digunakan untuk menghubungkan dua buah register di mana salah satunya sebagai register sumber, sedangkan yang lainnya sebagai tujuan.

- **Bus bidirectional** dapat mengirimkan data dalam kedua arah tetapi tidak secara simultan dan biasanya digunakan untuk sekumpulan register yang dapat berfungsi sebagai register sumber maupun sebagai register tujuan

- **Register Khusus**

Register merupakan memori khusus di dalam mikroprosesor. Untuk mengidentifikasikannya, register memiliki nama khusus yang mencerminkan fungsinya. Prosesor 8086 / 8088 mempunyai 14 register yang masing-masing mempunyai kapasitas 16 bit dan 9 flag. (dipelajari selanjutnya)

Komponen dasar CPU tersebut dibagi menjadi dua bagian :

1) BIU (Bus Interface Unit)

- Sebagai antarmuka dengan *peripheral* di luar mikroprosesor
- Bertanggung jawab terhadap semua operasi bus eksternal, seperti:
 - *Instruction Fetch* (IF)
 - Operasi baca-tulis memori atau I/O
 - Antrian instruksi dan penghitungan alamat (PA)

2) EU (Execution Unit)

- Menerjemahkan instruksi (*Instruction Decode* = ID)
- Menjalankan instruksi (*Execution* = EX)
- Mengambil instruksi dari *Queue*
- Mengirim informasi ke BIU untuk mengambil data / instruksi
- Cek dan update *flag*
- Penghitungan alamat *operand* (EA)
- Transfer data dari dan ke General Purpose Register

D. ORGANISASI MEMORI

Memori merupakan suatu jenis peralatan untuk menyimpan instruksi ataupun data yang diminta oleh suatu operasi pada sistem komputer. Ada 2 jenis memori yang utama yaitu :

1) Memori internal prosesor

Merupakan sekumpulan register berkecepatan tinggi yang berfungsi sebagai tempat penyimpanan sementara dari instruksi dan data selama proses di dalam CPU. Contoh: akumulator.

2) Memori utama (*main memory*) / memori primer

Merupakan jenis memori dengan kecepatan relatif tinggi untuk menyimpan instruksi dan data selama operasi komputer. Pada memori utama ini instruksi dan data akan disimpan dalam suatu lokasi alamat tertentu yang dapat dikenali dan dapat diakses secara langsung dan cepat oleh set instruksi dari CPU. Memori ini dapat dibedakan menjadi beberapa jenis seperti RAM, ROM. (dipelajari selanjutnya)

Dari penjelasan diatas dapat disimpulkan bahwa suatu memori mempunyai tiga fungsi utama, yaitu :

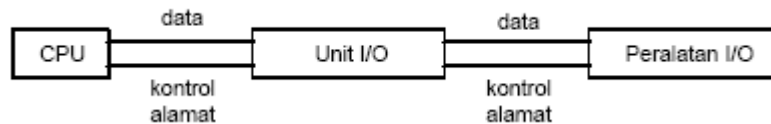
- Sebagai tempat penyimpanan kode-kode biner yang merupakan urutan instruksi yang ingin dikerjakan oleh sistem (berupa program).
- Melakukan operasi baca dan tulis data.
- Sebagai tempat menyimpan data-data yang ingin diolah sistem (berupa data).

E. INPUT / OUTPUT (I/O)

Organisasi I/O merupakan unit perantara yang memungkinkan suatu sistem mikroprosesor dapat berkomunikasi atau saling mentransfer informasi dengan dunia luar. Unit ini diperlukan karena seringkali data pada peralatan lain yang berada di luar sistem mikroprosesor mempunyai format dan kecepatan yang berbeda dengan CPU atau sistem. Unit

Input/masukan berfungsi untuk menyediakan informasi bagi ALU dan atau memori. Alat masukannya dapat berupa suatu sensor, seperti sensor tegangan atau daya, sensor temperatur, dan sebagainya. Sedangkan unit keluaran mempunyai fungsi untuk memberikan data yang datang dari ALU atau melaksanakan perintah-perintah. Alat keluaran dapat berupa suatu motor, rele atau dapat juga berupa lampu.

Hubungan antara CPU, I/O, dan peralatan I/O adalah sebagai berikut :



Gambar 5. Interface antara CPU dengan I/O

Metoda yang digunakan untuk mengendalikan operasi I/O dapat dibedakan dalam dua metoda berdasarkan cara kerja CPU dalam mengeksekusi operasi I/O. Kedua metoda pengendalian I/O tersebut adalah *metoda programmed I/O* dimana pengendalian operasi I/O sepenuhnya dilakukan oleh CPU, dan *metoda akses memori langsung* (DMA atau *Direct Memory Access*) yaitu metoda pengendalian operasi I/O tanpa melibatkan campur tangan CPU.

Teknik masukan dan keluaran pada sistem mikroprosesor dapat dibedakan menjadi dua sistem yaitu :

1). Sistem Paralel

Data masukan / keluaran dikirimkan dalam bentuk delapan bit paralel, digunakan

untuk mentransfer data dalam jarak beberapa kaki (contoh: *printer* dan *harddisk*),

tidak efisien untuk komunikasi jarak jauh karena membutuhkan banyak kabel .

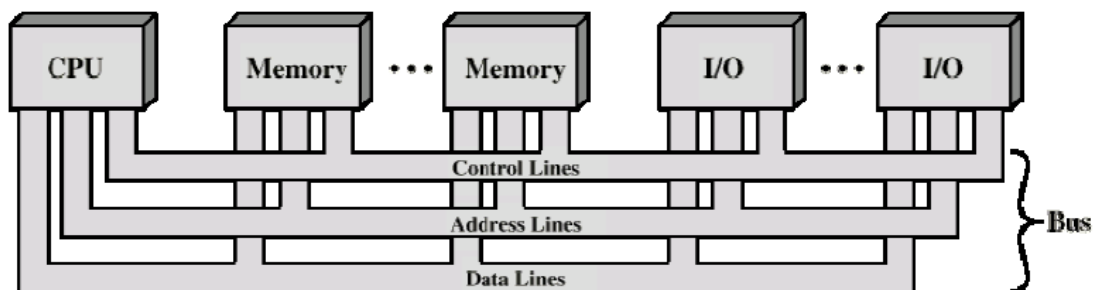
2). Sistem Serial

Data masukan/keluaran dikirim secara bit per bit berurutan melalui satu jalur, lebih sedikit membutuhkan kabel sehingga cocok untuk komunikasi jarak jauh.

Contoh: modem, *keyboard*, USB, dll.

F. ORGANISASI BUS / INTERKONEKSI

Interkoneksi yang umumnya berupa bus merupakan sarana pengirim informasi atau sinyal – sinyal yang dikelompokkan oleh fungsi-fungsi. Antara memori utama dengan prosesor dihubungkan oleh suatu bus. Bus ini terdiri dari beberapa jalur sinyal paralel. Untuk tiap fungsi ada dua jenis yang khas, yaitu : *intern* dan *ekstern*. *Busintern* merupakan bus-bus yang diimplementasikan pada serpih untuk menghubungkan semua unsur logika. Untuk menghubungkan sebuah sistem, bus yang pokok adalah *busekstern*.



Gambar 6. Struktur bus dalam sistem mikroprosesor

Ada 3 bus standar pada sistem mikroprosesor, yaitu :

- **Bus data**

Merupakan saluran untuk memuat data dari dan ke mikroprosesor. Pada mikroprosesor, standar bus data adalah bus dua arah 8-bit (artinya dapat dipakai pada kedua arah).

- **Bus alamat**

Merupakan bus yang berisi alamat – alamat yang datanya akan dikirim / diterima oleh mikroprosesor. Pada mikroprosesor standar, bus alamat adalah 16-bit yang memungkinkan pengiriman sampai 64 KB (2 MB) alamat-alamat *ekstern*.

- **Bus pengendali**

Bus pengendali membawa sinyal-sinyal penyerempak antara mikroprosesor dan semua alat yang dihubungkan kepada bus-bus. Sinyal-sinyal khas yang terdapat pada bus pengendali adalah sinyal-sinyal baca, tulis, interupsi, reset, dan berbagai macam pemberitahuan.

2. MIKROPROSESOR DAN PERBEDAAN MIKROPROSESOR 8088 DENGAN 8086 SECARA UMUM

A. MIKROPROSESOR

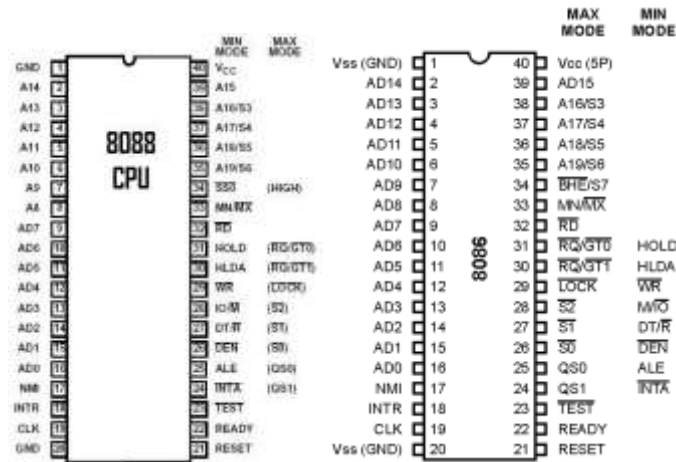
Mikroprosesor adalah komponen LSI (*LargeScale Integration* –IC dengan beberapa puluh ribu transistor per chip–) yang merupakan komponen sentral pada sistem mikrokomputer yang memiliki dimensi ukuran sangat kecil. Mikroprosesor biasa disebut sebagai otak dari komputer karena bertugas menghitung dan mengontrol peralatan lain di sekitarnya.

Bersama dengan komponen LSI lainnya, mikroprosesor dipakai untuk membuat sebuah sistem mikroprosesor. Mikroprosesor melakukan sebagian besar fungsi atau kerja unit pemroses, sedangkan komponen lainnya melaksanakan fungsi unit memori, I/O (Input Output) dan fungsi lain yang diperlukan.

B. PERBEDAAN MIKROPROSESOR 8088 DENGAN 8086

Perbedaan pokok antara 8086 dan 8088 terletak pada banyaknya saluran data yang dikeluarkan bus. Chip 8086 memiliki 16 saluran data pada busnya, dan 8088 hanya memiliki 8 saluran data. Hal ini dapat dilihat dari banyaknya saluran AD (alamat/data) terhadap saluran A (alamat) pada pin keluar untuk setiap chip. Chip 8086 memakai 16 saluran alamat untuk data, sehingga AD0 hingga AD15 dipakai untuk alamat dan data sedangkan 8088 hanya memiliki AD0 hingga AD7 untuk data dan

memakai A8 hingga A19 untuk alamat. Saluran data internal pada chipnya sama saja, dan masing-masing menambahkan, mengurangi, mengalikan atau membagi bilangan biner 16 bit.



Gambar 7 : Mikroprosesor 8086 dengan 8088

Keistimewaan mikroprosesor 8086 dan 8088 terletak pada antrian instruksi yang dipakai pada masing-masing. Chip 8086 dan 8088 membaca instruksi secara teratur dari memori mendahului operasi mereka, dan instruksinya diletakkan dalam suatu antrian yang terdiri dari sekumpulan register flip-flop. Cara ini mempercepat operasi karena pemroses dapat terus melakukan instruksi yang memakan waktu (misalnya perkalian) dan dalam waktu yang bersamaan membaca instruksi dari memori pemroses.

C. ORGANISASI DAN PENGALAMATAN MEMORI

Memori utama terdiri dari sejumlah sel yang masing-masing dapat menyimpan informasi sebesar 1 byte (8 bit). Masing-masing sel memori diberi suatu alamat (*address*) dimulai dari 0 sampai dengan jumlah dari sel memori dikurangi 1. Dengan adanya address ini, maka letak lokasi dari memori dapat dihubungi. Pada mikroprosesor intel 8086/8088 memori mencapai 1 Mbyte = 1048576 byte).

Dengan besarnya ruang memori sebesar 1 MB (1 MB = 2²⁰ = 20 saluran alamat → dari A0 sampai A19), yang dimulai alamat terendah yaitu 00000 sampai alamat tertinggi FFFFF, sedangkan kemampuan yang ada pada 8088 sebesar 16 bit yang berarti masih kekurangan 4 bit lagi untuk menampung suatu alamat memori. Dengan demikian

harus dilakukan penomoran dengan dua register. Sebuah register berisi 16 bit dihitung sebelah kiri (*register segment*) dan sebuah register lain lagi berisi 16 bit dari kanan (*register offset*). Tiap segment dapat menjangkau 64 KB. Antara segment dan offset ditulis dengan dipisahkan tanda titik dua (:) seperti berikut ini :

segment : offset

Contoh: 010E : 1406

D. PENGALAMATAN ORGANISASI MEMORI

Cara pengalamatan memori yang dilakukan oleh komputer sering disebut dengan pengalamatan relatif (*relative address*), sedangkan yang kita perlukan adalah kemampuan 20 bit sehingga pengalamatan yang dilakukan adalah pengalamatan mutlak atau absolut atau fisik.

Perlu diingat lagi bahwa 1 KB = 1024 byte sedangkan 1 MB = 1048576 byte. Dengan demikian alamat memori dari 0 sampai 1 MB memerlukan tempat lima digit angka *hexadecimal*, yaitu 00000 sampai dengan FFFFF H.

Register yang ada adalah register 16 bit yang berarti hanya dapat menampung 4 digit hexadesimal, yaitu dari 0000 sampai dengan FFFF H. Oleh karena itu, dilakukan *segmentasi* untuk memperoleh alamat fisik 20 bit atau 5 digit bilangan *hexadecimal*.

Segmentasi dilakukan dengan penggabungan antara segment register dengan offset register. Di antara register untuk mencatat alamat memori yang dipergunakan adalah *segment register* digabung dengan *offset register*. Aturan penulisan untuk segment register yaitu nilai digit terendah adalah 16 pangkat 1 dan digit tertinggi 16 pangkat 4 (hal ini akibat *segment register* digeser ke kiri satu digit), sedangkan pada *offset register* nilai digit terendah adalah 16 pangkat 0 dan tertinggi adalah 16 pangkat 3.

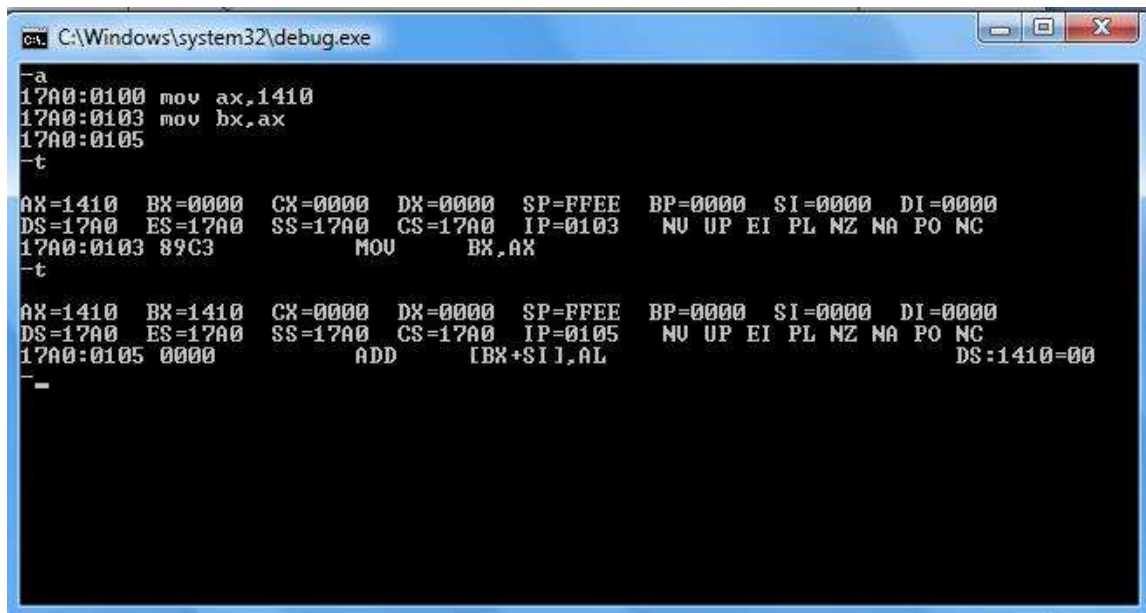
Contoh : alamat di memori adalah 2845 : FB00 yang berarti alamat segment 2845H dan offset adalah FB00H. maka cara untuk memperoleh alamat fisik digunakan aturan berikut :

	2845
	<u>FB00</u>
Alamat fisik	37F50

Seperti yang telah disebutkan diatas bahwa prosesor 8088 secara langsung dapat berhubungan dengan lokasi memori sebanyak 1 MB, yang dimulai 00000H sampai dengan FFFFFH dan keseluruhan lokasi memori tersebut terdapat pada dua jenis memori yaitu RAM dan ROM.

3. MACAM – MACAM COMMAND DALAM DEBUG

Command	Arti	Keterangan
A	Assemble	Menulis instruksi-instruksi yang akan dijalankan ke memori
C	Compare	Membandingkan data pada alamat tertentu
D	Dump	Melihat isi dari memori pada alamat tertentu
E	Enter	Memasukan atau mengisi ke dalam memori
F	Fill	Mengisi secara langsung suatu blok memori
G	Go	Menjalankan semua instruksi yang ada di memori
H	Hexadesimal	Menjumlah dan mengurangi dua bilangan hexadesimal
L	Load	Me-load file ke memori
M	Move	Mengcopy data tertentu dari alamat tertentu
N	Name	Memberikan nama file yang akan di edit atau disimpan
R	Register	Menampilkan isi semua register
Rxx	register xx	Mengubah isi suatu register xx
T	Trace	Menjalankan instruksi-instruksi yang ada di memori <i>instruction per instruction</i>
U	Unassemble	Melihat instruksi-instruksi yang ada di memori
W	Write	Menulis isi ke memori file



```

C:\Windows\system32\debug.exe
-a
17A0:0100 mov ax,1410
17A0:0103 mov bx,ax
17A0:0105
-t
AX=1410 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A0 ES=17A0 SS=17A0 CS=17A0 IP=0103  NU UP EI PL NZ NA PO NC
17A0:0103 89C3      MOV     BX,AX
-t
AX=1410 BX=1410 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A0 ES=17A0 SS=17A0 CS=17A0 IP=0105  NU UP EI PL NZ NA PO NC
17A0:0105 0000      ADD     [BX+SI],AL
DS:1410=00
-

```

4. PEMROGRAMAN INTERNAL REGISTER

A. DASAR TEORI

I. Jenis – Jenis Register di Mikroprosesor 8088

Register merupakan lokasi memori yang sangat khusus. register dalam mikroprosesor 8088 didesain untuk menampung data dan dapat diakses dalam kecepatan yang tinggi. Adapun jenis-jenis register yang terdapat dalam mikroprosesor 8088 antara lain sebagai berikut :

1) General Purpose Register

General register 8086/8088 adalah register 16 bit yang terdiri dari AX, BX, CX, dan DX. Sebagai pilihan, bagian tertinggi (AH, BH, CH, dan DH yang masing-masing 8 bit) atau bagian terendah (AL, BL, CL, dan DL yang masing-masing 8 bit) dari setiap register dapat dialamatkan tersendiri untuk operasi byte. Beberapa fungsi khusus dari General Purpose Register :

a) Akumulator register (AX)

Berfungsi sebagai akumulator dan berhubungan dengan jenis-jenis operasi khusus seperti aritmatika, IN, OUT, shift, logic, rotate dan operasi desimal berkode biner.

b) Base register (BX)

Dapat dipakai sebagai register base untuk mereferensikan alamat memori. Register ini dipisahkan menjadi BH dan BL yang masing-masing 8 bit. Operasi yang dapat dilakukan adalah rotate, aritmatika, shift dan logic.

c) Counter register (CX)

Dipakai sebagai pencacah implisit dengan instruksi tertentu, misalnya terhadap perintah LOOP dan operasi string. Cacahan naik bila direction flag 0 dan cacahan turun jika direction flag adalah 1. Register CX dipisahkan atas dua bagian yaitu CH dan CL yang masing-masing 8 bit.

d) Data register (DX)

Dipakai untuk menyimpan alamat port I/O selama operasi input dan output tertentu, baik alamat port 8 bit maupun alamat port 16 bit. Digunakan juga dalam operasi-operasi perkalian untuk menyimpan sebagian dari hasil kali 32 bit atau dalam operasi

pembagian untuk menyimpan suatu nilai sisa. Register ini dipisahkan menjadi DH dan DL masing-masing 8 bit.

General-Purpose Registers					
31	16-15	8-7	0	16-bit	32-bit
	AX	AL		AX	EAX
	BX	BL		BX	EBX
	CX	CL		CX	ECX
	DX	DL		DX	EDX
		BP			EBP
		SI			ESI
		DI			EDI
		SP			ESP

Gambar 8. General Purpose Register

2) Segmen Register

Register yang berfungsi untuk mengambil instruksi. Mengingat pada mikroprosesor 8088 untuk mengambil instruksi, pengalamatannya memerlukan segmen dan offset, maka register ini digunakan untuk menyimpan alamat segmennya. Register ini terdiri dari 16 bit, yaitu :

a) CS (Code Segment)

Digunakan untuk mencatat segment dari kode program atau instruksi.

b) DS (Data Segment)

Digunakan untuk menyimpan alamat dari segment letak data.

c) SS (Stack Segment)

Untuk menyimpan alamat segment memori yang dipergunakan menjadi stack.

d) ES (Extra Segment)

Digunakan untuk menyimpan alamat segment tambahan, misalnya alamat display, alamat sistem operasi dan sebagainya yang diinginkan oleh pemrogram.

3) Pointer Register

Register untuk menyimpan alamat offset, register ini menangani 16 bit yang biasanya mengakses memori *operand*. Register ini terdiri dari :

a) SP (Stack Pointer)

Menyimpan alamat offset dari stack.

b) BP (Base Pointer)

Digunakan untuk penunjuk base dalam *stack* yang disediakan sebagai daerah penyimpanan data, digunakan juga dalam komunikasi dengan bahasa komputer seperti bahasa C dengan assembler.

c) SI (Source Index) dan DI (Destination Index)

Digunakan untuk menyimpan nilai-nilai offset dalam segmen data memori pada saat bersangkutan

d) IP (Instruction Pointer)

IP adalah register 16 bit yang berfungsi menentukan alamat offset tempat tersimpannya kode instruksi berikutnya yang akan dilaksanakan oleh prosesor. IP merupakan Register pasangan CS, register utama untuk menunjukkan baris perintah program. Pada saat pertama program dijalankan, IP akan langsung menunjuk pada awal program. Code Segment dan Instruction Pointer berfungsi sebagai program counter dan ditulis dengan format CS:IP. Pada umumnya semua kode bahasa mesin ditaruh di code segment, semua data ditaruh pada data segment sedangkan operasi PUSH dan POP disediakan tempat di stack segment

CS	IP
DS	DI+SI+BX
SS	SP,BP
ES	DI+SI+BX

Gambar 9: Pasangan Segmen dan offset

4) Flag Register

Register 16-bit khusus dgn posisi bit tunggal yang dipekerjakan (assigned) untuk menunjukkan status CPU atau hasil-hasil operasi aritmatik. Dari 16 bit posisi hanya 9 bit posisi yang digunakan dan diberi nama yang diperlihatkan pada gambar dibawah ini :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	OF	DF	IF	TF	SF	ZF	*	AF	*	PF	*	CF

Gambar 10 : Flag Register

Terdapat 2 jenis tipe flag dasar :

A. Control Flag

1) DF (Direction Flag)

Dipergunakan untuk mengontrol arah daripada operasi string. Jika DF bernilai 1 maka register SI dan DI akan menurun (decrement). Sedang bila bernilai 0 maka register SI dan DI akan meningkat (increment). Pada program assembler, register ini digunakan untuk instruksi-instruksi MOVS, MOVSB, MOVSW, CMPS, CMPSB, dan CMPSW.

2) IF (Interrupt Flag)

Dimana jika diset (nilai 1) dapat melakukan interupsi dan sebaliknya bila bernilai 0, maka interupsi tidak dapat dilakukan.

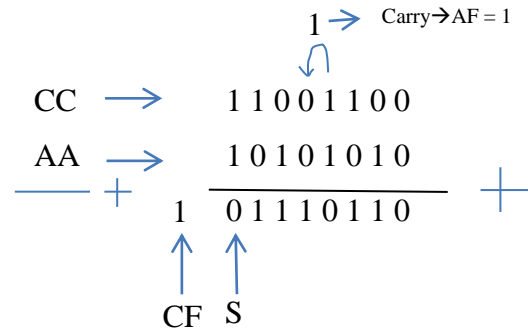
3) TF (Trap Flag)

Menentukan apakah CPU perlu dihentikan setelah setiap instruksi dilaksanakan. Digunakan pada saat Debugging program → TRACING

B. Status Flag

Flag Register	Arti	Nilai Bit 0	Nilai Bit 1
Carry Flag (CF)	Merupakan sebuah carry out atau borrow	Tidak ada carry	Ada carry
Overflow Flag (OF)	Menunjukkan sebuah operasi secara tidak benar, akibat perubahan tanda bit	Tidak ada perubahan tanda	Ada perubahan tanda
Zero Flag (ZF)	Diset jika hasil operasi aritmatik atau logika bernilai zero	Tidak semua bit nol	Jika semua bit nol
Auxiliary Flag (AF)	Diset jika suatu operasi mengakibatkan suatu carry/borrow dari bit 3 ke bit 4 dari sebuah operand	Tidak ada carry/borrow dari bit 3 ke bit 4	Ada carry/borrow dari bit 3 ke bit 4
Sign Flag (SF)	Menunjukkan tanda dari suatu bit	positif	Negative
Parity Flag (PF)	Ada 2 macam yaitu parity genap dan parity ganjil	Ganjil (jumlah bit 1 ganjil)	Genap (jumlah bit 1 genap)

Contoh :



CF = 1 ZF=0 (01110110 ≠ 00000000)
 AF = 1 PF=0 (01110110 → jumlah bit 1=5 → parity ganjil)
 SF = 1 OF= 1

II. Instruksi - instruksi pada mikroprosesor

Secara garis besar instruksi pada mikroprosesor dibagi menjadi :

Tipe Instruksi	instruksi	Format
Data Transfer	MOV	MOV destination,source
	PUSH	PUSH Source
	POP	POP Destination
	IN	IN Accumulator,port
	OUT	OUT Port, accumulator
Aritmatik	ADD	ADD destination, source
	SUB	SUB destination, source
	MUL	MUL source
	DIV	DIV source
	NEG	NEG destination
	DEC	DEC destination
	INC	INC destination

Logika	AND	AND destination,source
	OR	OR destination,source
	XOR	XOR destination,source
	NOT	NOT destination

III. Pola pengalaman pada mikroprosesor 8088

Pola Pengelamatan	Contoh	Arti
Register Addressing	MOV AX, CX	copy isi CX ke AX
Immediate Addressing	MOV CX, 500	Bilangan 500 dimasukan ke CX
Direct Addressing	MOV AL,[1234h]	copy isi memori dengan alamat 1234H ke AL
Register indirect Addressing	MOV AX,[BX]	copy isi memori yang ditunjukan BX ke AX
Base relative Addressing	MOV AX,[BX]+4	EA= Displacement + isi BX atau BP
Direct Indexed Addressing	MOV AX,[DI+5]	EA= Displacement + Index reg DI atau SI
Base Indexed Addressing	MOV AX,[BX+SI+7]	EA= Base reg + Index Reg +Disp

IV. PEMBUATAN KODE OPERASI (Op Code)

Seperti sudah dijelaskan pada tingkatan bahasa pemrograman, bahwa yang secara langsung dibaca dan dicerna oleh serpih mikroprosesor adalah instruksi dalam bentuk kode operasi (*op code*) atau biasa disebut dengan bahasa mesin. Kode operasi ini ditulis dalam bilangan heksadesimal dengan aturan tertentu, yang dihasilkan dengan menerjemahkan bahasa *assembly / mnemonic*.

1. Pembuatan op-code secara manual

- a. Tentukan perintah program yang akan dibuat kode operasinya.

Contoh : **MOV BX, DX**

- b. Lihat pada tabel instruction set (data sheet Intel 8088) perintah program yang kita buat. Dari banyak pilihan tambahan pada instruction set itu, ambil salah satu yang sesuai dengan perintah program. Dari contoh program diatas, dapat diketahui bahwa program diatas adalah program yang meng-copy data dari register DX ke register BX, sehingga dapat kita pilih **Register/Memory to/from Register**.

- c. Ambil kode instruksinya dan tuliskan. Contoh : kode instruksi **Register/Memory to/from**

Register adalah **100010dw mod reg r/m**.

- d. Gantilah variabel yang tidak diketahui seperti reg, w, data, dan lain-lain dengan digit biner yang sesuai. Variabel-variabel ini dapat dicari pada NOTES dari tabel instruction set, yaitu :

♦ **d (direction) :**

Menyatakan arah perpindahan data, d=1 bila data berpindah ke suatu register, d=0 bila data berpindah dari suatu register. Pada perintah diatas, d dapat berisi 1 atau 0, sehingga dapat dipilih salah satu. Misal kita pilih d=1, berarti data dinyatakan berpindah ke suatu register yaitu register BX.

♦ **w (word) :**

Menyatakan apakah operasi melibatkan data word (16 bit) atau tidak. Bila ya, maka w=1, bila tidak w=0. Perintah MOV BX,DX melibatkan word, karena baik BX maupun DX merupakan register 16 bit. Sehingga kita tentukan w=1.

♦ **mod (mode) :**

Merupakan mode lintas data, ada empat pilihan yang terdapat kata DISP melibatkan memori, sehingga kita pilih mod=11 yang menyatakan operasi antar register.

♦ **reg(register) :**

Menyatakan jenis register yang digunakan. Karena kita telah menentukan d=1, yang berarti data berpindah ke register BX, maka kita cari kode biner untuk register BX, yaitu 011.

♦ r/m (register/memory) :

Menyatakan cara perhitungan physical address dari memori. Karena perintah program kita tidak melibatkan memori, maka kita boleh sembarang memilih. Misalkan kita pilih r/m=111.

Dari keterangan NOTES diatas, kita dapatkan: **10001011 11011111**.

e. Konversikan kode biner yang telah diperoleh ke dalam bentuk heksadesimal.

1000 1011 1101 1111 = 8BDF

f. Kode operasi telah didapat, yaitu : **8BDF**

Perlu diketahui, karena memori mikroprosesor yang digunakan mempunyai lebar data 8 bit, maka untuk setiap satu alamat memori hanya dapat diisi dengan dua digit bilangan heksadesimal sehingga untuk kode operasi 8BDF (MOV BX,DX) memerlukan dua alamat memori (untuk data 8B dan data DF).

Contoh mencari offset :

Mov [1234],DL

Kode instruksi: 1 0 0 0 1 0 D W MOD REG R/M

1000	1000	00	010	110	34	12
└──┬──┘		└──┬──┘		└──┬──┘		
8		8		1		6

Jadi kode offsetnya adalah 88163412

Mov AX,BX

Kode instruksi: 1 0 0 0 1 0 D W MOD REG R/M

1000	1001	11	011	000
└──┬──┘		└──┬──┘		└──┬──┘
8		9		D

Jadi kode offsetnya adalah 89D8

2. Pembuatan op-code dengan menggunakan program Debug.exe

- Program ini dapat dijalankan di DOS, jadi kita dapat bekerja di MSDOS atau under DOS.
- Setelah berada pada program DOS (muncul prompt C), ketik debug lalu tekan enter, muncul tanda "-". Jika berada di lingkungan Windows, klik 'Start', pilih 'Run', ketik 'debug', lalu klik OK.

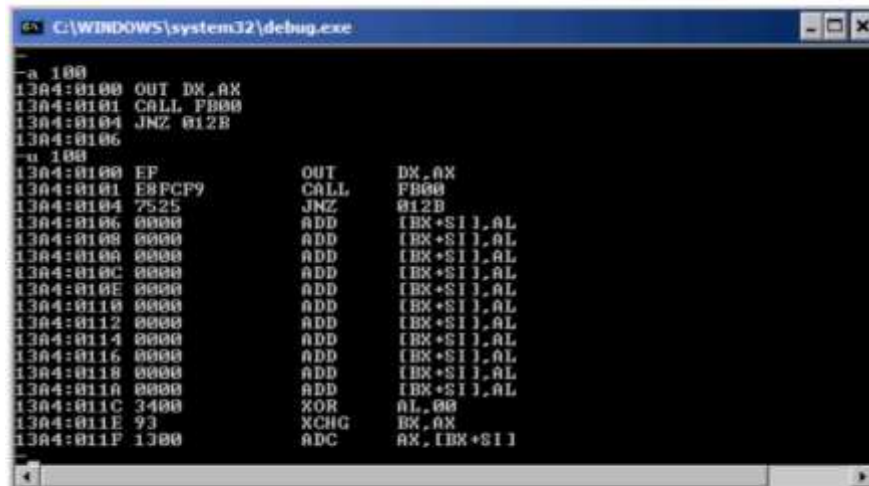
- c. Untuk mencari op-code dari suatu perintah program, ketik **a 100**, lalu tekan enter.
- d. Ketik semua perintah program yang akan dicari kode operasinya. Setiap ganti perintah program tekan enter satu kali, dan setelah semua program yang ingin diketahui op-codenya ditulis, tekan enter dua kali sampai muncul prompt debug -.
- e. Untuk melihat *op-code* dari perintah program, ketik **u 100** lalu enter.
- f. Format dari tampilannya adalah sebagai berikut :

-u 100

13A4:0100 EF OUT DX,AX

13A4:0101 E8FCF9 CALL FB00

13A4:0104 7525 JNZ 012B



```

C:\WINDOWS\system32\debug.exe
-a 100
13A4:0100 OUT DX,AX
13A4:0101 CALL FB00
13A4:0104 JNZ 012B
13A4:0106
-u 100
13A4:0100 EF OUT DX,AX
13A4:0101 E8FCF9 CALL FB00
13A4:0104 7525 JNZ 012B
13A4:0106 0000 ADD [EBX+SI],AL
13A4:0108 0000 ADD [EBX+SI],AL
13A4:010A 0000 ADD [EBX+SI],AL
13A4:010C 0000 ADD [EBX+SI],AL
13A4:010E 0000 ADD [EBX+SI],AL
13A4:0110 0000 ADD [EBX+SI],AL
13A4:0112 0000 ADD [EBX+SI],AL
13A4:0114 0000 ADD [EBX+SI],AL
13A4:0116 0000 ADD [EBX+SI],AL
13A4:0118 0000 ADD [EBX+SI],AL
13A4:011A 0000 ADD [EBX+SI],AL
13A4:011C 3400 XOR AL,00
13A4:011E 93 XCHG BX,AX
13A4:011F 1300 ADC AX,[EBX+SI]
  
```

Gambar 12. Format tampilan *debug.exe*

8088 Instruction Set

MOV - Move

Immediate to Reg	1011 wreg	data-LSB	data-MSB (w=1)	T=4
Register Direct	1000 10dw	modreg/r/m		T=2
Register Indirect	1000 10dw	modreg/r/m		T=13
Memory to Accumulator	1010 000w	addr-low	addr-high	T=14
Accumulator to Memory	1010 001w	addr-low	addr-high	T=14
Index	1000 10dw	modreg/r/m	disp	T=16

PUSH - Push

Register	0101 0reg	T=15
Segment Register	000reg110	T=14

POP - Pop

Register	0101 1reg	T=12
Segment Register	000reg111	T=12

XOR - Exclusive Or

Reg and Reg	0011 00dw	modreg/r/m		T=3
Immediate to Reg	1000 000w	mod110r/m	data-LSB	T=4
Immediate to AX/AL	0011 010w	data-LSB	data (if w=1)	T=4

ADD - Add

Reg with Reg	0000 00dw	modreg/r/m		T=3
Immediate to Reg	1000 000w	mod000r/m	data-LSB	T=4
Immediate to AX/AL	0000 010w	data-LSB	data (if w=1)	T=4

JMP - Unconditional Jump

Direct w/in Segment Short	1110 1011	disp	T=15
JZ - Jump on Zero	0111 0100	disp	T=16/4
JC - Jump on Carry	0111 0010	disp	T=16/4

d=direction: if d=1 then 'to' reg (Reg ← Mem)

if d=0 then 'from' reg (Reg ← Reg, Mem ← Reg)

w=word: if w=1 then word operation (1 word = 2 bytes)

if w=0 then byte operation

mod=mode:

if mod=00 then DISP=0, disp-low and disp-high are absent

if mod=01 then DISP=disp-low (signed), disp-high are absent

if mod=10 then DISP=disp-low and disp-high

if mod=11 then field r/m is translated as a register (REG field)

disp=displacement:

show how far should the CPU jump from recent point (reg. IP)

r/m: if r/m = 111 then EA = (BX) + DISP

REG is assigned according to the following table:

16-Bit (w=1)	8-Bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

note: remember that 8088 always exchange the place of the operand

MODUL II

PROGRAMMABLE PERIPHERAL INTERFACE (PPI 8255)

A. TUJUAN

1. Praktikan mampu memahami PPI 8255 dan port-port yang ada secara fisik
2. Praktikan mampu membuat program inisialisasi pada PPI sesuai dengan mode dan *Control Word* yang diinginkan, yaitu :
 - Mampu memprogram masing-masing port pada PPI 8255 sebagai input atau output pada BGC-8088 *Microengineer*
 - Mampu melakukan pilihan handshaking pada mode 1 untuk memastikan transfer data dari dan ke Port
 - Mampu memahami dan mengatur mode-mode operasi untuk PPI 8255 pada BGC 8088 *microengineer*

B. PERALATAN

1. BGC-8088 *Microengineer*
2. Blok indikator LED
3. Jumper kabel

C. TEORI

1. Pendahuluan

Dalam suatu sistem mikroprosessor dimungkinkan suatu mikroprosessor dapat berkomunikasi dengan peralatan yang lain. Karena seringkali data pada peralatan lain yang berada di luar mikroprosessor mempunyai format dan kecepatan yang berbeda maka diperlukan suatu antarmuka. Komunikasi data dapat dilakukan dengan dua cara yaitu :

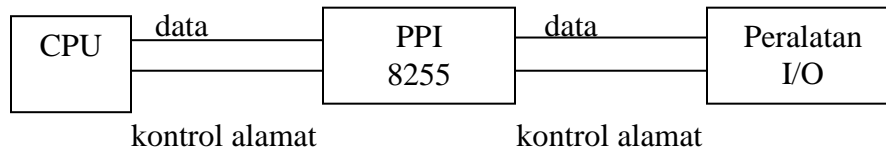
- a. Transfer data paralel (dengan PPI 8255)

Transfer data paralel memerlukan jumlah saluran sebanyak jumlah bit data yang dikirimkan. Pada umumnya, di dalam komputer semua data dikirim/ditransfer secara paralel melalui saluran data bus.

b. Transfer data serial (dengan PCI 8251-USART)

Transfer data serial hanya menggunakan satu buah saluran dalam pengiriman data. Untuk transfer data antar komputer pada umumnya menggunakan teknik transfer data serial agar lebih praktis dan efisien.

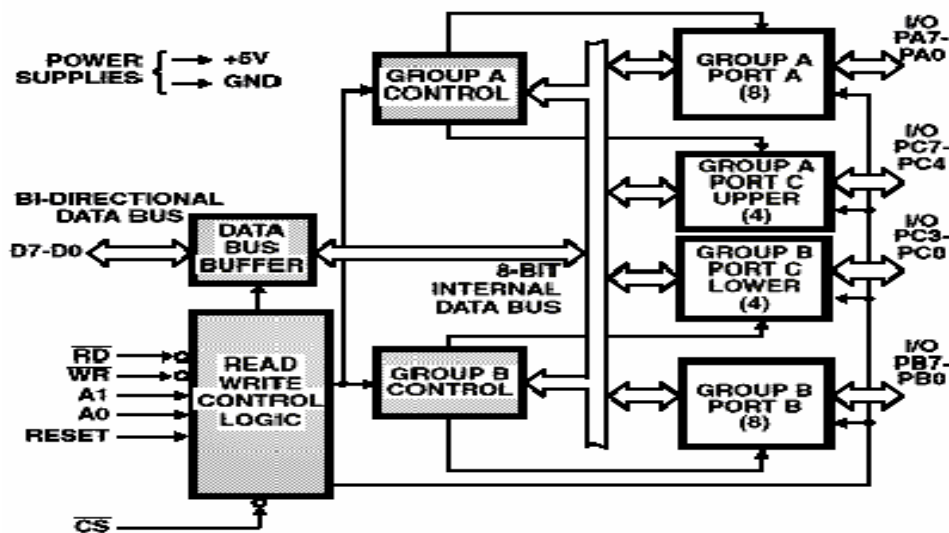
Programmable peripheral interface (PPI) 8255 adalah IC yang digunakan sebagai antarmuka yang menghubungkan sistem mikroprosesor dengan peralatan lain. Berikut digambarkan hubungan PPI dengan peralatan lain :



Gambar 16. Hubungan PPI dengan peralatan lain

2. Blok Diagram dan Port-Port pada PPI 8255.

Gambar blok diagram dan pin-pin keluaran untuk IC 8255 :



Gambar 17. Blok Diagram PPI 8255

IC 8255 merupakan suatu perangkat antarmuka yang dapat diprogram. Perangkat ini mempunyai empat port dan masing-masing mempunyai 8-bit jalur I/O. Setiap port dapat

diprogram masing-masing sebagai port input maupun port output, chip ini juga disiapkan untuk melakukan proses *handshaking* untuk memastikan transfer data dari dan ke port-port. Keempat port tersebut adalah:

a. Port A

Mempunyai 8 jalur yang dapat diprogram sebagai jalur input maupun output. Kedelapan jalur tersebut (A0-A7) harus diprogram seragam, baik sebagai input semua maupun output semua.

b. Port B

Mempunyai 8 jalur yang dapat diprogram sebagai jalur input maupun output. Kedelapan jalur tersebut (B0-B7) harus diprogram seragam, baik sebagai input semua maupun output semua.

c. Port C

Port ini terdiri dari dua bagian yaitu port C upper (PC upper) dan port C lower (PC lower), masing-masing terdiri dari 4 bit I/O dan dapat diprogram secara independen sebagai input atau output.

d. Port Control

Port control adalah port yang digunakan untuk mendefinisikan fungsi dari port-port lainnya. Port ini tidak bisa dijadikan sebagai input atau output.

Secara garis besar, PPI 8255 mempunyai 24 pin I/O yang terbagi menjadi:

- Group A terdiri dari: Port A (A0-A7) dan port C upper (C4-C7)
- Group B terdiri dari: Port B (B0-B7) dan port C lower (C0-C3)

3. Konfigurasi PPI pada BGC 8088

Pada praktikum PPI 8255 ini, digunakan BGC 8088 sebagai perangkat untuk pembelajaran sistem mikroprosesor. Pada BGC-8088 Microengineer, PPI 8255 mempunyai lokasi memori di alamat FF10 - FF1F. Masing – masing Port memiliki 4 lokasi memori. Sebagai contoh: Port A dapat diakses dengan lokasi memori di alamat FF10, FF14, FF18, atau FF1C, dan seterusnya.

Alokasi alamat bagi setiap Port dapat dilihat dengan lengkap pada tabel di bawah ini:

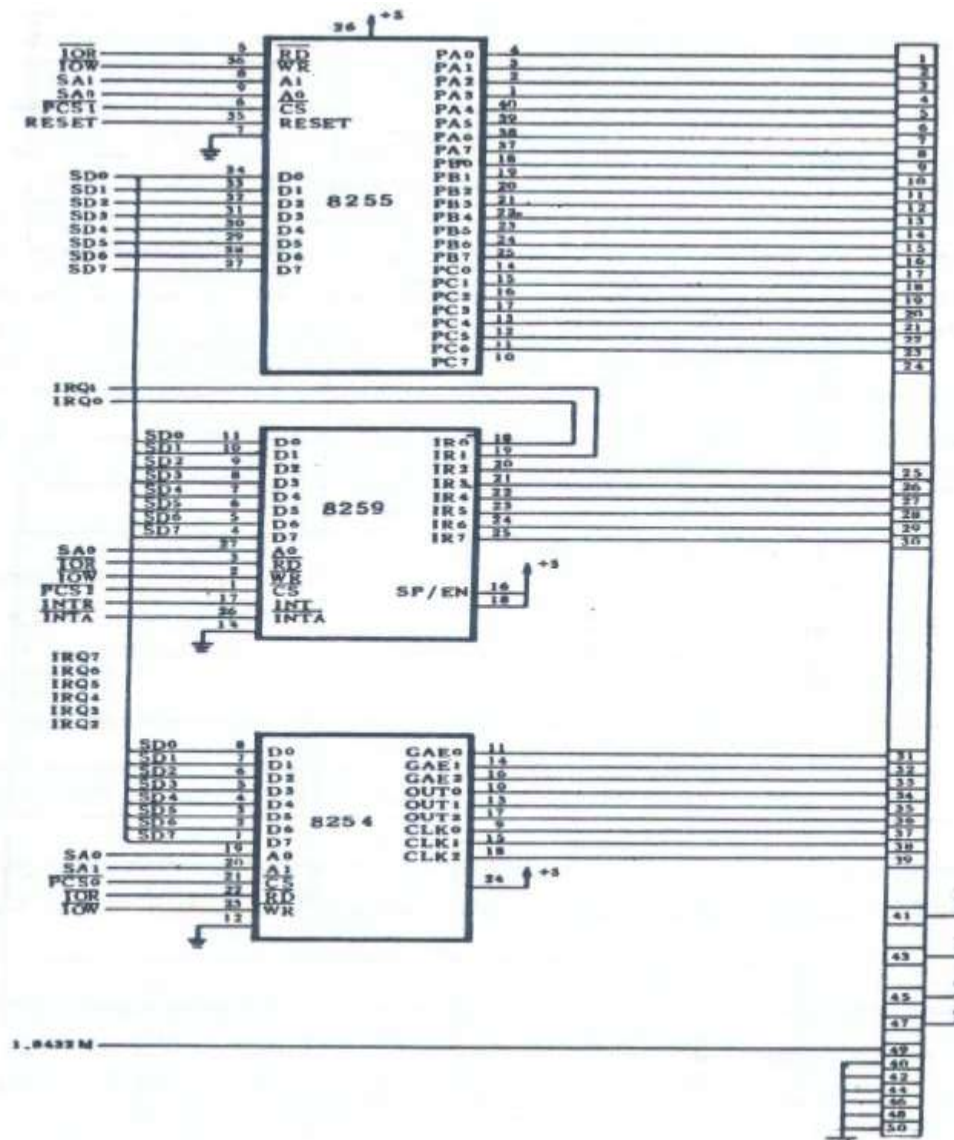
Port	Address
A	FF10, FF14, FF18, FF1C
B	FF11, FF15, FF19, FF1D
C	FF12, FF16, FF1A, FF1E
Control	FF13, FF17, FF1B, FF1F

Port	Alamat (hexadecimal)	Alamat (biner)						
		A15-A12	A11-A8	A7-A4	A3	A2	A1	A0
A	FF10	1 1 1 1	1 1 1 1	0 0 0 1	0	0	0	0
B	FF11	1 1 1 1	1 1 1 1	0 0 0 1	0	0	0	1
C	FF12	1 1 1 1	1 1 1 1	0 0 0 1	0	0	1	0
Control	FF13	1 1 1 1	1 1 1 1	0 0 0 1	0	0	1	1

Terlihat bahwa yang berbeda hanya dua bit terakhir yaitu A1 dan A0 hal ini dikarenakan address bus yang masuk hanya A1 dan A0 yang akan mengidentifikasi port yang diakses/digunakan.

Perhatikan bahwa alamat PPI 8255 tidak harus dari FF10 sampai FF1F. Penempatan alamat PPI 8255 bisa dimana saja tergantung dari perancangan hardware itu sendiri dan *address decoder* yang digunakan. Pada kenyataannya, PPI 8255 hanya menggunakan empat alamat (port A, B, C dan control), karena PPI 8255 hanya memiliki 2 kaki alamat yaitu A0 dan A1. Dan perhatikan bahwa konfigurasi hardware yang digunakan pada praktikum kali ini hanya untuk mempermudah perancangan saja, sehingga memungkinkan sebuah port pada PPI 8255 memiliki 4 alamat.

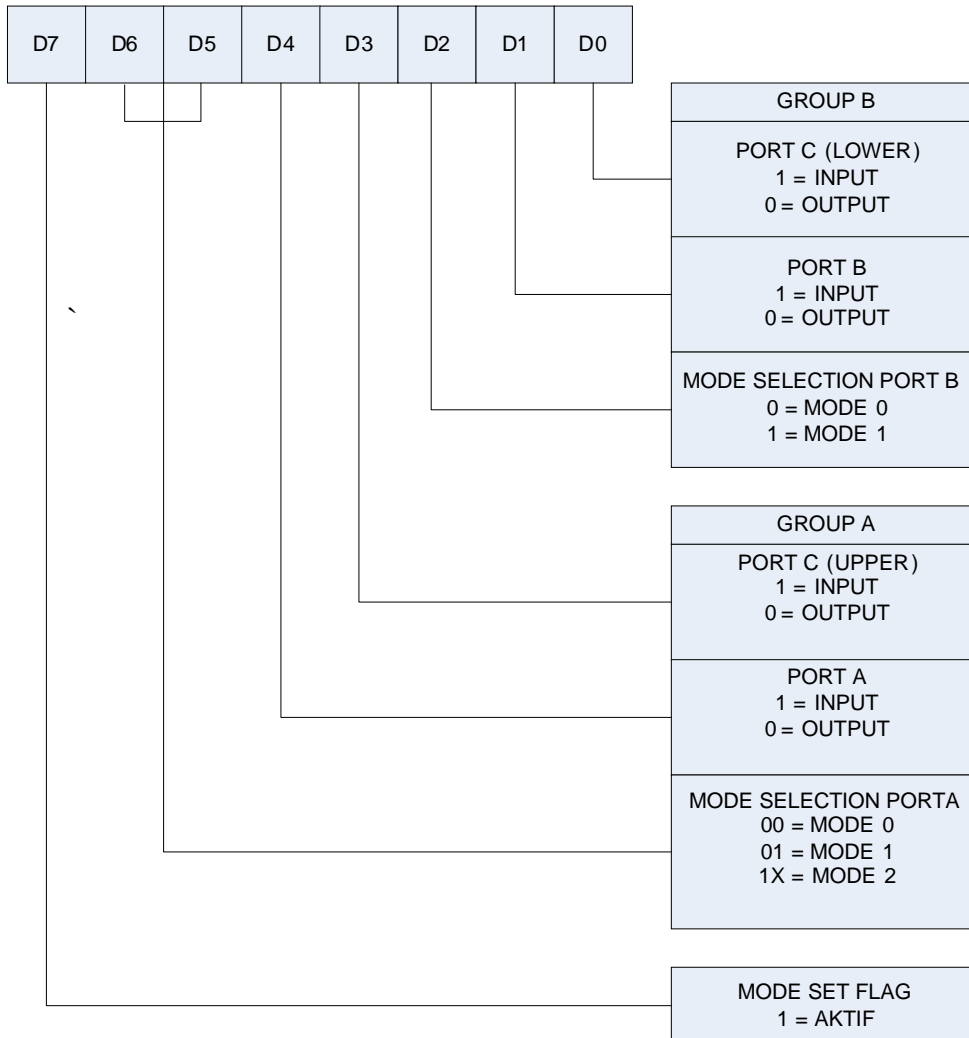
Sedangkan gambar berikut adalah rangkaian PPI yang berhubungan dengan perangkat paralel lainnya pada BGC 8088 Microengineer:



Gambar 18. Konfigurasi PPI Pada BGC 8088 Microengineer

4. Control Word

Ketika 8255 di-reset (dikembalikan ke kondisi awal), semua port-port difungsikan sebagai input. Tetapi jika port-port diinginkan tidak digunakan sebagai input, sebuah control word harus dikirim ke 8255 untuk mengubah mode-mode operasi port. *Control Word* merupakan data 8-bit yang mengandung informasi untuk menginisialisasikan fungsi PPI 8255. Format dari *Control Word* adalah:

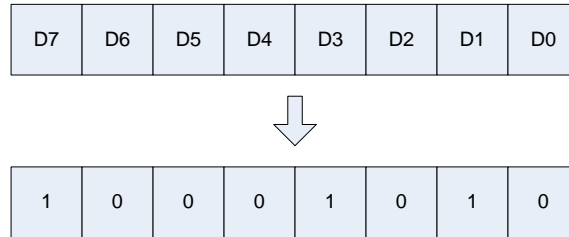


Gambar 19. Format Control Word

Contoh:

Nilai Control Word (CW) diset = $8A_H$

$8A_H = 10001010_2$



Gambar 20. Contoh Format Control Word

Penjelasan pada CW di atas misalnya pada instruksi: MOV AL, $8A_H$ maka dari data CW tersebut dapat diperoleh informasi bahwa PPI bekerja pada mode 0, Port A dan Port C lower menjadi output, Port B dan Port C upper menjadi input.

**cat : jika tidak ada keterangan apa-apa anggap bekerja pada mode 0*

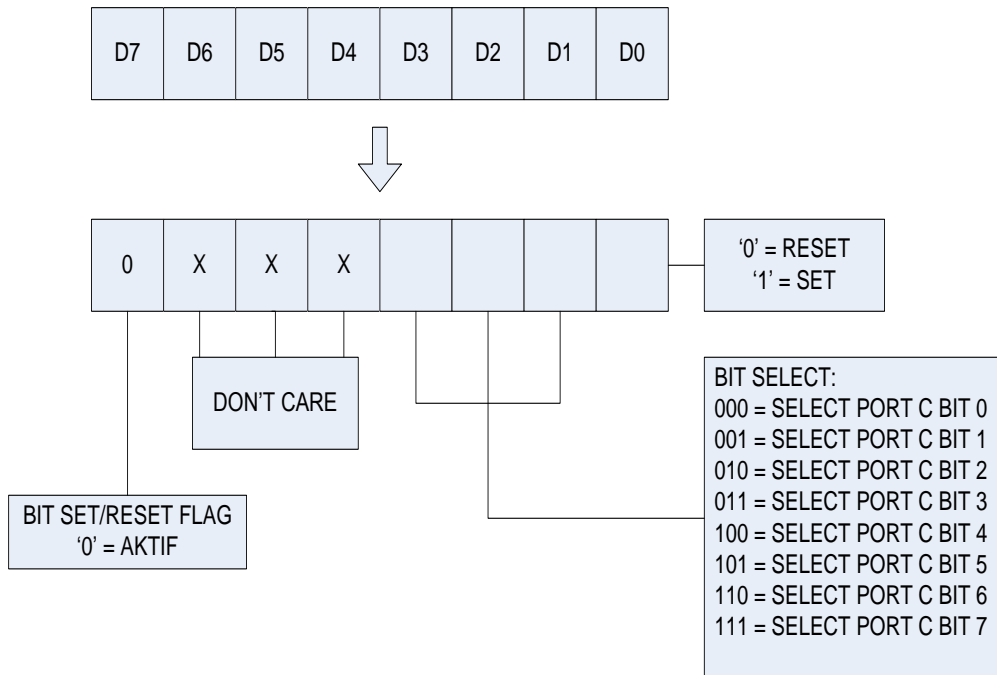
5. Set/Reset

Set/Reset merupakan operasi untuk mengeset atau me-reset bit-bit pada port C. Adapun syarat yang harus dipenuhi agar port C dapat menjalankan operasi khusus ini adalah:

- PPI bekerja pada mode 0
- Port C harus di-set sebagai output

Dari syarat di atas, maka control word (CW) yang mungkin digunakan untuk mengakses operasi Set/Reset yaitu 80_H , 82_H , 90_H , dan 92_H . CW spesial ini dipakai pada inisialisasi sebelum menggunakan operasi Set/Reset

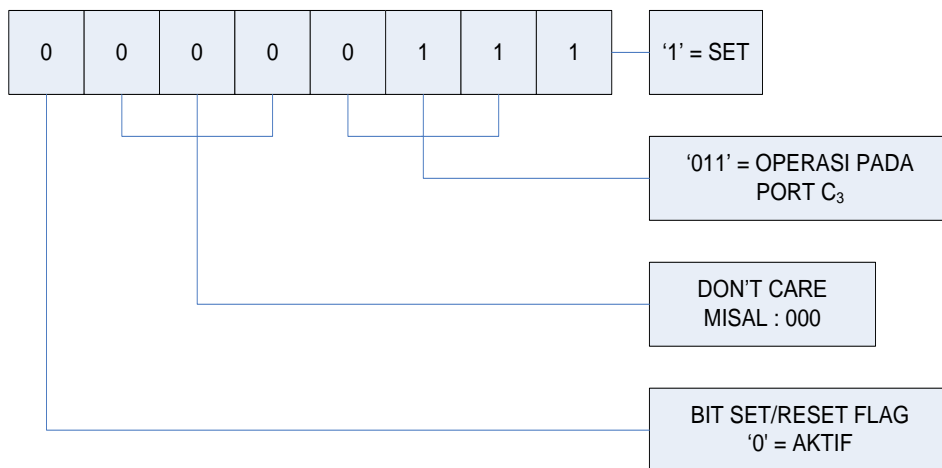
Format CW Set/Reset itu sendiri adalah sebagai berikut:



Gambar 21. Format Set/Reset

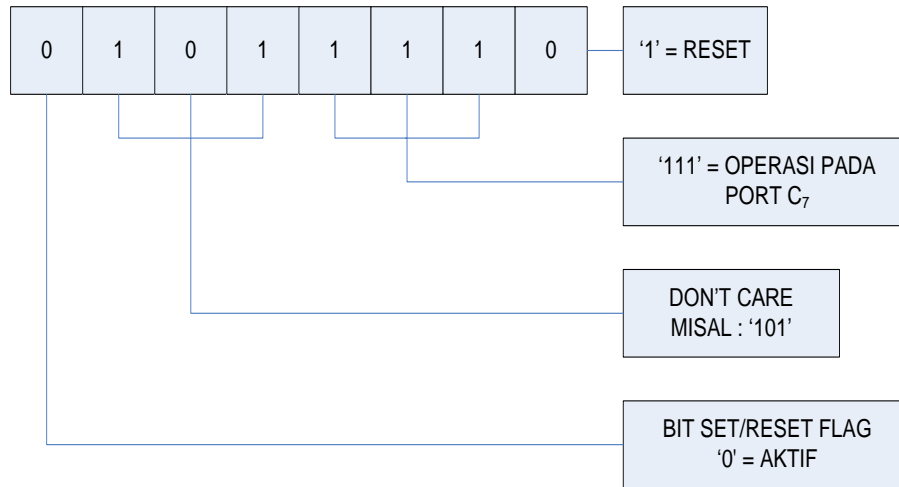
Contoh :

- mengesetport C3 :



Gambar 22.Contoh Format Set C3

- me-reset Port C7 :



Gambar 23.Contoh Format Reset C7

6. Mode-mode pada PPI 8255

Pada PPI terdapat tiga port yang dapat diprogram dan dioperasikan sebagai input/output. Perbedaan masing-masing mode ini terutama terletak pada proses *handshaking*. Handshaking adalah proses persetujuan antara dua piranti dalam pengiriman atau pertukaran data yang terjadi dalam mikroprocessor, handshaking secara umum diperlukan karena 2 piranti yang berhubungan beroperasi pada kecepatan yang berbeda.

Tiga pararel Port ini dapat diprogram dalam 3 macam mode, yaitu:

a. Mode 0, *Basic Input/Output*

Pada mode ini, setiap port bekerja secara independen. Masing-masing port bisa sebagai input atau output biasa tanpa proses handshaking. Jika port A dan port B digunakan pada mode 0, maka port C upper maupun port C lower dapat berfungsi bersama-sama sebagai port 8-bit ataupun mempunyai fungsi masing-masing sebagai port-port 4-bit.

Jika Port C digunakan sebagai output, maka bit-bit Port C dapat di-set atau di-reset secara individual dengan mengirimkan control word khusus ke kontrol register. Ketika control word dikirim ke register, PPI 8255 akan memeriksa kondisi logika bit D7 dari control word. Jika sebagai control word, D7 bernilai bit 1, sedangkan jika D7 bernilai 0, maka control word tersebut diartikan sebagai operasi bit set/reset yang digunakan untuk mengeset atau reset bit-bit pada port C.

Proses komunikasi pada mode ini yaitu pengirim akan langsung mengirim data pada penerima. Pengirim tidak akan mengecek apakah penerima telah siap atau tidak, Karena tidak terjadi proses handshaking pada mode ini.

b. Mode 1, *Strobe Input/Output*

Pada mode 1, jalur-jalur pada port C digunakan untuk mengontrol *handshaking* untuk transfer data (*single handshake*). Sinyal – sinyal kontrol ini mempunyai definisi yang berbeda disesuaikan dengan penggunaan port A dan port B apakah sebagai input atau output.

1) Jika Port A dan Port B diinisialisasikan sebagai input, maka

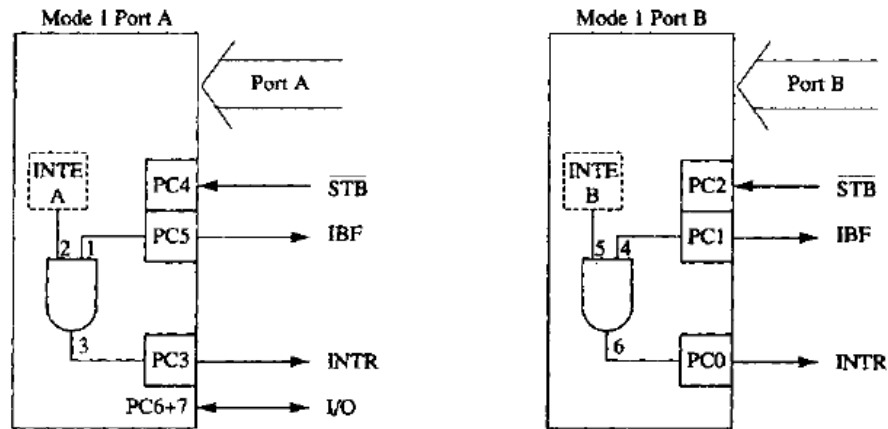
- PC0, PC1 dan PC2 digunakan sebagai jalur *handshake* untuk port B
- PC3, PC4 dan PC5 digunakan sebagai jalur *handshake* untuk port A
- PC6 dan PC7 digunakan sebagai jalur input atau output. Jika D3 pada CW = 1 maka PC6 dan PC7 adalah input

Tabel Konfigurasi jalur Port C pada saat Mode 1 Input

PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7
INTR _B	IBF _B	STB _B	INTR _A	STB _A	IBF _A	I/O	I/O

Proses hubungan pada mode 1 sebagai input bisa digambarkan sebagai berikut:

- Perangkat akan memberitahukan PPI 8255 dengan STB (strobe)
- Kemudian devais pengirimkan data ke PPI 8255
- Setelah pengiriman selesai, PPI 8255 merespon dengan IBF (*input buffer full*).
Proses pengiriman data dari perangkat ke PPI 8255 selesai
- Kemudian PPI 8255 mengirim sinyal INTR (*interrupt*) ke mikroprosessor
- Kemudian, mikroprosessor akan menanggapi dengan mengirim sinyal RD (*read*)
- Kemudian PPI 8255 akan mengirim data ke mikroprosessor.



Gambar 24. Struktur mode 1 sebagai Input

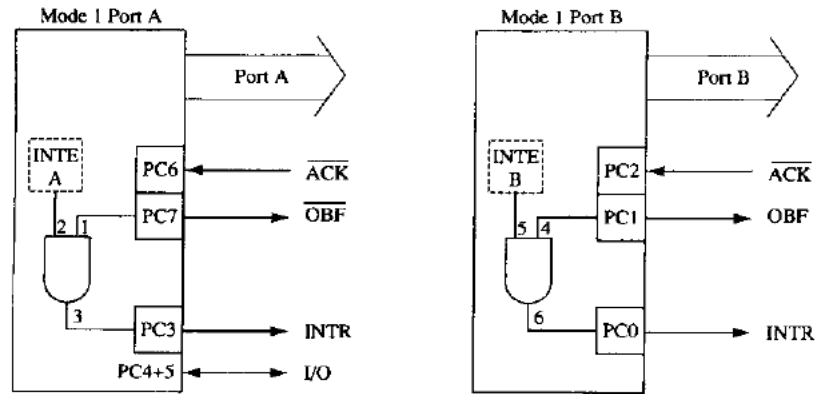
- 2) Jika Port A dan Port B diinisialisasikan sebagai output, maka
- PC0, PC1 dan PC2 digunakan sebagai jalur *handshake* untuk Port B
 - PC3, PC6 dan PC7 digunakan sebagai jalur *handshake* untuk Port A
 - PC4 dan PC5 digunakan sebagai jalur input atau output. Jika D3 pada CW = 1 maka PC4 dan PC5 adalah input

Tabel Konfigurasi jalur Port C pada saat Mode 1 Output

PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7
INTR _B	OBF _B	ACK _B	INTR _A	I/O	I/O	ACK _A	ACK _A

Proses hubungan pada mode 1 sebagai output bisa digambarkan sebagai berikut:

- Mikroprosessor akan mengirim sinyal WR (*write*) ke PPI 8255
- Kemudian mikroprosessor pengirimkan data ke PPI 8255. Proses pengiriman data dari mikroprosessor ke PPI 8255 selesai
- Setelah itu PPI 8255 akan mengirim sinyal OBF (output buffer full) keperangkat
- Kemudian PPI 8255 akan mengirim data ke perangkat
- Setelah menerima semua data, perangkat akan merespon dengan sinyal ACK (*acknowledge*) ke PPI 8255
- Kemudian PPI 8255 akan memberitahukan ke mikroprosessor bahwa data telah sampai ke devais dengan INTR (*interrupt*).



Gambar 25. Struktur mode 1 sebagai Output

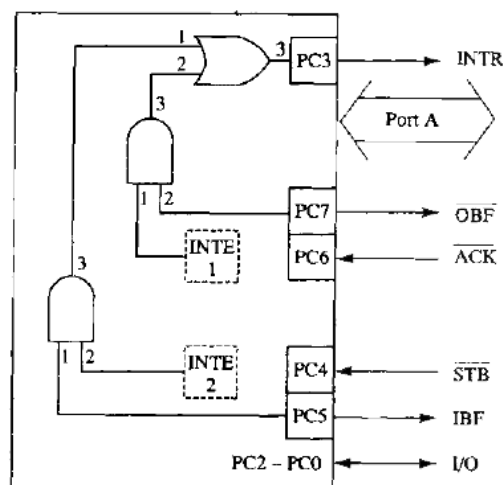
Ingat mode 1 hanya dapat digunakan pada port A dan port B saja. Sedangkan port C hanya dapat digunakan pada mode 0.

c. Mode 2, Bi-directional Bus

Mode 2 hanya dapat diinisialisasikan untuk port A. Pada mode ini port A dapat digunakan untuk *bidirectional handshake* dalam pengiriman data. Artinya output pada 8-bit jalur yang sama, port C digunakan untuk mengontrol handshaking.

Tabel Konfigurasi jalur Port C pada saat Mode 2

PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7
I/O	I/O	I/O	INTR _A	STB _A	IBF _A	ACK _A	ACK _A



Gambar 26. Struktur mode 2

Jika Port A diinisialisasikan sebagai mode 2 maka pin PC3 sampai PC7 digunakan sebagai jalur *handshake* untuk port A. PPI 8255 mempunyai mode dua ini untuk dapat memperlebar sistem bus sampai slave mikroprsesor atau untuk mengirim data ke dan dari *controller floppy disk*.

7. Pemrograman pada PPI 8255

- Pada pemrograman, alamat port PPI 8255 disimpan di register DX karena fungsinya digunakan untuk menyimpan alamat port I/O (ingat materi modul I).
- Data yang akan dikirimkan melalui port tertentu bisa di simpan terlebih dahulu di register AL (bagian low register AX).
- Contoh instruksi-instruksi pada pemrograman PPI 8255:

OUT DX,AL → perintah untuk mengeluarkan data yang ada di AL pada port yang alamatnya tersimpan di DX (perintah ini tidak mengubah isi register DX).

IN AL,DX → perintah untuk membaca data pada port yang alamatnya tersimpan di DX, dan register AL akan terisi sesuai data yang terbaca.

MOV DX,FF13 → FF13 akan dikenal sebagai alamat port karena di simpan di register DX.

- Pada setiap pemrograman yang menggunakan PPI 8255 harus dilakukan inisialisasi *control word*, contohnya:

Nilai *control word* (CW) = 8C_H, maka program inisialisasinya adalah:

```
MOV        DX,FF13
```

```
MOV        AL,8C
```

```
OUT DX,AL
```

Nilai 8C_H dikenal sebagai control word karena dikeluarkan di alamat FF13 (alamat control). Setelah proses inisialisasi inilah kita dapat menggunakan port A, B, dan C sesuai dengan yang kita butuhkan. Pada contoh CW=8C_H hal ini port A mode 0 sebagai output, port C upper sebagai input (ingat port C hanya menggunakan mode 0), port B mode 1 sebagai output, port C low sebagai output. Ingat bahwa saat menggunakan mode 1 ataupun mode 2, maka sebagian jalur pada port C digunakan. Untuk itu pengaturan input ataupun output hanya berlaku pada kaki I/O saja (lihat tabel konfigurasi mode 1 dan mode 2).

MODUL III

ADDRESS DECODER

A. TUJUAN

1. Memahami konsep address decoder
2. Dapat merancang suatu address decoder

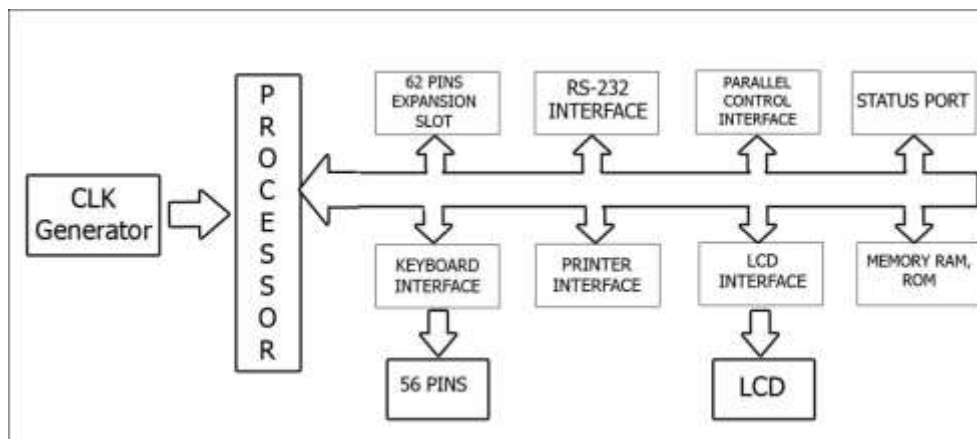
B. PERALATAN

1. BGC 8088
2. Software simulasi

C. DASAR TEORI

1. Pengenalan BGC 8088

BGC-8088 microengineer merupakan generasi kedua dari system mikroprosesor 8088 yang digunakan khusus untuk eksperimen yang menggunakan bahasa assembly (bahasa yang mempresentasikan kode mesin dalam bentuk simbol-simbol agar dapat dipahami oleh manusia).Perangkat ini terdiri dari hardware dan software yang memiliki spesifikasi tersendiri. Sistem ini memiliki seluruh komponen dari computer standar yang meliputi: printer interface, RS-232 interface, parallel port, slot ekspansi, keyboard 56 tombol, dan LCD 40x2. Berikut ini adalah blok diagram BGC 8088.



BGC 8088 menggunakan 2 jenis memori yaitu ROM dan RAM.

ROM merupakan jenis memory yang isinya tidak akan hilang ketika catu daya dimatikan (non volatile memory).

Jenis-jenis dari ROM :

a. PROM

Jenis ROM dimana user bisa burn data didalam IC tersebut. Burn artinya adalah blowing fuse dengan menggunakan alat yang disebut ROM burner atau ROM programmer. PROM hanya dapat ditulis sekali.

b. EPROM

Jenis ROM yang bisa diprogram dan dihapus berkali-kali. Penghapusan EPROM memerlukan waktu 20 menit dengan disinari oleh sinar UV

c. EEPROM

Jenis ROM yang metode penghapusannya menggunakan kelistrikan secara instan. EEPROM hanya bisa menghapus data pada salah satu bagian lokasi memory saja.

d. Flash EPROM

Jenis ROM yang bersifat user programmable, disebut flash karena untuk menghapus seluruh isi memory hanya membutuhkan waktu beberapa detik saja.

Sedangkan, RAM merupakan jenis memory dimana data akan hilang ketika catudaya dimatikan (bersifat volatile).

Jenis-jenis RAM :

a. Static RAM (SRAM)

Jenis RAM yang sel penyimpanan data pada memory RAM dibuat dari flip flop yang tidak perlu direfresh untuk menjaga data tersebut.

b. Dynamic RAM (DRAM)

Jenis RAM untuk keperluan baca dan tulis dengan menggunakan kapasitor untuk menyimpan informasi setiap bit. DRAM membutuhkan refresh untuk menjaga datanya akibat dari kebocoran kapasitor. Keuntungan menggunakan DRAM adalah kapasitas tinggi, biaya lebih rendah per bit, dan daya konsumsi lebih rendah per bit.

Memory digunakan dalam BGC adalah SRAM 6264 yang berkapasitas 8 KB (dapat ditingkatkan hingga 32 MB) dan EPROM 27128 yang kapasitasnya 16 KB (dapat ditingkatkan hingga 32 KB).

Tabel Pengalokasian ROM dapat dilihat pada table berikut:

EPROM Size	Address	Part Number	Usage
F8000-F9FFF or FA000-FBFFF	8K	2764	User Application
F8000-FBFFF	16K	27128A,27128	User Application
FC000-FFFFF	16K	27128	User Application

Untuk membuat suatu rangkaian address decoder dari pengalokasian ROM tersebut maka terlebih dahulu kita harus membuat mapping address decoder.

Berikut **mapping address decoder ROM**:

Jenis	EPROM size	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
EPROM 1	F8000 s/d FBFFF	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
EPROM 2	FC000 s/d FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

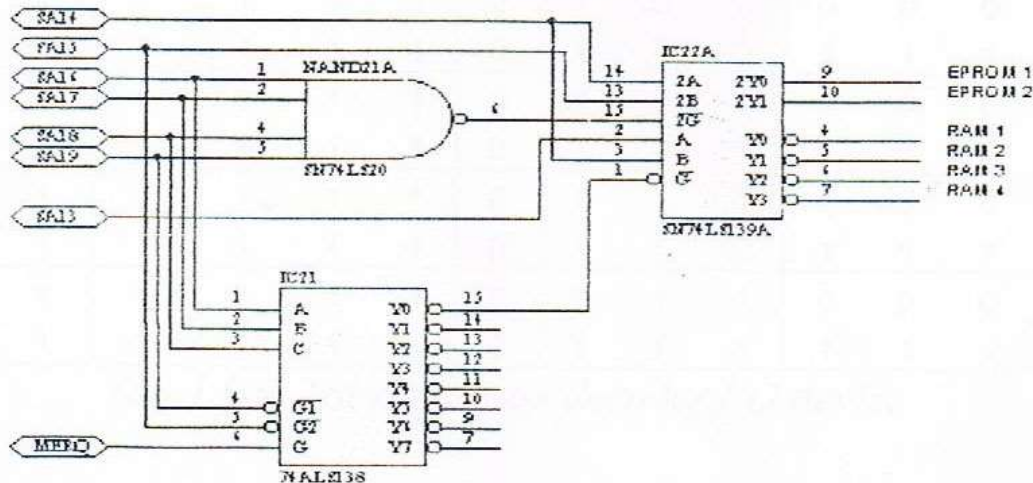
Tabel berikutnya menunjukkan rangkuman pengalokasian RAM pada BGC 8088.

Address	Size	Usage
00000-003FF	1K	256 Interrupt Vektor
00400-00FFF	3K	BIOS Data Area
01000-07FFF	28K	User Application
08000-F7FFF	960K	960K

Perhatikan pada table diatas bahwa alamat 08000H-F7FFFH tidak digunakan dan dapat digunakan pengguna dengan memakai slot ekspansi. Untuk RAM juga digunakan address decoder agar lebih mempermudah dalam mengakses RAM yang masing-masing telah dialokasikan pada alamat sendiri-sendiri.

Jenis	EPROM size	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RAM 1	0000 s/d 1FFF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM 2	2000 s/d 3FFF	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM 3	4000 s/d 5FFF	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM 4	6000 s/d 7FFF	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

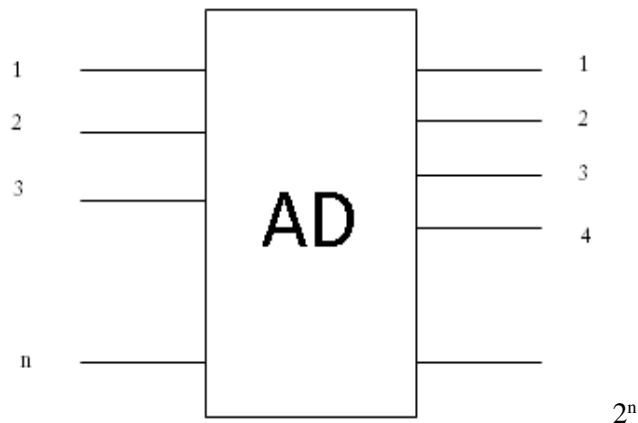
Dari tabel mapping address decoder ROM dan RAM dapat kita buat rangkaiannya address decodernya. Gambar rangkaiannya address decoder untuk RAM dan ROM dapat kita lihat pada gambar berikut:



2. Address decoder

Dalam suatu system mikroprocessor, mikroprocessor bekerja dengan dukungan dari device/peripheral lain seperti memori, I/O equipment, ADC, interrupt device, dan lain-lain. Dalam suatu perancangan system mikroprocessor, kadang dibutuhkan lebih dari satu

device agar system tersebut dapat terselesaikan. Masing-masing device perlu diberikan alamat yang unik agar mikroprosesor tidak salah identifikasi dalam mengakses salah satu device tersebut. Misalkan saja ada proses penambahan kapasitas dari memori yang memanfaatkan lebih dari satu keeping memori. Pada kondisi ini yang menjadi masalah adalah bagaimana mikroprosesor mengetahui keeping memori mana yang akan diakses. Salah satu solusi dari permasalahan perancangan tersebut menggunakan address decoder. Fungsi dari address decoder yaitu untuk mengaktifkan salah satu keeping memori.



Pada kondisi ini yang menjadi masalah adalah bagaimana mikroprosesor mengetahui keeping memori mana yang akan diakses. Salah satu solusi dari permasalahan perancangan tersebut adalah dengan menggunakan address decoder untuk mengaktifkan salah satu keeping memori. Decoder adalah suatu alat yang menerjemahkan kondisi input dengan mengaktifkan salah satu outputnya. Setiap output dari decoder akan dihubungkan ke masukan CS dari salah satu keeping memori. Karena ada satu output yang aktif, maka hanya ada satu keeping memori yang diaktifkan.

Ada 2 hal penting yang harus diperhatikan dalam perancangan address decoder untuk pengaksesan perangkat I/O, yaitu:

- **Jenis addressing**

- a. Fixed addressing**

Pada metode fixed addressing setiap perangkat I/O diberikan satu nomor yang disebut nomor port yang tetap(fixed). Sehingga saat mikroprosesor ingin mengakses perangkat yang diinginkan, mikroprosesor cukup mengakses nomor port dari perangkat tersebut(nomor port yang diakses dapat diketik langsung dalam instruksi).

- b. Variabel addressing**

Pada variabel addressing, nomor port I/O yang akan di akses terlebih dahulu ditulis kedalam sebuah register. Pada mikroprosesor 8088 register yang bertugas untuk menampung alamat I/O adalah register DX.

- **Peta Memori**

- a. **Isolated I/O**

Pada isolated I/O ini antara memory dan I/O dibuat terpisah baik dari segi pemetaannya maupun instruksi pengaksesannya. Peta alamat memori dan peta alamat I/O dipisahkan dan untuk membedakan antara kedua jenis data alamat tersebut di gunakan sinyal IO/M, dimana saat IO/M bernilai logika '1' data yang ada pada bus merupakan data dari perangkat I/O, sedangkan jika IO/M bernilai logika '0' maka data yang ada pada bus saat itu merupakan data dari memori. Itu jika dilihat dari pemetaannya, sedangkan jika dilihat dari cara pengaksesannya, pada metode isolated I/O ini untuk mengakses port I/O digunakan perintah IN atau OUT sedangkan untuk mengakses memori tetap menggunakan perintah MOV.

- b. **Memory-mapped I/O**

Pada metode ini pemetaan alamat I/O digabung dengan memori, dimana ada bagian tertentu dalam memori yang sengaja diperuntukkan untuk menyimpan data dari I/O, sehingga alamat memori tersebut berubah fungsi menjadi alamat I/O. Untuk pengaksesan data baik dari I/O maupun memori tetap menggunakan instruksi MOV.

Dalam perancangan address decoder sangat dipengaruhi sekali dengan pemetaan memori. Kapasitas berapa saja yang akan di-decoder-kan. Dalam perancangannya terdapat 4 jenis, yaitu:

1. **Fixed addressing + isolated I/O.**

Dalam perancangannya, design ini *menggunakan address line 8 bit (A0 s.d A7) dan menggunakan sinyal IO/M.*

2. **Fixed addressing + memory mapped I/O.**

Dalam perancangannya, design ini *menggunakan address line 8 bit (A0 s.d A7) dan tidak menggunakan sinyal IO/M.*

3. **Variable addressing + isolated I/O.**

Dalam perancangannya, design ini *menggunakan address line 16 bit (A0 s.d A15) dan menggunakan sinyal IO/M.*

4. Variable addressing + memory mapped I/O.

Dalam perancangannya, design ini *menggunakan address line 16 bit (A0 s.d A15)* dan *tidak menggunakan sinyal IO/M*.

Adapun langkah-langkah dalam perancangan address decoder menggunakan device memori adalah sebagai berikut:

a. Tentukan kapasitas memori.

Dalam aplikasinya pada mikroprosesor menggunakan sistem bilangan biner walaupun terlihat dalam penulisannya menggunakan sistem bilangan hexadecimal. Penggunaan sistem bilangan hexadecimal bermaksud untuk memudahkan dalam penulisan.

b. Buat memory map.

Tergantung dari jenis yang akan dibuat. Menggunakan empty space atau tidak. Biasanya jika menggunakan empty space maka penjumlahan kapasitas empty space dengan kapasitas memory yang *didecoderkan* akan berjumlah 256 Kbyte

c. Buat tabel pengamatan.

Tentukan terlebih dahulu alamat awalnya. Penjumlahan alamat awal dengan kapasitas satu memory akan menghasilkan alamat akhir untuk memory tersebut.

d. Implementasikan.

Pengimplementasian disini maksudnya adalah menggambarkan secara umum skematik address decoder hasil perancangan

Contoh perancangan address decoder:

Diberikan dua buah RAM dengan kapasitas 512 Byte dan dua buah ROM 256 Byte akan disusun tanpa empty space. Buatlah rangkaian address decodernya dengan susunan memori RAM 1, RAM 2, ROM 1, ROM 2 dan alamat awalnya 0000_H.

Langkah pengerjaan :

a. Menentukan kapasitas masing-masing memori :

- Kapasitas RAM = 512 byte = $2^9 = 1\ 1111\ 1111_b = 1FF_H$
- Kapasitas ROM = 256 byte = $2^8 = 1111\ 1111_b = FF_H$

b. Buat memory map-nya:

Apabila susunan tidak ditentukan, mulailah dengan memori berkapasitas paling besar diletakkan paling dekat dengan alamat 0000_H

ROM 2	05FF 0500
ROM 1	04FF 0400
RAM 2	03FF 0200
RAM 1	01FF 0000

c. Buat tabel pengalamatannya:

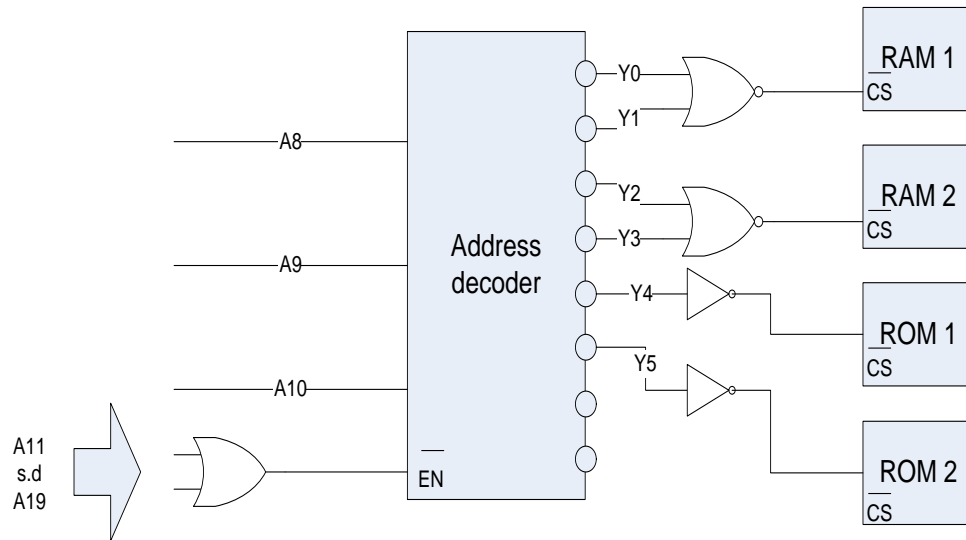
Jenis	EPROM size	Input																
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
RAM 1	0000 s/d	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y ₀
	01FF	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	Y ₁
RAM 2	0200 s/d	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	Y ₂
	03FF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	Y ₃
ROM 1	0400 s/d	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	Y ₄
	04FF	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1	Y ₄
ROM 2	0500 s/d	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	Y ₅
	05FF	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	Y ₅

Dengan demikian dapat ditentukan berapa inputan yang diperlukan. Dilihat motif bilangan yang berbeda. Maka inputan untuk address decoder diatas adalah A10, A9, dan A8. Karena memerlukan 3 input, maka address decoder yang dibutuhkan adalah decoder 3 to 8.

d. Implementasi

Cara 1 :

- Dengan menggunakan address decoder



Output dari address decoder dihubungkan ke chip select masing-masing memori. Pada pin Y₀ dan Y₁ dihubungkan ke gerbang logika NOR lalu outputnya dihubungkan ke chipselect RAM 1, demikian juga pin Y₂ dan Y₃. Untuk pin Y₄ dan Y₅ dapat dihubungkan ke inverter terhubung ke chipselect ROM 1 dan ROM 2.

Untuk A₁₁ sampai dengan A₁₉ tidak dibiarkan menggantung. Agar tidak terjadi pengulangan content memori maka sisa alamat ini semuanya dikode-kan untuk mengaktifkan chip address decoder tersebut.

Sedangkan untuk A₀ s.d A₈ langsung terhubung ke RAM 1 dan RAM 2 yg memiliki 9 pin alamat (hal ini tergantung kapasitas memori). Dan A₀ s.d A₇ langsung terhubung ke ROM 1 dan ROM 2 yang memiliki 8 pin alamat.

Penggunaan gerbang NOR pada output address decoder dimaksudkan agar output yang dihasilkan adalah 0 (nol) karena chipselect (CS) bersifat active low.

Cara 2 :

- Dengan K-map

Misal :

RAM 1 : 00

RAM 2 : 01

ROM 1 : 10

ROM 2 : 11

A10	A9	A8	A	B	\overline{En}
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	x	x	1
1	1	1	x	x	1

A

$\begin{matrix} A9 & A8 \\ A10 \end{matrix}$	00	01	11	10
0	0	0	0	0
1	1	1	x	x

$$A = A10$$

B

$\begin{matrix} A9 & A8 \\ A10 \end{matrix}$	00	01	11	10
0	0	0	1	1
1	0	1	x	x

$$B = A9 + A10 A8$$

\overline{En}

$\begin{matrix} A9 & A8 \\ A10 \end{matrix}$	00	01	11	10
0	0	0	0	0
1	0	$\overline{En} = A9$	1	1

Implementasinya

