Modul I/O

Fungsi modul I/O merupakan suatu entity didalam computer yang bertanggungjawab atas pengontrol sebuah perangkat eksternal atau lebih dan untuk pertukaran data antar perangkat tersebut dengan memori utama dan register-register CPU. Jadi modul I/O harus memiliki interface internal dengan computer (CPU dan main memory) dan interface eksternal dengan komputer (perangkat eksternal).

Fungsi Modul

Modul I/O merupakan suatu entity didalam komputer yang bertanggung jawab atas pengontrolan suatu perangkat eksternal atau lebih dan untuk pertukaran data antara perangkat-perangkat tersebut dengan memori utama dan/ atau register-register CPU. Jadi, modul I/O harus memiliki interface internal dengan komputer (CPU dan main memori) dan interface eksternal dengan komputer (perangkat eksternal).

Fungsi atau persyaratan utama bagi modul I/O dapat dibagi menjadi beberapa kategori seperti dibawah ini,

- Control dan timing
- Komunikasi CPU
- Komunikasi perangkat
- Data Buffering
- Deteksi error

Untuk mengkoordinasikan arus lalu-lintas antara sumber daya internal dan perangkat eksternal, fungsi I/O meliputi persyaratan *control dan timing*.

Apabila sistem menggunakan bus, maka setiap interaksi antara CPU dengan modul I/O akan melibatkan sebuah atau lebih arbitrasi bus.

Skenario yang telah disederhanakan seperti tersebut diatas menjelaskan juga bahwa modul I/O harus memiliki kemampuan untuk melaksanakan komunikasi dengan CPU dan perangkat eksternal. *Komunikasi CPU* meliputi:

- *Command Decoding*: Modul I/O menerima perintah-perintah dari CPU. Umumnya perintah-perintah ini dikirimkan sebagai signal bagi bus kontrol. Misalnya, sebuah

modul I/O untuk disk dapat menerima perintah-perintah berikut: READ SECTOR, WRITE SECTOR, SEEK nomor track, dan SCAN record ID. Kedua perintah terakhir meliputi parameter yang dikirimkan pada bus data.

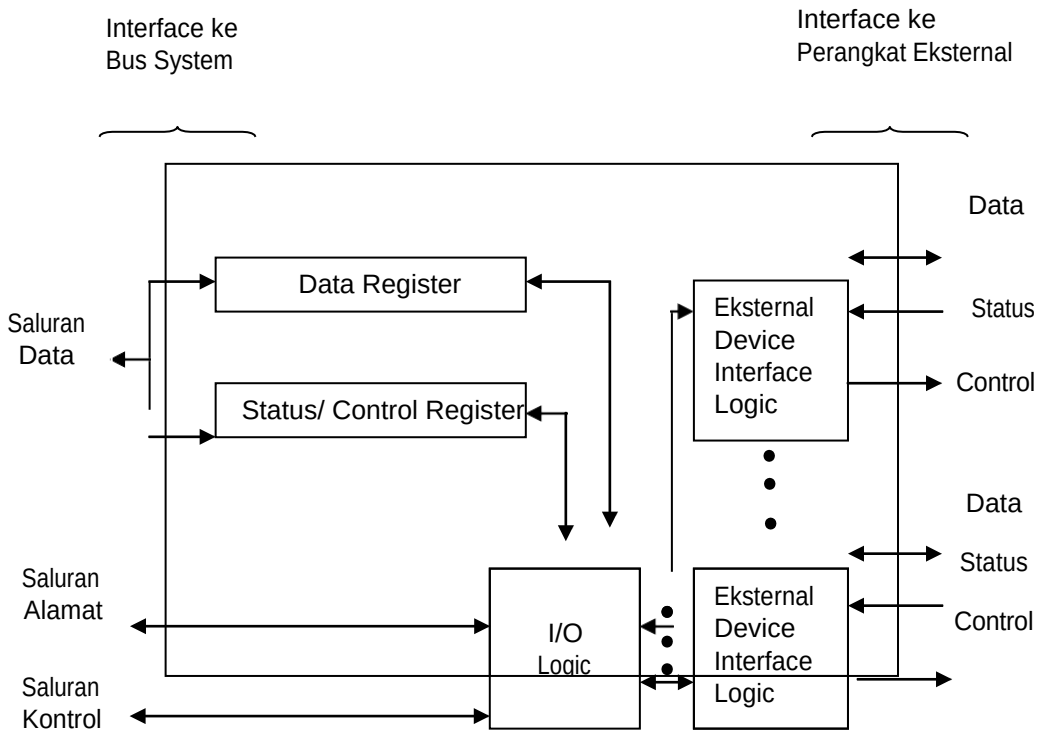
- *Data*: Data diperlukan antara CPU dengan modul I/O melalui bus data.
- *Status Reporting*: karena peripheral sangat lambat, maka status modul I/O perlu diketahui. Misalnya, bila sebuah modul I/O diminta untuk mengirimkan data ke CPU (read), maka mungkin modul tersebut berada dalam keadaan belum siap karena sedang melaksanakanperintah I/O lain. Kenyataan seperti ini perlu dilaporkan dengan menggunakan signal status. Signal status yang umum adalah BUSY dan READY.terdapat pula signal-signal status untuk melaporkan bermacam-macam kondisi error.
- *Address Recogntion*: Seperti halnya word memori memiliki alamat, demikian pula dengan perangkat I/O. Dengan demikian modul I/O harus mengetahui address unik seluruh peripheral yang dikontrolnya.

Disisi lain, modul I/O harus mampu membentuk komunikasi perangkat (*device communication)*. Komunikasi ini meliputi perintah, informasi status, dan data. Tugas utama modul I/O adalah *data buffering*.

Terakhir modul I/O sering kali harus bertanggung jawab atas deteksi error *(error detection)* dan pelaporan tentang terjadinya error terhadap CPU. Sebuah kelas error meliputi kesalahan mekanis dan elektris yang dilaporkan oleh perangkat (misalnya, kertas menggulung, track disk yang buruk). Kelas error lainnya terdiri perubahan pola bit yang tiba-tiba pada saat dikirimkan dari perangkat ke modul I/O. Beberapa kode error-detecting sering digunakan untuk mendeteksi error transmisi. Contohnya yang umum adalah penggunaan parity bit karakter data. Misalnya, kode karakter ASCII memakai 7 bit dari suatu byte. Bit kedelapan disetel sehingga jumlah total “bilangan satu”-nya dalam byte selalu genap (even parity) atau ganjil (odd parity). Ketika byte diterima, maka modul I/O memeriksa parity untuk menentukan apakah suatu error telah terjadi atau tidak.

Struktur Modul I/O

Modul I/O sangat berbeda dalam hal kompleksitas dan jumlah perangkat eksternal yang dikontrolnya. Disini kita hanya akan membahas masalah umumnya saja.



Gambar. Diagram blok dari suatu modul I/O

Gambar tersebut adalah diagram blok secara umum sebuah modul I/O. Modul dihubungkan dengan bagian-bagian komputer lainnya melalui saluran signal (misalnya, saluran bus sistem). Data yang dipindahkan ke modul dan dari modul di-buffer-kan dalam sebuah register data atau lebih. Mungkin juga terdapat sebuah register status atau lebih yang memberikan informasi status saat itu. Register status dapat juga berfungsi sebagai register kontrol, untuk menerima informasi kontrol secara detail dari CPU. Logik pada modul berinteraksi dengan CPU melalui sejumlah saluran kontrol. Saluran-saluran ini digunakan oleh CPU untuk memberikan perintah ke modul I/O. Beberapa saluran kontrol dapat digunakan oleh modul I/O (misalnya, untuk signal arbitrasi atau signal status). Modul juga dapat mengetahui dan menghasilkan alamat-alamat yang berkaitan dengan perangkat yang dikontrolnya. Setiap modul I/O memiliki

alamat yang unik, atau, apabila modul I/O mengontrol lebih dari sebuah perangkat eksternal, maka sekumpulan alamat yang unik. Terakhir modul I/O terdiri dari logik yang bersifat khusus bagi interface dengan setiap perangkat yang dikontrolnya.

Modul I/O berfungsi untuk memungkinkan CPU dapat mengetahui perangkat yang jumlahnya banyak dengan cara yang sederhana. Terdapat spektrum kemampuan yang dapat terjadi. Modul I/O dapat menyembunyikan detail pewaktuan, format, dan elektromekanis perangkat eksternal sehingga CPU dapat memberikan perintah pembacaan dan penulisan dengan musah, dan juga memungkinkan perintah-perintah membuka dan menutup file. Pada bentuk yang paling sederhananya, modul I/O masih dapat memberikan tugas pengontrolan perangkat dalam jumlah besar (misalnya, menggulung pita) yang dapat dketahui oleh CPU.

Modul I/O yang sering kalimendapatkan beban pengolahan yang detail,yang memberikan interface tingkat tinggi kepada CPU, dikenal sebagai I/O *channel* atau I/O *proccessor*. Modul I/O yang agak primitif dan membutuhkan kontrol detail sering kali disebut sebagai I/O *controller* atau *device controller*. Secara umum I/O controller dapat dilihat pada mikrokomputer, sedangkan I/O channel digunakan pada mainframe, sedangkan minikomputer menggunakan campuran keduannya.

Control dan Timing

Dalam periode waktu tertentu, CPU dapat berkomunikasi dengan satu buah atau lebih perangkat dengan pola yang tidak menentu, tergantung pada kebutuhan program I/O. Sumber daya internal, seperti memori utama dan sistem bus, harus dipakai bersama-sama oleh sejumlah aktivitas termasuk di antaranya I/O data. Dengan demikian, untuk mengkoordinasikan arus lalu-lintas antara sumber daya internal dan perangkat eksternal, fungsi I/O meliputi persyaratan *control dan timing*. Control dan Timing berfungsi untuk mengatur agar kecepatan transfer data yang berbeda-beda antar periferal dapat tersinkronisasi. Misalnya, kontrol pemindahan data dari sebuah perangkat eksternal ke CPU dapat meliputi langkah-langkah berikut ini:

1. CPU meminta modul I/O untuk memeriksa status perangkat yang terhubung.

2. Modul I/O memberikan jawabannya tentang status perangkat.
3. Bila perangkat sedang beroperasi dan berada dalam keadaan siap untuk mengirimkan, maka CPU meminta pemindahan data,dengan menggunakan perintah tertentu ke modul I/O.
4. Modul I/O akan memperoleh unit data (misalnya, 8 atau 16 bit) dari perangkat eksternal.
5. Data akan dipindahkan dari modul I/O ke CPU.

Buffering

Buffering adalah melembutkan lonjakan-lonjakan kebutuhan pengaksesan I/O, sehingga meningkatkan efisiensi dan kinerja sistem operasi. Terdapat beragam cara buffering, antar lain :

Single Buffering

Merupakan teknik paling sederhana. Ketika proses memberi perintah untuk perangkat I/O, sistem operasi menyediakan buffer memori utama sistem untuk operasi. Untuk perangkat berorientasi blok. Transfer masukan dibuat ke buffer sistem. Ketika transfer selesai,proses memindahkan blok ke ruang pemakai dan segera meminta blok lain. Teknik ini disebut reading ahead atau anticipated input. Teknik ini dilakukan dengan harapan blok akan segera diperlukan. Untuk banyak tipe komputasi, asumsi ini berlaku. Hanya di akhir pemrosesan maka blok yang dibaca tidak diperlukan.

Keunggulan :

Pendekatan ini umumnya meningkatkan kecepatan dibanding tanpa buffering. Proses pemakai dapat memproses blok data sementara blok berikutnya sedang dibaca. Sistem operasi dapat menswap keluar proses karena operasi masukan berada di memori sistem bukan memori proses pemakai.

Kelemahan :

- Merumitkan sistem operasi karena harus mencatat pemberian buffer-buffer sistem ke proses pemakai.
- Logika swapping juga dipengaruhi.

Jika operasi I/O melibatkan disk untuk swapping, maka membuat antrian penulisan ke disk yang sama yang digunakan untuk swap out proses. Untuk menswap proses dan melepas memori utama tidak dapat dimulai sampai operasi I/O selesai, dimana waktu swapping ke disk tidak bagus untuk dilaksanakan.Buffering keluaran serupa buffering masukan. Ketika data transmisi, data lebih dulu dikopi dari ruang pemakai ke buffer sistem. Proses pengirim menjadi bebas untuk melanjutkan eksekusi berikutnya atau di swap ke disk jika perlu.Untuk perangkat berorientasi aliran karakter. Single buffering dapat diterapkan dengan dua mode, yaitu :

1. Mode line at a time.

Cocok untuk terminal mode gulung (scroll terminal atau dumb terminal). Masukan pemakai adalah satu baris per waktu dengan enter menandai akhir baris. Keluaran terminal juga serupa, yaitu satu baris per waktu. Contoh mode ini adalah printer.

Buffer digunakan untuk menyimpan satu baris tunggal. Proses pemakai ditunda selama masukan, menunggu kedatangan satu baris seluruhnya. Untuk keluaran, proses pemakai menempatkan satu baris keluaran pada buffer dan melanjutkan pemrosesan. Proses tidak perlu suspend kecuali bila baris kedua dikirim sebelum buffer dikosongkan.

2. Mode byte at a time.

Operasi ini cocok untuk terminal mode form, dimana tiap ketikan adalah penting dan untuk peripheral lain seperti sensor dan pengendali.

Double Buffering

Peningkatan dapat dibuat dengan dua buffer sistem.Proses dapat ditransfer ke/dari satu buffer sementara sistem operasi mengosongkan (atau mengisi) buffer lain. Teknik ini disebut double buffering atau buffer swapping. Double buffering menjamin proses tidak menunggu operasi I/O. Peningkatan ini harus dibayar dengan peningkatan kompleksitas. Untuk berorientasi aliran karakter, double buffering mempunyai 2 mode alternatif, yaitu :

1. Mode line at a time.

Proses pemakai tidak perlu ditunda untuk I/O kecuali proses secepatnya mengosongkan buffer ganda.

2. Mode byte at a time.

Buffer ganda tidak memberi keunggulan berarti atas buffer tunggal. Double buffering mengikuti model producer-consumer.

Circular Buffering

Seharusnya melembutkan aliran data antara perangkat I/O dan proses. Jika kinerja proses tertentu menjadi fokus kita, maka kita ingin agar operasi I/O mengikuti proses. Double buffering tidak mencukupi jika proses melakukan operasi I/O yang berturutan dengan cepat. Masalah sering dapat dihindari denga menggunakan lebih dari dua buffer. Ketika lebih dari dua buffer yang digunakan, kumpulan buffer itu sendiri diacu sebagai circulat buffer. Tiap buffer individu adalah satu unit di circular buffer.

Programmed I/O

Pada I/O terprogram, data saling dipertukarkan antara CPU dengan modul I/O. CPU mengeksekusi program yang memberikan operasi I/O kepada CPU secara langsung, termasuk status perangkat pengindra, pengiriman perintah pembacaan atau penulisan, dan pemindahan data. Ketika CPU mengeluarkan perintah ke modul I/O, maka CPU harus menunggu sampai operasi I/O selesai. Apabila CPU lebih cepat dibandingkan modul I/O, maka hal ini akan membuang-buang waktu CPU. Dengan menggunakan interupt driven I/O, CPU mengeluarkan perintah I/O, dilanjutkan dengan mengeksekusi intstruksi-instruksi lainnya, dan diinterupsi oleh modul I/O apabila instruksi-instruksi tersebut telah selesai dilaksanakan. Dengan menggunakan I/O terprogram dan I/O interupt, maka CPU bertanggung jawab atas pengeluaran data dan memori utama untuk keperluan output dan penyimpanan data di dalam memori utama untuk keperluan input.

Bentuk lain dari programmed I/O adalah direct I/O. Pada teknik ini CPU tidak melakukan pemeriksaan status device I/O.

- **Direct I/O**

Sebagai contoh instrument berbasis computer dengan panel saklar dan indicator. Saklar berfungsi untuk pemilihan mode operasi oleh operator instrument. Indikator diimplementasikan menggunakan LED, yang berfungsi sebagai indicator status instrument.

Rangkaian tersebut menggunakan prosesor 8-bit dengan ruang alamat I/O terpisah. Untuk mengisolasi antara bus data dengan saklar diperlukan rangkaian penyangga. Penyangga ini diimplementasikan dengan octal buffer, yang dapat menampung delapan saklar. Bagian terpenting lainnya adalah address decoder, yang meng-enable hanya jika keluaran address decoder hanya dilewatkan pada saat RD* rendah. Selanjutnya, melihat apa yang terjadi ketika CPU mengeksekusi instruksi "INPUT". Dianggap alamat operand berhubungan dengan port input yang ditetapkan.

Pada awal I/O machine cycle keluaran address decoder adalah rendah. Kemudian, RD* diaktifkan dan SEL1* juga rendah. Selanjutnya keluaran penyangga bus di-enable.

Setiap rangkaian peyangga mengemudikan setiap jalur dari jalur bus data CPU untuk memberikan nilai tinggi atau rendah. Hal ini tergantung pada posisi saklar yang bersangkutan.

- **Polled I/O**

Pada polled I/O terlebih dahulu di perlukan pemeriksaan status device I/O. Setelah system diinisialisasi, program I/O yang berhubungan, yaitu I/O driver, membaca status device I/O. Kemudian menguji bit status untuk menentukan kondisi operasional device. Apabila device tidak siap, CPU dapat melompat ke tugas (task) yang lain. Beberapa saat kemudian, setelah beberapa interval waktu, mengulangi proses sepertidiuraikan di atas. Interval waktu ini tergantung pada kecepatan divais. Sebagai contoh printer dengan kecepatan 10 cps lamanya adalah 100 ms.

Pada langkah di atas apabila tidak ada task yang dikerjakan, CPU segera mengulang kembali untuk membaca dan memeriksa status device. Program I/O keluar "wait loop". Sekarang program melakukan transfer data atau bisa disebut melayani device. I/O Interface Controller hubungan device I/O dengan system computer disediakan melalui rangkaian standar dalam bentuk chip LSI (large scale integration). Rangkaian ini dikenal dengan I/O interface controller, pheripheral interface adapter, dan lain-lain. Dengan adanya komponen ini dapat memudahkan menghubungkan device I/O yang memiliki karakteristik berbeda dengan system bus umum, tanpa atau meminimumkan penggunaan perangkat keras yang khusus. Antarmuka yang paling sederhana adalah register penyangga satu word yang bertindak sebagai port I/O.

Interrupt Driven I/O

Permasalahan yang ditimbulkan oleh polled (programmed) I/O adalah waktu CPU terbuang untuk pengujian status. Interrupt-driven I/O menghilangkan permasalahan ini. Sebelum membahas lebih lanjut, berikut ini akan di diskusikan tentang interrupt-driven I/O.

Prinsip mekanisme transfer data pada interrupt-driven I/O sama dengan programmed I/O, yaitu CPU tetap mempertukarkan data dengan device I/O melalui beberapa register CPU. Maka antarmuka I/O menyediakan control, status, dan port data. Perbedaan yang prinsip adalah siapa yang bertanggung jawab untuk berinisiatif melakukan transfer data. Pada programmed I/O tanggung jawab pada CPU. Dengan kata lain, program I/O harus sering melakukan pemeriksaan (scan) status device I/O untuk menentukan apakah device siap melakukan transfer data. Pada interrupt-driven I/O, data transfer dimulai (atasinisiatif) device I/O, yang menggunakan mekanisme interupsi untuk memberitahukan CPU tentang kesiapannya. Mekanisme ini menghilangkan beban scanning status. Device I/O juga dapat menggunakan mekanisme interupsi untuk keperluan yang lain. Sebagai contoh untuk menarik perhatian CPU pada saat terjadi kesalahan/ kerusakkan, atau untuk menunjukkan selesainya operasi lokal.

Mekanisme interupsi tidak hanya penting untuk I/O, tetapi juga untuk yang lain. Interupsi yang disebabkan oleh sumber-sumber eksternal dapat maskable maupun non maskable. Maskable adalah interupsi yang disebabkan oleh device I/O. Disebut maskable karena interupsi ini dicegah melalui register status CPU. Non maskable interrupt, tidak dapat dicegah (disable), karena selalu memiliki prioritas yang lebih tinggi dibanding maskable. Fungsi utamanya adalah untuk memberitahu CPU apabila ada event-event yang berbahaya, seperti adanya kerusakkan catu daya atau error pada memori.

Biasanya, sumber interupsi eksternal tidak mengemudikan jalur interupsi CPU secara langsung. Tetapi melalui rangkian eksternal yang kompleksitasnya tergantung pada tipe CPU dan penggunaannya. Interupsi juga dapat disebabkan event-event internal, interupsi ini disebut dengan traps. Traps ini dibangkitkan selama eksekusi program yang disebabkan oleh hasil dari beberapa kondisi yang tidak diharapkan kesengajaannya.

Tanpa memperhatikan tipenya, apabila menerima interupsi, CPU akan:

1. Menagguhkan eksekusi program yang sedangberlangsung.
2. Menyimpan statusnya.
3. Melompatke interrupt service routine (ISR).
4. Kembali untuk melanjutkan eksekusi program yang diinterupsi.

- **Struktur Interupsi pada Prosesor**

Dari titik pandang antar muka eksternal, sebagian besar CPU menggunakan satu dari dua pola yang ditunjukkan pada gambar di bawahini.

a. Interupsi maskable dan nonmaskable dengan jalur terpisah.
b. Interupsi maskable dan nonmaskable dengan jalur bersama.

Pola pertama berbasis pada dua jalur permintaan interupsi (interrupt request) yang terpisah, satu untuk interupsi maskable (INTREQ) dan satu lagi untuk nonmaskable (NMINT). Jalur INTERQ biasanya berpasangan dengan jalur interrupt acknowledge (INTACK). Melalui jalur ini CPU menjawab penerimaan permintaan

interupsi dan menginstrusksikan device I/O untuk meletakkan kode alamat ke bus data. Kode alamat menunjukkan CPU ke ISR yang mentansfer data atau melakukan pelayanan yang lainya atas nama device I/O. INTERQ dan INTACK dapat dipandang sebagai pasangan jalur jabat tangan (handshaking). Beberapa CPU tidak menyediakan sinyal jawanan interupsi pada jalur yang terpisah, tetapi melakukan enscodesiklus ini pada jalur status. Dalam kasus ini sinyal INTACK diperoleh dengan cara mengkode jalur status CPU. Interupsi non maskable tidak memerlukan acknowledge karena interupsi ini selalu diterima dan karena tidak ada kode alamat yang diberikan secara eksternal. Dengan demikian tidak diperlukan jalur acknowledge untuk interupsi ini. Contoh prosesor yang menggunakan pola ini adalah 8086 dan 8088.

- **Pemrosesan Interupsi**

Sebelum masuk ke interrupt service routine (ISR), CPU harusmenyiapkan :

1. Program Counter (PC)
2. Register status
3. Informasi yang berkenaandengan program yang diinterupsi

Kumpulan informasi di atas yang selalu berubah dari waktu ke waktu disebut dengan context. Menyimpan context program yang ditunda merupakan kunci untuk menjamin kebenaran lanjutan eksekusi program yang diinterupsi.Disamping, ISR sendiri harus melakukan aksi tertentu dan kembali ke program yang di interupsi dengan semestinya. Urutan event yang dilakukan oleh CPU apabila menerima interupsi dan pelayanan interupsi eksternal ditunjukkan pada gambar berikut.

Setiap jalur INTR yang berhubungan dengan permintaan interupsi yang independent diberi prioritas yang unik.Sumber interupsi dapat dengan segera dikenali CPU, sehingga mengurangi untuk scan ke port I/O.

Jalur INTR (Interrupt Request) tunggal dipakai oleh semua port I/O. Untuk menjawab interupsi, CPU harus melakukan scan pada semua divais I/O untuk menentukan sumber interupasi. Hal ini dilaksanakan dengan mengaktifkan INTACK yang dihubungkan secara daisy chain keseluruh device I/O.

Klasifikasi Umum Perangkat I/O

Pendapat orang-orang mengenai I/O berbeda-beda. Seorang insinyur mungkin akan memandang perangkat keras I/O sebagai kumpulan chip-chip, kabel-kabel, catu daya, dan komponen fisik lainnya yang membangun perangkat keras ini. Seorang programmer akan memandangnya sebagai antarmuka yang disediakan oleh perangkat lunak atau perintah yang diterima perangkat keras, fungsi yang dikerjakannya, dan *error* yang ditimbulkan.Perangkat I/O dapat dibagi secara umum menjadi dua kategori, yaitu: perangkat blok (*block devices*), dan perangkat karakter (*character devices*). Perangkat blok menyimpan informasi dalam sebuah blok yang ukurannya tertentu, dan memiliki alamat masing-masing. Umumnya blok berukuran antara 512 bytes sampai 32.768 bytes.

Keuntungan dari perangkat blok ini ialah mampu membaca atau menulis setiap blok secara independen. Disk merupakan contoh perangkat blok yang paling banyak digunakan.Tipe lain perangkat I/O ialah perangkat karakter. Perangkat karakter mengirim atau menerima sebarisan karakter, tanpa menghiraukan struktur blok. Tipe ini tidak memiliki alamat, dan tidak memiliki kemampuan mencari (*seek*). *Printer* dan antarmuka jaringan merupakan contoh perangkat jenis ini.Pembagian ini tidaklah sempurna. Beberapa perangkat tidak memenuhi kriteria tersebut. Contohnya: *clock* yang tidak memiliki alamat dan juga tidak mengirim dan menerima barisan karakter. Yang ia lakukan hanya menimbulkan interupsi dalam jangka waktu tertentu.

Klasifikasi Perangkat I/O

Perangkat I/O dapat dikelompokkan berdasarkan :

a. Sifat aliran datanya, yang terbagi atas :
   - Perangkat berorientasi blok.

     Yaitu menyimpan, menerima, dan mengirim informasi sebagai blok-blok berukuran tetap yang berukuran 128 sampai 1024 byte dan memiliki alamat tersendiri, sehingga memungkinkan membaca atau menulis blok-blok secara

independen, yaitu dapat membaca atau menulis sembarang blok tanpa harus melewati blok-blok lain. Contoh : disk,tape,CD ROM,optical disk.

- Perangkat berorientasi aliran karakter.
  Perangkat yang menerima, dan mengirimkan aliran karakter tanpa membentuk suatu struktur blok. Contoh: terminal, line printer, pita kertas, kartu-kartu berlubang, interface jaringan, mouse.

b. Sasaran komunikasi, yang terbagi atas :

- Perangkat yang terbaca oleh manusia.
  Perangkat yang digunakan untuk berkomunikasi dengan manusia.
  Contoh : VDT (video display terminal) : monitor, keyboard, mouse.

- Perangkat yang terbaca oleh mesin.
  Perangkat yang digunakan untuk berkomunikasi dengan perangkat elektronik.
  Contoh : Disk dan tape, sensor, controller.

- Perangkat komunikasi.
  Perangkat yang digunakan untuk komunikasi dengan perangkat jarak jauh.
  Contoh : Modem.

Faktor-faktor yang membedakan antar perangkat :

- Kecepatan transmisi data (data rate).
- Jenis aplikasi yang digunakan.
- Tingkat kerumitan dalam pengendalian.
- Besarnya unit yang ditransfer.
- Representasi atau perwujudan data.
- Kondisi-kondisi kesalahan.

Prinsip Manajemen Perangkat I/O

Terdapat dua sasaran perancangan I/O, yaitu :

a. Efisiensi.

Aspek penting karena operasi I/O sering menimbulkan bottleneck.

b. Generalitas (device independence).

Manajemen perangkat I/O selain berkaitan dengan simplisitas dan bebas kesalahan, juga menangani perangkat secara seragam baik dari cara proses memandang maupun cara sistem operasi mengelola perangkat dan operasi I/O.

Software diorganisasikan berlapis. Lapisan bawah berurusan menyembunyikan kerumitan perangkat keras untuk lapisan-lapisan lebih atas. Lapisan lebih atas berurusan memberi antar muka yang bagus, bersih, nyaman dan seragam ke pemakai.

Masalah-masalah manajemen I/O adalah :

a. Penamaan yang seragam (uniform naming).
Nama berkas atau perangkat adalah string atau integer, tidak bergantung pada perangkat sama sekali.

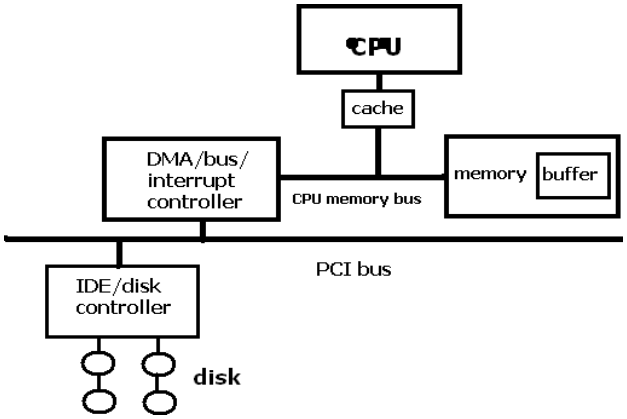
b. Penanganan kesalahan (error handling).
Umumnya penanganan kesalahan ditangani sedekat mungkin dengan perangkat keras.

c. Transfer sinkron vs asinkron.
Kebanyakan I/O adalah asinkron. Pemroses mulai transfer dan mengabaikan untuk melakukan kerja lain sampai interupsi tiba. Program pemakai sangat lebih mudah ditulis jika operasi I/O berorientasi blok. Setelah perintah read, program kemudian ditunda secara otomatis sampai data tersedia di buffer.

d. Sharable vs dedicated.
Beberapa perangk dapat dipakai bersama seperti disk, tapi ada juga perangkat yang hanya satu pemakai yang dibolehkan memakai pada satu saat. Contoh : printer.

Direct Memory Access

Dirrect Memory Acces atau biasa disebut DMA adalah suatu alat pengendali khusus yang disediakan untuk memungkinkan transfer blok data langsung antar perangkat eksternal dan memori utama, tanpa interversi terus menerus dari prosesor.Banyak sistem perangkat keras menggunakan DMA termasuk pengendali disk drive, VGA, LAN card dan Sound Card.DMA juga digunakan untuk interaksi chip transfer data dalam prosesor multi core, terutama pada system multiprosesor, di mana elemen pemrosesan dilengkapi dengan memori local (biasa disebut scratchpad memori) dan DMA digunakan untuk mentransfer data antara memori lokal dan memori utama.Komputer yang memiliki saluran DMA dapat mentransfer datanya ke suatu perangkat dengan proses yang jauh lebih kecil dari pada computer yang tidak memiliki saluran DMA.Demikian pula di dalam elemen pemrosesan multi-core prosesor yang dapat mentransfer data, dari memori lokal tanpa menduduki prosesor waktu dan memungkinkan komputasi dan transfer data yang pasti.



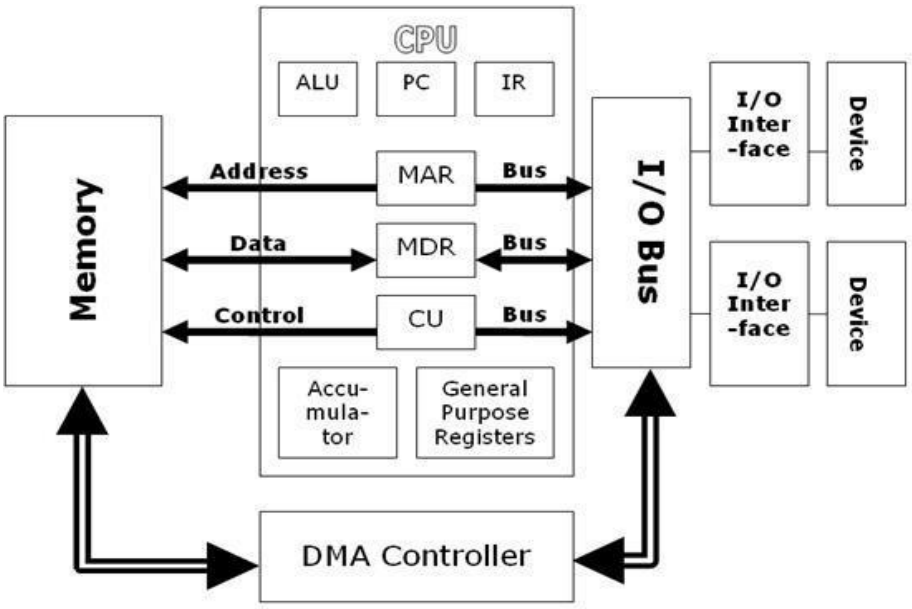
- Contoh Gambar DMA

Transfer DMA

Untuk memulai sebuah transfer DMA, *host* akan menuliskan sebuah DMA *command block* yang berisi *pointer* yang menunjuk ke sumber transfer, *pointer* yang menunjuk ke tujuan/destinasi transfer, dan jumlah *byte* yang di transfer ke memori CPU.CPU kemudian menuliskan alamat *command block* ini ke DMA *controller*.sehingga DMA *controller* dapat langsung mengperasikan *bus* memori secara langsung dengan menempatkan alamat-alamat pada *bus* tersebut untuk melakukan transfer tanpa bantuan CPU.Tiga langkah dalam transfer DMA :

1) Prosesor menyiapkan DMA transfer dengan menyediakan data-data dari devise, operasi yang akan ditampilkan, alamat memori yang menjadi sumber dan tujuan data, dan banyaknya byte yang di transfer.
2) DMA *controller* memulai operasi (menyiapkan bus, menyediakan alamat, menulis dan membaca data), sampai seluruh blok sudah di transfer.
3) DMA *controller* meng-interupsi prosesor, dimana selanjutnya akan ditentukan tindakan berikutnya.

- Contoh transfer data dengan menggunakan DMA *controller*



Dalam skenario ini blok besar data yang akan ditransfer menggunakan DMA hardware.CPU menerima interupsi mengawali DMA hardware dengan

- Sebuah memori mulai alamat
- Jumlah data yang di transfer
- Perangkat yang digunakan (harddisk, CD, DVD, dll)
- arah transfer (input atau output)

DMA controller kemudian mentransfer data.setelah selesai controller memberitahu CPU bahwa transfer telah selesai.

Pada dasarnya DMA mempunyai dua metode yang berbeda dalam mentransfer data,Metode yang ertama ialah metode yang sangat baku dan sederhana disebut *HALT* , atau *Burst Mode DMA*,karena pengendali DMA memegang control dari system bus dan mentransfer semua blok data kea tau dari memori pada *single burst*.selagi transfer masih dalam proses, system mikroprosessoe di-set *idle,* tidak melakukan intruksi operasi untuk menjaga internal *register*. Tipe operasi DMA seperti ini ada pada kebanyakan komputer.

Metode yang kedua adalah mengikut sertakan pengendali DMA untuk memegang control dari system bus untuk jangka waktu yang lebih pendek pada periode dimana mikroprosessoe sibuk dengan operasi internal dan tidak membutuhkan akses ke system bus. Metode DMA ini disebut dengan *cycle stealing mode*.*Cycle stealing modeDMA* lebih kompleks untuk diimplementasikan dibandingkan HALT DMA, karena pengendali DMA harus mempunyai kepintaran untuk merasakan waktu pada saat sistem bus terbuka.