**ORGANISASI DAN ARSITEKTUR KOMPUTER**

****

Satria Alief Putra Hidayat – 140810210051

Prames Ray Lapian – 140810210059

UNIVERSITAS PADJADJARAN

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

Program Studi INFORMATIKA

2022

**BAB 7**

**INPUT/OUTPUT**

**7.1 Perangkat Eksternal I/O**

Operasi dicapai melalui berbagai macam perangkat eksternal yang menyediakan sarana pertukaran data antara lingkungan eksternal dan komputer. Perangkat eksternal terhubung ke komputer melalui tautan ke modul I/O (Gambar 7.1). Tautan digunakan untuk bertukar kontrol, status, dan data antara modul I/O dan perangkat eksternal. Perangkat eksternal yang terhubung ke modul I/O sering disebut sebagai perangkat periferal atau, sederhananya, periferal. Kami secara luas dapat mengklasifikasikan perangkat eksternal ke dalam tiga kategori:

* Dapat dibaca manusia: Cocok untuk berkomunikasi dengan pengguna komputer;
* Dapat dibaca mesin: Cocok untuk berkomunikasi dengan peralatan;
* Komunikasi: Cocok untuk berkomunikasi dengan perangkat jarak jauh.

**Papan Ketik/Monitor**

Cara paling umum untuk interaksi komputer/pengguna adalah pengaturan keyboard/monitor. Pengguna memberikan input melalui keyboard, input tersebut kemudian ditransmisikan ke komputer dan juga dapat ditampilkan di monitor. Selain itu, monitor menampilkan data yang disediakan oleh komputer.

**Disk Drive**

Disk drive berisi elektronik untuk bertukar data, kontrol, dan sinyal status dengan modul I/O ditambah elektronik untuk mengontrol mekanisme baca/tulis disk. Dalam disk kepala tetap, transduser mampu mengubah antara pola magnetik pada permukaan disk yang bergerak dan bit dalam buffer perangkat (Gambar 7.2). Disk kepala yang bergerak juga harus dapat menyebabkan lengan disk bergerak secara radial masuk dan keluar melintasi permukaan disk.

**7.2 Modul I/O**

**Fungsi Modul**

Fungsi atau persyaratan utama untuk modul I/O jatuh ke dalam berikut ini:

kategori:

* Kontrol dan waktu
* Komunikasi prosesor
* Komunikasi perangkat
* Penyangga data
* Deteksi kesalahan

Selama periode waktu tertentu, prosesor dapat berkomunikasi dengan satu atau lebih perangkat eksternal dalam pola yang tidak terduga, tergantung pada kebutuhan program untuk I/O. Sumber daya internal, seperti memori utama dan bus sistem, harus dibagi di antara sejumlah aktivitas, termasuk I/O data. Dengan demikian, fungsi I/O mencakup persyaratan **kontrol dan pengaturan waktu**, untuk mengoordinasikan arus lalu lintas antara sumber daya internal dan perangkat eksternal.

Skenario sederhana sebelumnya juga menggambarkan bahwa modul I/O harus berkomunikasi dengan prosesor dan dengan perangkat eksternal. Komunikasi prosesor melibatkan hal-hal berikut:

* Penguraian kode perintah: Modul I/O menerima perintah dari prosesor, biasanya dikirim sebagai sinyal pada bus kontrol.
* Data: Data dipertukarkan antara prosesor dan modul I/O melalui bus data.
* Pelaporan status: Karena periferal sangat lambat, penting untuk mengetahui status modul I/O.
* Pengenalan alamat: Sama seperti setiap kata dari memori memiliki alamat, demikian juga setiap perangkat I/O. Jadi, modul I/O harus mengenali satu alamat unik untuk setiap periferal yang dikontrolnya

Di sisi lain, modul I/O harus mampu melakukan komunikasi perangkat. Komunikasi ini melibatkan perintah, informasi status, dan data

Tugas penting dari modul I/O adalah buffering data. Kecepatan transfer masuk dan keluar dari memori utama atau prosesor cukup tinggi, sementara kecepatannya jauh lebih rendah untuk banyak perangkat periferal dan mencakup rentang yang luas. Dengan kata lain, jika perangkat I/O beroperasi pada kecepatan yang lebih tinggi daripada kecepatan akses memori, maka modul I/O melakukan operasi buffering yang diperlukan.

Akhirnya, modul I/O sering bertanggung jawab untuk deteksi kesalahan dan yang selanjutnya melaporkan kesalahan ke prosesor.

**Struktur Modul I/O '**

Modul I/O sangat bervariasi dalam kompleksitas dan jumlah perangkat eksternal yang dikendalikan. Gambar 7.3 menyediakan diagram blok umum dari modul I/O. Modul terhubung ke seluruh komputer melalui serangkaian jalur sinyal (mis., jalur bus sistem). Data yang ditransfer ke dan dari modul di-buffer dalam satu atau lebih register data. Mungkin juga ada satu atau lebih register status yang menyediakan informasi status terkini. Sebuah register status juga dapat berfungsi sebagai register kontrol, untuk menerima informasi kontrol rinci dari prosesor. Logika di dalam modul berinteraksi dengan prosesor melalui serangkaian garis kontrol. Prosesor menggunakan jalur kontrol untuk mengeluarkan perintah ke modul I/O. Beberapa jalur kontrol dapat digunakan oleh modul I/O (misalnya, untuk arbitrase dan sinyal status). Modul juga harus dapat mengenali dan menghasilkan alamat yang terkait dengan perangkat yang dikontrolnya. Setiap modul I/O memiliki alamat unik atau, jika mengontrol lebih dari satu perangkat eksternal, satu set alamat unik. Terakhir, modul I/O berisi logika khusus untuk antarmuka dengan setiap perangkat yang dikontrolnya.

**7.3 I/O Terprogram**

**Gambaran Umum I/O Terprogram**

Ketika prosesor mengeksekusi program dan menemukan instruksi yang berkaitan dengan I/O, prosesor mengeksekusi instruksi tersebut dengan mengeluarkan perintah ke modul I/O yang sesuai. Dengan I/O terprogram, modul I/O akan melakukan tindakan yang diminta dan kemudian mengatur bit yang sesuai dalam register status I/O. Modul I/O tidak mengambil tindakan lebih lanjut untuk memperingatkan prosesor. Secara khusus, itu tidak mengganggu prosesor. Oleh karena itu, merupakan tanggung jawab prosesor untuk secara berkala memeriksa status modul I/O sampai ditemukan bahwa operasi telah selesai.

**Perintah I/O**

Untuk mengeksekusi instruksi terkait I/O, prosesor mengeluarkan alamat, menentukan modul I/O tertentu dan perangkat eksternal, dan perintah I/O. Ada empat jenis perintah I/O yang mungkin diterima modul I/O ketika ditangani oleh prosesor:

* Kontrol: Digunakan untuk mengaktifkan periferal dan memberi tahu apa yang harus dilakukan. Misalnya, unit pita magnetik dapat diinstruksikan untuk memundurkan atau memajukan satu rekaman. Perintah-perintah ini disesuaikan dengan jenis perangkat periferal tertentu.
* Test: Digunakan untuk menguji berbagai kondisi status yang terkait dengan modul I/O dan periferalnya. Prosesor akan ingin mengetahui bahwa periferal yang diinginkan dihidupkan dan tersedia untuk digunakan. Ia juga ingin mengetahui apakah operasi I/O terbaru telah selesai dan jika ada kesalahan yang terjadi.
* Baca: Modul I/O memperoleh item data dari periferal dan menempatkannya di buffer internal. Prosesor kemudian dapat memperoleh item data dengan meminta modul I/O menempatkannya pada bus data.
* Write: Modul I/O mengambil item data (byte atau word) dari bus data dan kemudian mengirimkan item data tersebut ke periferal.

Instruksi I/O

Dengan I/O terprogram, ada korespondensi yang erat antara instruksi terkait I/O yang diambil prosesor dari memori dan perintah I/O yang dikeluarkan prosesor ke modul I/O untuk mengeksekusi instruksi. Artinya, instruksi mudah dipetakan ke dalam perintah I/O, dan seringkali ada hubungan satu-ke-satu yang sederhana. Bentuk instruksi tergantung pada cara perangkat eksternal ditangani.

Biasanya, akan ada banyak perangkat I/O yang terhubung melalui modul I/O ke sistem. Setiap perangkat diberi pengenal atau alamat unik. Ketika prosesor mengeluarkan perintah I/O, perintah tersebut berisi alamat perangkat yang diinginkan. Jadi, setiap modul I/O harus menginterpretasikan baris alamat untuk menentukan apakah perintah itu untuk dirinya sendiri.

**7.4 I/O Berbasis Interupsi**

Masalah dengan I/O terprogram adalah bahwa prosesor harus menunggu lama agar modul I/O yang bersangkutan siap untuk penerimaan atau transmisi data. Prosesor, saat menunggu, harus berulang kali menginterogasi status modul I/O. Akibatnya, tingkat kinerja seluruh sistem sangat menurun.

Alternatifnya adalah prosesor mengeluarkan perintah I/O ke modul dan kemudian melanjutkan untuk melakukan beberapa pekerjaan berguna lainnya. Modul I/O kemudian akan menginterupsi prosesor untuk meminta layanan ketika siap untuk bertukar data dengan prosesor. Prosesor kemudian mengeksekusi transfer data, seperti sebelumnya, dan kemudian melanjutkan pemrosesan sebelumnya.

**Pemrosesan Interupsi**

Mari kita pertimbangkan peran prosesor dalam I/O yang digerakkan oleh interupsi secara lebih rinci. Terjadinya interupsi memicu sejumlah kejadian, baik pada perangkat keras prosesor maupun pada perangkat lunak. Gambar 7.6 menunjukkan urutan yang khas. Ketika perangkat I/O menyelesaikan operasi I/O, urutan kejadian perangkat keras berikut terjadi:

1. Perangkat mengeluarkan sinyal interupsi ke prosesor.
2. Prosesor menyelesaikan eksekusi instruksi saat ini sebelum menanggapi interupsi, seperti yang ditunjukkan pada Gambar 3.9.
3. Prosesor menguji interupsi, menentukan bahwa ada interupsi, dan mengirimkan sinyal pengakuan ke perangkat yang mengeluarkan interupsi. Pengakuan memungkinkan perangkat untuk menghapus sinyal interupsi.
4. Prosesor sekarang perlu bersiap untuk mentransfer kontrol ke rutin interupsi. Untuk memulai, perlu menyimpan informasi yang diperlukan untuk melanjutkan program saat ini pada titik interupsi. Informasi minimum yang diperlukan adalah (a) status prosesor, yang terdapat dalam register yang disebut program status word (PSW); dan (b) lokasi instruksi selanjutnya yang akan dieksekusi, yang terdapat pada program counter. Ini dapat didorong ke tumpukan kontrol sistem.2
5. Prosesor sekarang memuat penghitung program dengan lokasi entri dari program penanganan interupsi yang akan merespon interupsi ini. Tergantung pada arsitektur komputer dan desain sistem operasi, mungkin ada satu program; satu program untuk setiap jenis interupsi; atau satu program untuk setiap perangkat dan setiap jenis interupsi. Jika ada lebih dari satu rutin penanganan interupsi, prosesor harus menentukan mana yang akan dipanggil. Informasi ini mungkin telah dimasukkan dalam sinyal interupsi asli, atau prosesor mungkin harus mengeluarkan permintaan ke perangkat yang mengeluarkan interupsi untuk mendapatkan respons yang berisi informasi yang dibutuhkan.

Setelah penghitung program telah dimuat, prosesor melanjutkan ke siklus instruksi berikutnya, yang dimulai dengan pengambilan instruksi. Karena pengambilan instruksi ditentukan oleh isi pencacah program, hasilnya adalah kontrol dipindahkan ke program penangan interupsi. Eksekusi program ini menghasilkan operasi berikut:

* + 1. Pada titik ini, penghitung program dan PSW yang terkait dengan program yang terputus telah disimpan di tumpukan sistem. Namun, ada informasi lain yang dianggap sebagai bagian dari "keadaan" program yang dijalankan. Secara khusus, isi register prosesor perlu disimpan, karena register ini dapat digunakan oleh pengendali interupsi. Jadi, semua nilai ini, ditambah informasi status lainnya, perlu disimpan. Biasanya, handler interupsi akan mulai dengan menyimpan isi dari semua register pada stack. Gambar 7.7a menunjukkan contoh sederhana. Dalam hal ini, program pengguna diinterupsi setelah instruksi di lokasi N. Isi dari semua register ditambah alamat instruksi berikutnya (N + 1) didorong ke stack. Penunjuk tumpukan diperbarui untuk menunjuk ke atas tumpukan baru, dan penghitung program diperbarui untuk menunjuk ke awal rutin layanan interupsi.
    2. Penangan interupsi selanjutnya memproses interupsi. Ini termasuk pemeriksaan informasi status yang berkaitan dengan operasi I/O atau kejadian lain yang menyebabkan interupsi. Ini mungkin juga melibatkan pengiriman perintah atau pengakuan tambahan ke perangkat I/O.
    3. Ketika pemrosesan interupsi selesai, nilai register yang disimpan diambil dari stack dan dikembalikan ke register.
    4. Tindakan terakhir adalah mengembalikan PSW dan nilai penghitung program dari tumpukan. Akibatnya, instruksi berikutnya yang akan dieksekusi akan berasal dari program yang sebelumnya terputus.

**Masalah Desain**

Dua masalah desain muncul dalam mengimplementasikan I/O interupsi. Pertama, karena hampir selalu ada beberapa modul I/O, bagaimana prosesor menentukan perangkat mana yang mengeluarkan interupsi? Dan kedua, jika beberapa interupsi telah terjadi, bagaimana prosesor memutuskan mana yang akan diproses? Mari kita pertimbangkan identifikasi perangkat terlebih dahulu. Empat kategori umum teknik yang umum digunakan:

* Beberapa jalur interupsi
* Jajak pendapat perangkat lunak
* Rantai Daisy (jajak pendapat perangkat keras, di-vektor)
* Arbitrase bus (vektor)

Pendekatan yang paling mudah untuk masalah ini adalah dengan menyediakan beberapa jalur interupsi antara prosesor dan modul I/O. Namun, tidak praktis untuk mendedikasikan lebih dari beberapa jalur bus atau pin prosesor ke jalur interupsi. Akibatnya, bahkan jika beberapa baris digunakan, kemungkinan setiap baris akan memiliki beberapa modul I/O yang melekat padanya. Jadi, salah satu dari tiga teknik lainnya harus digunakan pada setiap baris.

Salah satu alternatif adalah software poll. Ketika prosesor mendeteksi interupsi, prosesor akan bercabang ke rutin layanan interupsi yang memeriksa setiap modul I/O untuk menentukan modul mana yang menyebabkan interupsi. Jajak pendapat bisa dalam bentuk baris perintah terpisah (mis., TESTI/O). Dalam hal ini, prosesor menaikkan TESTI/O dan menempatkan alamat modul I/O tertentu pada baris alamat. Modul I/O merespons secara positif jika mengatur interupsi.

Kerugian dari software poll adalah memakan waktu. Teknik yang lebih efisien adalah dengan menggunakan rantai daisy, yang pada dasarnya menyediakan polling perangkat keras. Ini menghindari kebutuhan untuk mengeksekusi rutin layanan interupsi umum terlebih dahulu. Teknik ini disebut interupsi vektor.

Ada teknik lain yang menggunakan interupsi vektor, dan itu adalah arbitrase bus. Dengan arbitrasi bus, modul I/O pertama-tama harus mendapatkan kendali atas bus sebelum dapat menaikkan jalur permintaan interupsi. Jadi, hanya satu modul yang dapat menaikkan garis pada satu waktu. Ketika prosesor mendeteksi interupsi, ia merespons pada jalur pengakuan interupsi. Modul yang meminta kemudian menempatkan vektornya pada jalur data.

**Pengontrol Interupsi Intel 82C59A**

Intel 80386 menyediakan satu Interrupt Request (INTR) dan satu baris Interrupt Acknowledge (INTA). Untuk memungkinkan 80386 menangani berbagai perangkat dan struktur prioritas, biasanya dikonfigurasi dengan arbiter interupsi eksternal, 82C59A. Perangkat eksternal terhubung ke 82C59A, yang pada gilirannya terhubung ke 80386.

**Antarmuka Periferal yang Dapat Diprogram Intel 8255A**

Sebagai contoh modul I/O yang digunakan untuk I/O terprogram dan I/O yang digerakkan oleh interupsi, kami mempertimbangkan Antarmuka Periferal yang Dapat Diprogram Intel 8255A. 8255A adalah modul I/O tujuan umum chip tunggal yang awalnya dirancang untuk digunakan dengan prosesor Intel 80386. Sejak itu telah dikloning oleh produsen lain dan merupakan chip pengontrol periferal yang banyak digunakan. Penggunaannya termasuk sebagai pengontrol untuk perangkat I/O sederhana untuk mikroprosesor dan dalam sistem tertanam, termasuk sistem mikrokontroler.

**7.5 Akses Memori Langsung**

**Kelemahan I/O Terprogram dan Berbasis Interupsi**

I/O Berbasis Interupsi, meskipun lebih efisien daripada I/O terprogram sederhana, masih memerlukan intervensi aktif prosesor untuk mentransfer data antara memori dan modul I/O, dan data apa pun transfer harus melintasi jalur melalui prosesor. Jadi, kedua bentuk I/O ini memiliki dua kelemahan yang melekat:

1. Kecepatan transfer I/O dibatasi oleh kecepatan prosesor dapat menguji dan melayani perangkat.
2. Prosesor terikat dalam mengelola transfer I/O; sejumlah instruksi harus dieksekusi untuk setiap transfer I/O.

Ada sedikit trade-off antara dua kelemahan ini. Pertimbangkan transfer blok data. Menggunakan I/O terprogram sederhana, prosesor didedikasikan untuk tugas I/O dan dapat memindahkan data pada kecepatan yang agak tinggi, tanpa melakukan hal lain. Interupsi I/O membebaskan prosesor sampai batas tertentu dengan mengorbankan kecepatan transfer I/O. Namun demikian, kedua metode tersebut memiliki dampak buruk pada aktivitas prosesor dan kecepatan transfer I/O.

**Fungsi DMA**

DMA melibatkan modul tambahan pada bus sistem. Modul DMA (Gambar 7.12) mampu meniru prosesor dan, memang, mengambil alih kendali sistem dari prosesor. Ini perlu dilakukan untuk mentransfer data ke dan dari memori melalui bus sistem. Untuk tujuan ini, modul DMA harus menggunakan bus hanya ketika prosesor tidak membutuhkannya, atau harus memaksa prosesor untuk menghentikan operasi sementara. Teknik yang terakhir lebih umum dan disebut sebagai pencurian siklus, karena modul DMA pada dasarnya mencuri siklus bus.

Ketika prosesor ingin membaca atau menulis blok data, prosesor mengeluarkan perintah ke modul DMA, dengan mengirimkan ke modul DMA informasi berikut:

* Apakah membaca atau menulis diminta, menggunakan jalur kontrol baca atau tulis antara prosesor dan modul DMA.
* Alamat perangkat I/O yang terlibat, dikomunikasikan pada jalur data.
* Lokasi awal dalam memori untuk membaca atau menulis, dikomunikasikan pada jalur data dan disimpan oleh modul DMA dalam register alamatnya.
* Jumlah kata yang akan dibaca atau ditulis, dikomunikasikan kembali melalui jalur data dan disimpan dalam register penghitung data.

Prosesor kemudian melanjutkan dengan pekerjaan lain. Ini telah mendelegasikan operasi I/O ini ke modul DMA. Modul DMA mentransfer seluruh blok data, satu kata pada satu waktu, langsung ke atau dari memori, tanpa melalui prosesor. Ketika transfer selesai, modul DMA mengirimkan sinyal interupsi ke prosesor. Dengan demikian, prosesor hanya terlibat di awal dan akhir transfer

**7.6 Direct Cache Access**

DMA telah membuktikan cara yang efektif untuk meningkatkan kinerja I/O dengan perangkat periferal dan lalu lintas I/O jaringan. Namun, untuk peningkatan dramatis dalam kecepatan data untuk jaringan I/O, DMA tidak mampu menskalakan untuk memenuhi permintaan yang meningkat. Permintaan ini terutama datang dari penyebaran luas sakelar Ethernet 10-Gbps dan 100-Gbps untuk menangani transfer data dalam jumlah besar ke dan dari server basis data dan sistem berkinerja tinggi lainnya [STAL14a]. Sumber lalu lintas sekunder tetapi semakin penting berasal dari Wi-Fi dalam jangkauan gigabit. Perangkat Wi-Fi jaringan yang menangani 3,2 Gbps dan 6,76 Gbps menjadi tersedia secara luas dan menghasilkan permintaan pada sistem perusahaan.

**DMA Using Shared Last-Level Cache**

DMA penggunaan cache Dalam operasi DMA tradisional, data dipertukarkan antara memori utama dan perangkat I/O melalui struktur interkoneksi sistem, seperti bus, ring, atau matriks titik-ke-titik QPI. Jadi, misalnya, jika Xeon E5-2600/4600 menggunakan teknik DMA tradisional, output akan diproses sebagai berikut. Driver I/O yang berjalan pada inti akan mengirimkan perintah I/O ke pengontrol I/O (diberi label PCIe pada Gambar 7.16) dengan lokasi dan ukuran buffer di memori utama yang berisi data yang akan ditransfer. Kontroler I/O mengeluarkan permintaan baca yang dirutekan ke hub pengontrol memori (MCH), yang mengakses data pada memori DDR3 dan meletakkannya di ring sistem untuk dikirimkan ke pengontrol I/O. Cache L3 tidak terlibat dalam transaksi ini dan satu atau lebih pembacaan memori off-chip diperlukan. Demikian pula, untuk input, data datang dari pengontrol I/O dan dikirimkan melalui ring sistem ke MCH dan ditulis ke memori utama. MCH juga harus membatalkan semua baris cache L3 yang terkait dengan lokasi memori yang diperbarui. Dalam hal ini, satu atau lebih penulisan memori off-chip diperlukan. Selanjutnya, jika aplikasi ingin mengakses data baru, diperlukan pembacaan memori utama.

**Cache-Related Performance Issues**

Untuk memperjelas masalah kinerja dan untuk menjelaskan manfaat DCA sebagai cara untuk meningkatkan kinerja, mari kita lihat pemrosesan lalu lintas protokol secara lebih rinci untuk lalu lintas masuk. Secara umum, langkah-langkah berikut terjadi:

* Paket tiba
* DMA
* NIC menginterupsi host
* Ambil deskriptor dan header
* Cache miss terjadi
* Header diproses
* Muatan ditransfer

Urutan langkah yang serupa terjadi untuk lalu lintas paket keluar, tetapi ada beberapa perbedaan yang memengaruhi cara cache dikelola. Untuk lalu lintas keluar, langkah-langkah berikut terjadi:

* Transfer paket diminta
* Paket dibuat
* Operasi keluaran dipanggil
* transfer DMA
* sinyal NIC selesai
* Pengemudi membebaskan buffer

**Direct Cache Access Strategies**

Strategi paling sederhana adalah yang diimplementasikan sebagai prototipe pada sejumlah prosesor Intel Xeon antara tahun 2006 dan 2010 [KUMA07, INTE08]. Bentuk DCA ini hanya berlaku untuk lalu lintas jaringan yang masuk. Fungsi DCA di pengontrol memori mengirimkan petunjuk pengambilan ke inti segera setelah data tersedia di memori sistem. Hal ini memungkinkan inti untuk mengambil paket data dari buffer sistem, sehingga menghindari kesalahan cache dan pemborosan terkait siklus inti.

**Direct Data I/O**

Ingatlah bahwa ada dua teknik untuk menangani pembaruan ke baris cache:

* Write through
* Write back
* Write allocate
* Non-write allocate

**7.7 I/O Channels and Processors**

**The Evolution of the I/O Function**

Langkah-langkah evolusi dapat diringkas sebagai berikut:

1. CPU secara langsung mengontrol perangkat periferal.
2. Kontroler atau modul I/O ditambahkan.
3. Konfigurasi yang sama seperti pada langkah 2 digunakan, tetapi sekarang interupsi digunakan.
4. Modul I/O diberikan akses langsung ke memori melalui DMA
5. Modul I/O ditingkatkan untuk menjadi prosesor tersendiri, dengan set instruksi khusus yang disesuaikan untuk I/O
6. Modul I/O memiliki memori lokalnya sendiri dan, pada kenyataannya, merupakan komputer yang berdiri sendiri

**Characteristics of I/O Channels**

Kanal I/O merepresentasikan perluasan dari konsep DMA. Saluran I/O memiliki kemampuan untuk mengeksekusi instruksi I/O, yang memberikan kendali penuh atas operasi I/O. Dalam sistem komputer dengan perangkat seperti itu, CPU tidak menjalankan instruksi I/O. Instruksi tersebut disimpan dalam memori utama untuk dieksekusi oleh prosesor tujuan khusus di saluran I/O itu sendiri. Dengan demikian, CPU memulai transfer I/O dengan menginstruksikan saluran I/O untuk mengeksekusi program di memori. Program akan menentukan perangkat atau perangkat, area atau area memori untuk penyimpanan, prioritas, dan tindakan yang harus diambil untuk kondisi kesalahan tertentu. Saluran I/O mengikuti instruksi ini dan mengontrol transfer data.

**7.8 External Interconnection Standards**

**Universal Serial Bus (USB)**

USB banyak digunakan untuk koneksi periferal. Ini adalah antarmuka default untuk perangkat berkecepatan lebih lambat, seperti keyboard dan perangkat penunjuk, tetapi juga biasa digunakan untuk I/O berkecepatan tinggi, termasuk printer, drive disk, dan adaptor jaringan. Sistem USB dikendalikan oleh pengontrol host root, yang menempel pada perangkat untuk membuat jaringan lokal dengan topologi hierarki hierarki.

**FireWire Serial Bus**

FireWire was developed as an alternative to the small computer system interface (SCSI) to be used on smaller systems, such as personal computers, workstations, and servers. The objective was to meet the increasing demands for high I/O rates on these systems, while avoiding the bulky and expensive I/O channel technologies developed for mainframe and supercomputer systems. The result is the IEEE standard 1394, for a High Performance Serial Bus, commonly known as FireWire.

**Small Computer System Interface (SCSI)**

SCSI adalah standar yang pernah umum untuk menghubungkan perangkat periferal (disk, modem, printer, dll.) ke komputer berukuran kecil dan menengah. Meskipun SCSI telah berevolusi ke kecepatan data yang lebih tinggi, SCSI telah kehilangan popularitas dari pesaing seperti USB dan FireWire dalam sistem yang lebih kecil. Namun, versi SCSI berkecepatan tinggi tetap populer untuk dukungan memori massal pada sistem perusahaan. Misalnya, IBM zEnterprise EC12 dan mainframe IBM lainnya menawarkan dukungan untuk SCSI, dan sejumlah sistem hard drive Seagate menggunakan SCSI

**Thunderbolt**

Teknologi koneksi periferal terbaru dan tercepat yang tersedia untuk penggunaan umum adalah Thunderbolt, yang dikembangkan oleh Intel dengan kolaborasi dari Apple. Satu kabel Thunderbolt dapat mengatur pekerjaan yang sebelumnya diperlukan untuk beberapa kabel. Teknologi ini menggabungkan data, video, audio, dan daya ke dalam satu koneksi berkecepatan tinggi untuk periferal seperti hard drive, array RAID (Redundant Array of Independent Disks), kotak pengambilan video, dan antarmuka jaringan.

**InfiniBand**

InfiniBand adalah spesifikasi I/O yang ditujukan untuk pasar server kelas atas. Versi pertama dari spesifikasi dirilis pada awal 2001 dan telah menarik banyak vendor. Misalnya, seri mainframe IBM zEnterprise sangat bergantung pada InfiniBand selama beberapa tahun. Standar tersebut menjelaskan arsitektur dan spesifikasi aliran data di antara prosesor dan perangkat I/O cerdas.

**PCI Express**

PCI Express adalah sistem bus berkecepatan tinggi untuk menghubungkan periferal dari berbagai jenis dan kecepatan.

**SATA**

Serial ATA (Serial Advanced Technology Attachment) adalah antarmuka untuk sistem penyimpanan disk.

**Ethernet**

Ethernet adalah teknologi jaringan kabel utama, digunakan di rumah, kantor, pusat data, perusahaan, dan jaringan area luas. Karena Ethernet telah berkembang untuk mendukung kecepatan data hingga 100 Gbps dan jarak dari beberapa meter hingga puluhan km, Ethernet telah menjadi penting untuk mendukung komputer pribadi, workstation, server, dan perangkat penyimpanan data besar dalam organisasi besar dan kecil.

**Wi-Fi**

Wi-Fi adalah teknologi akses Internet nirkabel yang dominan, digunakan di rumah, kantor, dan ruang publik. Wi-Fi di rumah sekarang menghubungkan komputer, tablet, ponsel pintar, dan sejumlah perangkat elektronik, seperti kamera video, TV, dan termostat.

**7.9 IBM zEnterprise EC12 I/O Structure**

**Channel Structure**

Dari 96 prosesor inti, hingga 4 di antaranya dapat didedikasikan untuk penggunaan I/O, menciptakan 4 subsistem saluran (CSS). Setiap CSS terdiri dari elemen-elemen berikut:

* System assist processor (SAP)
* Hardware system area (HSA)
* Logical partitions
* Subchannels
* Channel path
* Channel

**I/O System Organization**

Sistem ini terdiri dari dua ruang besar, yang disebut bingkai, yang menampung berbagai komponen zEnterprise EC12. Rangka A sebelah kanan mencakup dua sangkar besar, ditambah ruang untuk pemasangan kabel dan komponen lainnya. Sangkar atas adalah sangkar prosesor, dengan empat slot untuk menampung hingga empat buku prosesor yang saling terhubung sepenuhnya. Setiap buku berisi modul multichip (MCM), kartu memori, dan koneksi sangkar I/O. Setiap MCM adalah papan yang menampung enam chip multicore dan dua chip kontrol penyimpanan.

Sangkar bawah dalam bingkai A adalah sangkar I/O, yang berisi perangkat keras I/O, termasuk multipleksor dan saluran. Sangkar I/O adalah unit tetap yang dipasang oleh IBM untuk spesifikasi pelanggan di pabrik. Rangka Z sebelah kiri berisi baterai internal dan catu daya serta ruang untuk satu atau lebih elemen pendukung, yang digunakan oleh manajer sistem untuk manajemen platform. Bingkai Z juga berisi slot untuk dua atau lebih laci I/O.

**PUSTAKA**

Stailings, W. (2016). *Computer Organization and Architecture Designing For Performa 10th Edition.* PEARSON Education.