HANDOUT MATA KULIAH

PENERAPAN MIKROPROSESOR

(TKF2818 & TKN541I)

Oleh:

Ir. Balza Achmad, M.Sc.E.

Jurusan Teknik Fisika
Fakultas Teknik
Universitas Gadjah Mada

Yogyakarta

PENGANTAR

Dalam rangka meningkatkan kualitas pelaksanakan proses belajar mengajar di perguruan tinggi, khususnya di Jurusan Teknik Fisika, Fakultas Teknik, Universitas Gadjah Mada Yogyakarta, Penulis merasa perlu menulis diktat ini sebagai buku pegangan dalam penyampaian materi pada tatap muka di dalam kelas dan sebagai sarana bantu mahasiswa dalam memahami konsep ilmu pengetahuan yang tercakup dalam mata kuliah Penerapan Mikroprosesor.

Diktat mata kuliah Penerapan Mikroprosesor ini ditulis berdasarkan silabus dalam kurikulum yang ada. Tentunya dengan imbuhan di sana-sini untuk menyesuaikan dengan perkembangan ilmu pengetahuan yang mutahir dan kebutuhan yang ada. Namun sudah barang tentu diktat ini tidak dimaksudkan untuk menggantikan buku teks yang ada, melainkan hanya sebagai bahan pelengkap saja.

Diktat ini membahas aspek-aspek yang berhubungan dengan arsitektur dan prinsip kerja mikroprosesor serta antarmuka dengan peralatan luar. Penjelasan diberikan bagian per bagian secara detil yang meliputi: arsitektur dasar mikroprosesor, sistem koneksi, interupsi, transfer data, antarmuka digital, antarmuka analog, antaramuka melalui port paralel serta contoh-contoh penerapannya.

Diktat ini bukanlah benda mati dan tidak pula berharga mati, dalam artian: setiap saat, dari waktu ke waktu, akan selalu dilakukan revisi dan koreksi. Dengan demikian Penulis mengharapkan adanya saran dan masukan yang tentunya akan membuat semakin sempurnanya diktat ini.

Demi kemajuan bangsaku,
Yogyakarta, Oktober 2003
Penulis,

Ir. Balza Achmad, M.Sc.E.

DEDIKASI

Untuk keluargaku
yang telah memberi dukungan dalam menulis diktat ini:

Kartika Firdausy
Nuansha Thufaila
dan
Mikail Achmad

DAFTAR ISI

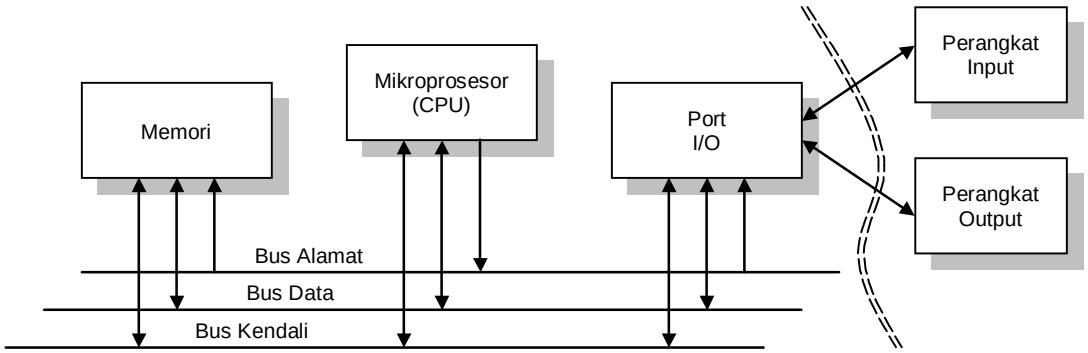
BAB I. ARSITEKTUR DASAR MIKROPROSESOR

A. Sistem Komputer

Apakah beda antara komputer dengan mikroprosesor? Bab ini akan menjelaskannya secara ringkas, dengan fokus pada segi perangkat keras dan pemrograman yang mendukungnya.

1. Arsitektur Komputer

Arsitektur komputer secara umum, sesuai dengan von Neumann, seperti terlihat dalam Gambar I-1. terdiri atas beberapa komponen, yakni: mikroprosesor sebagai pusatnya (CPU = *Central Processing Unit*), memori, port input/output, serta bus.



Gambar I-1. Diagram skematis arsitektur komputer

2. Mikroprosesor/CPU

Mikroprosesor atau CPU adalah “otak” yang merupakan pengendali utama semua operasi dalam sistem komputer. Mikroprosesor mengambil instruksi biner dari memori, menerjemahkannya menjadi serangkaian aksi dan menjalankannya. Aksi tersebut bisa berupa transfer data dari dan ke memori, operasi aritmatika dan logika, atau pembangkitan sinyal kendali.

3. Memori

Memori adalah komponen yang digunakan untuk menyimpan instruksi-instruksi biner yang akan dijalankan oleh mikroprosesor, serta data yang digunakan untuk bekerja. Dalam pandangan Penulis, yang dimaksud sebagai memori dalam diktat ini berupa memori yang dapat langsung diakses oleh mikroprosesor, yaitu RAM (*random access memory*) yang dapat dibaca-tulis dan ROM (*read only memory*) yang hanya dapat dibaca saja.

Sedangkan komponen penyimpan data yang lain, seperti floppy disk, harddisk, CDROM, dll., dikelompokkan sebagai perangkat (*device*) input/output. Setiap lokasi data dalam memori diberi alamat tertentu sehingga dapat secara khusus dituju oleh mikroprosesor. Dalam sistem komputer, memori tidak harus berupa sebuah komponen tunggal, tapi bisa lebih asalkan tidak ada alamat yang saling bertindihan. Satuan memori menentukan ukuran data pada setiap lokasi di memori, pada personal komputer satuan memori biasanya adalah 8 bit (1 byte), sedangkan pada mainframe ada yang bersatuan 12 bit atau 16 bit.

Memori dapat berupa memori statik yang tersusun atas matriks flip-flop yang masing-masing menyimpan bit-bit biner. Bisa juga berupa memori dinamik yang tersusun atas susunan banyak kapasitor yang ada-tidaknya muatan listriknya menandakan isyarat biner. Karena pada kapasitor terjadi peluruhan muatan, maka pada setiap selang waktu tertentu (misalnya setiap 2 milidetik) harus di*refresh* agar kembali ke keadaan semula.

4. Port input/output

Port input/output adalah komponen yang menghubungkan mikroprosesor dengan perangkat luar (harddisk printer, keyboard, monitor, dll.). Jadi port disini berlaku sebagai "pintu" ke perangkat luar. Sebagaimana memori, port I/O juga bukan merupakan komponen tunggal (artinya ada banyak port di dalam sistem komputer) yang masing-masing diberi alamat tertentu. Dengan demikian mikroprosesor tahu, misalnya, ke mana untuk mengirim data ke printer, mengambil data dari mouse dsb.

5. Bus

Bus adalah kumpulan jalur yang menghubungkan ketiga komponen di atas. Bus dapat dianalogikan sebagai jalan umum di muka rumah kita yang dapat kita lewati jika hendak menuju rumah tetangga, kantor, dsb. Bedanya, di jalan umum pada suatu waktu bisa terdapat banyak orang atau kendaraan yang melewatinya; sedangkan untuk bus, pada suatu saat hanya bisa ada satu keadaan (biner) untuk setiap jalurnya. Dengan kata lain, ada banyak komponen yang terhubung ke bus, tapi hanya sebuah komponen yang akan mengisi bus tersebut pada suatu saat. Bus dalam sistem komputer dibagi menjadi 3 kelompok:

Bus alamat (a*ddress bus*), yang digunakan oleh mikroprosesor untuk mengirim informasi alamat memori atau port I/O yang akan dihubungi olehnya. Ukuran bus alamat menentukan berapa kapasitas memori yang ada, misalnya ukuran bus alamat 16 bit (16 jalur alamat) akan mampu mengalamati 2^{16} atau 65536 (64 kb) lokasi memori. Perhatikan arah panah ke dan dari bus alamat pada Gambar I-1.

Bus data (d*ata bus*), yang digunakan untuk lewatnya data dari dan ke masing-masing komponen di atas. Bus data mempunyai ukuran tertentu misalnya 8, 16, atau 32 jalur. Ukuran ini tidak harus sama dengan ukuran data pada setiap lokasi memori. Misalnya apabila berukuran memori adalah 8 bit, maka dengan bus data 32 bit akan dapat memindahkan 4 data (menulis/membaca 4 lokasi memori) sekaligus.

Bus kendali (*control bus*), yang berisi jalur-jalur untuk keperluan pengiriman sinyal kendali antar komponen, misalnya sinyal yang menandakan isyarat untuk membaca, atau menulis, pemilihan memori atau port, interupsi, dll. Isyarat-isyarat ini yang kemudian menentukan aksi apa yang harus dilakukan oleh masing-masing komponen.

6. Eksekusi program

Program adalah urutan instruksi yang akan dijalankan oleh mikrorposesor. Program ini terletak di dalam memori. Mikroprosesor melakukan *fetch and execute* dengan cara mengambil instruksi yang hendak dijalankan dari lokasi memori tersebut (*fetch*), menerjemahkannya, dan kemudian menjalankannya (*execute*). Secara praktis hal di atas terjadi dengan cara berikut: mikroprosesor mengisi bus alamat dengan alamat instruksi berikutnya di dalam memori, lalu memori mengirimkan instruksi yang ada di alamat tersebut melalui bus data. Karena ukuran instruksi tidak mesti hanya 1, bisa juga suatu instruksi terdiri atas 3 byte misalnya[1], maka operasi *fetch* ini diulang sampai instruksi yang diambil dari memori lengkap, setelah itu mikroprosesor menerjemahkan instruksi tersebut ke dalam aksi yang harus dijalankan. Selesai menjalankannya lantas melakukan *fetch and execute* untuk instruksi berikutnya. Demikian dilakukan berulang-ulang, satu instruksi demi satu instruksi.

B. Sejarah Mikroprosesor

Iklan pertama untuk mikroprosesor muncul di Electronic News. Federico Faggin, Ted Hoff, dan timnya di Intel Corporation mendesain mikroprosesor 4004 ketika membuat sebuah IC pesanan untuk Busicom, sebuah perusahaan kalkulator Jepang. Mikroprosesor 4004 mempunyai 2.250 transistor PMOS, menangani data 4 bit, dan dapat mengeksekusi 60 ribu operasi per detik. Mikroprosesor 4004 ini adalah salah satu dari seri IC untuk

---

[1] Contoh instruksi berukuran 1: Tidurlah
-“ - 2: Makanlah roti
-“ - 3: Makanlah roti yang ada di lemari

komponen kalkulator tersebut: 4001: memori ROM 2.048 bit; 4002: memori RAM 320 bit; serta 4003: register geser I/O 10 bit.

Pada tahun 1972, 8008 dengan bus data 8 bit digunakan oleh Don Lancaster untuk membuat cikal-bakal personal komputer. 8008 membutuhkan 20 komponen tambahan untuk dapat bekerja penuh sebagai CPU. Lalu tahun 1974, 8080 menjadi otak personal pertama komputer, Altair, diduga merupakan nama tujuan pesawat Starship Enterprise di film TV Star Trek. 8080 hanya membutuhkan 2 perangkat tambahan untuk bekerja. Selain itu 8080 terbuat dari transistor NMOS yang bekerja lebih cepat. 8080 diseut sebagai mikroprosesor generasi kedua. Segera sesudah itu Motorolla membuat MC6800 yang juga merupakan CPU multiguna. MC6800 sangat populer karena menggunakan catu daya +5V, dibanding 8080 dengan catu daya –5V, +5V, -12V, dan +12V. Mikroprosesor lain yang muncul adalah 6502 sebagai CPU komputer Apple II, dan Zilog Z80 untuk CPU Radio Shack TRS-80.

Tahun 1978, IBM menciptakan personal komputer PC-XT yang sangat populer menggunakan mikroprosesor 8086 dan 8088. Keduanya mampu menangani data 16 bit. Bedanya hanya pada ukuran bus data yang hanya 8 bit untuk 8088 (operasi internal 16 bit), dan 16 bit untuk 8086. Kemudian Intel membut 80186 dan 80188 yang juga berisi perangkat *peripheral* terprogram. Tahun 1982, 80286 adalah prosesor pertama yang dapat menjalankan perangkat lunak yang ditulis untuk pendahulunya, karena instruksi yang dimiliki oleh seri sebelumnya semuanya dimiliki dan ditambahi dengan instruksi lain. Kompatibilitas ke atas ini kemudian menjadi ciri khas mikroprosesor Intel. Dalam 6 tahun, ada 15 juta PC-AT yang menngunakan 80286 sebagai CPU.

Tahun 1985, Intel membuat 80386 (386TM) yang mengandung 275 ribu transistor, dan merupakan mikroprosesor 32 bit yang dapat melakukan *multi tasking* (menjalankan beberapa program dalam waktu yang bersamaan). Tahun 1989, Intel 486TM adalah prosesor pertama yang mempunyai *math coprosesor* secara *built*-in di dalamnya.

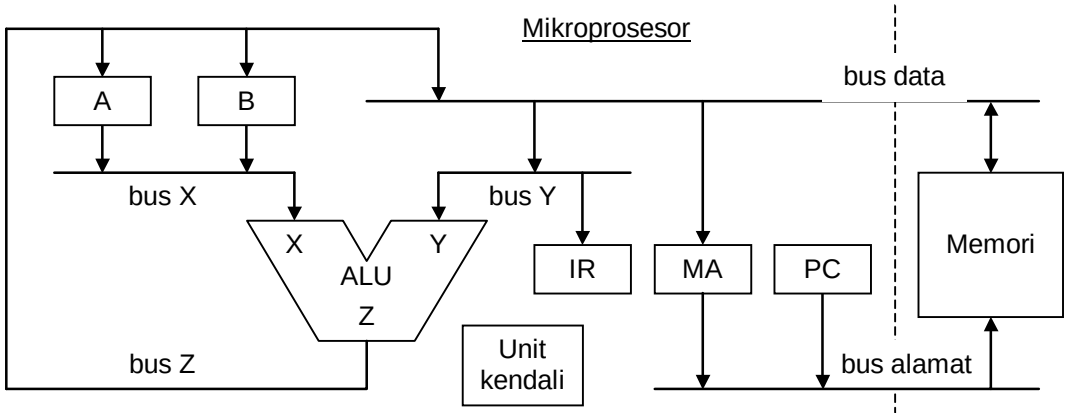
Tahun 1993, lahir keluarga prosesor Pentium®. Tahun 1995, prosesor Pentium® Pro didesain untuk server 32-bit, mengandung 5,5 juta transistor dan mempunyai *chip* memori *cache* kedua di dalamnya. Tahun 1997, dibuat prosesor Pentium® II dengan 7,5 juta transistor dan teknologi MMX, yang didesain khusus untuk memproses data video, audio and grafik secara efisien. Prosesor ini juga diperkenalkan dengan bentuk *cartridge* Single Edge Contact (S.E.C). Seiring dengan itu bermunculan seri Celeron yang merupakan versi Pentium dengan beberapa fitur yang dihilangkan untuk menekan biaya produksi.

Tahun 1999 muncul Pentium !!! dengan 70 instruksi baru yang mendukung Internet Streaming SIMD. Processor ini berisi 9,5 juta transistor, dan mengintroduksi teknologi 0,25-micron. Pada saat ini sedang dikembangkan mikroprosesor 64 bit, sehingga operasi-operasi matematis yang dilakukan dapat lebih cepat.

C. Arsitektur Internal Mikroprosesor

1. Arsitektur umum mikroprosesor

Secara umum, mikroprosesor berisi unit aritmetika/logika (ALU), register, bus internal, serta unit kendali, seperti terlihat pada Gambar I-2. Register dan ALU dihubungkan dengan bus internal dalam mikroprosesor sehingga register dan memori (melalui bus data) dapat mensuplai data ke ALU dan menerima hasilnya. Dalam contoh ini, terdapat 2 buah register, A dan B, yang digunakan untuk secara temporer menyimpan hasil komputasi. Bus internal X dan Y digunakan untuk mentransfer data sebagai *operand* yang akan diolah ALU. Bus internal *Z* digunakan untuk mentransfer hasil operasi ALU ke register atau memori (melalui bus data). Register MA[(2)] (Memory Address) berisi informasi alamat memori yang akan diakses. Unit kendali mengendalikan semua operasi dalam mikroprosesor. Perhatikan kepala panah yang menunjukkan arah aliran data.



Gambar I-2. Arsitektur umum mikroprosesor

Sebagai contoh. misalkan kita hendak menjumlahkan data dari suatu lokasi di memori dengan data dari register A serta menyimpannya di register B. Register MA diisi

[2] Ingat, ini adalah nama umum, setiap mikroprosesor mungkin mempunyai nama register yang berbeda meskipun fungsinya sama

dengan alamat memori yang akan dibaca, lalu register A dihubungkan ke bus X, bus data dihubungkan ke bus Y, dan bus Z dihubungkan dengan register B, kemudian ALU melakukan operasi penjumlahan.

Instruksi yang dijalankan oleh mikroprosesor ada di memori, berupa urutan data-data biner yang merupakan bahasa mesin mikroprosesor. Mikroprosesor mengambil instruksi biner tersebut dari memori yang ditunjuk oleh sebuah register yang bernama Program Counter atau register PC. Mula-mula bus alamat diisi dengan informasi alamat di mana letak instruksi berikutnya yang hendak dijalankan dengan register PC. Lalu mikroprosesor mengambil instruksi tersebut melalui bus data dan menyimpannya di Instruction Register atau register IR. Selanjutnya isi register PC ditambah satu, dengan demikian akan menunjuk ke alamat memori berikutnya di mana instruksi berikutnya akan dijalankan lagi. Secara simbolik kejadian di atas dapat dituliskan sebagai berikut:

$$\text{Mem(PC)} \rightarrow \text{IR}$$
$$\text{PC} + 1 \rightarrow \text{PC}$$

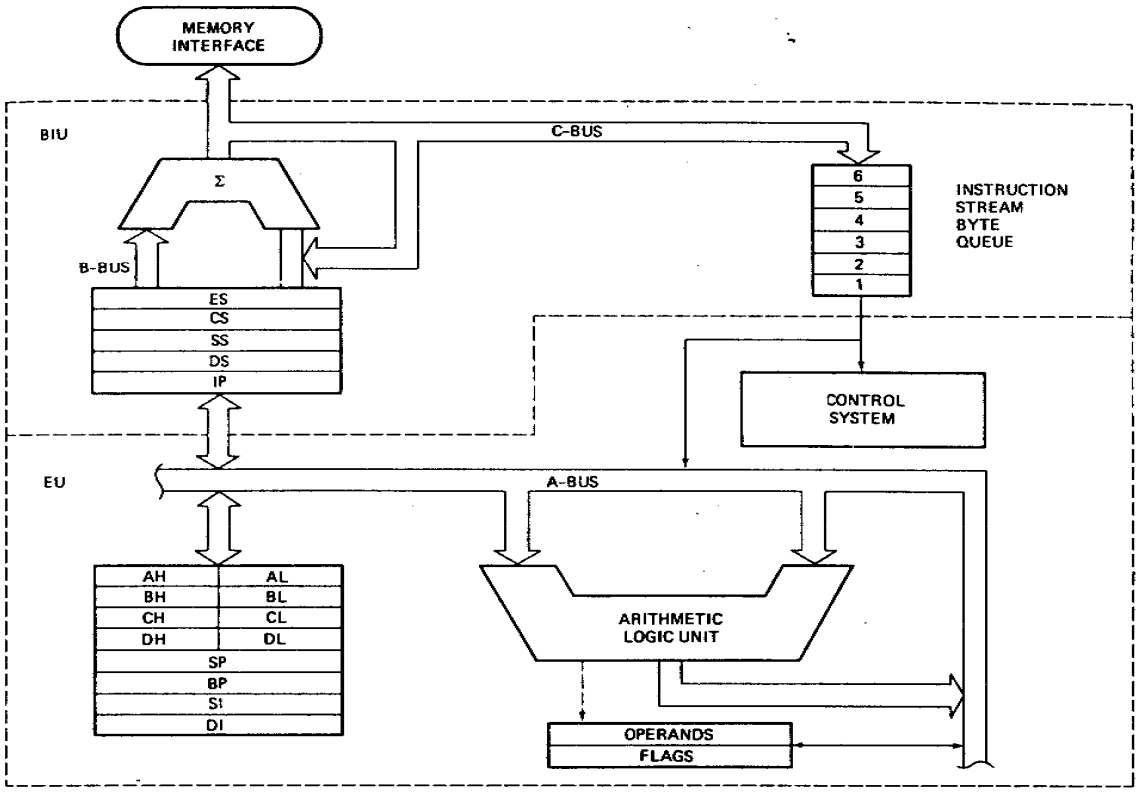
Apabila instruksi yang sudah terambil belum merupakan instruksi yang utuh (setiap instruksi bisa tersusun atas lebih dari 1 byte) maka kejadian di atas diulang lagi.

Setelah register IR berisi instruksi biner, unit kendali lalu menerjemahkannya dan mengeksekusinya. Apa yang dilakukan oleh mikroprosesor tergantung dari instruksi yang diberikan tersebut. Misalnya instruksinya adalah operasi menjumlahan isi register B dengan isi suatu memori dan hasilnya disimpan di dalam register B lagi (alamat memori yang hendak ditambahkan merupakan bagian dari instruksi), maka operasi yang akan dijalankan adalah oleh mikroprosesor adalah:

$$\text{Mem(PC)} \rightarrow \text{MA}$$
$$\text{PC} + 1 \rightarrow \text{PC}$$
$$\text{B} + \text{Mem(MA)} \rightarrow \text{B}$$

2. Intel 8086

Intel 8086 adalah mikroprosesor 16 bit, di mana dia dapat bekerja secara internal menggunakan operasi 16 bit dan secara eksternal dapat mentransfer data 16 bit melalui bus data. Prosesor 8086 dapat dihubungkan dengan bus alamat yang berukuran 20 bit, sehingga mampu mengalamati memori maksimal 2^{20} = 1.048.576 byte (1 MB). Diagram blok arsitektur 8086 dapat dilihat pada Gambar I-3. Mikroprosesor 8086 terbagi atas 2 unit, yaitu unit antarmuka bus (*bus interface unit*, BIU) dan unit pengeksekusi (*execution unit*, EU).



Gambar I-3. Diagram blok internal mikroprosesor 8086

Unit Antarmuka Bus (BIU)

Unit ini merupakan bagian yang berhubungan langsung dengan "pihak luar": bus alamat dan bus data. BIU mengirim alamat ke bus alamat, mengambil instruksi (*fetch*) dari memori, membaca data dari port dan memori, serta menulis data ke port dan memori (menangani transfer data antara bus dan unit eksekusi). BIU tersusun atas:

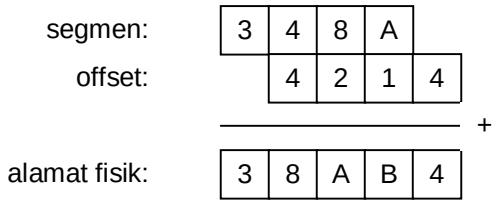
Instruction Stream Byte Queue (ISBQ). BIU mem*fetch* instruksi dari memori sebanyak-banyaknya 6 buah instruksi ke depan. Hal ini dilakukan agar eksekusi progam menjadi lebih cepat. Instruksi yang sudah diambil ini ditaruh di ISBQ yang berupa 6 buah register *first-in-first-out*. BIU dapat melakukan *fetching* selagi EU menerjemahkan dan mengeksekusi instruksi yang tidak membutuhkan penggunaan bus (misalnya operasi matematis menggunakan register internal). Ketika EU selesai melaksanakan suatu instruksi, maka dia tinggal mengambil perintah berikutnya di ISBQ, tanpa harus mengirim alamat ke memori untuk mengambil instruksi berikutnya, sehingga eksekusi akan lebih cepat. Kegiatan *fetching* instruksi berikutnya selagi menjalankan suatu instruksi disebut sebagai: *pipelining*. Pada mikroprosesor yang lebih baru, ukuran ISBQ tidak hanya 6 byte tetapi

mencapai 512 byte, ini efektif untuk program yang mempunyai banyak kalang (struktur program yang berulang).

Register segmen. BIU berisi 4 buah register segmen 16 bit, yaitu: *code segment* (CS), *data segment* (DS), *extra segment* (ES), dan *stack segment* (SS). Sistem komputer 8086 mempunyai bus alamat 20 bit, tetapi ukuran register - termasuk register alamat (*memory address register*) – yang dimilikinya hanya 16 bit, lantas bagaimana cara mengatasinya. Cara pemberian alamat 20 bit dilakukan menggunakan 2 komponen alamat: segmen dan offset, yang masing-masing berukuran 16 bit. BIU akan menggeser ke kiri nilai segmen sebanyak 4 bit (mengalikan dengan 16), kemudian menambahkan offset untuk memperoleh alamat fisik memori yang dikirimkan melalui bus alamat. Untuk lebih jelasnya, diberi contoh untuk memberi alamat fisik $38AB4[3], segmen dapat diisi dengan angka $348A, dan offset diisi dengan angka $4214, lihat Gambar I-4. Cara penulisan kombinasi segmen dan offset adalah:

`segmen:offset`

Sehingga untuk contoh ini, penulisannya adalah $348A:$4214. Perlu diingat bahwa kita bisa menggunakan kombinasi nilai segmen dan offset yang bervariasi untuk memberi alamat fisik yang sama, misalnya $38AB:$0004, $3800:$0AB4, dsb.



Gambar I-4. Contoh cara pengalamatan memori pada mikroprosesor 8086

Secara umum, suatu program terdiri atas 4 bagian: segmen *code* yang berisi instruksi; segmen data, berisi data yang telah dialokasikan sebelumnya (statik); segmen ekstra, untuk variabel dinamik; serta segmen *stack* yang dipakai untuk menyimpan informasi pada saat pemanggilan subrutin. Informasi segmen disimpan dalam keempat register segmen sesuai dengan namanya.

[3] Data dan alamat biasanya dinyatakan dalam bentuk heksadesimal dan diberi simbol $ di depannya, sehingga angka 8 bit ditulis dalam 2 digit/karakter heksadesimal, angka 16 bit ditulis dalam 4 digit heksadesimal, dst.

Instruction Pointer (IP), adalah register berisi informasi offset yang bersama-sama CS menunjuk posisi dalam memori di mana instruksi berikutnya berada.

Unit Eksekusi (EU)

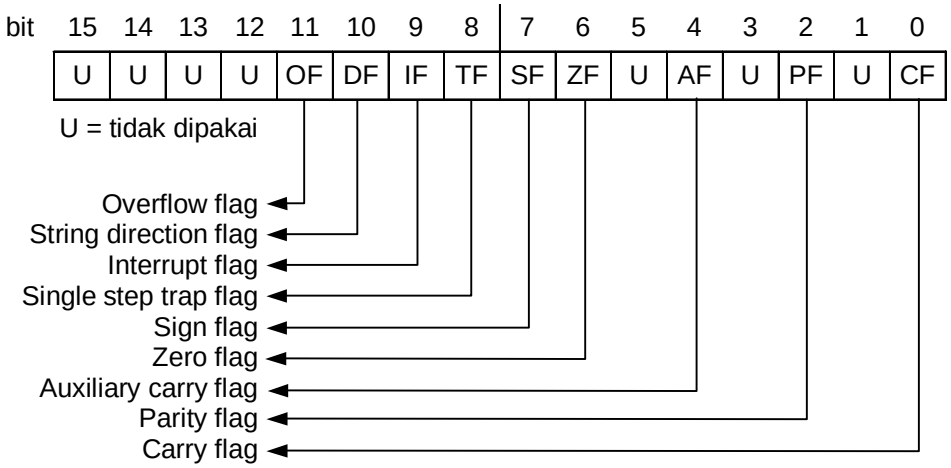
Unit ini memberitahu BIU di mana mengambil instruksi dan data, menerjemahkan kode instruksi, dan menjalankannya. EU tersusun atas:

Dekoder instruksi, yang mengambil urut-urutan instruksi dari ISBQ kemudian menerjemahkannya ke runtutan aksi yang harus dikerjakan oleh EU.

Sistem kontrol, merupakan rangkaian yang mengendalikan kerja mikroprosesor berdasarkan instruksi yang telah diterjemahkan oleh dekoder instruksi tadi.

Arithmetic Logic Unit (ALU), yaitu bagian dari mikroprosesor yang dapat melakukan operasi matematis (misalnya operasi penjumlahan, pengurangan, perkalian, dan pembagian) dan logika (misalnya operasi AND, OR, XOR, geser, dan rotasi) 16 bit.

Register flag (bendera), yaitu register flip-flop 16 bit yang menunjukkan kondisi yang dihasilkan oleh eksekusi suatu operasi oleh EU. Selain itu flag juga mengatur beberapa operasi tertentu. Terdapat 9 flag dalam register flag 8086, seperti terlihat pada Gambar I-5.



Gambar I-5. Format register flag pada mikroprosesor 8086

Sebanyak 6 buah flag merupakan flag kondisi yang menunjukkan keadaan setelah eksekusi suatu instruksi, yaitu: Carry Flag (CF), Parity Flag (PF), Auxiliary Carry Flag (AF), Zero Flag (ZF), Sign Flag (SF), dan Overflow Flag (OF). Sedangkan, 3 buah flag sisanya berupa flag kontrol yang mengendalikan operasi tertentu, yaitu: Single Step Trap Flag (TF), Interrupt Flag (IF), dan String Direction Flag (DF).

CF akan diset (bernilai 1) jika sebuah operasi menghasilkan simpanan (*carry*) melebihi bit terpenting (*most significat* bit, MSB, atau bit 15), dan sebaliknya direset (bernilai 0) apabila tidak ada simpanan. PF diset jika suatu operasi memberikan hasil dengan parity genap, dan direset jika hasilnya berparity ganjil. AF mirip dengan CF, namun diset oleh operasi BCD (*binary coded decimal*). ZF diset jika suatu operasi menghasilkan nol. SF merupakan nilai MSB hasil operasinya, yang menunjukkan tanda; diset jika hasil bertanda negatif dan direset jika hasil bertanda positif. OF diset jika hasil operasi melebihi tempat yang disediakan.

Flag kondisi akan digunakan oleh perintah tertentu untuk menentukan pencabangan atau lompatan. Sedangkan flag kontrol dapat diatur dengan perintah tertentu. Jika TF diset bernilai 1, maka mikroprosesor akan bekerja langkah demi langkah, sehingga dapat digunakan untuk mencek jalannya suatu program. IF digunakan untuk mengatur apakah kerja mikroprosesor dapat diinterupsi atau tidak. Hal yang menyangkut interupsi akan dijelaskan secara lebih detil pada bab berikutnya. DF digunakan untuk menentukan arah operasi string.

Register serbaguna, merupakan register yang dapat digunakan untuk menyimpan data yang akan diolah atau hasil suatu operasi oleh ALU. Terdiri atas 8 buah register 8 bit, yaitu AH, AL, BH, BL, CH, CL, DH, dan DL. Register-register ini juga dapat digunakan secara berpasangan sehingga membentuk register 16 bit, yaitu; AX (gabungan dari AH dan AL), BX, CX, dan DX. AX biasanya digunakan untuk menyimpan hasil operasi, sehingga disebut akumulator. CX biasanya digunakan untuk pencacah untuk keperluan perulangan/kalang (*loop*), sehingga disebut *counter*. BX dan DX biasanya digunakan sebagai offset dari alamat data di memori (dengan segmen DS).

Register pointer dan indeks, terdiri atas Stack Pointer (SP), Base Pointer (BP), Source Index (SI), dan Destination Index (DI). *Stack* (tumpukan) adalah bagian dari memori yang digunakan untuk menyimpan informasi alamat program yang ditinggalkan pada saat terjadi pemanggilan subrutin/subprogram. Demikian juga apabila subrutin tersebut berupa fungsi yang menggunakan parameter, maka data parameter akan disimpan pula di stack. Alamat tumpukan terluar dari stack ditunjuk oleh SS:SP. Sedangkan BP digunakan sebagai offset yang menunjuk ke parameter-parameter fungsi yang dipanggil. SI dan DI biasanya digunakan sebagai offset (masing-masing berpasangan dengan ES dan DS) yang menunjuk ke suatu variabel/data untuk operasi string (larik data).

D. Bahasa Mesin dan Bahasa Assembly

Instruksi yang di*fetch* dari memori untuk kemudian diseksekusi oleh mikroprosesor berformat biner (kombinasi angka 0 dan 1), yang disebut bahasa mesin. Sebagai contoh, perintah untuk memindahkan data dalam register BX ke register CX adalah 10001011 11001011 ($8B CB), sedangkan bahasa mesin untuk menjumlahkan data dalam register AL dengan angka 7 adalah 00000100 00000111 ($04 07), dan perintah membaca dari port 5 diberikan dengan 11100100 00000101 ($E4 05).

Seperti terlihat pada ketiga contoh di atas, bahasa mesin tidak mudah untuk dimengerti dan dihapalkan oleh seorang programer, apalagi jumlah instruksi yang tersedia berkisar ribuan perintah. Di samping itu, akan mudah sekali terjadi kesalahan ketika menuliskan angka-angka biner yang tersusun atas angka 0 dan 1 yang banyak sekali. Oleh karena itu biasanya kita tidak memprogram komputer langsung dalam bahasa mesin, namun dalam bahasa assembly.

Dalam bahasa assembly, setiap instruksi diberi kata (*mnemonic*) yang sesuai dengan maksud perintah itu, sehingga dapat membantu pemrogram dalam mengingat instruksi kepada mikroprosesor tersebut. Kata yang dipakai biasanya berupa singkatan atau beberapa huruf awal dari kata dalam bahasa Inggris untuk perintah tersebut. Misalnya, mnemonic untuk perintah penjumlahan adalah ADD, untuk perintah pengurangan adalah SUB (dari kata *subtract*), dan untuk memindahkan data[4] dari suatu register atau memori ke lokasi lain adalah MOV (dari kata *move*). Sebagian besar instruksi terdiri atas mnemonic dan operand yang merupakan parameter dari instruksi tersebut, yang dituliskan di belakang mnemonic tersebut. Contoh bahasa assembly dari perintah-perintah dengan bahasa mesin di atas diberikan dalam Tabel I-1.

Tabel I-1. Contoh Instruksi dalam bahasa mesin dan bahasa assembly

Instruksi	Bahasa Mesin	Bahasa Assembly
Memindahkan data dari register BX ke register CX	$8B CB	MOV CX, BX
Menjumlahkan data dalam register AL dengan angka 7	$04 07	ADD AL, 7
Membaca port 5	$E4 05	IN AL, 5

[4] Lebih tepat dikatakan operasi pengkopian/punduplikatan data, karena data di lokasi asal (baik memori maupun register) tidak serta-merta menjadi hilang karena telah dipindahkan, namun tetap seperti semula.

Secara umum instruksi mikroprosesor dikelompokkan menjadi (beserta contoh):

1. **Operasi transfer data**, yaitu operasi pemindahan (pengkopian) data antara register, memori, dan port. Di antaranya:

```
MOV AX, BX       ; mengkopi isi BX ke AX
MOV BL, [437AH]  ; mengkopi data byte dari DS:437AH ke BL
IN AL, 34H       ; membaca input byte dari port 34H ke AL
OUT 220H, AX     ; menulis output word ke port 220H
PUSH CX          ; menyimpan register CX ke stack (SS:SP)
```

2. **Operasi aritmatika**, yaitu operasi matematis antara register/memori (dilakukan ALU).

```
ADD DL, BL       ; menambahkan isi BL ke DL
SUB CX, 437AH    ; mengurangi isi CX dengan angka 437AH
MUL CX           ; mengalikan AX dengan CX, hasil disimpan di AX
```

3. **Operasi bit**, yaitu operasi logika antara register/memori (dilakukan ALU).

```
AND BH, CL       ; meng-AND-kan isi BH dengan CL
NOT DX           ; menginvers isi register DX
```

4. **Operasi string**, yaitu operasi yang melibatkan sekumpulan data yang berurutan dalam memori/port.

```
REP MOVSB        ; mengkopi isi byte dari DS:SI ke ES:DI sebanyak CX
```

5. **Operasi kontrol aliran program**, untuk mengatur *loop*, lompatan, dan perulangan.

```
CALL 2323H       ; memanggil subrutin di alamat CS:2323H
RET              ; kembali ke pemanggil subrutin
JZ LBL1          ; lompat ke instruksi yang diberi label LBL1 jika
                   flag ZF bernilai 1 (set)
```

6. **Operasi kontrol prosesor**, yang mengatur kerja mikroprosesor.

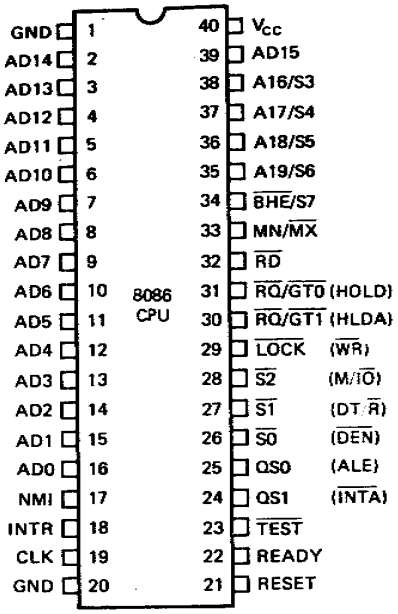
```
STI              ; menset flag interupsi (TF), membolehkan interupsi
CLI              ; mereset TF, menghambat interupsi
```

Setiap instruksi mungkin merubah nilai flag tergantung dari instruksi itu sendiri dan hasil operasinya. Lebih detilnya dapat dilihat di Bab 6 buku *Mikroprosesor and Interfacing, Programming and Hardware*, Douglas V. Hall.

BAB II. SISTEM KONEKSI

A. Diagram Pin Mikroprosesor 8086

Sebuah mikroprosesor dihubungkan dengan komponen lain untuk membentuk sebuah sistem komputer (seperti terlihat pada Gambar I-1 pada bab sebelumnya) melalui bus-bus data, alamat, dan kendali. Untuk dapat memahami bagaimana mikroprosesor dihubungkan dengan komponen lain, maka perlu dijelaskan terlebih dahulu sinyal apa saja yang ada di mikroprosesor tersebut. Diagram pin mikroprosesor 8086 ditunjukkan oleh Gambar II-1 berikut ini.



Gambar II-1. Diagram pin pada mikroprosesor 8086

Mikroprosesor 8086 mempunyai 40 kaki (pin) yang masing-masing digunakan untuk melewatkan sinyal tertentu. Setiap pin sinyal diberi nama berupa mnemonic yang sesuai dengan fungsinya. Sistem komputer 8086 mempunyai bus data selebar 16 bit dan bus alamat selebar 20 bit, sehingga dapat mengalamati memori sampai dengan 2^{20} atau 1 Mb. Untuk menghemat jumlah pin, maka antara pin untuk data dan pin untuk alamat digabungkan dengan diberi nama AD0-AD15 (dari kata *address data*), sedangkan 4 bit alamat sisanya diberi nama A16-A19 (pin-pin ini juga digunakan untuk sinyal status).

Terdapat juga pin-pin untuk catu daya yang disuplaikan, yaitu VCC dan GND, masing-masing untuk tegangan catu daya dan pentanahan. Untuk dapat bekerja, selain

membutuhkan catu daya, mikroprosesor 8086 juga memerlukan sinyal detak (*clock*) secara eksternal dengan frekuensi sampai 10 MHz. Sinyal clock ini dilewatkan ke pin CLK yang ada pada kaki nomor 19.

Pin-pin lainnya digunakan untuk sinyal kendali. Mikroprosesor 8086 dapat digunakan dalam 2 mode, minimum dan maksimum, yang masing-masing menggunakan pin kendali secara berbeda. Mode ini ditentukan dengan memberi nilai pada pin MN/$\overline{\text{MX}}$ [1], nilai 1 (dihubungkan dengan Vcc) untuk mode minimum dan nilai 0 (ditanahkan) untuk mode maksimum. Kebanyakan aplikasi menggunakan mode minimum. Pada mode ini, nama pin yang dipakai pada kaki nomor 24 sampai dengan 31 adalah yang berada di dalam tanda kurung (sebelah kanan).

Sinyal RESET digunakan untuk memerintah mikroprosesor agar melakukan inisialisasi dengan cara memberi nilai 0 pada register DS, SS, ES, IP, dan flag; serta nilai $FFFF untuk CS[2]. Pin INTR dan NMI digunakan untuk menginterupsi kerja mikroprosesor. Jika ada sinyal pada kedua pin itu, maka mikroprossor akan menghentikan eksekusi program yang sedang dijalankannya, kemudian menjalankan subrutin sesuai yang dikehendaki, dan setelah selesai kembali ke tempat semula di mana program diinterupsi. Sinyal INTR (*interrupt*) berupa permintaan untuk melakukan interupsi yang dapat dianulir /tidak dipenuhi jika flag IF direset, sedangkan sinyal NMI (*non maskable interrupt*) tidak dapat ditutup/ditolak, artinya interupsi harus dilakukan. Pin $\overline{\text{INTA}}$ (*interrupt acknowledge*) digunakan oleh mikroprosesor untuk menjawab bahwa permintaan interupsi dari sinyal INTR dapat diterima/dijalankan

Pin M/$\overline{\text{IO}}$ (*memory/IO*), $\overline{\text{RD}}$ (*read*), dan $\overline{\text{WR}}$ (*write*) digunakan untuk mengendalikan memori dan port pada saat pemindahan data. Sinyal M/$\overline{\text{IO}}$ digunakan untuk memilih apakah memori atau port yang akan diakses oleh mikroprosesor. Jika hendak menghubungi memori, maka mikroprosesor memberi nilai tinggi (1) pada sinyal ini dan jika port yang hendak diakses maka sinyal ini diberi nilai rendah (0). Sinyal $\overline{\text{RD}}$ akan diaktifkan (bernilai rendah) jika operasi yang dilakukan adalah membaca, yaitu transfer data dari memori/port ke mikroprosesor. Sementara sinyal $\overline{\text{WR}}$ digunakan untuk menulis, tranfer data dari mikroprosesor ke memori/port, jika aktif. Sinyal-sinyal lain adalah DT/$\overline{\text{R}}$

---

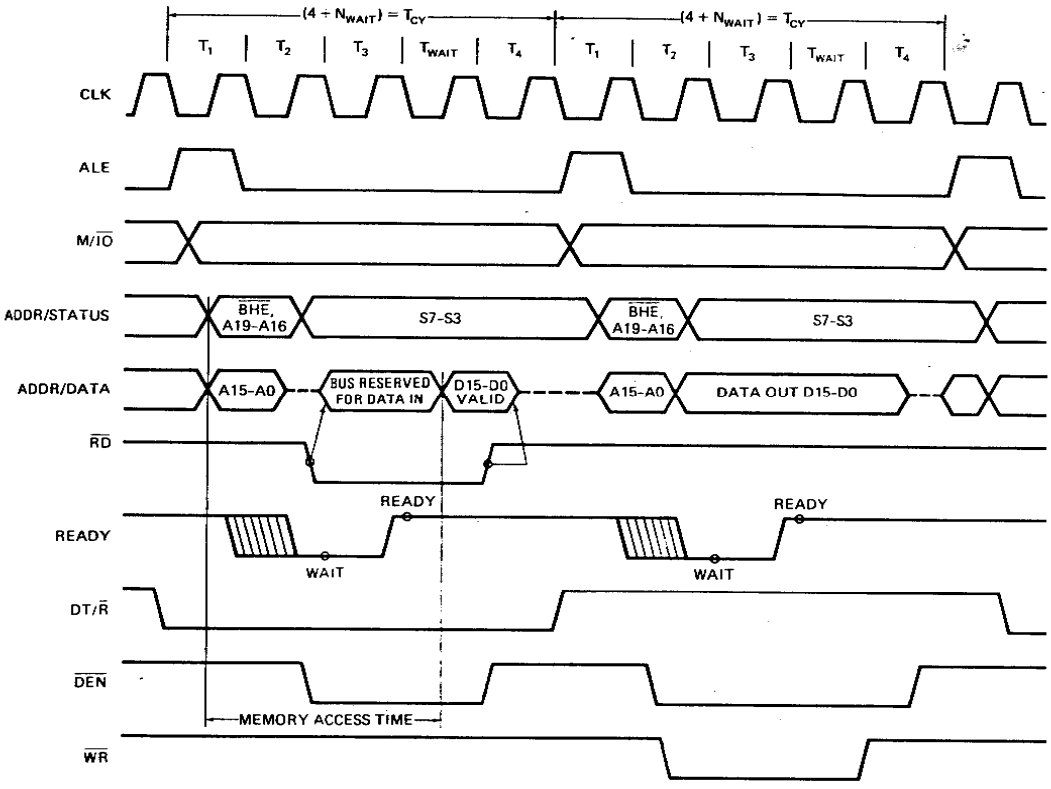
1 Sinyal/pin yang diberi tanda bar (garis di atasnya) merupakan sinyal aktif rendah, di mana sinyal tersebut dikatakan aktif jika bernilai 0. Sebaliknya sinyal/pin tanpa bar adalah aktif tinggi (pada nilai 1).

2 Dengan demikian, instruksi yang pertama kali akan dijalankan mikroprosesor 8086 pada saat *booting* terletak di alamat memori $FFFF0.

(*data transmit/receive*), $\overline{\text{DEN}}$ (*data enable*), ALE (*address latch enable*), dan $\overline{\text{BHE}}$ (*bus high enable*) yang akan dibahas kemudian.

B. Pewaktuan (*Timing*) pada 8086

Sinyal-sinyal dalam bus data, alamat, dan kendali berubah sepanjang waktu sesuai dengan operasi mikroprosesor. Kerja sistem disinkronkan dengan detal (*clock*) yang disuplaikan ke CLK. Setiap sebuah gelombang penuh pada CLK disebut sebagai keadaan atau *state*, misalnya T_1 dan T_2 pada Gambar II-2. Setiap operasi pada bus memerlukan sejumlah state tertentu yang disebut sebagai siklus mesin (*machine cycle*). Pada 8086, setiap siklus mesin terdiri atas empat state (T_1 sampai dengan T_4) dengan kemungkinan ditambah dengan satu atau lebih *wait state* (T_{WAIT}). Dengan demikian setiap siklus mesin membutuhkan detak sebanyak $4+N_{WAIT}$, di mana N_{WAIT} adalah jumlah wait state yang dibutuhkan. Setiap instruksi memerlukan satu atau lebih siklus mesin untuk menyelesaikannya, yang disebut sebagai siklus instruksi. Berikut ini akan diberikan contoh apa yang terjadi dalam bus data, alamat, dan kendali saat dilakukan operasi pembacaan dari memori atau port.



Gambar II-2. Sistem pewaktuan pada 8086

Selama state T_1 dalam operasi baca tersebut, mikroprosesor 8086 memberikan nilai pada $M/\overline{IO}$ (tinggi untuk membaca dari memori atau rendah untuk membaca dari port). Kemudian 8086 mengaktifkan sinyal ALE yang diikuti dengan pengiriman alamat ke pin-pin AD0-AD15 dan A16-A19. Sinyal tersebut digunakan oleh komponen lain untuk memegang nilai alamat ke dalam bus alamat[3]. Sesaat kemudian sinyal ALE direndahkan karena alamat sudah dipegang dalam bus alamat. Sekarang pin AD dapat digunakan untuk menampung data dari memori atau port.

Mikroprosesor sekarang siap untuk membaca data dari memori atau port dengan alamat yang sudah diberikan tadi, sehingga menjelang state T_2 berakhir, sinyal $\overline{RD}$ direndahkan yang akan mengaktifkan komponen memori/port tersebut. Sinyal $\overline{RD}$ tersebut akan memicu memori/port yang diakses untuk memberikan data melalui bus data yang kemudian masuk ke pin AD. Memori/port yang dialamati harus dapat menyediakan data sebelum waktu akses memori habis, supaya mikroprosesor memperoleh data yang valid.

Apabila komponen memori atau port yang diakses tersebut jauh lebih lambat kerjanya, maka 8086 harus diminta menunggu sampai data tersedia, yaitu dengan memberi satu atau lebih *wait state* (T_{WAIT}). Ini dapat dilakukan dengan mengubah sinyal READY. Pada saat sinyal READY bernilai tinggi, maka 8086 akan beroperasi secara normal. Namun jika sinyal READY dibuat rendah (oleh suatu komponen) sebelum tepi naik pada CLK dalam state T_2, maka 8086 akan menyisipkan sebuah *wait state* di antara T_3 dan T_4. Selama *wait state*, sinyal pada pin 8086 akan dipertahankan tetap sama dengan saat memasuki *wait state*. Hal ini dimaksudkan agar memori/port mendapat waktu tambahan untuk menyediakan data. Jika sinyal READY tetap rendah menjelang akhir sebuah *wait state*, maka 8086 akan menyisipkan sebuah T_{WAIT} lagi, sampai READY bernilai tinggi, di mana mikroprosesor kemudian akan masuk ke state T_4. Sinyal READY ini biasanya dibangkitkan oleh pewaktu sistem dan dapat diatur melalui BIOS komputer.

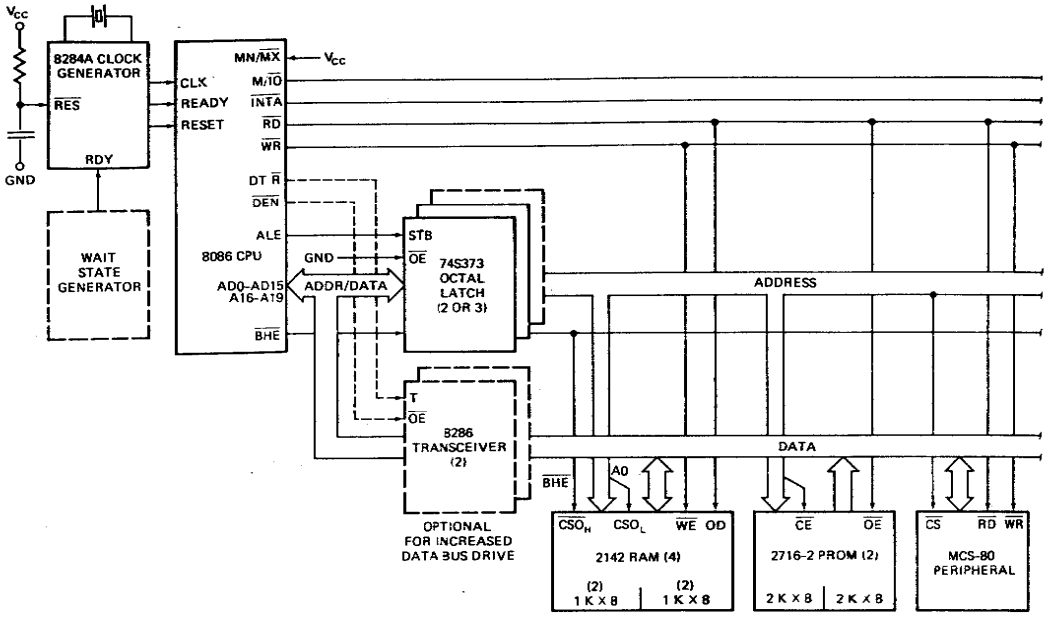
Pin $\overline{DEN}$ digunakan untuk memindahkan data dari bus data ke mikroprosesor atau sebaliknya dari mikroprosesor ke bus data. Sinyal ini dibuat aktif saat sinyal $\overline{RD}$ diberikan, dan dikembalikan ke tinggi ketika data sudah dipindahkan. Pada operasi tulis, nilai sinyal $\overline{DEN}$ aktif selama sinyal $\overline{WR}$ diberikan.

---

[3] Ini perlu dilakukan karena pin AD0-AD15 digunakan secara bergantian untuk lewatnya informasi alamat dan data dari/ke mikroprosesor, sehingga harus diberitahukan kepada komponen luar kapan pin-pin tersebut berisi alamat dan kapan berisi data.

C. Koneksi Mikroprosesor dengan Komponen Lain

Sebagian pin pada mikroprosesor 8086 dihubungkan ke komponen lain secara langsung, misalnya pin M/$\overline{IO}$, $\overline{RD}$, $\overline{WR}$, dan $\overline{INTA}$. Namun sebagian lain dihubungkan melalui komponen *buffer* dan *latch*, seperti terlihat pada Gambar II-3.



Gambar II-3. Koneksi mikroprosesor 8086

Komponen buffer dan latch[4] tersebut dipakai untuk memisahkan informasi dari pin AD dari mikroprosesor ke dalam bus data dan bus alamat sistem komputer. Seperti telah dibahas di depan, mikroprosesor 8086 mempunyai pin yang sama untuk data (D0-D15) dan 16 bit terkecil dari alamat (A0-A15). Informasi alamat dari pin AD harus tersedia sampai akhir state T_1, saat di mana ALE bergerak turun. Sinyal ALE dihubungkan dengan *strobe* (STB) pada latch, sehingga tepi turun ALE akan memicu latch untuk memindahkan informasi alamat dari pin AD ke latch, Nilai tersebut akan dipegang oleh latch sampai ada picuan dari ALE lagi. $\overline{OE}$ (*output enable*) pada latch dihubungkan dengan pentanahan (GND), sehingga latch akan selalu aktif dan mengeluarkan informasi alamat yang ada di latch tadi ke bus alamat.

---

4 Buffer (misalnya 8286) digunakan untuk melewatkan data dari dan ke bus data sistem, berupa *three-state-buffer* (dapat bersifat sebagai masukan, keluaran, atau putus/tidak terhubung). Sedangkan latch dipakai untuk memegang nilai alamat keluaran pin mikroprosesor dan hanya bersifat satu arah, yaitu dari mikroprosesor ke bus alamat sistem.

Informasi alamat dalam bus alamat tersebut dikombinasikan dengan sinyal-sinyal dari bus kendali (yaitu $M/\overline{IO}$ dan $\overline{BHE}$) pada gilirannya akan dideteksi oleh komponen-komponen yang hendak diakses oleh mikroprosesor (memori atau port I/O). Apabila kombinasi alamat dan sinyal kontrol tersebut sesuai dengan alamatnya sendiri, maka komponen tersebut akan memberikan respon sesuai perintah dari mikroprosesor yang ditentukan oleh sinyal dari bus kendali, antara lain $\overline{RD}$ dan $\overline{WR}$.

Pin AD0-AD15 pada mikroprosesor juga dihubungkan ke buffer untuk mengekstrak informasi data. Dari diagram pewaktuan pada saat mengakses memori atau port pada Gambar II-2, data yang ada di pin-pin tersebut valid pada saat $\overline{DEN}$ berubah dari aktif (nilai rendah) menjadi tidak aktif (nilai tinggi), atau dengan kata lain pada saat tepi naiknya, yaitu sekitar permulaan state T_4. Oleh karena itu, pin $\overline{DEN}$ itu dihubungkan dengan $\overline{OE}$ pada buffer data, sehingga nilai data akan dipindahkan dari mikroprosesor ke bus data atau sebaliknya dari bus data ke mikroprosesor, pada saat sinyal $\overline{DEN}$ aktif. Arah transfer data dari dan ke bus data ditentukan oleh sinyal $DT/\overline{R}$. Jika sinyal ini bernilai tinggi maka mikroprosesor akan memberikan data ke bus data (*data transmit*). Sebaliknya, jika sinyal ini bernilai rendah maka mikroprosesor akan menerima data dari bus data (*data receive*). Oleh karena itu, pin $DT/\overline{R}$ dihubungkan dengan pin T (*transmit*) dari buffer yang ada[5].

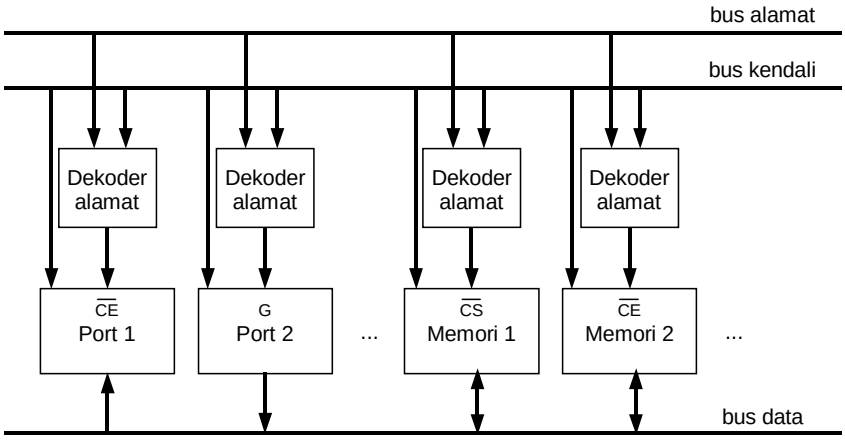
D. Dekoder Alamat

Informasi dalam bus alamat digunakan untuk mengaktifkan memori atau port yang diakses oleh mikroprosesor. Komponen yang mempunyai alamat yang sama dengan nilai bus alamat akan merespon dengan membaca data yang ada di bus data atau menulis data ke dalam bus data. Hal ini dapat dilakukan dengan memasang dekoder alamat pada setiap komponen yang terhubung dengan bus alamat seperti terlihat pada Gambar II-4. Pada gambar itu terlihat jalur-jalur pada bus alamat digunakan sebagai masukan dekoder alamat. Demikian juga sebagian jalur dari bus kendali disambungkan ke dekoder alamat, sedangkan sebagian lagi langsung terhubung ke komponen port atau memori (di antaranya adalah $\overline{RD}$ dan $\overline{WR}$). Keluaran dekoder alamat dihubungkan dengan pin yang digunakan untuk mengaktifkan komponen, misalnya $\overline{CS}$ (*chip select*), $\overline{CE}$ (*chip enable*), atau G (*gate*).

---

5 Ingat, nama sinyal T (*transmit*) yang ada di buffer ini adalah khusus untuk buffer seri 8286 yang dipakai dalam contoh ini. Apabila buffer 3 keadaan yang digunakan berbeda, maka nama pin yang bersesuaian (yang digunakan untuk menentukan arah transfer data dalam buffer) bisa jadi akan berbeda pula.

Dekoder alamat ini pada dasarnya adalah sebuah rangkaian kombinatorial gerbang logika yang memberikan sinyal keluaran aktif[6] saat nilai yang ada dalam bus alamat bersesuaian dengan alamat yang telah ditetapkan untuk komponen tersebut.



Gambar II-4. Dekoder alamat pada port dan memori

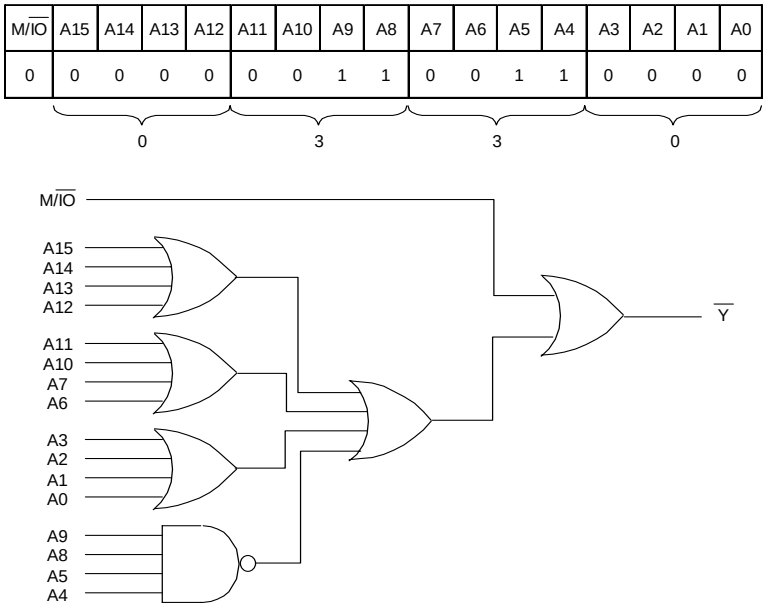
1. Contoh dekoder alamat sebuah port

Sebagai contoh sebuah port diberi alamat $330[7], maka dekoder alamat yang dipakai harus dapat memberi keluaran yang rendah (anggap sinyal yang digunakan untuk menghidupkan port tersebut adalah aktif rendah) apabila nilai yang ada dalam bus alamat adalah $330. Selain itu nilai sinyal $M/\overline{IO}$ adalah rendah juga, karena yang dialamati adalah port I/O, bukan memori. Dengan demikian nilai masukan ke dekoder alamat yang akan memberikan keluaran aktif adalah: A15 = A14 = A13 = A12 = A11 = A10 = A7 = A6 = A3 = A2 = A1 = A0 = $M/\overline{IO}$ = 0 dan A9 = A8 = A5 = A4 = 1. Implementasi ke dalam rangkaian logikanya sangat bervariasi serta menyesuaikan dengan komponen gerbang logika yang tersedia. Salah satu contohnya adalah dengan me-NAND-kan semua jalur alamat yang bernilai tinggi serta meng-OR-kan jalur-jalur alamat yang bernilai rendah, kemudian menggabungkan semua keluaran gerbang-gerbang tersebut dengan gerbang OR yang keluarannya di-OR-kan lagi dengan jalur $M/\overline{IO}$ untuk menghasilkan output bernilai rendah, seperti terlihat pada Gambar II-5 berikut ini.

---

6 Yang dimaksud dengan keluaran aktif adalah keluaran yang bernilai sama dengan nilai aktif dari pin yang digunakan untuk menghidupkan komponen yang dialamati. Kalau pin tersebut aktif rendah (misalnya $\overline{CE}$ atau *chip enable*), maka keluaran dekoder alamat juga dibuat aktif rendah, begitu juga sebaliknya.

7 Nilai dalam bus alamat untuk memori menggunakan 20 bit (A0-A19) sedangkan untuk port hanya dipakai 16 bit (A0-A15), sehingga jumlah port yang bisa dialamati adalah 2^{16} = 65535 buah.



Gambar II-5. Contoh dekoder alamat untuk port $330

2. Contoh dekoder alamat banyak port sekaligus

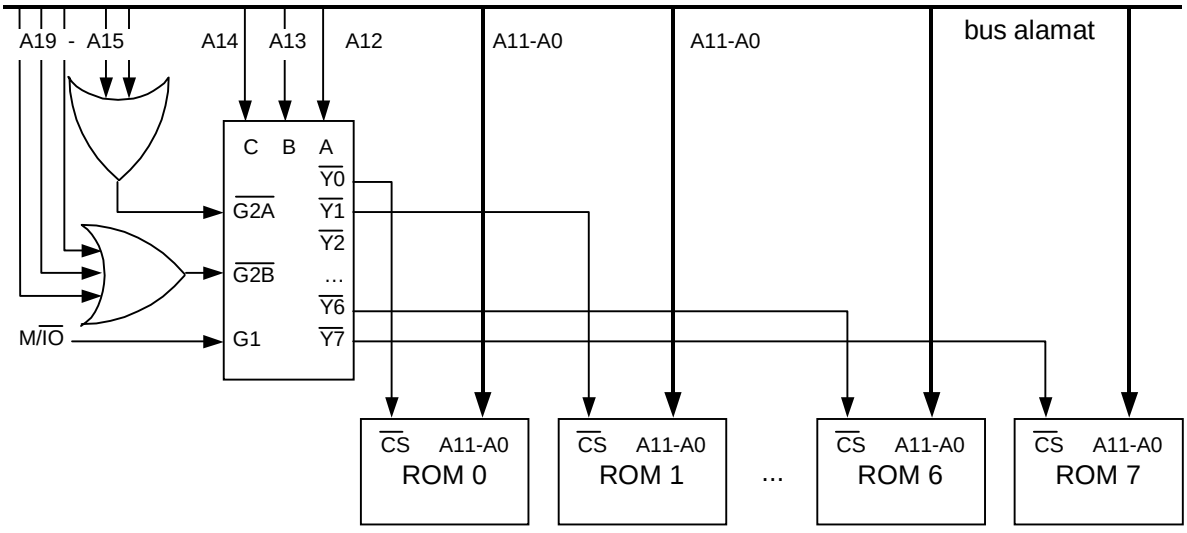
Sebuah dekoder alamat dapat juga digunakan sekaligus oleh beberapa port yang berbeda, asalkan alamat-alamatnya berurutan (bisa merupakan alamat yang berurutan langsung, misalnya: $330, $331, $332, dan $333; maupun yang mempunyai selisih alamat yang sama, misalnya: $330, $334, $338, dan $33C). Untuk keperluan itu, dapat kita manfaatkan sebuah IC dekoder, misalnya IC dekoder 3-ke-8 74LS138 yang dapat digunakan untuk mengalamati 8 port sekaligus.

Sebagai contoh di sini akan dibuat dekoder alamat untuk mengalamati 8 buah EPROM 2732 yang masing-masing berkapasitas 4096 byte. Setiap ROM mempunyai alamat internal sebesar 4096 atau 2^{12}, sehingga harus ada 12 jalur dari bus alamat yang langsung dihubungkan ke masing-masing ROM. Sisanya dihubungkan dengan IC 74LS138. Dari Tabel II-1 terlihat bahwa ROM 0 yang beralamat $00000 sampai dengan $00FFF, akan aktif apabila nilai $M/\overline{IO}$ = 1, dan nilai A19 = A18 = A17 = A16 = A15 = A14 = A13 = A12 = 0. Sedangkan nilai A11 sampai dengan A0 tidak berpengaruh, oleh karena itu jalur-jalur tersebut langsung dihubungkan ke ROM 0. ROM 1 yang beralamat $01000 sampai dengan $01FFF, akan aktif apabila nilai $M/\overline{IO}$ = A12 = 1, dan nilai A19 = A18 = A17 = A16 = A15 = A14 = A13 = 0. Demikian seterusnya sampai ROM yang terakhir.

Tabel II-1. Pengalamatan pada 8 buah ROM berkapasitas 4 kB

ROM	Alamat	M/IO	A19	...	A15	A14	A13	A12	A11	A10	...	A1	A0
0	00000-00FFF	1	0	…	0	0	0	0	X	X	…	X	X
1	01000-01FFF	1	0	…	0	0	0	1	X	X	…	X	X
2	02000-02FFF	1	0	…	0	0	1	0	X	X	…	X	X
3	03000-03FFF	1	0	…	0	0	1	1	X	X	…	X	X
4	04000-04FFF	1	0	…	0	1	0	0	X	X	…	X	X
5	05000-05FFF	1	0	…	0	1	0	1	X	X	…	X	X
6	06000-06FFF	1	0	…	0	1	1	0	X	X	…	X	X
7	07000-07FFF	1	0	…	0	1	1	1	X	X	…	X	X
		1	0	…	0	C	B	A	Langsung ke ROM				

Dari tabel tersebut dapat disimpulkan bahwa nilai M/$\overline{IO}$ selalu 1, sementara A19-A15 selalu bernilai 0. Dengan demikian jalur-jalur tersebut digunakan untuk mengaktifkan IC 74LS138 pada pin G1, $\overline{G2A}$, dan $\overline{G2B}$. Sedangkan sinyal-sinyal yang membedakan antara ketujuh ROM tersebut (yaitu A14, A13, dan A12) digunakan untuk memilih salah satu di antara keluaran 74LS138 yang akan diaktifkan, sehingga masing-masing dihubungkan dengan pin C, B, dan A pada 74LS138. Apabila nilai CBA adalah 000, maka 74LS138 akan memberikan keluaran rendah pada $\overline{Y0}$ dan nilai tinggi pada output yang lain. ROM 0 akan terpilih/aktif. Jika nilai CBA adalah 110, maka 74LS138 akan memberikan keluaran rendah pada $\overline{Y6}$ sehingga mengaktifkan ROM 6. Demikian seterusnya. Implementasinya terlihat pada Gambar II-6.



Gambar II-6. Dekoder alamat untuk 8 buah ROM sekaligus

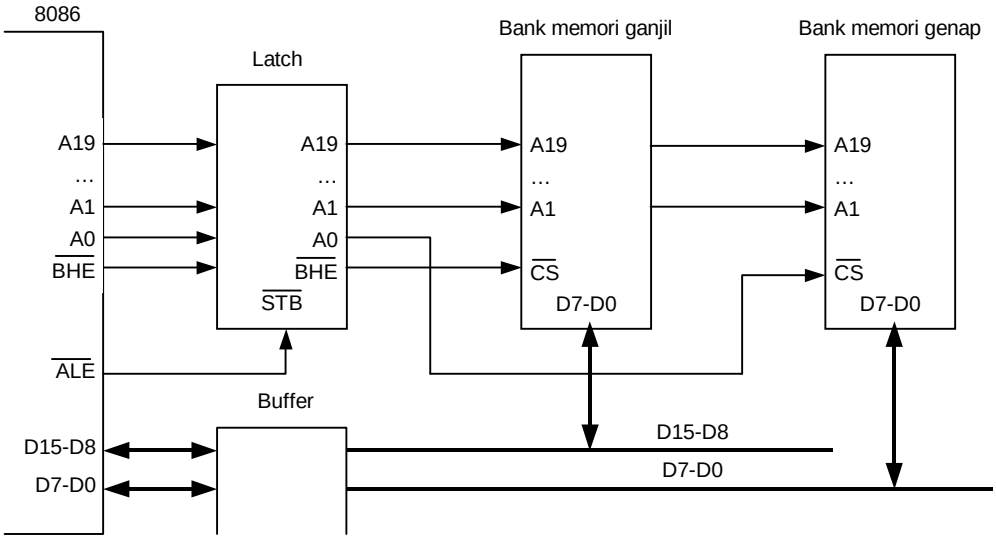
E. Pengalamatan Memori

Lebar bus data pada sistem mikroprosesor 8086 adalah 16 bit (jalur), sehingga dimungkinkan untuk melakukan akses ke memori secara langsung sebanyak 2 byte dengan alamat yang berurutan. Sebagai contoh, perintah MOV DS:[100],BX akan memindahkan isi register BX ke memori dengan alamat $FA100 dan $FA101, jika diasumsikan DS = $FA00. Jika kedua alamat tersebut terletak pada sebuah IC memori, maka pemindahan data dari register ke memori tidak dapat dilakukan dalam 1 siklus mesin, karena tidak mungkin mengaktifkan 2 alamat secara bersamaan dalam sebuah komponen. Oleh karena itu, kedua alamat tersebut harus dipisahkan. Secara praktis hal ini dilakukan dengan menggunakan 2 buah *memory bank* yang memisahkan antara memori dengan alamat genap (00000, 00002, 00004, dst) dengan memori yang beralamat ganjil (00001, 00003, 00005, dst). Bank memori yang beralamat genap dihubungkan dengan byte (8 bit) bus data yang rendah, D0 sampai D7, sedangkan bank memori yang beralamat ganjil dihubungkan dengan byte (8 bit) bus data yang tinggi, D8 sampai D15.

Jalur A0 pada bus alamat digunakan untuk mengaktifkan komponen memori yang berada pada bank memori genap. Jika A0 bernilai rendah maka komponen tersebut akan diaktifkan, dengan demikian jalur A0 langsung dihubungkan dengan $\overline{CS}$ pada komponen tersebut (Gambar II-7). Sedangkan $\overline{CS}$ pada komponen memori yang berada pada bank memori ganjil dihubungkan dengan sinyal $\overline{BHE}$ (*bus high enable*). Sinyal ini diberikan bersamaan dengan sinyal alamat oleh mikroprosesor. Sinyal $\overline{BHE}$ bernilai rendah jika sebuah data berukuran 1 byte (8 bit) diakses pada alamat memori yang ganjil, atau jika sebuah data berukuran 1 word (2 byte, atau 16 bit) diakses pada alamat genap.

Jika sebuah data byte dibaca dari memori beralamat genap, misalnya dengan instruksi MOV AH, [100H], jalur A0 akan bernilai rendah sementara jalur $\overline{BHE}$ bernilai tinggi. Bank genap menjadi aktif, sedangkan bank ganjil tidak aktif. Sebuah data byte lantas akan ditransfer dari komponen yang aktif ke jalur D7-D0 pada bus data. Kemudian 8086 akan memasukkan data dari byte rendah jalur data tersebut ke AH.

Kemudian jika kita hendak memindahkan sebuah data byte dari memori yang beralamat ganjil, misalnya dengan instruksi MOV AL, [101H], maka sinyal A0 menjadi tinggi dan $\overline{BHE}$ menjadi rendah, sehingga bank memori ganjil akan aktif, sementara bank memori genap tidak aktif. Data dari memori pada bank ganjil akan ditransfer ke byte tinggi pada bus data (D15-D8) dan kemudian akan dipindahkan ke register AL.



Gambar II-7. Pengalamatan memori menggunakan bank memori

Jika sebuah perintah lain, misalnya MOV AX, [100H], digunakan untuk memindahkan sebuah data word dari memori yang beralamat genap, maka kedua sinyal A0 dan $\overline{BHE}$ akan bernilai rendah, sehingga kedua bank memori akan aktif. Data dari memori pada bank genap akan ditransfer ke byte rendah pada bus data (D7-D0), sementara data dari memori pada bank ganjil akan dipindahkan ke byte tinggi pada bus data (D15-D8), dan 8086 akan memindahkan keduanya ke register AX. Pada kasus ini, sebenarnya ada 2 alamat yang diakses secara bersamaan oleh 8086 (yaitu 100H dan 101H), namun perbedaannya hanya pada nilai A0.

Kasus terakhir adalah apabila kita mentransfer sebuah data word dari memori beralamat ganjil, misalnya dengan instruksi MOV AX, [101H]. Pada kasus ini, 8086 tidak dapat memberikan 2 alamat yang berurutan tadi (101H dan 102H) secara bersamaan karena kedua alamat tersebut berbeda pada 2 jalur (A1 dan A0), sehingga dibutuhkan 2 siklus mesin untuk melakukannya. Pada siklus pertama, 8086 memberi alamat 101H (ganjil), $\overline{BHE}$ bernilai rendah dan A0 tinggi. Data pertama lalu akan ditransfer ke jalur D15-D8 dan masuk ke AL. Kemudian pada siklus mesin kedua, CPU memberi alamat 102H (genap), A0 akan bernilai rendah, namun $\overline{BHE}$ tinggi. Data kedua kemudian akan dipindahkan ke jalur D7-D0 dan dimasukkan ke AH. Untuk menghindari pemakaian 2 siklus mesin untuk operasi pengaksesan memori, maka pada saat memprogram, variabel-variabel yang berukuran word harus dialokasikan pada memori yang beralamat genap.

F. Standar Pengalamatan Port

Pengalamatan port dapat dilakukan secara bebas namun tidak sembarangan. Yang penting adalah dipastikan bahwa tidak ada 2 port atau lebih yang mempunyai alamat yang sama. Sistem komputer personal (IBM *compatible*) telah mempunyai standar pengalamatan untuk port-port yang umum dipakai, seperti ditunjukkan pada Tabel II-2. Untuk alat yang dirancang sendiri, alamat port harus dipilihkan di luar alamat standar tersebut.

Tabel II-2. Standar pengalamatan port pada komputer personal

Alamat port	Komponen
0000 - 000F	DMA (*Direct memory access*)
0020 - 0021	Pengontrol interupsi (*Programmable interrupt controller*) #1
0040 - 0041	Pewaktu sistem
0060, 0064	Keyboard
0061	Speaker
0070 - 0073	CMOS/real time clock
0080 - 0090	DMA
00A0 - 00A1	Pengontrol interupsi #2
00F0 - 00FF	Numeric coprocessor
0170 - 01F7	Hard disk controller
0200 - 02FF	Joystick
0200 - 020F	Soundcard
02F8 - 02FF	Port serial COM2
0330 - 0331	Soundcard MIDI
0378 - 037F	Port paralel printer
03F2 - 03F5	Floppy disk controller
03F8 - 03FF	Port serial COM3

BAB III. INTERUPSI

A. Polling & Interupsi

Tinjau ilustrasi berikut, yaitu sebuah sistem mikroprosesor yang dipasang pada sebuah sistem kendali suatu proses. Proses tersebut mempunyai beberapa parameter atau variabel yang diukur menggunakan sensor dan tranduser. Sinyal analog dari tranduser kemudian dikonversi ke bentuk digital menggunakan ADC (*analog to digital converter*) sehingga dapat dibaca oleh komputer. Proses konversi menggunakan ADC biasanya membutuhkan waktu (waktu konversi) yang cukup signifikan dibandingkan dengan kecepatan mikroprosesor dalam menjalankan suatu instruksi [1]. Dalam masa konversi, belum tersedia data hasil konversi. Oleh karena itu mikroprosesor harus menunggu sampai konversi selesai. Masalahnya adalah apabila komputer dibiarkan menganggur (*idle*) saat menunggu konversi selesai maka akan banyak waktu yang terbuang sementara sebetulnya kecepatan mikroprosesor yang cukup tinggi masih bisa dimanfaatkan untuk mengerjakan hal yang lain sembari menunggu [2].

Ketika data hasil konversi tersedia, maka ada 2 kemungkinan yang berhubungan dengan transfer data, yakni ADC secara pasif menunggu sampai mikroprosesor meminta data darinya atau ADC itu sendiri yang secara aktif memberitahu mikroprosesor bahwa data konversi telah tersedia. Yang pertama menggunakan cara polling dan yang kedua menggunakan cara interupsi.

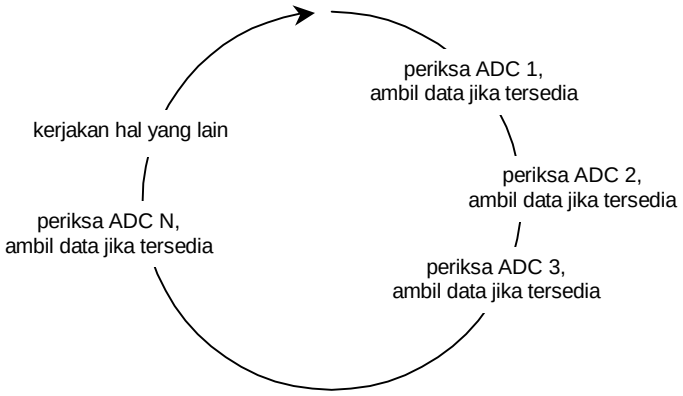
1. Polling

Dengan cara ini, komputer diprogram untuk secara berkala dan bergiliran memeriksa apakah konversi telah selesai dan data di ADC telah tersedia. Biasanya dilakukan dengan cara memeriksa salah satu sinyal atau pin yang ada di ADC yang menandakan bahwa data hasil konversi telah siap. Apabila sinyal tersebut sudah aktif maka data diambil oleh mikroprosesor, sedangkan bila belum aktif maka akan dibiarkan saja oleh mikroprosesor. Kemudian komputer memeriksa ADC berikutnya dan melakukan hal yang sama. Demikian seterusnya seperti diilustrasikan dalam Gambar III-1. Keuntungan cara polling adalah program yang diperlukan dan prinsip kerjanya relatif lebih sederhana, tidak memerlukan koneksi khusus dengan ADC yang dipakai. Sedangkan kekurangannya adalah

---

1 Kecuali ADC jenis flash yang dapat memberikan hasil konversi secara instan.
2 Terutama jika komputer dipakai dalam lingkungan sistem operasi yang mendukung multitasking

kalau konversinya lambat maka seringkali komputer mendapati data belum tersedia ketika memeriksa sebuah ADC, sehingga dapat dikatakan banyak pekerjaan yang sia-sia.



Gambar III-1. Pengambilan data secara polling

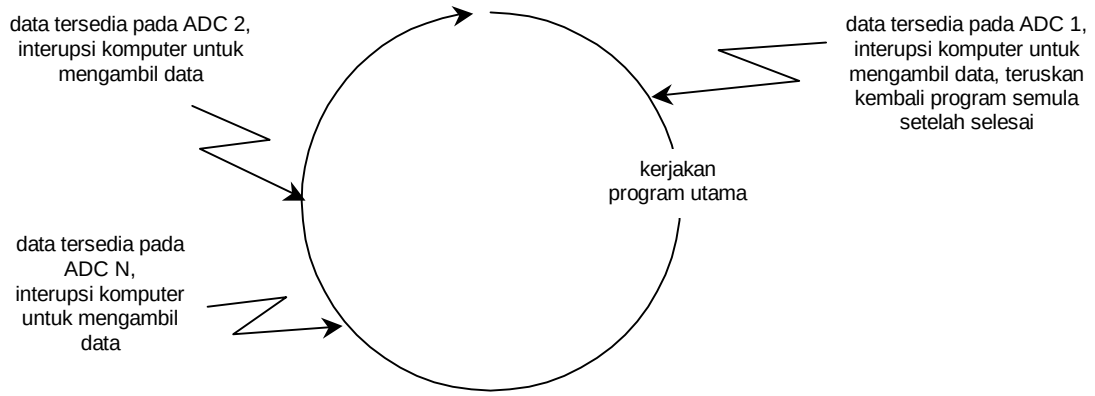
2. Interupsi

Pada cara ini, kita memanfaatkan pin INTR atau NMI yang telah disinggung di bab sebelumnya yang digunakan untuk menginterupsi kerja mikroprosesor. Komputer dibiarkan melakukan pekerjaan yang telah diprogramkan tanpa harus memeriksa setiap ADC apakah data telah siap atau belum. Ketika data telah tersedia, sinyal yang menandakannya pada ADC dimanfaatkan untuk mengaktifkan sinyal INTR atau NMI sehingga komputer menghentikan apa yang sedang dikerjakannya pada saat itu. Kemudian komputer mengambil data hasil konversi dan kemudian meneruskan kembali pekerjaannya yang tadi diinterupsi. Keuntungan cara ini adalah komputer dapat lebih efisien dalam memanfaatkan waktu serta pada program utamanya tidak perlu merisaukan untuk memeriksa piranti luar setiap saat. Sedangkan kekurangannya adalah diperlukan koneksi secara khusus dengan piranti yang dipakai serta pemrograman yang lebih kompleks untuk menangani interupsi [(3)].

Gambar III-2 mengilustrasikan prinsip kerja pengambilan data dengan cara interupsi. Perlu diingat bahwa interupsi oleh suatu piranti dapat muncul kapan saja, waktunya tidak dapat ditentukan sebelumnya. Demikian juga urutan interupsi tidak mesti urut dari suatu piranti ke piranti berikutnya. Pada kenyataanya kebanyakan piranti bekerja menggunakan sistem interupsi untuk memberitahukan ketersediaan data untuk ditransfer ke mikroprosesor, misalnya: keyboard, mouse, printer, kartu suara, modem, dan lain-lain.

---

[3] Bagian program yang akan dilaksanakan oleh komputer pada saat diinterupsi dinamakan rutin pelayanan interupsi



Gambar III-2. Pengambilan data dengan cara interupsi

B. Tipe Interupsi

Pada mikroprosesor 8086, interupsi dapat berasal dari 3 kemungkinan:

1. dari sinyal yang diberikan oleh perangkat luar melalui pin NMI atau INTR, yang disebut interupsi secara perangkat keras (*hardware interrupt*)
2. berasal dari eksekusi instruksi interupsi (INT), disebut interupsi secara perangkat lunak (*software interrupt*)
3. dari suatu kondisi yang ditimbulkan karena suatu eksekusi, misalnya ketika prosesor diminta membagi suatu bilangan dengan nol, maka akan muncul interupsi, hal ini disebut interupsi kondisional

Terdapat sebanyak 256 buah interupsi yang disediakan oleh 8086 yang diberi nomor atau tipe 0 sampai dengan 255. Beberapa tipe interupsi sudah dipakai oleh sistem, antara lain: nomor 0 untuk interupsi kesalahan pembagian dengan nol, tipe 1 untuk *single step*, tipe 2 untuk NMI, nomor 4 untuk interupsi *pointer overflow*. Sisanya dapat dipakai untuk keperluan yang lain.

Setiap nomor interupsi mempunyai rutin pelayanan interupsi sendiri. Alamat-alamat dari rutin tersebut ditaruh dalam tabel pointer interupsi yang berlokasi di memori 0000H-03FFH, dalam bentuk pasangan segmen-offset. Pada saat terjadi interupsi, pasangan segmen-offset ini akan dimasukkan ke CS dan IP sehingga mikroprosesor dapat menjalankan rutin tersebut. Untuk interupsi nomor N, segmen rutin pelayanan interupsinya ditaruh di memori dengan alamat 4N dan 4N+1, sedangkan offsetnya di alamat 4N+2 dan 4N+3 [(4)]. Jadi interupsi nomor 8, alamat rutinnya disimpan di memori 020H - 023H. Jika

[4] Segmen dan offset berukuran 2 byte sehingga menempati 2 alamat pada memori untuk menyimpannya.

kita membuat sistem mikroprosesor yang mempergunakan interupsi, maka selain koneksi hardware, kita juga harus membuat program untuk rutin pelayanan interupsi. Kemudian menyimpan alamat rutin tersebut di tabel alamat interupsi yang sesuai dengan nomor interupsi yang dipakai.

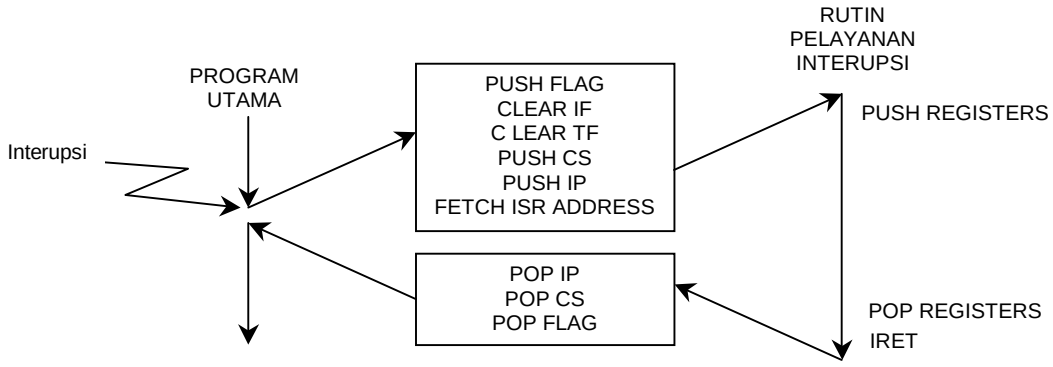
C. Respon Interupsi

Prinsip kerja interupsi cukup rumit karena ketika ada interupsi, sebelum mikroprosesor mengerjakan rutin pelayanan interupsi, mikroprosesor harus mencatat dulu alamat dari instruksi yang sedang dikerjakannya pada saat itu, sehingga dia bisa kembali ke tempat yang tepat ketika selesai mengerjakan ruting pelayanan interupsi dan meneruskan pekerjaanya yang tertunda.

Pada setiap selesai melaksanakan suatu instruksi, 8086 akan memeriksa apakah ada permintaan interupsi. Jika ada dan flag interupsi pada mikroprosesor (IF) mempunyai nilai 1 atau set, artinya mikroprosesor mengijinkan adanya interupsi, maka dia akan mengerjakan hal-hal berikut (Gambar III-3):

1. menyimpan register flag ke dalam stack
2. mereset flag interupsi (IF) untuk mencegah interupsi berikutnya
3. mereset flag TF
4. menyimpan register CS dan IP ke dalam stack untuk mencatat posisi program yang sedang dijalankan sekarang
5. mengambil alamat rutin pelayanan interupsi dan kerjakan rutin tersebut sampai ditemui instruksi IRET (*interupt return*), yang menandai akhir dari rutin tersebut untuk kembali ke program semula
6. mengambil kembali posisi program semula (IP dan CS) dari stack
7. mengambil kembali nilai flag semula
8. jalankan instruksi berikutnya dari posisi sebagaimana sebelum adanya interupsi

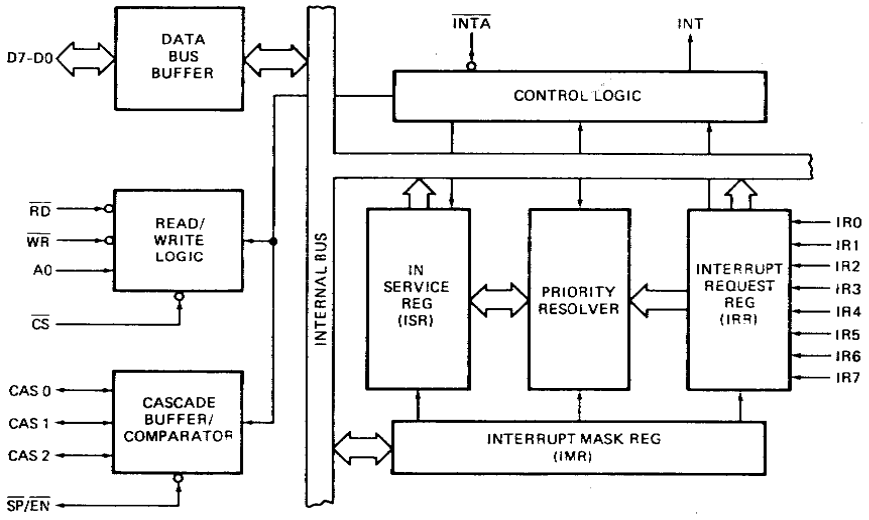
Khusus pada interupsi hardware, setelah menerima sinyal INTR, mikroprosesor menjawabnya dengan mengirimkan sinyal melalui $\overline{\text{INTA}}$ untuk memberitahu piranti yang mnginterupsinya tadi untuk bersiap karena permintaan interupsinya diterima. Lalu mikroprosesor mengirimkan lagi pulsa $\overline{\text{INTA}}$ yang direspon piranti luar dengan mengirimkan nomor interupsi melalui bus data D7-D0. Kemudian mikroprosesor melanjutkan dengan kedelapan langkah di atas.



Gambar III-3. Urut-urutan respon mikroprosesor terhadap suatu interupsi

D. Priority Interrupt Controller (PIC 8259A)

8086 hanya mempunyai sebuah pin INTR (di samping NMI). Namun seringkali terdapat lebih dari satu piranti yang perlu untuk menggunakan interupsi. Oleh karena itu diperlukan komponen bantu untuk meningkatkan jumlah sinyal interupsi yang dapat dipakai oleh bermacam piranti. Dalam hal ini, fungsi tersebut dilakukan oleh IC 8259A Priority Interrupt Controller (PIC) yang mempunyai arsitektur dalam seperti terlihat pada Gambar III-4.



Gambar III-4. Diagram blok internal PIC 8259

Pin D7-D0 terhubung ke bus data, A0 dan CS dihubungkan ke bus alamat dan dekoder alamat, sementara $\overline{RD}$, $\overline{WR}$, INT, dan $\overline{INTA}$ dihubungkan ke mikroprosesor pada sinyal yang bersesuaian. Pin CAS2-CAS0 dan $\overline{SP}/\overline{EN}$ digunakan untuk keperluan konstruksi 8259A bertingkat (*master-slave*) sehingga dapat menambah lagi jumlah

interupsi yang tersedia. Dengan sebuah PIC, kita akan mempunyai 8 interupsi yang dilewatkan pin IR0-IR7. Sebelum kita memakai PIC, terlebih dahulu kita harus memprogramnya dengan mengirimkan sebuah kata kendali (*control word*) inisialisasi dan operasional yang akan mengatur kerja IC tersebut, misalnya nomor-nomor interupsi berapa saja yang dipakai, bagaimana urutan prioritasnya, apakah sinyal interupsi berupa *level*-trigger atau *edge*-trigger, serta apakah diperbolehkan terjadinya interupsi secara bertingkat.

E. Standar Sinyal Interupsi

Pemilihan nomor sinyal interupsi dilakukan dengan mempertimbangkan nomor IRQ yang sudah dipakai oleh komputer seperti terlihat pada Tabel III-1. Sebaiknya tidak mempergunakan sinyal interupsi yang sama untuk lebih dari satu piranti untuk menghindari konflik (kecuali menggunakan teknik tersendiri untuk memakai sebuah sinyal interupsi untuk 2 atau lebih piranti, yang dinamakan *shared interrupt*). Lebih detil tentang interupsi dapat dilihat di Bab 8 buku *Mikroprosesor and Interfacing, Programming and Hardware*, Douglas V. Hall.

Tabel III-1. Standar sinyal interupsi pada komputer personal

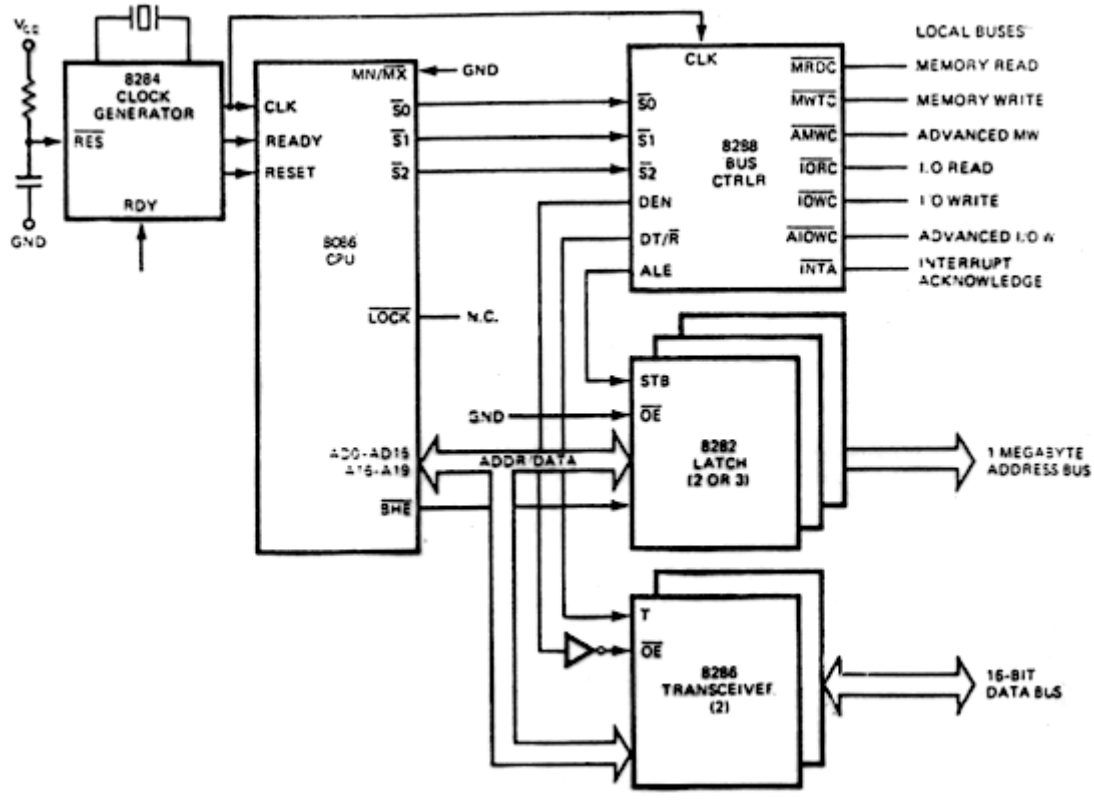
Nomor IRQ	Komponen
IRQ0	DMA (*Direct memory access*)
IRQ1	Pengontrol interupsi (*Programmable interrupt controller*) #1
IRQ2	Pewaktu sistem
IRQ3	Keyboard
IRQ4	Speaker
IRQ5	CMOS/real time clock
IRQ6	DMA
IRQ7	Pengontrol interupsi #2
IRQ8	Numeric coprocessor
IRQ9	Hard disk controller
IRQ10	Joystick
IRQ11	Soundcard
IRQ12	Port serial COM2
IRQ13	Soundcard MIDI
IRQ14	Port paralel printer
IRQ15	Port serial COM3

BAB IV. DMA DAN SLOT EKSPANSI

Pada bagian ini dijelaskan mekanisme transfer data secara langsung tanpa melalui mikroprosesor, yaitu *direct memory access* (DMA), dan slot ekspansi yang banyak dijumpai dalam aristektur komputer personal. Pembicaraan dikhususkan pada sistem komputer personal yang kompatibel dengan IBM PC dengan mikroprosesor 8086. Dengan memahami konsep dasarnya maka mekanisme tersebut dapat pula diterapkan ke sistem komputer yang lain.

A. Mode Maksimum pada 8086

Pada Bab II telah dibahas tentang deskripsi pin-pin pada mikroprosesor 8086 untuk mode minimum, di mana mode operasi ini dipilih dengan memberi nilai tinggi pada pin MN/$\overline{\text{MX}}$. Pada mode tersebut, pin nomor 24 sampai dengan 31 membangkitkan sinyal-sinyal pada bus kendali dengan nama yang terdapat di dalam kurung (Gambar II-1), yaitu secara berurutan: $\overline{\text{INTA}}$ (*interrupt acknowledge*), ALE (*address latch enable*), $\overline{\text{DEN}}$ (*data enable*), DT/$\overline{\text{R}}$ (*data transmit/receive*), M/$\overline{\text{IO}}$ (*memory/IO*), $\overline{\text{WR}}$ (*write*), HLDA (*hold acknowledge*), dan HOLD.

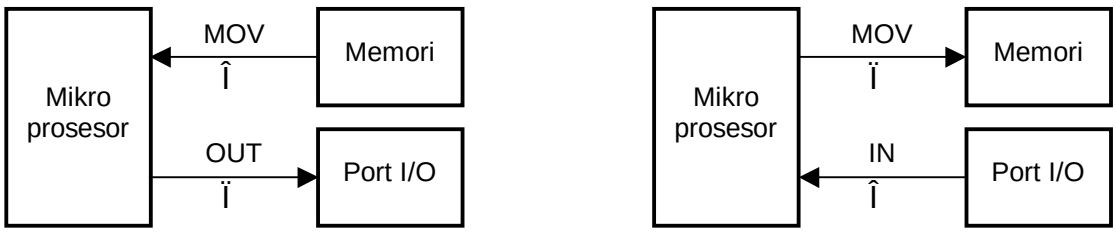


Gambar IV-1. Sinyal kendali pada mode maksimum

Pada mode maksimum, yang dipilih dengan memberi nilai rendah ke pin MN/$\overline{\text{MX}}$, pin sinyal-sinyal kendali dikirimkan dalam bentuk terkode dengan kombinasi sinyal $\overline{\text{S0}}$, $\overline{\text{S1}}$, dan $\overline{\text{S2}}$. Sinyal-sinyal tersebut diperoleh mempergunakan komponen pembantu *Bus Controller* 8288 seperti terlihat dalam (Gambar IV-1). Sinyal-sinyal yang berfungsi sama dengan pada mode minimum adalah $\overline{\text{DEN}}$ dan DT/$\overline{\text{R}}$ yang dihubungkan dengan latch dan buffer untuk memisahkan bus data dan bus alamat. Sedangkan sinyal-sinyal tambahan sesuai namanya adalah $\overline{\text{MRDC}}$ (*memory read*), $\overline{\text{MWTC}}$ (*memory write*), $\overline{\text{AMWC}}$ (*advanced memory write*), $\overline{\text{IORC}}$ (*I/O read*), $\overline{\text{IOWC}}$ (*I/O write*), dan $\overline{\text{AIOWC}}$ (*advanced I/O write*). Pada mode maksimum ini, sinyal untuk mengaktifkan memori atau port (M/$\overline{\text{IO}}$ dalam mode minimum) sudah digabungkan langsung dengan sinyal untuk membaca atau menulis.

B. *Direct Memory Access*

Instruksi pemindahan/transfer data yang tersedia dalam mikroprosesor 8086 hanya dapat dipergunakan untuk memindahkan data dari register dalam mikroprosesor ke memori atau sebaliknya (MOV), serta dari register ke port (OUT) dan sebaliknya dari port ke register (IN). Dengan demikian, untuk memindahkan data dari memori ke port dapat dilakukan dengan kombinasi instruksi MOV dan OUT (Gambar IV-2a), sedangkan transfer data dari port ke memori dilakukan dengan kombinasi instruksi IN dan MOV (Gambar IV-2b).

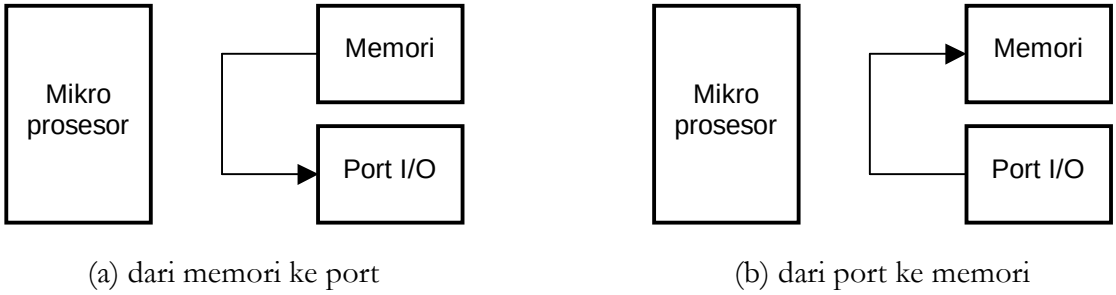


(a) dari memori ke port (b) dari port ke memori

Gambar IV-2. Mekanisme transfer antara port dan memori melalui mikroprosesor

Dengan mekanisme di atas, transfer data antar memori dan port melibatkan mikroprosesor sebagai perantara. Pada aplikasi tertentu, terutama untuk transfer data yang berukuran sangat besar misalnya pemindahan data file dari harddisk ke memori, mekanisme

di atas menjadi tidak efisien. Transfer data akan menjadi lebih cepat apabila dapat dilakukan secara langsung dari memori ke port atau sebaliknya, tanpa melalui mikroprosesor (Gambar IV-3). Mekanisme ini disebut *direct memoru access (DMA).*



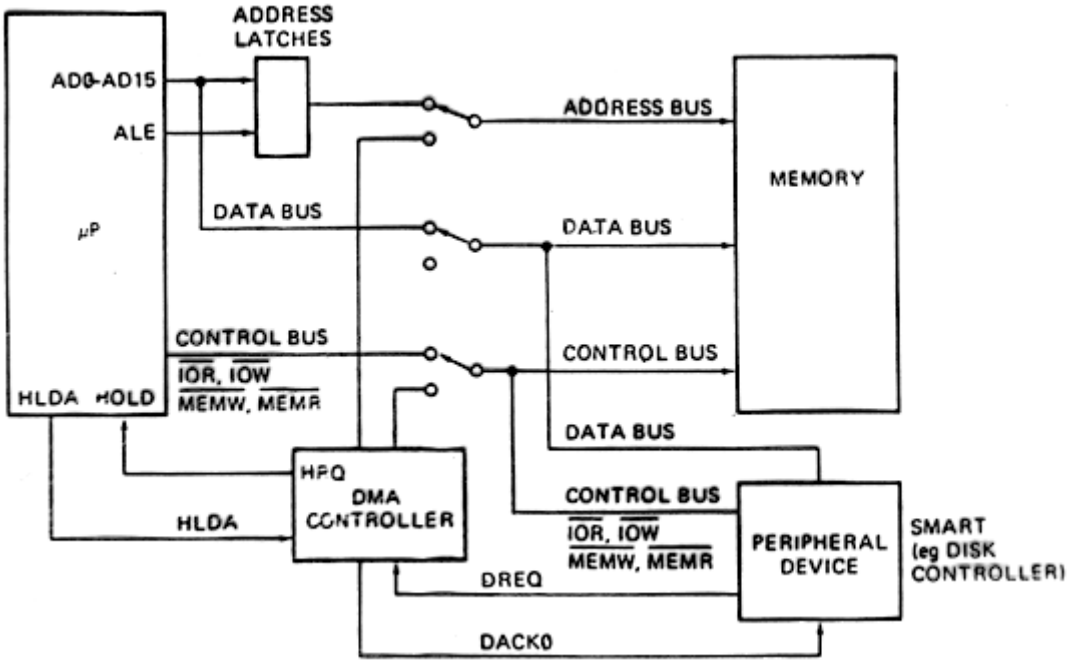
(a) dari memori ke port (b) dari port ke memori

Gambar IV-3. Mekanisme transfer antara port dan memori secara langsung (DMA)

Pada operasi normal, pemakaian bus alamat, bus data dan bus kendali diatur oleh mikroprosesor. Pada saat terjadi transfer data dengan DMA, mikroprosesor melepaskan kendali atas bus-bus tersebut agar dapat dipergunakan secara langsung oleh memori dan port, yaitu dengan sinyal HOLD dan HLDA. Mekanisme tersebut dapat dilakukan dengan memanfaatkan sebuah komponen bantu yang dapat menangani sinyal HOLD dan HLDA secara tepat dan mampu memegang kendali atas bus-bus yang dipinjam secara temporer atau sementara dari mikroprosesor. Komponen bantu tersebut disebut DMA *controller*.

Salah satu komponen bantu yang banyak dipakai dalam sistem 8086 adalah Intel 8237 DMA *controller*. Gambar IV-4 menunjukkan bagaimana 8237 dihubungkan dengan mikroprosesor, bus, memori dan port. Tiga buah *switch* di dalam gambar tersebut dipergunakan untuk memindahkan koneksi ketiga bus (data, alamat, dan kendali) dari mikroprosesor ke DMA *controller*. Pada keadaan normal, ketiga bus terhubung ke mikroprosesor seperti terlihat pada gambar tersebut, sehingga apabila dilakukan operasi transfer data, akan terjadi perpindahan data dari mikroprosesor ke port dan memori atau sebaliknya.

Pada gambar tersebut, perangkat input/output yang akan diperlakukan mode transfer DMA, dalam contoh ini adalah kendali *harddisk* (*disk controller*), selain mempunyai sinyal-sinyal kendali seperti yang telah dibahas sebelumnya, juga mempunyai sinyal DREQ (*DMA request*) dan DACK (*DMA acknowledge*) yang dihubungkan dengan DMA *controller*. Sebuah IC 8237 sendiri mempunyai 4 pasang sinyal DREQ dan DACK yang diberi nomor 0 sampai dengan 3.



Gambar IV-4. Operasi DMA dalam sistem komputer

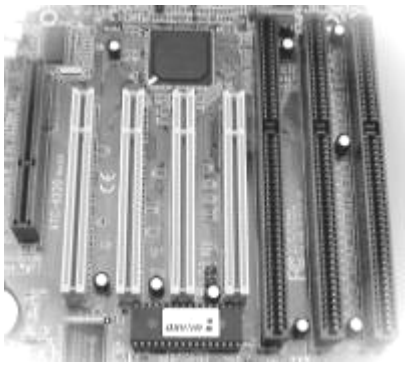
Pada saat data akan diambil dari *harddisk*, *disk controller* mengirimkan sinyal DREQ ke 8237. DMA *controller* kemudian mengirimkan sinyal HRQ (*hold request*), yaitu permintaan untuk meminjam bus, kepada mikroprosesor melalui kaki HOLD. Mikroprosesor merespon permintaan tersebut dengan memutuskan hubungan dirinya ke bus dan mengirimkan sinyal HLDA (*hold acknowledge*) ke 8237. Setelah menerima sinyal tersebut, 8237 kemudian memindahkan *switch* ke bawah sehingga bus sekarang terhubung ke 8237. Dengan demikian kendali terhadap bus berada di tangan 8237.

DMA *controller* kemudian mengirimkan alamat memori di mana data dari *harddisk* akan disimpan. Selanjutnya, 8237 mengirimkan sinyal DACK ke *disk controller* untuk memberitahu agar siap mengirimkan data. Kemudian, 8237 mengaktifkan sinyal pada bus kendali, yaitu $\overline{\text{MEMW}}$ (*memory write*), yang akan mengaktifkan memori dengan alamat yang dituju untuk menerima data, dan $\overline{\text{IOR}}$ (*I/O read*), yang akan mengaktifkan *disk controller* untuk mengirimkan data. Data kemudian ditransfer secara langsung dari port I/O ke memori tanpa melalui mikroprosesor maupun DMA *controller*. Setelah jumlah data yang ditransfer sesuai dengan yang telah diprogram sebelumnya (pada saat inisialisasi 8237), 8237 menonaktifkan sinyal HRQ ke mikroprosesor dan membebaskan bus dengan cara menaikkan kembali ketiga *switch* tadi. Pada saat itu, kendali terhadap bus kembali dipegang oleh mikroprosesor.

Transfer secara DMA dari memori ke port I/O dapat dilakukan dengan cara yang mirip dengan di atas, namun kali ini DMA *controller* mengaktifkan sinyal $\overline{\text{MEMR}}$ (*memory read*), yang akan mengaktifkan memori dengan alamat yang dituju untuk mengirimkan data, dan $\overline{\text{IOW}}$ (*I/O write*), yang akan mengaktifkan port I/O untuk menerima data.

C. Sistem Bus dan Slot Ekspansi

Apabila kita merancang sendiri suatu perangkat yang akan dihubungkan dengan mikroprosesor, maka sinyal-sinyal yang bersesuaian harus dihubungkan antara perangkat tersebut dengan mikroprosesor melalui bus data, bus alamat, dan bus kendali. Misalnya, ketika menghubungkan perangkat DAC (*digital to analog converter*), maka kaki-kaki data digital pada DAC harus dihubungkan dengan bus data. Untuk menghubungkan ke bus yang ada di sistem komputer (pada *motherboard*), adalah tidak mungkin atau sangat sulit apabila dilakukan secara langsung, misalnya dengan cara menyolder. Selain tidak praktis, hal ini juga dapat mengakibatkan kerusakan pada *motherboard*. Oleh karena itu sistem komputer yang standar, misalnya IBM PC, telah menyediakan mekanisme untuk menghubungkan perangkat tambahan ke sistem bus yang ada, yaitu melalui slot ekspansi.



Gambar IV-5. Slot ekspansi

Slot ekspansi ini berupa soket untuk menyisipkan kartu antarmuka (*interface card*) yang terletak di bagian belakang *casing* komputer (Gambar IV-5). Melalui slot ekspansi ini, sinyal-sinyal dari bus data, bus alamat, dan bus kendali siap dihubungkan dengan sinyal-sinyal pada perangkat (kartu antarmuka) yang kita pasangkan. Posisi sinyal pada slot ekspansi sudah ditentukan, sehingga apabila kita merancang sebuah kartu antarmuka, maka

kita harus mengikuti standar yang ada. Terdapat beberapa standar slot ekspansi sesuai dengan bus yang ada.

1. Slot ekspansi ISA

Gambar IV-6a menunjukkan posisi dan nama sinyal yang dikeluarkan melalui slot ekspansi ISA (*Industry Standard Architecture*), yaitu standar bus yang dikeluarkan oleh IBM untuk komputer personal PC-XT (*Personal Computer-Extended Technology*). Slot in mempunyai 62 sinyal yang terbagi menjadi 2 sisi yang diberi nomor A1-A31 dan B1-B31 dengan nama yang sesuai dengan fungsinya[(1)]. Pada slot ekspansi ISA, lebar bus data yang dikeluarkan adalah 8 bit (D0-D7), sedangkan bus alamat selebar 20 bit (A0-A19). Koneksi untuk catu daya, yaitu +5V, -5V, +12V, -12V, dan GND, serta pembangkit gelombang OSC dan CLOCK.

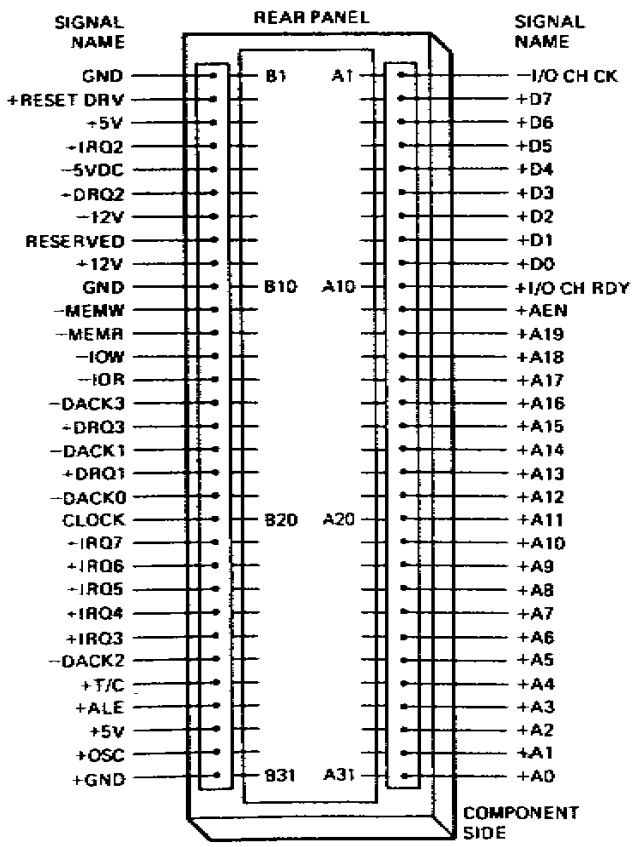
Terdapat pula sinyal kendali IOW, IOR, MEMW, dan MEMR untuk operasi pembacaan atau penulisan memori dan port, ALE untuk pengalamatan. Sinyal RESETDRV akan diberikan oleh mikroprosesor pada saat komputer pertama kali dinyalakan atau direset. Sinyal I/OCHRDY dipergunakan untuk menyisipkan *wait state* melalui sinyal READY pada mikroprosesor.

Pada sistem ini, sinyal permintaan interupsi yang telah dicabangkan oleh *interrupt* controller 8259 yang dapat dipergunakan adalah IRQ2-IRQ7. Tidak perlu ada sinyal INTA karena mekanisme interupsi telah ditangani oleh 8259. Sementara untuk mekanisme DMA, tersedia 3 sinyal permintaan DMA (DRQ1-DRQ3) dan jawaban dari 8237 (DACK0-DACK3). Sinyal AEN (*address enable*) untuk dekoder alamat pada mekanisme DMA, dan sinyal T/C (*transfer complete*) untuk memberitahu perangkat luar bahwa transfer DMA telah selesai.

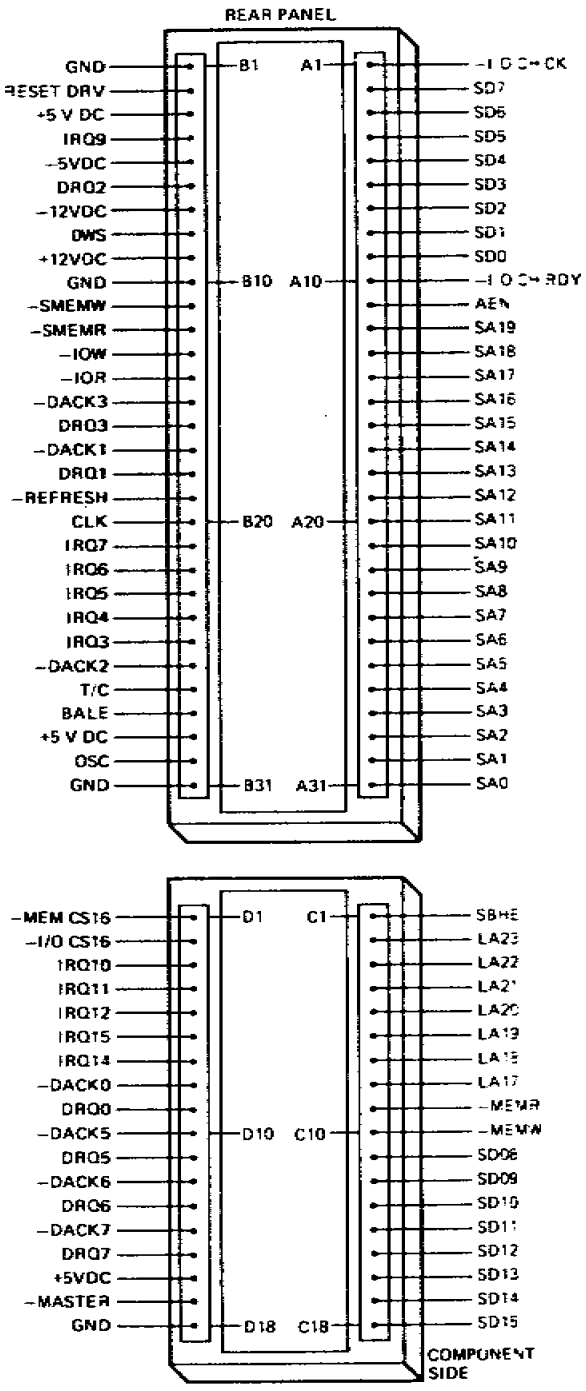
Gambar IV-6b adalah pengembangan standar bus ISA untuk sistem komputer IBM PC-AT (*Advanced Technology*). Terdapat tambahan 36 sinyal yang terbagi menjadi 2 sisi dengan nomor C1-C18 dan D1-D18. Slot ekspansi ini memiliki bus data dengan lebar 16 bit, byte rendah SD0-SD7 dan byte tinggi SD08-SD15. Lebar bus alamat yang tersedia adalah 24 bit, yaitu SA0-SA19 ditambah dengan LA20-LA23.

---

[1] Tanda + di depan nama sinyal menunjukkan sinyal tersebut aktif tinggi, sedangkan tanda – menunjukkan aktif rendah.



(a) pada IBM PC-XT



(b) pada IBM PC-AT

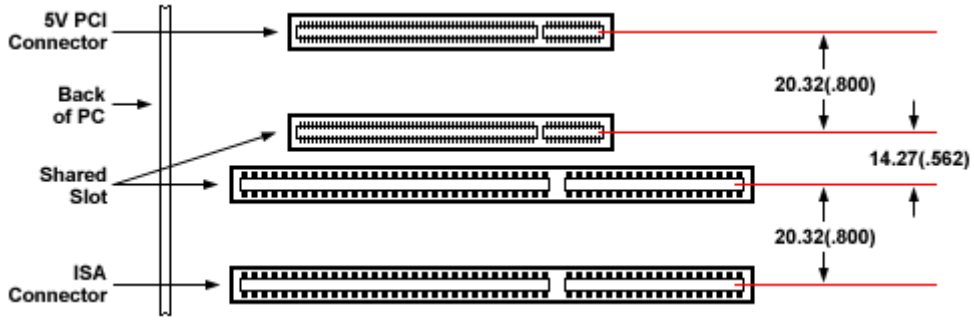
Gambar IV-6. Slot ekspansi ISA

IBM PC-AT mempunyai 2 buah pengontrol interupsi 8259 yang bekerja secara *master-slave*, sehingga jumlah sinyal permintaan interupsi pada slot ekspansi meningkat menjadi 11 buah, yaitu IRQ-3-IRQ7 dan IRQ9-IRQ15. Demikian juga untuk mekanisme DMA, PC-AT memiliki 2 buah pengontrol DMA, sehingga sinyal DMA juga meningkat menjadi 7 kanal, yaitu DRQ0-DRQ3 dan DRQ5-DRQ7 serta DACK0-DACK3 dan

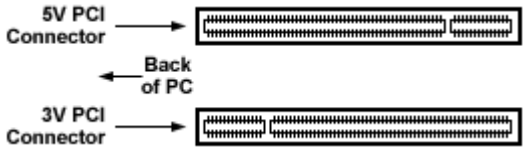
DACK5-DACK7. Sinyal lain adalah 0WS (*zero wait state*) yang memberitahu bahwa transfer data yang terjadi tidak memerlukan *wait state*. Sinyal MEMCS16 dan IOCS16 untuk memberitahu mikroprosesor bahwa operasi transfer data yang akan dilakukan adalah operasi 16 bit (2 byte sekaligus sehingga lebih cepat). Operasi ini memanfaatkan pula sinyal SBHE yang mempunyai fungsi seperti telah dijelaskan pada bab sebelumnya.

2. Slot ekspansi PCI

Pada tahun 1993, Intel memperkenalkan standar bus baru yang diberi nama PCI (*Peripheral Component Interconnect*) yang sangat populer sampai saat ini dan banyak dipergunakan dalam sistem komputer dengan mikroprosesor Pentium. Slot ekspansi PCI memiliki bus data dengan lebar 32 atau 64 bit, namun mempunyai ukuran fisik yang lebih kecil dibandingkan ISA (Gambar IV-7). Jarak antar kaki pada slot ekspansi PCI lebih kecil dibandingkan pada ISA. Intel juga mengembangkan prosesor dengan tegangan kerja lebih rendah untuk mengurangi disipasi panas yang timbul, yaitu sebesar 3V. Untuk menghindarkan kesalahan dalam memasang kartu antarmuka PCI dengan tegangan yang berbeda, maka orientasi dari slot ekspansi dengan tegangan rendah dibalik (Gambar IV-8). Slot ekspansi PCI terdiri atas 188 sinyal yang terbagi menjadi 2 sisi yang diberi nomor A1-A94 dan B1-B94. Deskripsi sinyal-sinyal yang terdapat pada slot ekspansi PCI diberikan dalam Tabel IV-1.



Gambar IV-7. Ukuran slot ekspansi PCI dan ISA



Gambar IV-8. Slot ekspansi PCI dengan tegangan yang berbeda

Tabel IV-1. Deskripsi sinyal slot ekspansi PCI

Pin No.	*Signal Name*	*Pin No.*	*Signal Name*
A1	Test Logic Reset	B1	-12 VDC
A2	+12 VDC	B2	Test Clock
A3	Test Mde Select	B3	Ground
A4	Test Data Input	B4	Test Data Output
A5	+5 VDC	B5	+5 VDC
A6	Interrupt A	B6	+5 VDC
A7	Interrupt C	B7	Interrupt B
A8	+5 VDC	B8	Interrupt D
A9	Reserved	B9	PRSNT1#
A10	+V I/O *	B10	+V I/O *
A11	Reserved	B11	PRSNT2#
A12	Ground or Key *	B12	Ground or Key *
A13	Ground or Key *	B13	Ground or Key *
A14	Reserved	B14	Reserved
A15	Reset	B15	Ground
A16	+V I/O *	B16	Clock
A17	Grant PCI use	B17	Ground
A18	Ground	B18	Request
A19	Reserved	B19	+V I/O *
A20	Address/Data 30	B20	Address/Data 31
A21	+3.3 VDC	B21	Address/Data 29
A22	Address/Data 28	B22	Ground
A23	Address/Data 26	B23	Address/Data 27
A24	Ground	B24	Address/Data 25
A25	Address/Data 24	B25	+3.3VDC
A26	Initialization Device Select	B26	Command/Byte Enable 3
A27	+3.3 VDC	B27	Address/Data 23
A28	Address/Data 22	B28	GND
A29	Address/Data 20	B29	Address/Data 21
A30	Ground	B30	Address/Data 19
A31	Address/Data 18	B31	+3.3 VDC
A32	Address/Data 16	B32	Address/Data 17
A33	+3.3 VDC	B33	Command/Byte Enable 2
A34	Frame	B34	Ground
A35	Ground	B35	Initiator Ready
A36	Target Ready	B36	+3.3 VDC
A37	Ground	B37	Device Select
A38	Stop Transfer Cycle	B38	Ground
A39	+3.3 VDC	B39	Lock bus
A40	Snoop Done	B40	Parity Error
A41	Snoop Backoff	B41	+3.3 VDC
A42	Ground	B42	System Error
A43	Parity	B43	+3.3 VDC
A44	Address/Data 15	B44	Command/Byte Enable 1
A45	+3.3 VDC	B45	Address/Data 14
A46	Address/Data 13	B46	Ground
A47	Address/Data 11	B47	Address/Data 12
A48	Ground	B48	Address/Data 10
A49	Address/Data 9	B49	Ground
A50	Ground or Key *	B50	Ground or Key *
A51	Ground or Key *	B51	Ground or Key *
A52	Command/Byte Enable 0	B52	Address/Data 8
A53	+3.3 VDC	B53	Address/Data 7
A54	Address/Data 6	B54	+3.3 VDC
A55	Address/Data 4	B55	Address/Data 5
A56	Ground	B56	Address/Data 3
A57	Address/Data 2	B57	Ground
A58	Address/Data 0	B58	Address/Data 1
A59	+V I/O *	B59	+V I/O *
A60	Request 64 bit	B60	Acknowledge 64 bit
A61	+5 VDC	B61	+5 VDC
A62	+5 VDC	B62	+5 VDC

START OF 64bit CONNECTOR			
A63	Ground	B63	Reserved
A64	Command/Byte Enable 7	B64	Ground
A65	Command/Byte Enable 5	B65	Command/Byte Enable 6
A66	+V I/O *	B66	Command/Byte Enable 4
A67	Parity 64	B67	Ground
A68	Address/Data 62	B68	Address/Data 63
A69	Ground	B69	Address/Data 61
A70	Address/Data 60	B70	+V I/O *
A71	Address/Data 58	B71	Address/Data 59
A72	Ground	B72	Address/Data 57
A73	Address/Data 56	B73	Ground
A74	Address/Data 54	B74	Address/Data 55
A75	+V I/O *	B75	Address/Data 53
A76	Address/Data 52	B76	Ground
A77	Address/Data 50	B77	Address/Data 51
A78	Ground	B78	Address/Data 49
A79	Address/Data 48	B79	+V I/O *
A80	Address/Data 46	B80	Address/Data 47
A81	Ground	B81	Address/Data 45
A82	Address/Data 44	B82	Ground
A83	Address/Data 42	B83	Address/Data 43
A84	+V I/O *	B84	Address/Data 41
A85	Address/Data 40	B85	Ground
A86	Address/Data 38	B86	Address/Data 39
A87	Ground	B87	Address/Data 37
A88	Address/Data 36	B88	+V I/O *
A89	Address/Data 34	B89	Address/Data 35
A90	Ground	B90	Address/Data 33
A91	Address/Data 32	B91	Ground
A92	Reserved	B92	Reserved
A93	Ground	B93	Reserved
A94	Reserved	B94	Ground

Sebenarnya, sebelum munculnya PCI, terdapat 2 buah standar bus yang memiliki bus data selebar 32 bit, yaitu EISA (*Extended ISA*) yang dikembangkan oleh IBM dan MCA (*Micro Channel Architecture*) yang dikembangkan oleh Compaq. Namun kedua standar bus tersebut tidak populer dan hanya dipergunakan pada sistem komputer tertentu. Selain itu juga terdapat jenis bus lokal, yaitu bus yang menghubungkan mikroprosesor dengan komponen tertentu, misalnya dengan kartu display pada VESA[(2)] *local bus* (VLB) dan AGP (*Accelerated Graphics Port*). Namun yang terakhir ini lebih tepat disebut sebagai port alih-alih bus. Perbandingan antara sistem bus yang ada dapat dilihat dalam

[2] Video Electronics Standards Association

Tabel IV-2.

Tabel IV-2. Perbandingan sistem bus

Bus	*Width (bits)*	*Bus Speed (MHz)*	*Bus Bandwidth (MBytes/sec)*
8-bit ISA	8	8,3	7,9
16-bit ISA	16	8,3	15,9
EISA	32	8,3	31,8
VLB	32	33	127,2
PCI	32	33	127,2
64-bit PCI 2.1	64	66	508,6
AGP	32	66	254,3
AGP (x2 mode)	32	66x2	508,6
AGP (x4 mode)	32	66x4	1.017,3

BAB V. TRANSFER DATA PARALEL DAN ANTARMUKA DIGITAL

Salah satu fungsi dari mikroprosesor adalah untuk melakukan pemrosesan terhadap data, baik berupa operasi matematika maupun operasi logika. Data tersebut merupakan data digital yang berasal dari memori maupun port yang diambil (dibaca) atau disimpan (ditulis) ke kedua komponen tersebut dengan instruksi MOV, IN, dan OUT serta memanfaatkan jalur-jalur bus kendali, yaitu $\overline{RD}$ dan $\overline{WR}$. Mekanisme transfer data melibatkan 2 pihak, yaitu pengirim dan penerima. Agar tidak terjadi kesalahan dalam pemindahan data, maka perlu dilakukan pengaturan bagaimana data tersebut ditransfer[1].

A. Mode Transfer Data Paralel

Terdapat 4 mode transfer data paralel yang dapat dilakukan di mana masing-masing mempunyai protokol tersendiri, yaitu:

1. Sederhana (*simple*)
2. Sederhana dengan *strobe*
3. Jabat tangan tunggal (*single handshake*)
4. Jabat tangan ganda (*double handshake*)

1. Input/Output sederhana

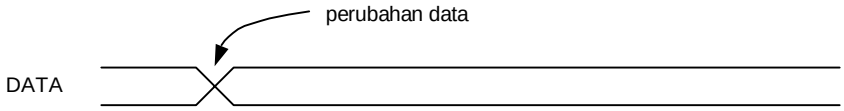
Mode ini digunakan untuk transfer data dari atau ke perangkat yang selalu siap. Kapan saja data akan ditulis ke perangkat tersebut, dia selalu siap untuk menerima data. Contohnya adalah ketika kita hendak menulis ke perangkat display sederhana seperti lampu led, *seven segment*, dll, yang selalu siap menerima data untuk ditampilkan, maka mikroprosesor bisa mengirimkan data kapan saja. Demikian pula kapan saja data akan dibaca dari suatu perangkat dengan mode sederhana, maka perangkat tersebut harus selalu siap menyediakan data yang diminta. Misalnya *switch* atau *relay* yang selalu siap untuk diubah nilainya oleh mikroprosesor.

---

[1] Pengaturan yang berupa prosedur urut-urutan yang harus dijalani sering disebut sebagai protokol. Contoh sebuah protokol adalah pada saat kita hendak menelpon seseorang, maka urutan yang harus diikuti adalah: mengangkat gagang telepon, lalu memutar/menekan tombol nomor telpon yang akan dituju, menunggu sampai tersambung dan ada suara dari seberang, kemudian memperkenalkan diri, dst. Jika protokol tidak diikuti, misalnya setelah mengangkat gagang telepon langsung memperkenalkan diri tanpa memutar nomor, maka komunikasi akan terganggu atau bahkan tidak dapat terlaksana.

Diagram waktu (*timing diagram*) untuk mode ini diberikan dalam Gambar I-1, di mana sumbu mendatar adalah waktu sedangkan sumbu vertikal adalah nilai tegangan atau nilai biner. Garis yang bersilangan pada DATA menunjukkan perubahan data yanga dapat terjadi di mana saja pada sumbu waktu, artinya data yang valid dapat diberikan untuk ditulis ke atau tersedia untuk dibaca perangkat luar kapan saja. Tidak adanya sinyal yang lain menunjukkan bahwa waktu perubahan data tersebut tidak dipengaruhi/ditentukan oleh sinyal lain.

Apabila perangkat tidak selalu siap menerima atau menyediakan data, maka transfer data sederhana tidak dapat dilakukan. Jika data dikirimkan oleh mikroprosesor ketika perangkat yang ditulis tidak siap, maka data tersebut tidak akan diterima oleh perangkat tersebut atau dapat dikatakan data tersebut hilang. Di pihak lain, apabila mikroprosesor membaca input dari perangkat luar sementara data belum tersedia (contohnya jika ADC belum selesai mengkonversi sinyal analog ke digital), maka data yang terbaca akan salah (mungkin data yang sebelumnya akan terbaca lagi atau bisa pula data acak yang terambil). Dengan demikian mode transfer data sederhana atau *simple* tidak cocok untuk kasus-kasus tersebut.



Gambar V-1. Mode transfer data sederhana (simple)

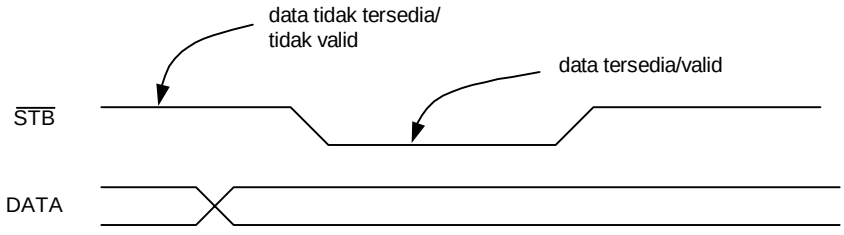
2. Input/Output sederhana dengan strobe

Untuk mengatasi kelemahan transfer data sederhana di atas, sebuah sinyal strobe[(2)] dimanfaatkan untuk menandai kapan data yang valid tersedia. Apabila mikroprosesor hendak mengambil data dari suatu perangkat luar, maka pertama-tama dia harus memeriksa dulu nilai sinyal strobe yang diberikan oleh perangkat tadi, apakah sinyal tersebut aktif atau tidak. Jika sinyal strobe tersebut ($\overline{\text{STB}}$) aktif, dalam hal ini bernilai rendah, maka mikroprosesor dapat mengambil data yang ada karena data valid sudah tersedia. Namun apabila mikroprosesor mengetahui sinyal tersebut tidak aktif, dalam hal ini bernilai tinggi, maka dia tidak boleh mengambil data karena nilai data tidak valid. Mikroprosesor dapat

menunggu atau mengerjakan hal lain, kemudian memeriksa $\overline{\text{STB}}$ lagi sampai nilainya aktif dan ketika itu data baru bisa diambil atau ditransfer. Dalam hal ini, mikroprosesor menggunakan cara polling.

$\overline{\text{STB}}$ dapat juga dihubungkan ke INTR atau IRQ untuk membangkitkan sinyal permintaan interupsi. Contohnya pada koneksi keyboard ke sistem mikroprosesor, jika sebuah tombol di keyboard ditekan, maka rangkaian di dalamnya akan memberikan sebuah nilai yang sesuai dengan tombol yang ditekan (yang dikenal sebagai kode ASCII) ke dalam buffer keyboard yang dihubungkan dengan bus data. Selanjutnya keyboard juga mengaktifkan sebuah sinyal yang berfungsi sebagai strobe yang dihubungkan dengan sinyal interupsi (biasanya IRQ3), sehingga dengan asumsi mikroprosesor mengijinkan adanya interupsi, sebuah rutin pelayanan interupsi kemudian dijalankan yang berisi instruksi untuk mengambil data dari keyboard tersebut.

Diagram waktu mode transfer data sederhana dengan strobe diberikan dalam Gambar V-2. Di sini terlihat bahwa data disediakan terlebih dahulu, baru sinyal $\overline{\text{STB}}$ diaktifkan. Sebagai contoh, jika kita menggunakan mode transfer data sederhana dengan strobe untuk konversi data analog ke digital dengan ADC, sinyal $\overline{\text{EOC}}$ (*end of conversion*) yang terdapat pada ADC dapat dipergunakan sebagai sinyal $\overline{\text{STB}}$. Pada saat konversi, sinyal $\overline{\text{EOC}}$ bernilai tinggi. Ketika konversi selesai maka data digital hasil konversi sudah tersedia di *buffer* keluaran ADC. ADC kemudian menurunkan sinyal $\overline{\text{EOC}}$ sehingga strobe aktif.



Gambar V-2. Mode transfer data *simple strobe*

[2] *Strobe* dalam artian sebenarnya adalah perangkat yang digunakan untuk menghasilkan cahaya kilat yang sangat terang untuk memberikan tanda atau sinyal.

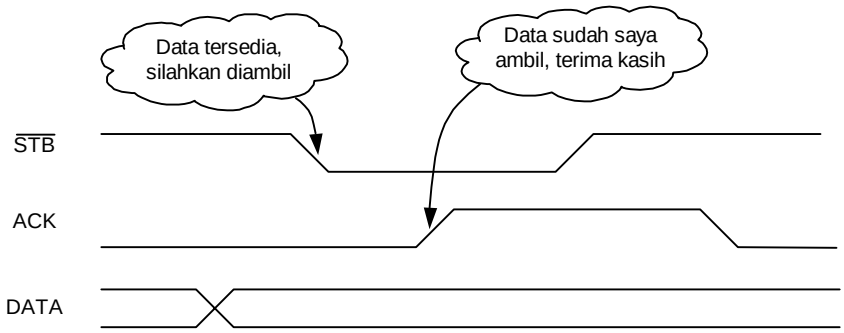
3. Transfer data jabat tangan tunggal

Mode transfer data sederhana dengan strobe masih mempunyai kelemahan. Meskipun pihak penerima data dapat mengetahui kapan adanya data valid, namun pihak pengirim tidak mampu mengetahui apakah data yang disediakannya sudah diambil oleh penerima atau belum. Komponen pengirim data mungkin mempergunakan waktu untuk memperkirakan bahwa data sudah diambil, yaitu dalam waktu tertentu, misalnya sekian mikrodetik, dianggap data sudah diambil, sehingga dia boleh menaikkan sinyal strobe dan mempersiapkan data berikutnya. Namun hal ini bukan merupakan penyelesaian yang tepat, karena kapan data akan diambil oleh pihak penerima data tidak dapat ditentukan secara pasti. Hal ini adalah hak prerogratif pemrogram.

Terdapat 2 kesalahan yang mungkin terjadi. Pertama, ada kemungkinan data hilang atau tidak terambil. Pada kasus mekanisme polling, jika sinyal strobe sudah terlanjur dinaikkanketika mikroprosesor melakukan pemeriksaan sinyal tersebut, maka mikroprosesor akan menganggap data yang valid belum tersedia. Padahal sebenarnya data tersebut sudah tersedia sebelum mikroprosesor melakukan polling ke komponen tersebut, sehingga data tadi akan hilang, karena data tersebut belum diambil oleh mikroprosesor sementara data berikutnya sudah disiapkan lagi oleh komponen pengirim data.

Kesalahan kedua yaitu ada kemungkinan sebuah data yang valid akan terambil lebih dari sekali oleh penerima data. Ketika mikroprosesor memeriksa komponen tersebut dan mendapati sinyal strobe aktif, maka data akan diambil. Apabila waktu aktifnya strobe relatif lama, maka pada saat polling berikutnya ke komponen tersebut, mikroprosesor masih mendapati sinyal strobe aktif, oleh karena itu data tersebut akan diambil kembali. Pada beberapa kasus, terambilnya sebuah data sebanyak lebih dari sekali tidak mempunyai efek yang besar, tetapi pada kebanyakan aplikasi hal ini dapat berakibat fatal, misalnya pada transfer file antara 2 komputer atau koneksi melalui modem.

Untuk menghindarkan terjadinya data yang hilang maupun data yang sama terambil lebih dari sekali, diperlukan komunikasi (dalam bahasa manusia adalah percakapan) antara pihak pengirim dan penerima data. Seperti pada mode transfer sebelumnya, pihak pengirim memberitahukan bahwa data sudah tersedia di jalur data dengan mempergunakan sinyal strobe. Sementara pihak penerima data mempergunakan sebuah sinyal yang lain, yaitu ACK (*acknowledge*) untuk memberitahu pihak pengirim bahwa data tersebut sudah diambilnya.



Gambar V-3. Mode transfer data jabat tangan tunggal

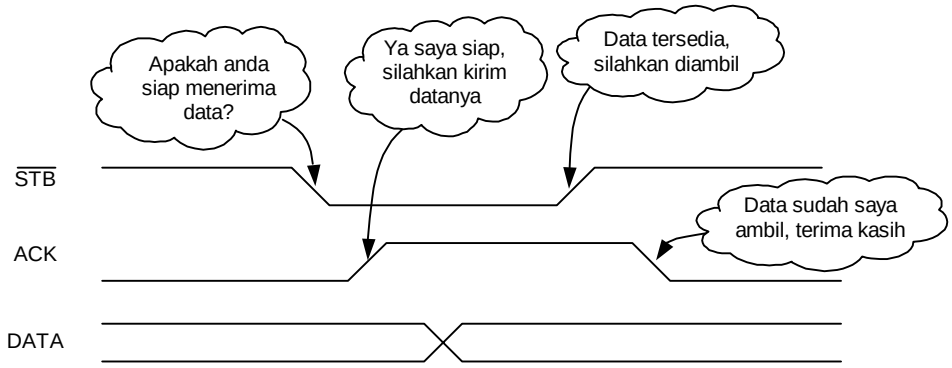
Gambar V-3 menunjukkan komunikasi atau percakapan yang terjadi pada mode jabat tangan tunggal. Setelah menyediakan memasukkan data ke jalur data, pihak pengirim kemudian mengaktifkan sinyal $\overline{\text{STB}}$, seolah-olah berkata: "*Data sudah saya sediakan sediakan, silahkan diambil*". Pihak penerima ketika mendapat informasi ini, dapat melalui polling ataupun interupsi, kemudian mengambil data dari jalur data serta kemudian mengaktifkan sinyal ACK, seolah-olah berkata: "*Data sudah saya ambil, terima kasih*". Ketika mendeteksi sinyal ini, pengirim dapat menaikkan lagi sinyal strobe (menonaktifkan), sehingga proses pengiriman data dapat dilanjutkan untuk data berikutnya tanpa khawatir kehilangan data atau data terambil lebih dari satu kali.

4. Transfer data jabat tangan ganda

Meskipun dengan menggunakan jabat tangan tunggal, kemungkinan data terambil lebih dari sekali dapat dihilangkan, pada beberapa kasus masih ada kemungkinan data akan hilang. Hal ini dapat terjadi apabila ketika pihak pengirim memasukkan data ke jalur data, ternyata pihak penerima belum siap. Kemungkinan lain adalah apabila jalur data masih dipakai oleh pihak penerima data untuk keperluan lain, maka data yang dimasukkan oleh pengirim akan menjadi kacau. Untuk menangani hal itu, diperlukan suatu komunikasi sebelum data diberikan.

Pada Gambar V-4 terlihat sebelum memberikan datanya, pihak pengirim menanyakan kesiapan pihak penerima dengan menurunkan sinyal strobe, seolah-olah berkata: "*Apakah anda siap untuk menerima data?*". Setelah mendeteksi aktifnya sinyal strobe tersebut dan menyiapkan diri untuk menerima data, pihak penerima kemudian mengaktifkan sinyal ACK, seolah-olah menjawab dengan kalimat: "*Ya, saya sudah siap untuk menerima data. Silahkan kirim datanya*". Mengetahui hal itu, pihak pengirim

kemudian memasukkan data ke jalur data, kemudian menaikkan kembali (menonaktifkan) sinyal strobe, seolah-olah berkata: "*Data sudah tersedia, silahkan diambil*". Setelah mengambil data, penerima menurunkan sinyal acknowledge, seolah-olah berkata: "*Data sudah saya ambil, terima kasih*".



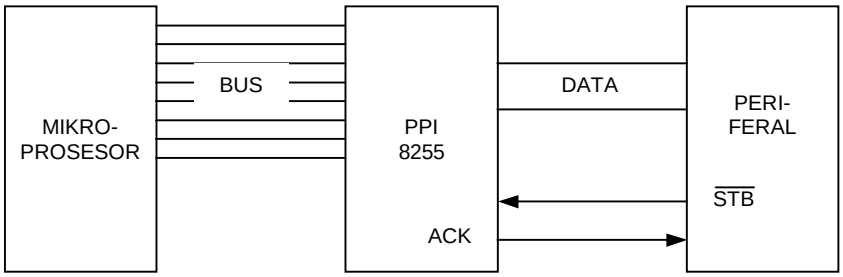
Gambar V-4. Mode transfer data jabat tangan ganda

Pada gambar tersebut terlihat bahwa pada mode ini terjadi 2 kali percakapan, yaitu sebelum data dikirmkan dan setelah data diterima, oleh karena itu mode transfer ini disebut jabat tangan ganda. Penggunaan sinyal $\overline{\text{STB}}$ dan ACK pada kedua percakapan tersebut namun dengan posisi yang berkebalikan (strobe diaktifkan pada percakapan pertama dan dinonaktifkan pada percakapan kedua, serta acknowledge dinaikkan pada percakapan pertama dan diturunkan pada percakapan kedua), dapat menghemat banyaknya sinyal yang diperlukan. Jadi untuk komunikasi cukup memakai 2 buah sinyal, tidak perlu 4 sinyal.

B. Programmable Peripheral Interface (PPI 8255A)

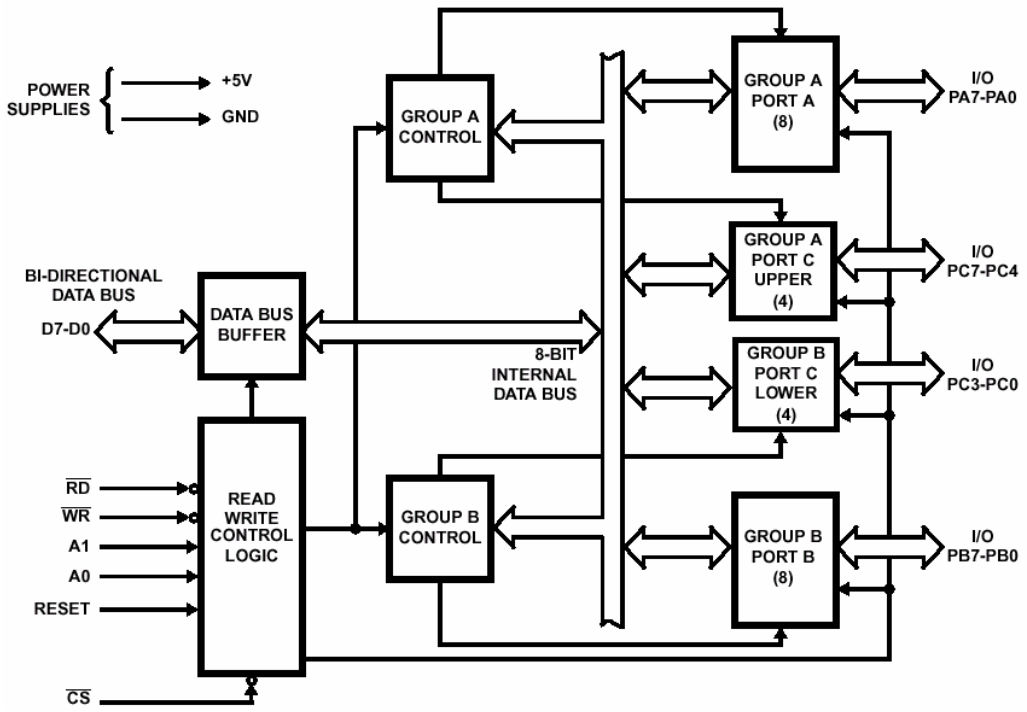
Syarat yang diperlukan pada mode jabat tangan (baik tunggal maupun ganda) adalah kemampuan untuk melakukan percakapan, baik oleh pihak pengirim maupun penerima data. Kemampuan yang dimaksud adalah kemampuan pihak pengirim untuk mengaktifkan sinyal strobe dan mendeteksi aktif tidaknya sinyal acknowledge, dan sebaliknya kemampuan pihak penerima untuk mendeteksi aktif tidaknya sinyal strobe dan mengaktifkan sinyal acknowledge. Hal in tidak mudah untuk dilakukan. Oleh karena itu, Intel membuat sebuah komponen yang mampu melakukan komunikasi sehingga dapat dipergunakan untuk membantu mikroprosesor atau komponen yang kita rancang untuk melakukan transfer data dengan mode jabat tangan. Komponen tersebut adalah IC PPI (*Programmable Peripheral Interface*) 8255. Gambar V-5 menunjukkan posisi PPI 8255

dalam membantu mikroprosesor melakukan transfer data dengan sebuah perangkat luar (periferal).

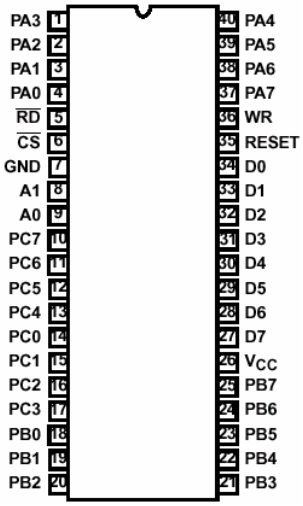


Gambar V-5. PPI bertindak sebagai perantara mikroprosesor dengan periferal

PPI 8255 adalah IC dengan 40 kaki yang memiliki 3 buah port yang dapat dipergunakan untuk input maupun output, yaitu port A, port B, dan port C (Gambar V-6 dan Gambar V-7). Kaki D0-D7 menghubungkan bus data pada sistem mikroprosesor dengan sebuah buffer 3 keadaan (*3-state buffer*). Kaki $\overline{CS}$ dihubungkan dengan dekoder alamat, sedangkan kaki A1 dan A0 langsung dihubungkan dengan bus alamat. RESET, $\overline{WR}$, dan $\overline{RD}$ dihubungkan dengan bus kendali.



Gambar V-6. Diagram blok internal PPI 8255



Gambar V-7. Konfigurasi kaki IC PPI 8255

Di bagian dalam 8255 terdapat bus data internal selebar 8 bit yang dipergunakan untuk melewatkan data dari D0-D7 ke port A, B, dan C, atau sebaliknya. Apabila sebuah port dijadikan input, maka aliran data terjadi dari port ke D0-D7. Sebaliknya jika sebuah port dijadikan output, maka aliran data yang terjadi adalah dari D0-D7 ke port. Arah aliran data ditentukan oleh kaki $\overline{WR}$, dan $\overline{RD}$. $\overline{WR}$ diaktifkan (rendah) untuk mengirimkan data ke port, dan $\overline{RD}$ diaktifkan untuk mengambil data dari port. Port yang dituju dipilih dengan menggunakan kombinasi A1 dan A0, sedangkan $\overline{CS}$ yang dihubungkan ke dekoder alamat biasanya menunjuk ke alamat dasar (*base address*) dari 8255. Contoh: jika alamat dasar dari dekode alamat menunjuk ke alamat port H340, maka alamat port A adalah H340, port B H341, port C H342 dan alamat CW adalah H343. CW (*control word*), atau kata kendali, adalah register dalam 8255 yang dipergunakan untuk memprogam (sesuai dengan namanya) mode operasi dari 8255.

Kombinasi sinyal kendali dan aksi yang dilakukan diberikan dalam Tabel V-1. Untuk memprogram 8255, maka sebuah data dikirimkan ke alamat CW. Bergantung pada data yang dikirimkan, terdapat 2 hal yang dapat dilakukan ketika menuliskan data tersebut ke alamat CW dari 8255. Pertama untuk memilih arah (input atau output) dan mode dari masing port (A, B, dan C), dan kedua untuk melakukan operasi bit set/reset pada port C.

Tabel V-1. Kombinasi sinyal kendali dan alamat pada PPI 8255

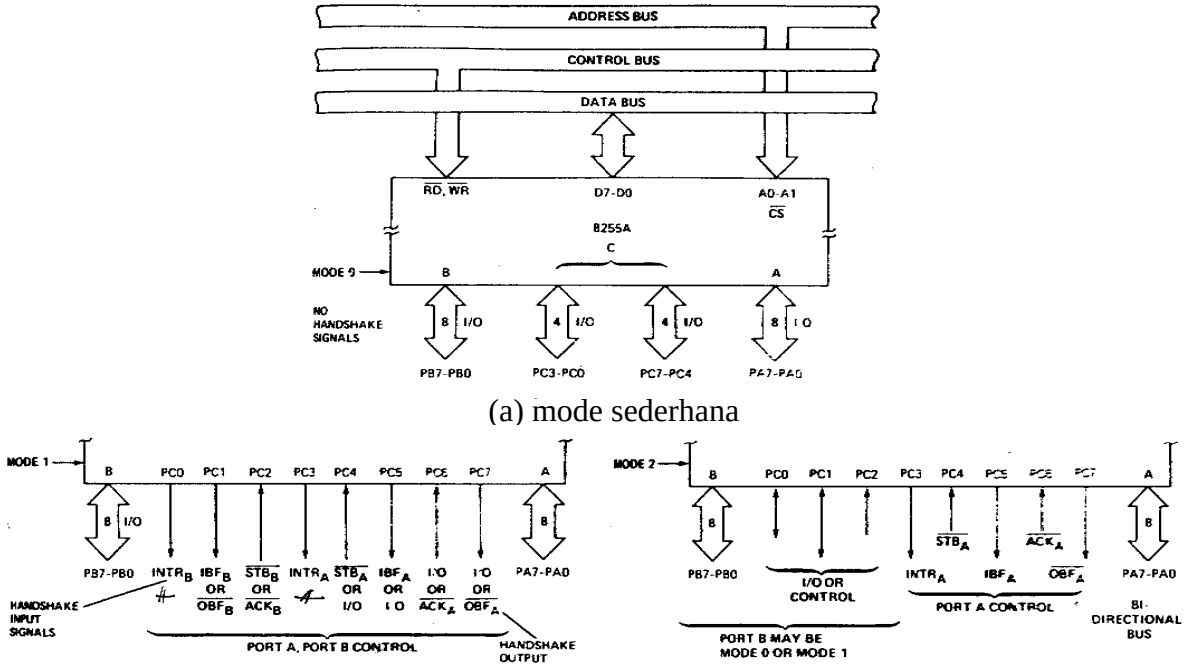
$\overline{CS}$	A1	A0	$\overline{WR}$	$\overline{RD}$	Aksi
0	0	0	0	1	Menulis data ke port A (bus data → Port A)
0	0	1	0	1	Menulis data ke port B (bus data → Port B)
0	1	0	0	1	Menulis data ke port C (bus data → Port C)
0	1	1	0	1	Menulis data ke CW (memprogram)
0	0	0	1	0	Membaca data dari port A (Port A → bus data)
0	0	1	1	0	Membaca data dari port B (Port B → bus data)
0	1	0	1	0	Membaca data dari port C (Port C → bus data)
0	1	1	1	0	Membaca mode sekarang (dari CW)
1	X	X	X	X	Tidak beroperasi (bus data terputus)
X	X	X	1	1	Tidak beroperasi (bus data terputus)

C. Mode Operasi PPI 8255

Ketiga port dalam 8255 dapat dikelompokkan menjadi 2 grup, yaitu grup A yang terdiri atas port A (PA7-PA0) dan sebagian port C (PC7-PC4), serta grup B yang terdiri atas port B (PB7-PB0) dan sebagian lain port C (PC3-PC0). Masing-masing grup dikendalikan oleh register kendali yang ada di dalam 8255 (Gambar V-6). Ketiga port tersebut dapat diprogram untuk bekerja pada mode transfer data simple dan jabat tangan. Jika dipergunakan untuk mode transfer sederhana maka ketiga port tersebut dapat dioperasikan secara independen baik sebagai input atau output (Gambar V-8a). Pada mode ini, port C dapat dipergunakan sebagai port 8 bit atau sebagai 2 buah port 4 bit (PC7-PC4 dan PC3-PC0).

Untuk mode jabat tangan, karena diperlukan beberapa sinyal jabat tangan untuk percakapan ($\overline{STB}$ dan ACK), maka sebagian dari port C yang menjadi kelompoknya akan dipergunakan untuk keperluan itu. Terlihat pada Gambar V-8b, jika port A dipergunakan sebagai input dengan mode jabat tangan, maka PC4 berfungsi sebagai $\overline{STB_A}$, PC5 berfungsi untuk memberikan sinyal acknowledge yang diberi nama IBF_A (*input buffer full*) yang menandakan data sudah masuk ke buffer pada port A. Apabila operasi transfer data melibatkan mekanisme interupsi, maka kaki PC3 ($INTR_A$) dapat digunakan sebagai sinyal permintaan interupsi dengan menghubungkannya ke IRQ yang ada di bus kendali. PC6 dan PC7 tidak dipakai, sehingga dapat dipergunakan sebagai input atau output bit tunggal. Jika port A dipergunakan sebagai output dengan mode jabat tangan, maka kaki PC7

dipergunakan sebagai sinyal strobe dengan nama $\overline{OBF_A}$ (*output buffer full*) yang menandakan buffer pada port A sudah berisi data. Sinyal acknowledge $\overline{ACK_A}$ dihubungkan ke PC6, sementara PC4 dan PC5 dapat dipergunakan secara bebas. Untuk port B, sinyal jabat tangan adalah PC1 untuk IBF_B atau $\overline{OBF_B}$, PC2 sebagai $\overline{STB_B}$ atau $\overline{ACK_B}$, dan kaki PC0 untuk $INTR_B$.



(a) mode sederhana

(b) mode jabat tangan

(c) mode jabat tangan 2 arah

Gambar V-8. Sinyal yang digunakan pada ketiga mode operasi PPI 8255

Sebuah port pada dasarnya dipergunakan hanya sebagai input atau sebagai output saja,. Namun pada aplikasi tertentu sebuah port dapat berrfungsi sekaligus sebagai input dan sebagai output. Port B dan C hanya dapat berfungsi sebagai input atau output pada suatu saat, namun port A dapat berfungsi sebagai input dan output sekaligus (*bidirectional*). Untuk mode jabat tangan transfer 2 arah ini, fungsi kaki-kaki pada port C adalah sebagaimana terlihat dalam Gambar V-8c. Pada 8255, mode sederhana diberi nomor 0, sedangkan mode 1 adalah untuk transfer data jabat tangan, dan mode 2 untuk transfer data 2 arah dengan jabat tangan.

D. Pemrograman pada PPI 8255

Untuk memilih mode transfer data dan arah (input atau output) pada ketiga port, 8255 harus diprogram terlebih dahulu. Pemrograman PPI 8255 dilakukan dengan mengirimkan sebuah data berukuran 1 byte (disebut kata kendali atau *control word*) ke alamat CW. Data tersebut mengikuti format yang telah ditentukan. Format kata kendali untuk inisialisasi terlihat pada Gambar V-9a. Nilai dari setiap bit (D7-D0) harus disesuaikan dengan aplikasi dari 8255. Untuk inisialisasi, D7 selalu bernilai 1. Kombinasi D6-D5 serta D2 untuk memilih mode transfer data, sedangkan D4, D3, D1, dan D0 digunakan untuk menentukan arah transfer data. Contoh: jika Port A untuk input dengan mode sederhana, Port C atas (PC7-PC4) sebagai output sederhana, kemudian Port B sebagai output dengan jabat tangan, maka nilai kata kendalinya adalah 10010100 atau H94.

Selain untuk melakukan inisialisasi, pemrograman pada 8255 juga dapat dipergunakan untuk melakukan operasi bit set-reset pada Port C. Hal ini hanya berlaku apabila Port C dipergunakan sebagai output. Format kata kendali untuk operasi bit ini diberikan dalam Gambar V-9b. D7 bernilai 0, D6-D4 tidak dipergunakan (biasanya diberi nilai 0), kombinasi D3-D1 menentukan Port C nomor berapa yang akan dioperasikan, sedangkan D0 menunjukkan nilai yang akan diberikan (1 = set, 0 = reset). Contoh: untuk menset (memberi nilai 1) PC4, maka nilai kata kendalinya adalah 00001001 atau H09, sedangkan untuk mereset PC7 digunakan kata kendali 00001110 atau H0E.

Contoh pemrograman dalam bahasa assembly: pandang kasus berikut ini. Alamat port A adalah H340, port B H341, port C H342 dan alamat CW adalah H343. Port A digunakan untuk input dengan mode sederhana, Port C atas (PC7-PC4) sebagai output sederhana, dan Port B sebagai output dengan jabat tangan (mode 1).

Potongan program untuk inisialisasi 8255:

```
MOV DX, 343H      ; alamat CW
MOV AL, 94H       ; kata kendali mode operasi 8255
OUT DX, AL
```

Potongan program untuk membaca input dari Port A:

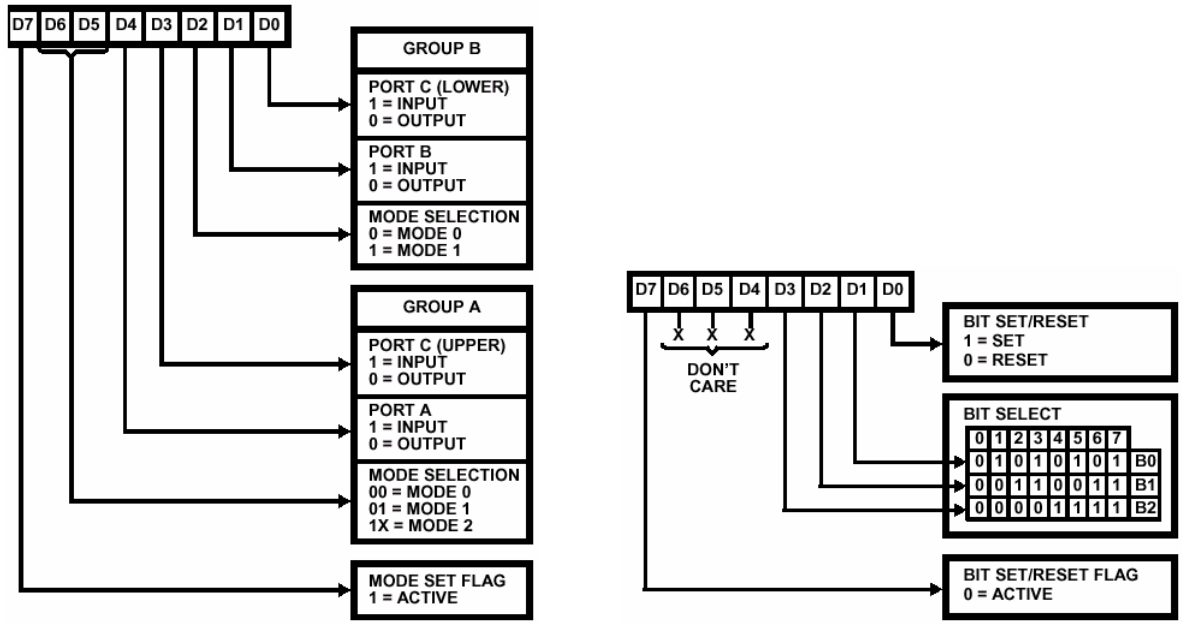
```
MOV DX, 340H      ; alamat port A
IN  AL, DX
```

Potongan program untuk menulis ke Port B:

```
MOV DX, 341H      ; alamat port B
MOV AL, 99H       ; data yang hendak dituliskan
OUT DX, AL
```

Potongan program untuk menset PC4 dan mereset PC7:

```
MOV DX, 343H      ; alamat CW
MOV AL, 9H        ; kata kendali set PC4
OUT DX, AL
MOV AL, 0EH       ; kata kendali reset PC7
OUT DX, AL
```



(a) untuk inisialisasi (b) untuk operasi bit pada Port C

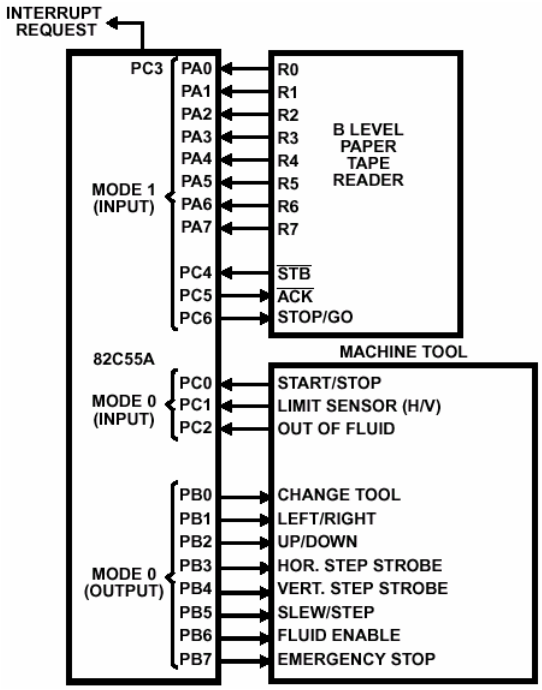
Gambar V-9. Format kata kendali 8255

E. Contoh Aplikasi PPI 8255

Pada contoh pertama, 8255 dipergunakan dalam mesin CNC (*computer numerical control*), yaitu mesin bubut terkontrol komputer yang dipergunakan untuk membuat berbagai komponen mesin: mur, baut, roda gigi, dan lain-lain. Instruksi untuk memotong dan melakukan hal lain diberikan di dalam pita kertas selebar ¾ inci yang dilobangi sesuai dengan intruksi yang diinginkan. Sebuah alat pembaca (*tape reader*) memutar pita tersebut dan mendeteksi lubang pada pita dengan mempergunakan sumber cahaya dan sensor.

Gambar V-10 menunjukkan koneksi yang dapat dilakukan dengan mempergunakan IC PPI 8255. Perhatikan bahwa yang ditunjukkan di dalam gambar tersebut hanya koneksi antara 8255 dengan *tape reader*, sedangkan koneksi antara mikroprosesor atau slot ekspansi dengan 8255 tidak ditampilkan. Port A dipergunakan sebagai input dari *tape reader* dengan mekanisme jabat tangan (mode 1), dengan demikian PC3-PC5 dipergunakan sebagai sinyal

jabat tangan. Karena PC6 bebas, maka akan dipergunakan untuk menyala-matikan *tape reader*, sehingga Port C tinggi berlaku sebagai output. Port B dipergunakan untuk mengatur gerakan mesin bubut, sehingga berlaku sebagai output dengan mode transfer sederhana. Mikroprosesor juga perlu untuk mendeteksi kemungkinan habisnya material yang hendak dipotong dan fluida untuk pendingin serta gerakan alat melebihi batas yang ditentukan. Oleh karena itu dipasang sensor yang dihubungkan dengan Port C rendah. Dalam hal ini, port tersebut menjadi input. Dengan demikian kata kendali yang dipergunakan untuk aplikasi ini adalah 10110001 atau HB1.



Gambar V-10. Antarmuka pada CNC menggunakan PPI 8255

REFERENSI

1. Douglas V. Hall, *Microprocessors and Interfacing: Programming and Hardware*, McGraw Hill Book Company, Singapore, 1987.
2. K.J., Breeding, *Microprocessor Systems Design Fundamentals*, Prentice Hall, New Jersey, 1995.
3. Agfianto Eko Putra, *Teknik Antarmuka Komputer: Konsep dan Aplikasi*, Graha Ilmu, Yogyakarta, 2002.
4. Dwi Sutadi, *I/O Bus & Motherboard*, Penerbit Andi, Yogyakarta, 2003

FEEDBACK

Untuk kesempurnaan Diktat ini, mohon halaman ini diisi dengan masukan, koreksi, dan saran yang bermanfaat untuk edisi berikutnya, kemudian disampaikan ke Penulis di Jurusan Teknik Fisika FT-UGM.

Koreksi:

No.	Bab/hal.	Yang salah	Seharusnya

Usulan tambahan materi:

No.	Bab/hal.	Sebaiknya ditambah dengan materi berikut

Komentar dan usulan lain

No.	Usulan

_____, _____

(_____)