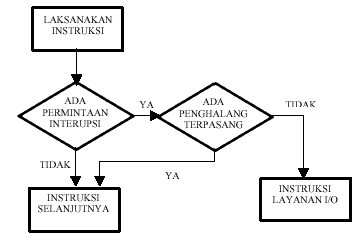
Nama :Tiyssa

NIM :L200150047

**Metode Interupsi**  
  
          Pengendalian I/O dengan metode polling mempunyai dua kelemahan :  
- Pemborosan waktu prosesor karena status semua periferal diperiksa terus menerus secara berurutan.  
- Karena harus memeriksa status semua alat I/O maka waktu kerjanya menjadi lambat. Ini merupakan kelemahan dalam sistem waktu nyata (Real Time), dimana satu periferal mengharap layanan dalam satu waktu tertentu.  
          Kelemahan ini diatasi dengan menggunakan layanan waktu tak sinkron menggunakan interupsi. Tiap alat I/O atau pengendalinya dihubungkan ke sebuah saluran interupsi. Saluran interupsi menggerbangkan sebuah permintaan interupsi ke mikroprosesor. Bilamana sebuah alat I/O memerlukan layanan , alat akan membangkitkan pulsa interupsi atau status suatu tingkatan saluran untuk menarik perhatian mikroprosesor. Mikroprosesor akan memberikan layanan pada alat I/O jika ada interupsi dan jika tidak ada interupsi mikroprosesor melakukan instruksi selanjutnya. Logika pengendalian I/O dengan metode interupsi ditunjukkan pada diagram alir Gambar 2.

[](http://3.bp.blogspot.com/-bXkYpGd1x3Q/UXY4b0Xh4tI/AAAAAAAABPY/dpqqY2Zsqcw/s1600/gambar9.2.jpg)

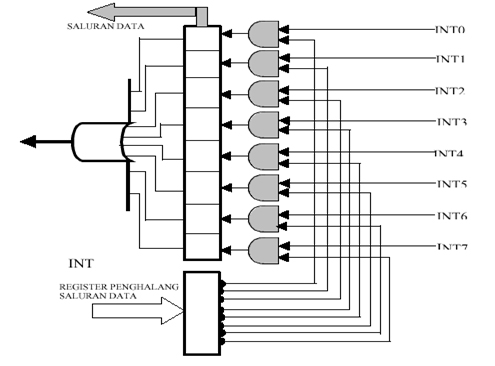
Gambar 2. Diagram Alir Pengendalian I/O Metode Interupsi

          Begitu permintaan interupsi diterima dan disetujui oleh mikroprosesor, alat I/O harus dilayani. Untuk melayani alat I/O, maka mikroprosesor melaksanakan suatu routin pelayanan khusus. Ada dua masalah yang muncul pada saat melakukan layanan interupsi :

- Bagaimana status program yang dilaksanakan pada mikroprosesor pada saat interupsi harus diperilahara dalam *stack.*

*-*Bagaimana mikroprosesor dapat mengenali secara tepat alat I/O mana yang membangkitkan interupsi. Identifikasi ini dapat dilakukan dengan perangkat keras, perangkat lunak, atau kombinasi perangkat keras dan perangkat lunak. Pencabangan ke alamat alat I/O disebut **Pemvektoran Interupsi**. Secara rutin perangkat lunak menetapkan identitas alat yang meminta layanan interupsi. Rutin identifikasi interupsi akan memilih saluran setiap alat yang dihubungkan dengan sistem. Setelah dikenal alat mana yang mencetuskan interupsi maka ia kemudian bercabang ke alamat rutin penanganan interupsi yang sesuai. Metode ke dua yang digerakkan oleh perangkat lunak, tetapi dengan pertolongan beberapa perangkat keras tambahan. Metode ini menggunakan rantai beranting (daisy chain) untuk mengenal alat yang mencetuskan interupsi. Metode tercepat adalah interupsi yang divektorkan. Adalah menjadi tanggung jawab pengendali alat I/O untuk memberikan baik interupsi maupun pengenal alat yang menyebabkan interupsi atau lebih baik lagi alamat pencabangan bagi rutin penanganan interupsi. Bila pengendali hanya memberikan pengenal alat, adalah tugas perangkat lunak mencari tabel alamat pencabangan bagi tiap alat. Ini sederhana bagi perangkat keras tapi tak mencapai performansi tertinggi.

- Prioritas, beberapa interupsi dapat dibangkitkan serentak. Mikroprosesor diberi tugas untuk memutuskan bagaimana urutan pelayanannya. Setiap alat diberikan suatu prioritas. Mikroprosesor melayani setiap alat sesuai prioritasnya. Dalam dunia komputer prioritas 0, menurut konvensi memiliki prioritas. tertinggi, prioritas 1 yang kedua demikian seterusnya. Prioritas dapat diset baik pada perangkat keras maupun perangkat lunak. Pengaturan prioritas dengan perangkat keras dikerjakan oleh suatu piranti yang disebut Programmable Interrupt Controller (PIC). Struktur dasar logika PIC dapat digambarkan seperti Gambar 3.

[](http://2.bp.blogspot.com/-D9BWAYM8-Hw/UXY5jS6SOqI/AAAAAAAABPo/RUmfs1lLtuk/s1600/gambar9.3.jpg)

Gambar 3. Struktur Dasar Logika PIC

**Interupsi**

Interupsi terjadi bila suatu perangkat M/K ingin memberitahu prosesor bahwa ia siap menerima perintah, *output* sudah dihasilkan, atau terjadi *error*.

**Penanganan Interupsi**

Ada beberapa tahapan dalam penanganan interupsi:

1. *Controller* mengirimkan sinyal interupsi melalui *interrupt-request-line*
2. Sinyal dideteksi oleh prosesor
3. Prosesor akan terlebih dahulu menyimpan informasi tentang keadaan *state*-nya (informasi tentang proses yang sedang dikerjakan)
4. Prosesor mengidentifikasi penyebab interupsi dan mengakses tabel vektor interupsi untuk menentukan *interrupt handler*
5. Transfer kontrol ke *interrupt handler*
6. Setelah interupsi berhasil diatasi, prosesor akan kembali ke keadaan seperti sebelum terjadinya interupsi dan melanjutkan pekerjaan yang tadi sempat tertunda.

**Gambar 10.2. Siklus penanganan interupsi**

|  |
| --- |
| Siklus penanganan interupsi |

**Interrupt Request Line**

Pada kebanyakan CPU, ada dua *interrupt request line*. Pertama, interupsi *nonmaskable*, interupsi ini biasanya berasal dari perangkat keras dan harus segera dilaksanakan, seperti terjadinya *error* pada memori.

Kedua, interupsi *maskable*, jenis interupsi ini bisa dilayani oleh prosesor atau bisa tidak dilayani. Kalaupun dilayani, harus dilihat keadaan prosesor saat itu. Ada kemungkinan prosesor langsung menangani bila saat itu, prosesor *preemptive*, bila *nonpreemptive*, maka harus menunggu proses yang sedang dikerjakan selesai.

**Vektor Interupsi dan *Vector Chaining***

Bila ada sebuah sinyal interupsi pada *interrupt request line*, bagaimana sebuah *interrupt handler* mengetahui sumber dari interupsi itu? Apakah harus menelusuri semua sumber interupsi satu-persatu? Tidak perlu, karena setiap *interrupt handler* mempunyai alamat memori masing-masing. Alamat ini adalah *offset* pada sebuah tabel yang disebut dengan vektor interupsi.

**Tabel 10.1. Tabel Vector-Even pada Intel Pentium**

| **Vector number** | **Description** |
| --- | --- |
| 0 | Divide error |
| 1 | Debug Exception |
| 2 | Null Interrupt |
| 3 | Breakpoint |
| 4 | INTO-detected overflow |
| 5 | Bound range exception |
| 6 | Invalid opcode |
| 7 | Device not available |
| 8 | Double fault |
| 9 | Compressor segment overrun (reserved) |
| 10 | Invalid task state segment |
| 11 | Segment not present |
| 12 | Stack fault |
| 13 | General protection |
| 14 | Page fault |
| 15 | (Intel reserved, do not use) |
| 16 | Floating point error |
| 17 | Alignment check |
| 18 | Machine check |
| 19-31 | (Intel reserved, do not use) |
| 32-255 | Maskable interrupt |

Sesuai dengan perkembangan zaman, komputer mempunyai lebih banyak perangkat (dan lebih banyak *interrupt handlers*) daripada elemen alamat di vektor. Hal ini bisa diatasi dengan teknik *vector chaining*. Setiap elemen di vektor interupsi menunjuk ke kepala dari sebuah daftar *interrupt handlers*, sehingga bila ada interupsi, *handler* yang terdapat pada daftar yang ditunjuk akan dipanggil satu persatu sampai didapatkan *handler* yang bisa menangani interupsi yang bersangkutan.

**Prioritas Interupsi**

Mekanisme interupsi juga menerapkan sistem level prioritas interupsi. Sistem ini memungkinkan interupsi berprioritas tinggi menyela eksekusi interupsi berprioritas rendah. Sistem ini juga memungkinkan perangkat M/K yang membutuhkan pelayanan secepatnya didahulukan daripada perangkat lainnya yang prioritasnya lebih rendah. Pengaturan prioritas dan penanganan perangkat berdasarkan prioritasnya diatur oleh prosesor dan *controller* interupsi.

**Penyebab Interupsi**

Mekanisme interupsi tidak hanya digunakan untuk menangani operasi yang berhubungan dengan perangkat M/K. Sistem operasi menggunakan mekanisme interupsi untuk beberapa hal, di antaranya:

1. Menangani *exception*

*Exception* adalah suatu kondisi dimana terjadi sesuatu, atau dari sebuah operasi didapatkan hasil tertentu yang dianggap khusus sehingga harus mendapat perhatian lebih, contohnya, pembagian dengan nol, pengaksesan alamat memori yang *restricted* atau tidak valid, dll.

1. Mengatur *virtual memory paging*.
2. Menangani perangkat lunak interupsi.
3. Menangani alur kontrol kernel.

Jika interupsi yang terjadi merupakan permintaan untuk transfer data yang besar, maka penggunaan interupsi menjadi tidak efisien, untuk mengatasinya digunakanlah DMA.