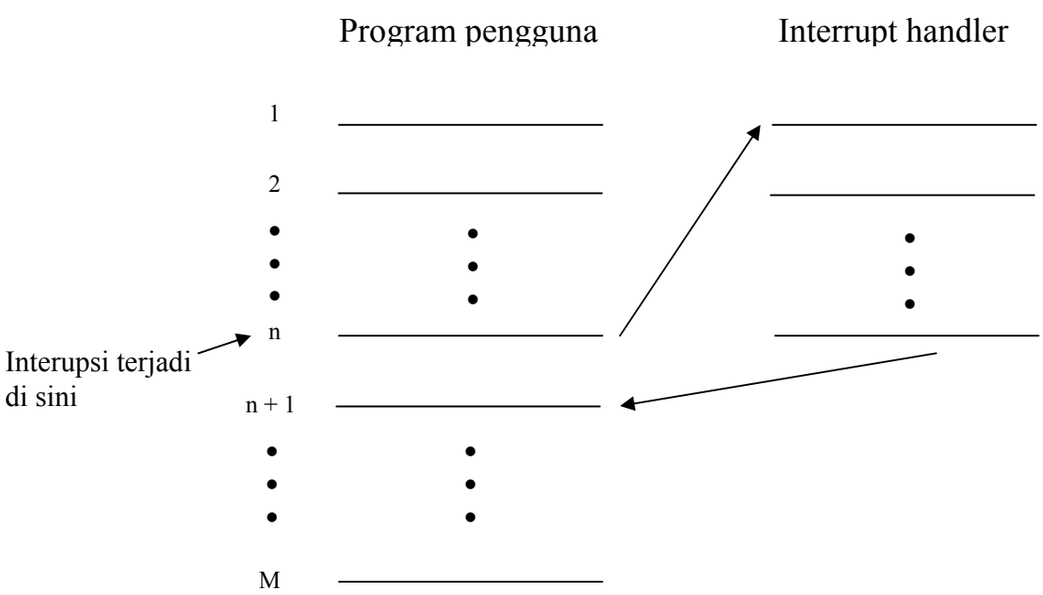
INTERUPSI

- Semua komputer menyediakan mekanisme yang membuat modul-modul lainnya (I/O, memori) dapat menginterupsi pengolahan normal CPU.

- Jenis-jenis interupsi :
  - Program : dibangkitkan sebagai hasil eksekusi instruksi, seperti arithmetic overflow, pembagian dengan nol, usaha mengeksekusi instruksi mesin yang ilegal, dan referensi ke luar ruang memori pengguna yang diperbolehkan.
  - Timer : dibangkitkan oleh timer di dalam prosesor. Hal ini memungkinkan sistem operasi menjalankan fungsi-fungsi tertentu secara regular.
  - I/O : dibangkitkan oleh I/O controller, untuk memberi sinyal penyelesaian normal suatu operasi atau memberi sinyal berbagai kondisi error.
  - Hardware failure : dibangkitkan oleh kegagalan daya atau memori parity error.

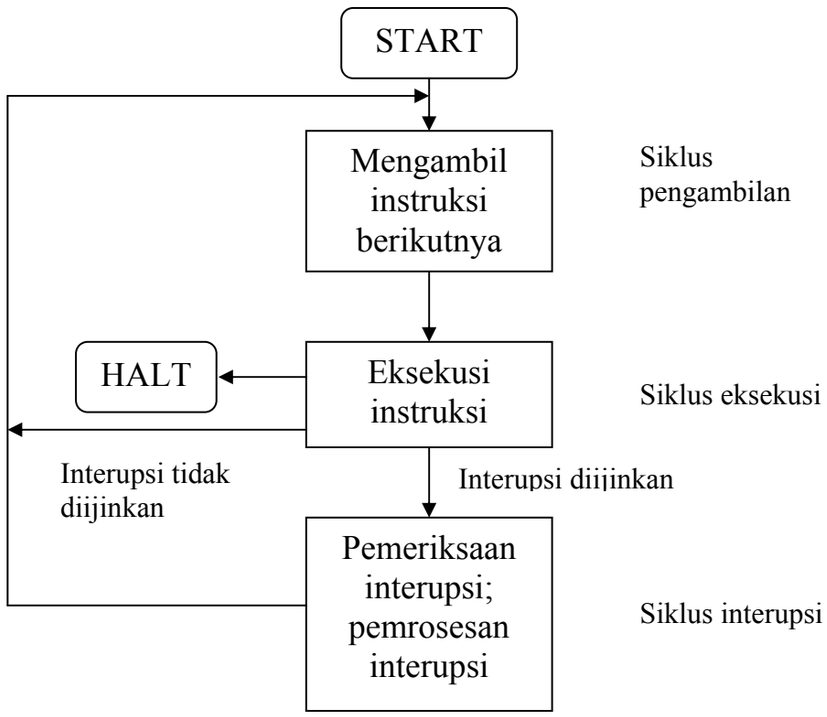
- Interupsi disediakan terutama sebagai cara untuk meningkatkan efisiensi prosesor, karena sebagian besar perangkat eksternal jauh lebih lambat dibanding prosesor.
  - Misalkan prosesor sedang melakukan pemindahan data ke printer dengan menggunakan pola siklus instruksi seperti pada gambar siklus instruksi dasar. Setiap kali setelah melakukan operasi penulisan, CPU akan berhenti dan berada dalam keadaan idle sampai printer menerima seluruh data. Lamanya berhenti dapat ratusan bahkan ribuan siklus instruksi. Jelas hal ini sangat menyia-nyiakan kemampuan prosesor. Dengan adanya interupsi, prosesor dapat diperintahkan untuk mengeksekusi instruksi-instruksi lainnya pada saat operasi-operasi I/O sedang dilaksanakan.

- Dari sudut pandang program pengguna, sebuah interupsi adalah : sebuah interupsi terhadap eksekusi normal program. Bila pengolahan interupsi telah selesai, maka eksekusi program pengguna kembali dilakukan.



Pemindahan kontrol via interupsi

- Untuk mengakomadasi interupsi, maka sebuah siklus interupsi ditambahkan ke siklus instruksi seperti pada gambar berikut :



Siklus instruksi dengan interupsi

- Eksekusi instruksi dimulai dengan siklus pengambilan dan siklus eksekusi. Pada siklus interupsi, CPU memeriksa apakah telah terjadi interupsi. Bila tidak ada interupsi, prosesor akan melanjutkan ke siklus

pengambilan berikutnya. Bila ada interupsi, prosesor melakukan hal-hal di bawah ini :

1. Prosesor menangguhkan eksekusi program yang sedang dieksekusi dan menyimpan konteksnya. Ini berarti menyimpan alamat instruksi berikutnya yang akan dieksekusi (isi PC saat itu) dan data lainnya yang relevan dengan aktivitas prosesor.
2. Prosesor menyetel PC ke alamat awal interrupt handler.

FORMAT INSTRUKSI

Op Code field	Address field

- Format instruksi di atas terdiri dari 2 bagian :
  - Bagian kode operasi (Op-code field) : digunakan untuk mengkodekan instruksi
  - Bagian alamat (address field) : digunakan untuk menunjukkan alamat operand (data).
- Jumlah bit pada op code field menentukan jumlah instruksi yang dapat dikodekan.
  - Jumlah instruksi yang dapat dikodekan = 2^n , n adalah jumlah bit pada op code field.
  - Misalkan jumlah bit pada op code field = 4, maka ada 16 instruksi yang dapat dikodekan dengan format instruksi tersebut.

- Jumlah bit pada address field menentukan jumlah alamat yang dapat diakses.
  - Jumlah alamat yang dapat diakses = 2^n , n adalah jumlah bit pada address field.
  - Misalkan jumlah bit pada address field = 4, maka ada 16 alamat yang dapat diakses dengan format instruksi tersebut.

- Agar jumlah instruksi yang dapat dikodekan lebih banyak, jumlah bit pada op code field harus lebih banyak. Agar alamat yang dapat diakses lebih banyak, jumlah bit pada address field harus lebih banyak.
- Op code dan address field yang lebih panjang akan menyebabkan instruksi yang lebih panjang.

- Untuk menyimpan format instruksi yang lebih panjang dibutuhkan lokasi memori yang lebih banyak.
  - Contoh :
    - ✓ Jumlah data yang bisa disimpan pada tiap lokasi memori = 8 bit =1 byte.
    - ✓ Misalkan panjang instruksi 1 = 16 bit = 2 byte, maka instruksi tersebut membutuhkan 2 lokasi memori.
    - ✓ Misalkan instruksi 2 panjangnya 32 bit = 4 byte, maka instruksi tersebut membutuhkan 4 lokasi memori.

Format instruksi 1-alamat

Op Code field	Address field

- Pada format instruksi 1-alamat jumlah address field adalah satu, berarti ada 1 alamat operand (data) yang dapat secara langsung dinyatakan pada instruksi.

Format instruksi 2-alamat

Op Code field	Address field 1	Address field 2

- Pada format instruksi 2-alamat, ada 2 address field, berarti ada 2 alamat operand (data) yang dapat secara langsung dinyatakan pada instruksi.

- Dengan format instruksi yang lebih kompleks (jumlah alamat lebih banyak), jumlah instruksi yang diperlukan untuk melakukan suatu operasi akan lebih sedikit.

Contoh-contoh instruksi 1-alamat

- Pada instruksi 1-alamat, ada 1 operand yang dinyatakan secara eksplisit. Operand lainnya secara implisit terletak di Accumulator (AC). Accumulator juga berfungsi menyimpan hasil operasi.
  - ❖ ADD X $\Longleftrightarrow$ AC $\longleftarrow$ AC + [X]

- SUB X ⟺ AC ⟵ AC - [X]
- MUL X ⟺ AC ⟵ AC * [X]
- DIV X ⟺ AC ⟵ AC / [X]
- LOAD X ⟺ AC ⟵ [X]
- STORE X ⟺ X ⟵ [AC]

Contoh-contoh instruksi 2-alamat

- Pada instruksi 2-alamat, ada 2 operand yang dinyatakan secara eksplisit.
  - ADD X,Y ⟺ X ⟵ [X] + [Y]
  - SUB X,Y ⟺ X ⟵ [X] - [Y]
  - MUL X,Y ⟺ X ⟵ [X] * [Y]
  - DIV X,Y ⟺ X ⟵ [X] / [Y]
  - MOVE X,Y ⟺ [X] ⟵ [Y]

Contoh-contoh instruksi 3-alamat

- Pada instruksi 3-alamat, ada 3 operand yang dinyatakan secara eksplisit.
  - ADD X,Y,Z ⟺ X ⟵ [Y] + [Z]
  - SUB X,Y,Z ⟺ X ⟵ [Y] - [Z]
  - MUL X,Y,Z ⟺ X ⟵ [Y] * [Z]
  - DIV X,Y,Z ⟺ X ⟵ [Y] / [Z]