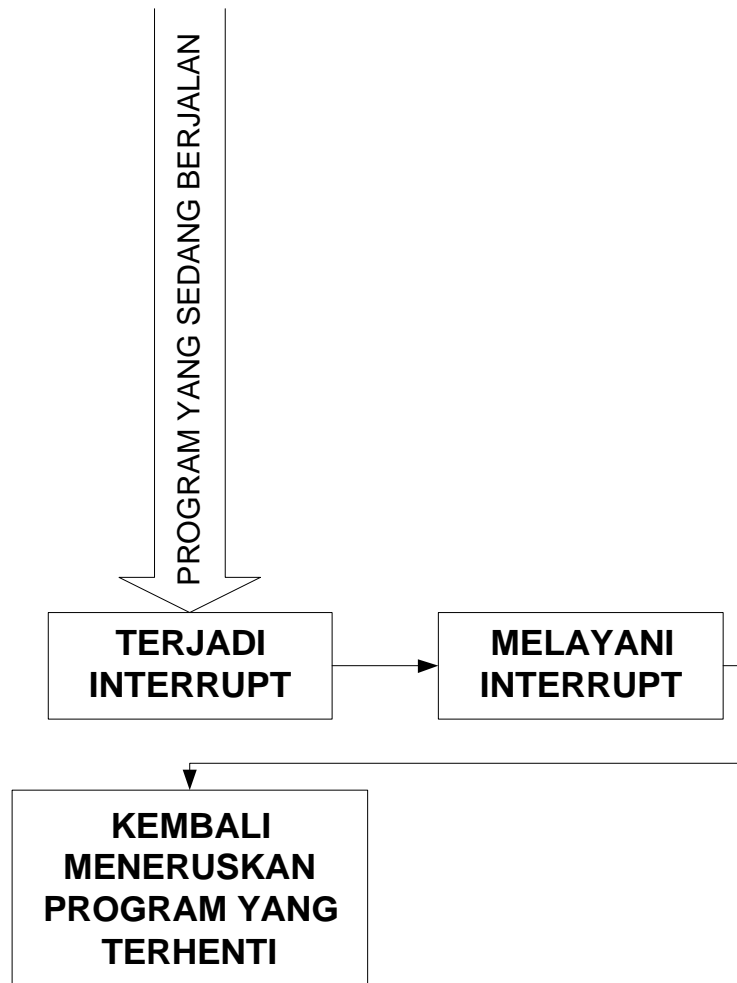


1. Interrupt

Interrupt adalah suatu kejadian atau peristiwa yang menyebabkan mikrokontroler berhenti sejenak untuk melayani interrupt tersebut. Program yang dijalankan pada saat melayani interrupt disebut **Interrupt Service Routine**. Analoginya adalah sebagai berikut, seseorang sedang mengetik laporan, mendadak telephone berdering dan menginterrupsi orang tersebut sehingga menghentikan pekerjaan mengetik dan mengangkat telephone. Setelah pembicaraan telephone yang dalam hal ini adalah merupakan analogi dari Interrupt Service Routine selesai maka orang tersebut kembali meneruskan pekerjaannya mengetik.

Demikian pula pada sistem mikrokontroler yang sedang menjalankan programnya, saat terjadi interrupt, program akan berhenti sesaat, melayani interrupt tersebut dengan menjalankan program yang berada pada alamat yang ditunjuk oleh vektor dari interrupt yang terjadi hingga selesai dan kembali meneruskan program yang terhenti oleh interrupt tadi. Seperti yang terlihat Gambar 4.1, sebuah program yang seharusnya berjalan terus lurus, tiba-tiba terjadi interrupt dan harus melayani interrupt tersebut terlebih dahulu hingga selesai sebelum ia kembali meneruskan pekerjaannya.



Gambar 4.1
Interrupt

Proses yang dilakukan oleh mikrokontroler saat melayani interrupt adalah sebagai berikut:

Proses yang terjadi saat mikrokontroler melayani interrupt adalah sebagai berikut:

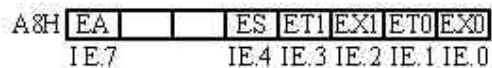
- Instruksi terakhir yang sedang dijalankan diselesaikan terlebih dahulu
- Program Counter (alamat dari instruksi yang sedang berjalan) disimpan ke stack
- Interrupt Status disimpan secara internal
- Interrupt dilayani sesuai peringkat dari interrupt (lihat Interrupt Priority)
- Program Counter terisi dengan alamat dari vector interrupt (lihat Interrupt Vector) sehingga mikrokontroler langsung menjalankan program yang terletak pada vector interrupt

Program pada vector interrupt biasanya diakhiri dengan instruksi RETI di mana pada saat ini proses yang terjadi pada mikrokontroler adalah sebagai berikut:

- Program Counter diisi dengan alamat yang tersimpan dalam stack pada saat interrupt terjadi sehingga mikrokontroler kembali meneruskan program di lokasi saat interrupt terjadi
- Interrupt Status dikembalikan ke kondisi terakhir sebelum terjadi interrupt

4.1. Enable Interrupt

Dalam suatu kondisi dapat juga dibutuhkan suatu program yang sedang berjalan tidak boleh diinterrupt, untuk itu 89C51 mempunyai lima buah interrupt yang masing-masing dapat dienable ataupun disable satu per satu. Pengaturan enable dan disable interrupt dilakukan pada Register Interrupt Enable yang terletak pada alamat A8H.



EA: Disable semua interrupt apabila bit ini clear. Bila bit ini clear, maka apapun kondisi bit lain dalam register ini, semua interrupt tidak akan dilayani, oleh karena itu untuk mengaktifkan salah satu interrupt, bit ini harus set

ES: Enable/disable Serial Port Interrupt, set = enable, clear = disable
Apabila Serial Port Interrupt aktif maka interrupt akan terjadi setiap ada data yang masuk ataupun keluar melalui serial port yang membuat Flag RI (Receive Interrupt Flag) ataupun TI (Transmit Interrupt Flag).

ET1: Enable/disable Timer 1 Interrupt, set = enable, clear = disable
Apabila interrupt ini enable maka interrupt akan terjadi pada saat Timer 1 overflow.

EX1: Enable/disable External Interrupt 1, set = enable, clear = disable
Apabila interrupt ini enable maka interrupt akan terjadi pada saat terjadi pulsa low pada INT1

ET0: Enable/disable Timer 0 Interrupt, set = enable, clear = disable

Apabila interrupt ini enable maka interrupt akan terjadi pada saat Timer 0 overflow.

EX0: Enable/disable External Interrupt 0, set = enable, clear = disable
Apabila interrupt ini enable maka interrupt akan terjadi pada saat terjadi pulsa low pada INT0

4.2. Status Interrupt

Status-status interrupt dari 89C51 terletak pada Register TCON yaitu
INT0: Bit IE0, clear oleh hardware saat interrupt terjadi pada mode aktif level
INT1: Bit IE1, clear oleh hardware saat interrupt terjadi pada mode aktif level
Timer 0: Bit TF0, clear oleh hardware saat interrupt terjadi
Timer 1 : Bit TF1, clear oleh hardware saat interrupt terjadi
Serial Port (TXD) : Bit TI, clear oleh software
Serial Port (RXD) : Bit RI, clear oleh software

External Interrupt 0 maupun External Interrupt 1 dapat diatur menjadi aktif level maupun aktif transisi dengan mengubah bit IT0 atau IT1 pada Register TCON
External Interrupt akan bekerja secara aktif level bila bit IT_x (x = 0 untuk INT0 dan x =1 untuk INT1) berkondisi low dan bekerja secara aktif transisi bila bit IT_x berkondisi high.

4.3. Interrupt Priority

Dalam melayani interrupt, mikrokontroler bekerja berdasarkan prioritas yang dapat diatur pada Register Interrupt Priority.

IP0 atau PX0 untuk External Interrupt 0
IP1 atau PT0 untuk Timer 0 Interrupt
IP2 atau PX1 untuk External Interrupt 1
IP3 atau PT1 untuk Timer 1 Interrupt
IP4 atau PS untuk Serial Interrupt

Bit-bit ini akan berkondisi set apabila interrupt yang diaturnya ditempatkan pada prioritas yang tinggi. Interrupt dengan prioritas yang tinggi dapat meng-interrupt interrupt lain yang mempunyai prioritas lebih rendah, sedangkan interrupt dengan prioritas tinggi itu sendiri tidak dapat di interrupt oleh interrupt lain.

Apabila terjadi lebih dari satu interrupt yang mempunyai prioritas yang sama secara bersamaan, maka prioritas akan diatur secara *polling* mulai dari:

- External Interrupt 0
- Timer 0 Interrupt
- External Interrupt 1
- Timer 1 Interrupt
- Serial Interrupt

4.3. Interrupt Vector

Interrupt Vector adalah harga yang disimpan ke Program Counter pada saat terjadi interrupt sehingga program akan menuju ke alamat yang ditunjukkan oleh Program Counter. Pada saat program menuju ke alamat yang ditunjuk oleh Interrupt Vector maka flag-flag yang set karena terjadinya interrupt akan di-clear kecuali RI dan TI.

Kelima interrupt dan system reset dari 89C51 mempunyai Vector masing-masing yang dapat dilihat pada Tabel 4.1

Table 4.1
Interrupt Vector

INTERRUPT	FLAG	ALAMAT VEKTOR
System Reset	RST	0000H
External Interrupt 0	I E0	0003H
Timer 0	TF0	000BH
External Interrupt 1	I E1	0013H
Timer 1	TF1	001BH
Serial Port	RI atau TI	0023H

Masing-masing alamat vektor mempunyai jarak yang berdekatan sehingga akan timbul masalah bila diperlukan sebuah Interrupt Service Routine yang cukup panjang, misalnya hendak digunakan External Interrupt 0 dan Timer 0 dalam satu sistem, maka bila Interrupt Service Routine untuk External Interrupt 0 diletakkan pada alamat 0003H dan Interrupt Service Routine untuk Timer 0 diletakkan pada alamat 000BH akan terjadi bentrok alamat antara kedua Interrupt Service Routine ini apabila tidak dilakukan suatu trik berikut yang terlihat pada listing berikut.

Listing 4.1

```

ORG 0000H
LJMP Start
ORG 0003H
LJMP Int0
ORG 000BH
LJMP Timer0
.....
.....
.....
Start:
.....
..... ;Main Program
.....

Int0:
.....
..... ;Int0 ISR
.....
RETI

Timer0:
.....
..... ;Timer 0 ISR
.....
RETI

```

Jadi pada listing ini, saat terjadi interrupt Program Counter memang tetap berisi nilai dari Interrupt Vector sehingga program juga meloncat ke alamat tersebut, namun karena di alamat tersebut sudah diletakkan instruksi untuk meloncat ke label yang lain seperti Int0 untuk External Interrupt 0 Service Routine maka tidak akan terjadi bentrok alamat antara kedua Interrupt Service Routine ini.

