**HARDWARE INTERRUPT**

**SISTEM OPERASI**

Dosen Pengampu : Ike Yunia Pasa, S.T., M.KOM.



Disusun Oleh:

Aji Sukriana Wahid 5230411336

Halim Arifsanjani Syaila 5230411360

Radityo Haryo Bismoko 5230411376

Albin Rhamdani Wibowo 5230411324

Kelas G

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS & TEKNOLOGI**

**UNIVERSITAS TEKNOLOGI YOGYAKARTA**

**YOGYAKARTA**

**2024**

**DAFTAR ISI**

[**I. Pengertian Hardware Interrupt 1**](#_Toc185326476)

[**II. Cara Kerja Hardware Interrupt 2**](#_Toc185326477)

[a) Penerbitan Interrupt 2](#_Toc185326478)

[b) Interupsi Eksekusi Prosesor 2](#_Toc185326479)

[c) Identifikasi Sumber Interrupt 2](#_Toc185326480)

[d) Eksekusi Interrupt Handler 2](#_Toc185326481)

[e) Kembali ke Eksekusi Sebelumnya 2](#_Toc185326482)

[**III. Jenis-jenis Hardware Interrupt 3**](#_Toc185326483)

[a) Interrupt I/O (Input/Output): 3](#_Toc185326484)

[b) Interrupt Timer: 3](#_Toc185326485)

[c) Interrupt Kesalahan (Fault): 3](#_Toc185326486)

[d) Interrupt Perangkat Jaringan 3](#_Toc185326487)

[e) Interrupt DMA (Direct Memory Access): 4](#_Toc185326488)

[**IV. Keunggulan Hardware Interrupt 4**](#_Toc185326489)

[**V. Kekurangan Hardware Interrupt 5**](#_Toc185326490)

[**VI. Contoh Implementasi Hardware Interrupt 5**](#_Toc185326491)

[a) Keyboard Interrupt: 5](#_Toc185326492)

[b) Disk I/O Interrupt: 5](#_Toc185326493)

[c) Timer Interrupt 6](#_Toc185326494)

[d) Perangkat Jaringan 6](#_Toc185326495)

[**VII. Kesimpulan 6**](#_Toc185326496)

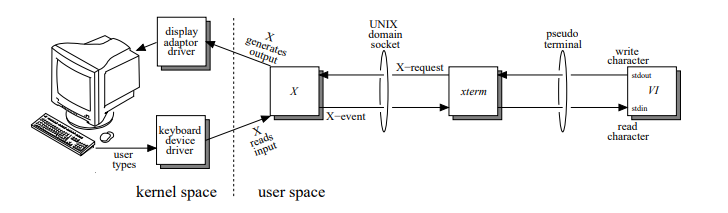
[**VIII. Referensi 7**](#_Toc185326497)

# Pengertian Hardware Interrupt

Interupsi atau interrupt adalah suatu permintaan khusus pada mikroprocessor untuk melakukan sesuatu, jika terjadi interupsi maka komputer akan menghentikan dahulu apa yang sedang dikerjakan dan melakukan apa yang di minta oleh yang menginterupsi, setelah selesai maka aliran program akan kembali ke pernyataan program sebelum terjadinya interupsi. Interupsi merupakan sub rutin yang sudah tersedia dalam memori komputer. (Fathuddin Yazid, 2015)

Hardware interrupt adalah sinyal yang dikirimkan oleh perangkat keras (hardware) kepada prosesor untuk menunjukkan bahwa sebuah kejadian tertentu membutuhkan perhatian segera. Proses ini memungkinkan perangkat keras seperti keyboard, mouse, atau perangkat I/O lainnya untuk berkomunikasi dengan CPU tanpa perlu menunggu CPU selesai dengan tugas saat ini. Seperti yang dijelaskan oleh (Tsafrir, 2007), hardware interrupt memungkinkan CPU untuk merespons kejadian dengan cepat, tanpa harus terus-menerus memeriksa status perangkat keras. Hal ini sangat penting untuk meningkatkan efisiensi dalam pengelolaan sumber daya dan multitasking.

Interrupt ini membantu dalam pengelolaan sumber daya sistem dan meningkatkan efisiensi dengan memungkinkan CPU untuk menangani tugas-tugas kritis secara cepat dan dinamis, yang mendukung sistem operasi untuk menangani beberapa tugas secara bersamaan (Liu et al., 2007). Dalam konteks arsitektur komputer, hardware interrupt merupakan salah satu mekanisme yang mendukung konsep multitasking, real-time processing, dan manajemen perangkat keras



# Cara Kerja Hardware Interrupt

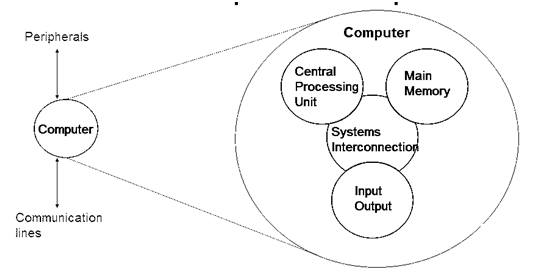
Saat sebuah perangkat keras memicu interrupt, langkah-langkah berikut terjadi:

1. Penerbitan Interrupt: Perangkat keras mengirim sinyal interrupt ke prosesor melalui jalur khusus yang disebut interrupt request line (IRQ). Sinyal ini biasanya berupa pulsa listrik yang diterima oleh kontroler interrupt di CPU.
2. Interupsi Eksekusi Prosesor: CPU menghentikan sementara eksekusi instruksi saat ini dan menyimpan status kerja (register, flag, dan program counter) ke dalam memori untuk memastikan tugas sebelumnya dapat dilanjutkan setelah interrupt selesai ditangani.
3. Identifikasi Sumber Interrupt: CPU menggunakan interrupt vector table (IVT), yaitu tabel yang berisi alamat dari interrupt service routine (ISR) untuk menentukan handler (penangan) yang sesuai dengan jenis interrupt yang terjadi.
4. Eksekusi Interrupt Handler: Prosesor menjalankan interrupt service routine (ISR), yaitu kode program khusus yang dirancang untuk menangani kejadian yang dipicu oleh interrupt. ISR dapat berupa rutinitas singkat atau fungsi yang lebih kompleks tergantung pada kebutuhan.
5. Kembali ke Eksekusi Sebelumnya: Setelah ISR selesai, CPU memulihkan status kerja sebelumnya dari memori dan melanjutkan tugas yang sempat dihentikan. Proses ini memastikan integritas data dan kelancaran eksekusi. Fenomena ini, meskipun sangat efisien dalam hal multitasking, memerlukan manajemen yang hati-hati agar tidak terjadi overhead yang berlebihan, seperti yang dibahas oleh (Tsafrir, 2007)

# Jenis-jenis Hardware Interrupt

Hardware interrupt dapat dibagi menjadi beberapa jenis berdasarkan sumber pemicu:

1. Interrupt I/O (Input/Output): Dihasilkan oleh perangkat seperti keyboard, mouse, atau disk drive. Misalnya, menekan tombol keyboard akan memicu interrupt untuk memproses input tersebut. Perangkat keras ini biasanya menggunakan jalur IRQ tertentu untuk mengirim sinyal ke CPU.
2. Interrupt Timer: Timer interrupt digunakan untuk mengatur waktu pada sistem. Contohnya adalah clock interrupt yang mengatur fungsi sistem operasi seperti multitasking. Timer interrupt juga berperan dalam memastikan bahwa CPU dapat membagi waktu dengan efisien antara proses yang berjalan.
3. Interrupt Kesalahan (Fault): Digunakan untuk memberi tahu CPU tentang kondisi kesalahan, seperti page fault atau pembagian oleh nol. Interrupt jenis ini memungkinkan sistem operasi untuk menangani masalah secara dinamis tanpa memengaruhi kinerja keseluruhan sistem.(Scheler et al., 2009)
4. Interrupt Perangkat Jaringan: Digunakan oleh perangkat jaringan seperti kartu Ethernet atau modul Wi-Fi untuk memberi tahu CPU bahwa paket data telah diterima atau ada kesalahan dalam transmisi data. Interrupt ini memungkinkan komunikasi jaringan berjalan lancar.
5. Interrupt DMA (Direct Memory Access): Digunakan oleh perangkat DMA untuk memberi tahu CPU bahwa proses transfer data antar perangkat keras telah selesai, tanpa campur tangan prosesor secara langsung.



# Keunggulan Hardware Interrupt

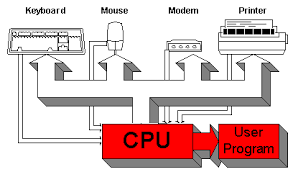
* **Efisiensi**: Membantu CPU merespons kejadian penting tanpa harus terus-menerus memeriksa status perangkat keras (polling), yang dapat membebaskan sumber daya untuk tugas lain. (Liu et al., 2007)
* **Waktu Respons Cepat**: Memungkinkan sistem untuk menangani situasi kritis dengan segera, seperti menangani input pengguna atau error perangkat keras. (Feng et al., 2008)
* **Penggunaan Sumber Daya Optimal**: Meminimalkan penggunaan CPU untuk tugas-tugas non-kritikal sehingga meningkatkan efisiensi sistem secara keseluruhan.
* **Komunikasi Real-time**: Interrupt memungkinkan sistem untuk merespons kejadian dalam waktu nyata, menjadikannya ideal untuk aplikasi real-time seperti kontrol robotik dan otomasi industri, sebagaimana dijelaskan dalam kajian (Scheler et al., 2009)

# Kekurangan Hardware Interrupt

* **Kerumitan Desain Sistem**: Sistem berbasis interrupt memerlukan desain perangkat lunak dan perangkat keras yang kompleks, termasuk manajemen prioritas interrupt dan sinkronisasi antarproses. Seperti yang ditunjukkan dalam penelitian (Liu et al., 2007), sistem dengan banyak interrupt dapat menambah beban pengelolaan dan memerlukan perencanaan yang hati-hati.
* **Overhead Waktu**: Memproses interrupt memerlukan waktu untuk menyimpan dan memulihkan status CPU, yang dapat memengaruhi kinerja jika terlalu sering terjadi. Hal ini disoroti oleh (Tsafrir, 2007) yang menunjukkan bahwa terlalu sering terjadi interrupt dapat menurunkan efisiensi sistem.
* **Interrupt Storm**: Jika terlalu banyak interrupt yang terjadi dalam waktu singkat, ini dapat menyebabkan sistem menjadi lambat atau bahkan gagal berfungsi dengan baik.
* **Masalah Prioritas**: Jika prioritas interrupt tidak diatur dengan benar, tugas penting dapat tertunda karena interrupt dengan prioritas lebih rendah.

# Contoh Implementasi Hardware Interrupt

1. Keyboard Interrupt: Saat tombol ditekan, keyboard mengirimkan interrupt ke CPU untuk memproses input. Interrupt ini memungkinkan sistem untuk mengenali dan merekam input tanpa jeda.
2. Disk I/O Interrupt: Ketika proses membaca atau menulis data ke/dari disk selesai, perangkat keras disk memicu interrupt untuk memberi tahu CPU bahwa operasi telah berhasil. Hal ini menghindari polling terus-menerus pada perangkat disk. seperti yang dijelaskan oleh (Scheler et al., 2009)
3. Timer Interrupt: Digunakan oleh sistem operasi untuk mengatur pengalihan konteks (*context switching*) dalam multitasking. Timer interrupt juga dapat digunakan untuk menjalankan tugas periodik dalam sistem real-time.
4. Perangkat Jaringan: Kartu jaringan seperti NIC (Network Interface Card) menggunakan interrupt untuk memberi tahu sistem operasi bahwa data baru telah diterima dari jaringan atau bahwa transmisi telah selesai.
5. **Kontroler DMA**: Perangkat DMA memanfaatkan interrupt untuk memberi tahu CPU bahwa transfer data antarperangkat keras telah selesai tanpa intervensi langsung dari CPU.



# Kesimpulan

Hardware interrupt adalah mekanisme penting dalam sistem komputer modern yang memungkinkan perangkat keras dan CPU untuk berkomunikasi secara efisien. Dengan interrupt, CPU dapat merespons kejadian penting tanpa harus terus-menerus memantau perangkat keras, sehingga meningkatkan kinerja sistem secara keseluruhan. Namun, penggunaannya memerlukan desain yang hati-hati untuk memastikan stabilitas dan efisiensi sistem, seperti yang dikemukakan oleh (Liu et al., 2007) dan (Tsafrir, 2007). Keunggulan seperti waktu respons yang cepat dan efisiensi sumber daya menjadikannya esensial untuk aplikasi real-time, meskipun tantangan seperti *interrupt storm* dan kompleksitas desain tetap harus diatasi dengan solusi yang tepat.

# Referensi

1. Fathuddin Yazid. (2015). Interupsi Pada Komputer. *Jurnal Informatika*, *1*(1).
2. Feng, X., Shao, Z., Dong, Y., & Guo, Y. (2008). Certifying low-level programs with hardware interrupts and preemptive threads. *ACM SIGPLAN Notices*, *43*(6), 170–182. https://doi.org/10.1145/1379022.1375603
3. Liu, M., Shao, Z., Wang, M., Wei, H., & Wang, T. (2007). Implementing hybrid operating systems with two-level hardware interrupts. *Proceedings - Real-Time Systems Symposium*, 244–253. https://doi.org/10.1109/RTSS.2007.10
4. Scheler, F., Hofer, W., Oechslein, B., Pfister, R., Schröder-Preikschat, W., & Lohmann, D. (2009). Parallel, hardware-supported interrupt handling in an event-triggered real-time operating system. *Embedded Systems Week 2009 - 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, CASES’09*, 167–174. https://doi.org/10.1145/1629395.1629419
5. Tsafrir, D. (2007). The context-switch overhead inflicted by hardware interrupts (and the enigma of do-nothing loops). *Proceedings of the 2007 Workshop on Experimental Computer Science*, *June*, 13–14. https://doi.org/10.1145/1281700.1281704