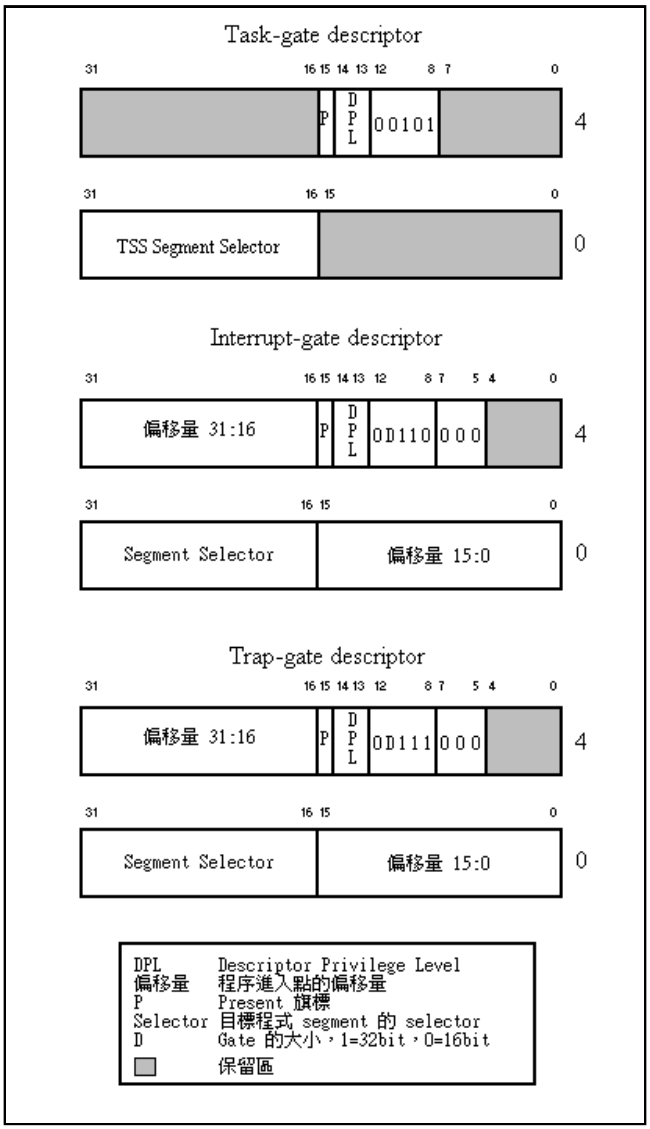
中斷描述表

概論

在發生中斷（或例外）時，處理器由中斷描述表（Interrupt Descriptor Table，IDT）中查出該中斷向量的 gate descriptor，才能找到該中斷向量的處理程序。IDT 就如同 GDT 或 LDT 一樣，是由一堆 descriptor 組成的陣列。IDT 的基底位址存放在 IDTR 中，包含一個 32 bit 的線性位址和一個 16 bit 的邊界值。邊界值存放的是最後一個有效位址的值，因此，若邊界值為 0，則有一個有效位元組。因為 descriptor 必定是 8 bytes，因此 IDTR 中的邊界值一定要是 8N-1 的形式。IDT 中並不一定要存放 256 個 descriptor，只需要存放有用到的數目就可以了。如果有一中斷的中斷向量超出 IDT 的範圍，則處理器會發出 general-protection（#GP）的例外。
利用 LIDT 指令可以把記憶體中的值載入到 IDTR 中，而用 SIDT 指令可以把 IDTR 中的位址存到記憶體中。這兩個指令只能在 ring 0 中執行。

IDT 中的 descriptor

在 IDT 中可以有三種 descriptor：task-gate descriptor、interrupt-gate descriptor、和 trap-gate descriptor。在 IDT 中的 gate descriptor 的格式就和在 GDT 中所使用格式相同。其中，interrupt-gate 和 trap-gate 的 descriptor 和 call-gate descriptor 的格式很像（參考「保護機制」的「特權等級」）。而 task-gate descritor 則是指向中斷處理程序的 TSS segment。它們的格式如下：



在中斷發生時，若在 IDT 中相對映的 descriptor 是 interrupt-gate 或 trap-gate 的 descriptor，則處理器的行為會和用 CALL 在呼叫一個 call-gate 的時候很類似。而若相對映的 descriptor 是 task-gate descriptor，則處理器的行為會類似用 CALL 呼叫一個 task-gate（參考「多

工處理」）。

Interrupt gate 和 Trap gate

如果中斷的 gate 是 interrupt gate 或 trap gate，則表示這是一個在目前執行的工作中執行的中斷處理程序。也就是說，這個中斷處理程序並不會進行工作切換。這時候，處理器就如同是經由 call gate 來呼叫一個程序一般。
發生中斷後，處理器會依序將 EFLAGS、CS、和 EIP 推入堆疊中。如果這是一個有錯誤碼的例外，則還會把錯誤碼推入堆疊中。若中斷處理程序的特權等級和 CPL 相同，則不會有堆疊切換。而如果中斷處理程序的特權等級較高，則處理器會切換堆疊，並把 SS、SP 也推入堆疊中（會在推入 EFLAGS 等之前推入），並把所有推入堆疊中的資料複製到新的堆疊中。
由中斷處理程序中返回時，必須使用 IRET 指令。IRET 和 RET 指令不同的地方，即在於它會取回 EFLAGS 的值。不過，EFLAGS 中的 IOPL 只有在 CPL 為 0 時，才會改變。而 EFLAGS 中的 IF 旗標只有在 CPL 小於或等於 IOPL 時，才會改變。如果在中斷時有切換堆疊，IRET 也會把堆疊切換回來。
Trap gate 和 interrupt gate 的唯一不同點，是在處理旗標的方式。當中斷（或例外）是經由 interrupt gate 時，處理器會把 IF 設為 0，以避免在
中斷處理程序中再度發生中斷。在返回時，處理器會恢復 IF 的值。但是，若中斷是經由 trap gate，則處理器不會改變 IF 的值。也就是說，在中斷處理程序中有可能會再度發生中斷。

Task gate

若中斷是經由 task gate，則在中斷發生時，會導致工作切換。利用工作切換的中斷處理程序有幾個好處：

- 在工作切換時，被中斷的程式或工作的狀態會自動被儲存起來。
- 切換到新的 TSS 可以讓中斷處理程式一律使用新的堆疊。如果在 CPL 為 0 的程序中的堆疊已經損毀了，那麼一般的中斷處理程序並不會切換堆疊（因為特權等級相同），而可能會使得中斷處理程序無法執行。
- 中斷處理程序可以指定新的 LDT，而擁有自己的定址空間。

經由 task gate 的中斷處理程序的主要缺點，就是在每次中斷時的工作切換會花費很多時間，而減低了系統的效率。

中斷處理程序的保護

和一般的程序一樣，處理器不允許進入特權等級較低的中斷處理程序中。如果發生中斷時，會進入特權等級比 CPL 更低的中斷處理程序中，則處理器會發出 general-protection（#GP）例外。
和處理一般的程序不同的是，中斷向量中並沒有 RPL，所以處理器並不會檢查 RPL。而且，處理器只有在中斷（或例外）是由 INT n、INT3、INTO 指令產生的時候，才會檢查 gate 的 CPL 是否比 DPL 小，這可以避免等級較低的程序利用這些指令破壞系統的中斷處理程序（如分頁錯誤的處理程序）。對硬體產生的中斷（或例外），處理器並不會檢查 gate 的 DPL。
因為中斷隨時都可能發生，因此這些限制可能會導致不適當的錯誤。下面有兩個方法，都可以避免這個問題：

- 如果中斷處理程序只需要處理堆疊中的資料（例如，除法錯誤的處理程序），則可以把這個程序放到 conforming 的程式 segment 中。
- 如果中斷處理程序需要處理內部的資料 segment，則可以把它放到 nonconforming 且權限為 0 的程式 segment 中。這樣一來，任何情形下發生中斷，都不會有違反保護規則的情形。