5 Interrupts

Interrupts dienen der schnellen Reaktion auf ***externe*** Ereignisse durch Unterbrechung eines gerade laufenden Programms; sie sind in „Single Task"-Mikroprozessor-Systemen schon seit langem bekannt.

Darüber hinaus erlaubt die Multitasking-Architektur des 80486 jeder einzelnen Task, auf alle Informationen über externe Ereignisse wie ein Single Task-System zu reagieren.

Beim 80486 kann ein Interrupt sowohl durch ein ***externes*** als auch durch ein ***internes*** Ereignis ausgelöst werden, das unabhängig von der Ausführung des augenblicklichen Programms auftritt. Dabei werden

- ***externe Interrupts*** entweder durch den INTR (Interrupt Request)-Eingang oder den NMI (Non-maskable Interrupt)-Eingang der CPU erzeugt, während
- ***interne Interrupts*** durch die Befehle INT n, INT 3, INTO und BOUND ausgelöst werden können.

Die Prozedur zur Beantwortung eines Interrupts kann entweder

- im „Context der Task", in der der Interrupt aufgetreten ist oder
- in einer separaten Task

ausgeführt werden.

Im ersten Fall bedeutet dies, daß das gleiche Task-Status-Segment, die gleiche LDT und dieselben Stack-Segmente benützt werden.

Die Wahl der Task zur Interruptbehandlung hängt davon ab,

- welche Funktion mit der Interruptbehandlung erfüllt werden soll
- welche Isolations-Ebene gefordert ist.

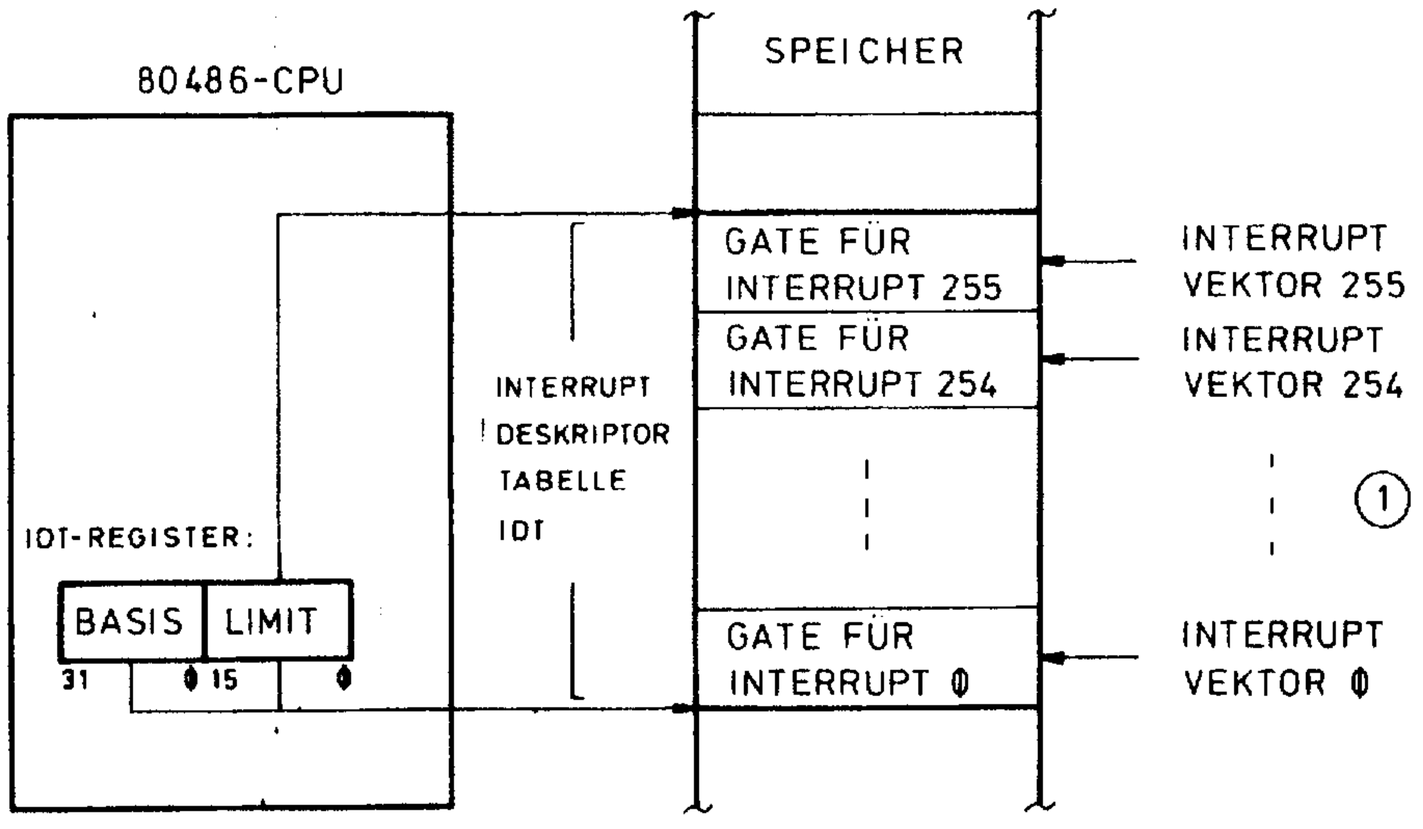
5.1 Interrupt-Deskriptor-Tabelle IDT

Da verschiedenartige Ereignisse einen Interrupt auslösen können, ist es erforderlich, jede Interrupt-Anforderung eindeutig zu identifizieren. Aus diesem Grund erhält jede Interrupt-Quelle eine Kennziffer, die als ***Interrupt-Vektor*** bezeichnet wird. Insgesamt werden 256 Interrupt-Vektoren unterschieden, wobei jeder auf einen bestimmten Eintrag in der sogenannten ***I***nterrupt-***D***eskriptor-***T***abelle (IDT) zeigt ①.

Diese Tabelle wird vom 32-Bit-Basisadreß-Feld und vom 16-Bit-Limit-Feld des On-Chip-IDT-Registers beschrieben.

Da nur eine einzige IDT im System existieren kann, wird normalerweise das IDT-Register während der System-Initialisierung mit dem LIDT-Befehl in der Privileg-Ebene 0 geladen. Allerdings kann das Betriebssystem den LIDT-Befehl benützen, um die augenblickliche IDT durch eine andere IDT zu ersetzen.

Jeder IDT-Eintrag ist ein 4-Wort Gate-Deskriptor, der einen Zeiger zur Interrupt-Prozedur („Interrupt Handler“) enthält. Das folgende Bild zeigt das Layout einer kompletten Interrupt-Deskriptor-Tabelle mit 256 Gate-Deskriptoren.



Die IDT muß nicht alle 256 Gate-Deskriptoren enthalten. Das Limit-Feld im IDT-Register erlaubt es, daß auch weniger eingetragen werden können.

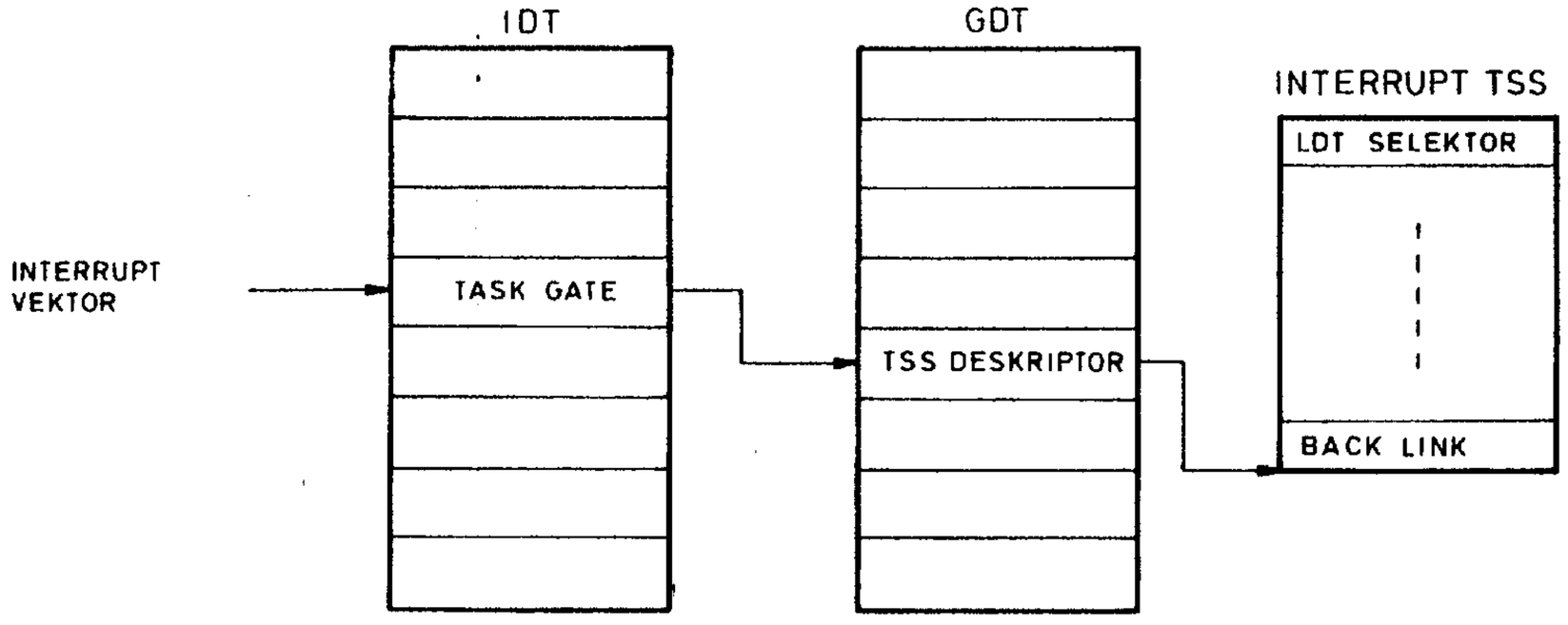
5.2 Interrupt Task

Als Antwort auf ein Ereignis unterbricht der Prozessor die augenblicklich aktive Task und beginnt die Ausführung eines vom zugehörigen IDT-Gate-Deskriptor identifizierten Befehls.

Die vom Ereignis aktivierten Befehle können

- entweder von einer ***anderen*** Task als der augenblicklich ausgeführten oder
- von einer Prozedur ***innerhalb*** der augenblicklich ausgeführten Task bearbeitet werden.

An dieser Stelle soll zunächst der erste Fall behandelt werden. Wenn die Interrupt-Quelle einen Task-Gate-Deskriptor identifiziert, der auf ein Task-Status-Segment zeigt, hat dies zur Folge, daß der Prozessor einen Task-Wechsel auslöst. Das folgende Bild illustriert einen solchen Vorgang:

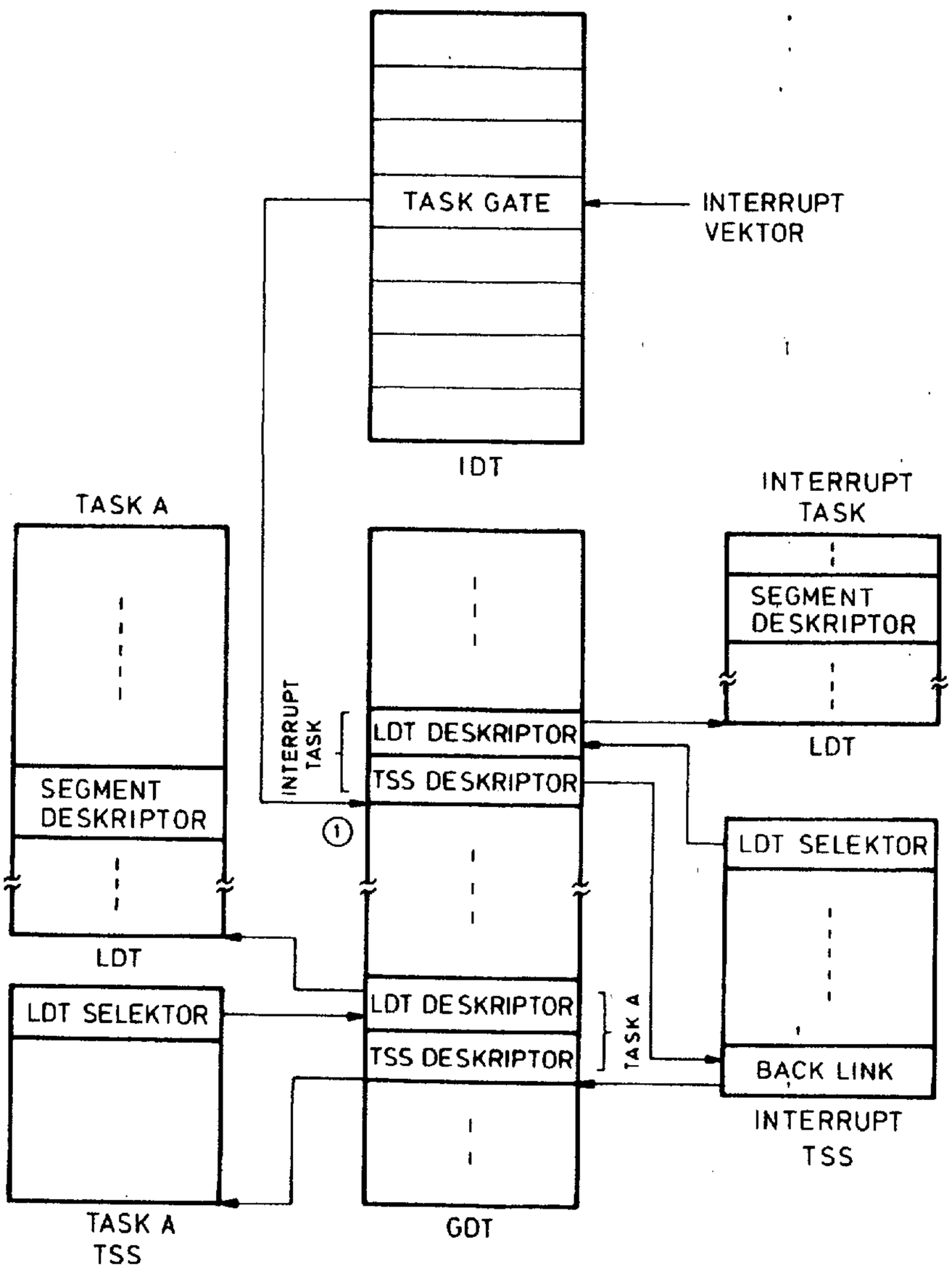


Es ist ersichtlich, daß der Selektor im Task-Gate-Deskriptor (in der IDT) einen TSS-Deskriptor (in der GDT) auswählt, der dann das Task-Status-Segment (Interrupt-TSS) einer sogenannten *„Interrupt Task"* beschreibt. Die Felder der selektierten Interrupt-TSS werden nun benützt, um die eintretende „Interrupt Task" auszuführen.

Die Benützung eines Task Gate-Deskriptors hat ***zwei*** Vorteile:

1. Als Teil der Antwort auf ein externes oder internes Ereignis wird eine Task-Wechsel-Operation durchgeführt. Dies bedeutet, daß automatisch der augenblickliche Status der unterbrochenen Task in ihrem Task-Status-Segment abgelegt wird, so daß er nach einer Rückkehr wieder komplett zur Verfügung steht.
2. Die neue Task („Interrupt Task") ist vollständig von der unterbrochenen und ausscheidenden Task isoliert und wird von der Privileg-Ebene der ausscheidenden Task ***nicht*** beeinflußt.

Das folgende Bild illustriert als Beispiel eine Unterbrechung von Task A durch einen Interrupt-Vektor und den daraus resultierenden Wechsel in die „Interrupt Task". Dabei benützt der Interrupt-Vektor einen Task-Gate-Deskriptor.



Hat der Interrupt-Vektor ein gültiges Task-Gate in der IDT selektiert, wird der im Gate enthaltene Selektor mit TI = 0 (GDT) den TSS-Deskriptor der eintretenden „Interrupt Task“ auswählen ①. Ab diesem Zeitpunkt laufen die üblichen Aktionen ab. Es wird zunächst der augenblickliche Task-A-Status im Task-A-TSS zwischengespeichert und anschließend der Inhalt der Interrupt-TSS in die CPU übertragen, so daß sie mit der Ausführung der „Interrupt Task“ beginnen kann.

5.3 Interrupt-Prozedur

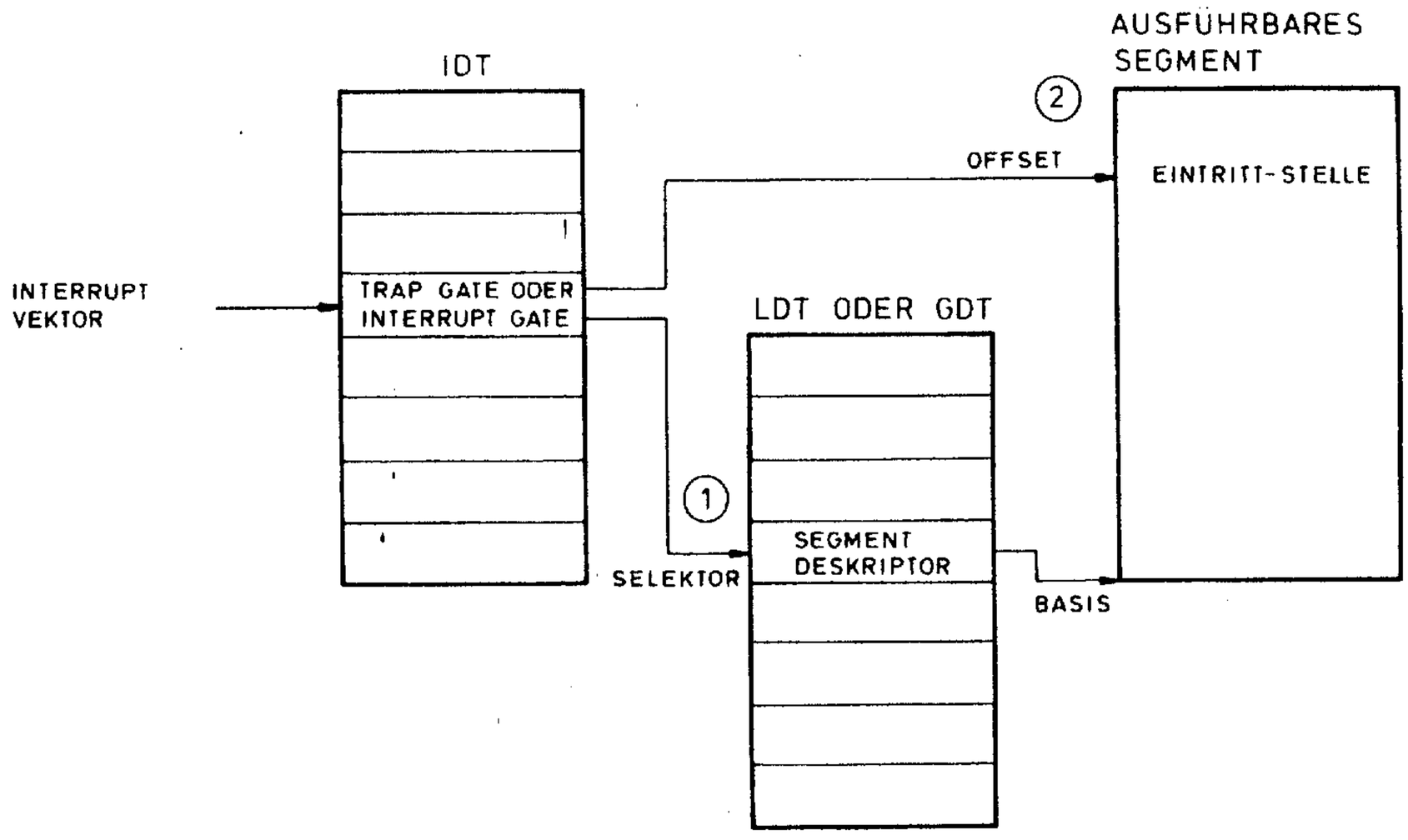
Neben Task-Gate-Deskriptoren kann die IDT noch zwei weitere Deskriptor-Typen, nämlich

- „Interrupt Gates“
- „Trap Gates“

enthalten.

Immer dann, wenn ein Interrupt-Vektor einen derartigen Deskriptor-Typ identifiziert, wird ***kein*** Task-Wechsel ausgelöst. Statt dessen verhält sich der Prozessor so, als wenn er innerhalb der augenblicklichen Task über ein „Call Gate“ eine Prozedur aufrufen würde.

Die über ein „Interrupt Gate“ oder „Trap Gate“ aufgerufene Prozedur wird als ***Interrupt-Prozedur*** bezeichnet. Die Hardware-Verbindungen einer so identifizierten Interrupt-Prozedur zeigt das folgende Bild:

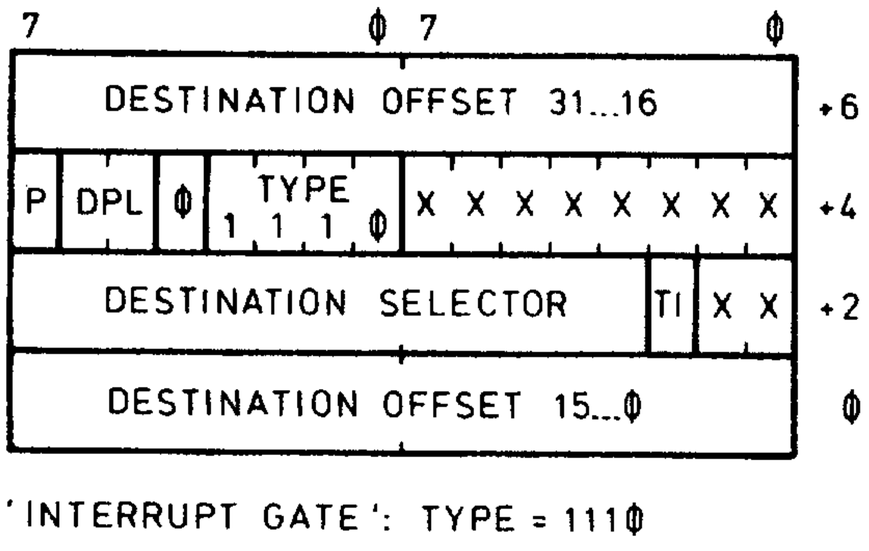


Es ist ersichtlich, daß sowohl der Interrupt-Gate- als auch der „Trap Gate“-Deskriptor je ein Selektor- und ein Offset-Feld enthalten.

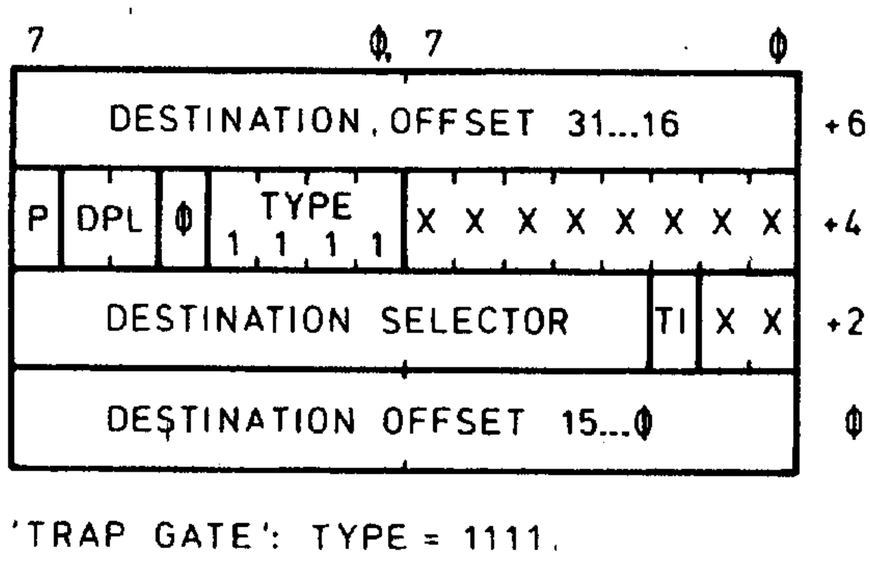
Hat ein Interrupt-Vektor einen solchen Gate-Deskriptor identifiziert, wird die Selektor-Komponente in der LDT oder GDT den Deskriptor eines ausführbaren Segments auswählen ① und die Offset-Komponente die Eintritt-Stelle in die Prozedur markieren ②.

5.4 Interrupt Gate- und Trap Gate-Deskriptoren

Die folgenden beiden Bilder zeigen die kompletten Layouts der Interrupt-Gate- und Trap-Gate-Deskriptoren des 80486:



'INTERRUPT GATE': TYPE = 1110



'TRAP GATE': TYPE = 1111

HINWEIS:
Die in den Deskriptoren angegebenen X-Felder werden nicht benützt.

Destination-Selector-Feld
Dieses Feld enthält einen Selektor, der entweder in der GDT (TI = 0) oder in der LDT (TI = 1) einen Deskriptor für ein ausführbares Code-Segment auswählt. Das RPL-Feld in diesem Selektor wird ***nicht*** benützt (XX).

Destination-Offset-Felder
Beide Felder enthalten zusammen einen Zeiger, der die Anfangsadresse der auszuführenden Prozedur im ausgewählten Ziel-Code-Segment kennzeichnet.

Present-Bit
Wie Call-Gate-Deskriptoren beziehen sich auch Interrupt Gate- oder Trap-Gate-Deskriptoren ***nicht*** auf ein „normales" Segment.
Daher hat das Present-Bit in diesen Deskriptoren die gleiche Bedeutung wie das Present-Bit im Call-Gate-Deskriptor (siehe Kap. 3.10.3).

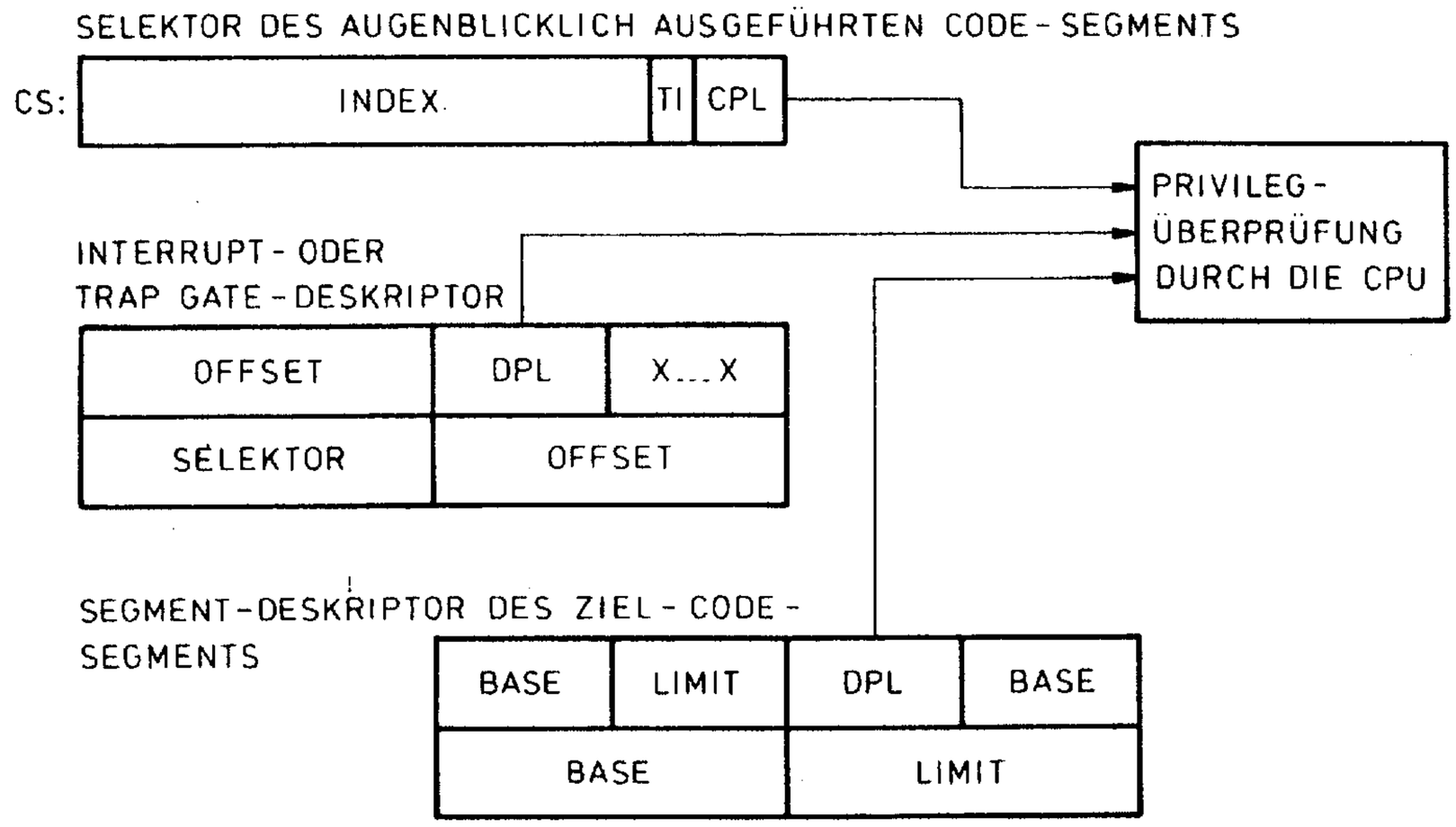
5.5 Interrupt und Gate-DPL

Das DPL-Feld eines Deskriptors in der IDT bestimmt die geforderte Privileg-Ebene, um einen der Interrupt-Befehle wie INTO, INT 3, INT n oder BOUND auszuführen. Jeder dieser Befehle liefert eine Kennziffer n im Bereich 0 ... 255 und identifiziert damit in der IDT

- einen Interrupt-Gate-Deskriptor oder
- einen Trap-Gate-Deskriptor oder
- einen Task-Gate-Deskriptor.

Der Schutzmechanismus des 80486 benützt deswegen die Privileg-Ebenen DPL der Gate-Deskriptoren, um nicht-autorisierte Programme daran zu hindern, über Interrupt-Befehle privilegierte Programme aufzurufen.

Wenn solche Aufrufe erfolgen, prüft der Prozessor ***drei*** verschiedene Privileg-Ebenen-Felder.



Wie das angegebene Bild zeigt, sind dies:

- Die augenblickliche Privilegebene (CPL) des augenblicklich ausgeführten Code-Segments.
- Die Privilegebene (DPL) des Interrupt- oder Trap-Gate-Deskriptors und
- die Privilegebene (DPL) im Segment-Deskriptor des auszuführenden Ziel-Code-Segments.

Eine Interrupt-Prozedur kann nur dann erfolgreich aufgerufen werden, wenn die folgenden Beziehungen zwischen den Privileg-Kennziffern erfüllt sind:

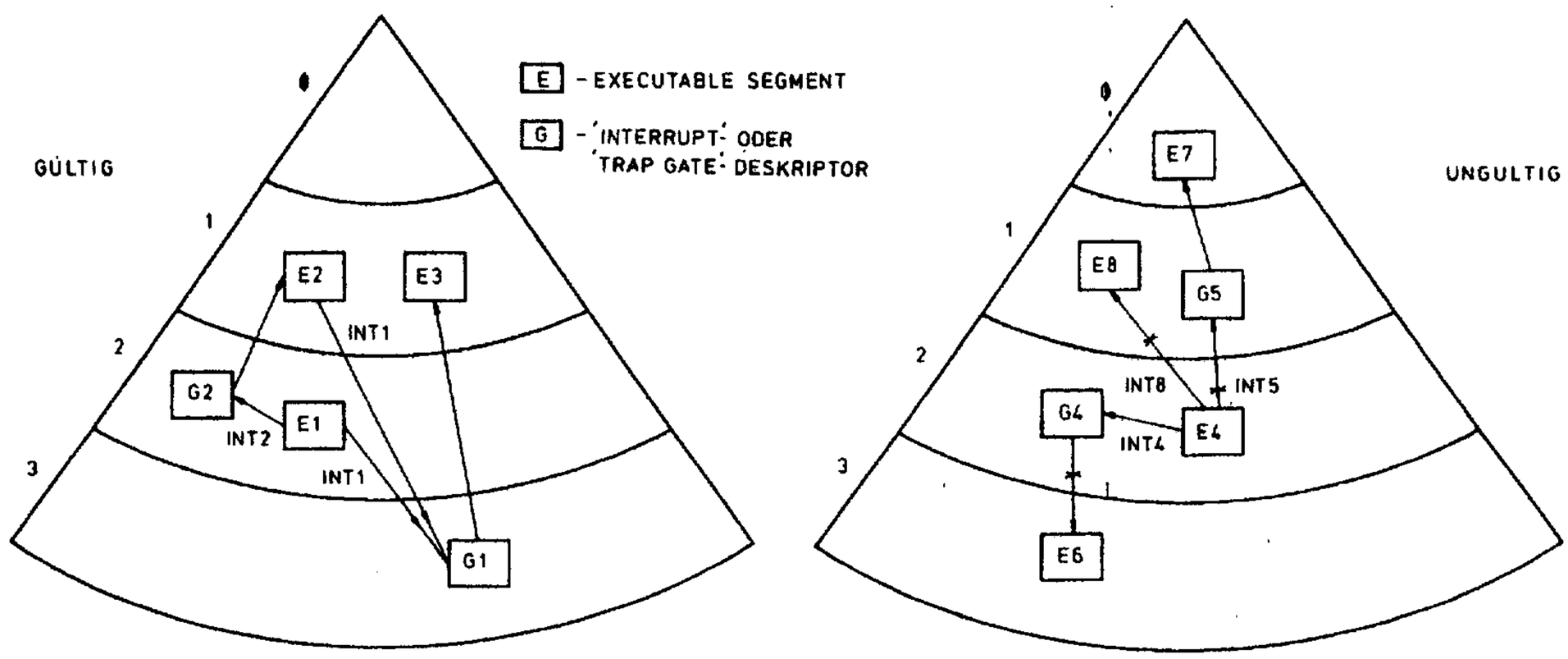
„Task“ CPL	**≤ „Interrupt- oder Trap-Gate“ DPL**
Zielsegment DPL	**≤ „Task“ CPL**

Die angegebenen Regeln zeigen, daß ein Interrupt-Befehl einen Interrupt- oder Trap-Gate-Deskriptor immer dann problemlos benützen kann, wenn sich

- die Interrupt-Prozedur auf einer numerisch ***kleineren*** oder auf der ***gleichen*** Privileg-Ebene befindet, wie das durch den Interrupt-Befehl unterbrochene Code-Segment.

Sollte ein Interrupt-Befehl diese Privileg-Regeln nicht einhalten, führt das zwangsläufig zur sogenannten „allgemeinen Schutzverletzung“ und damit zur Unterbrechung des Programms.

Die angegebenen Bilder illustrieren gültige und ungültige Versuche, durch Befehle INT n Interrupt-Prozeduren über Interrupt- und Trap-Gate-Deskriptoren aufzurufen.



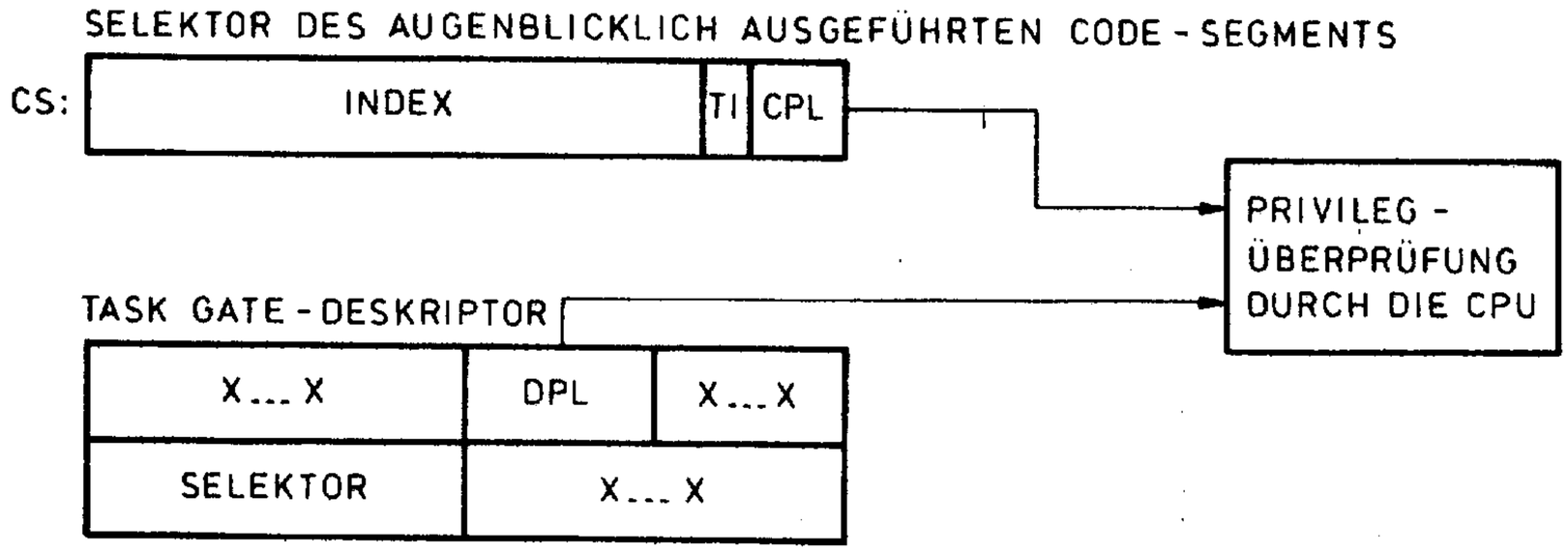
Der Befehl INT 5 in E4 kann ***nicht*** ausgeführt werden, da DPL des Gate-Deskriptors G5 numerisch kleiner ist als CPL des unterbrochenen Code-Segments E4.

Der Befehl INT 8 in E4 kann ebenfalls ***nicht*** ausgeführt werden, da der „Interrupt Handler" in E8 nur über einen Gate-Deskriptor zu erreichen ist.

Auch der „Interrupt Handler" in E6 ist vom INT 4-Befehl ***nicht*** zu erreichen, da DPL von E6 numerisch größer ist als CPL des unterbrochenen Code-Segments E4.

Nur die von den Befehlen INT 2 und INT 1 initiierten Pfade ***E***1-***G***2-***E***2, ***E***1-***G***1-***E***3 und ***E***2-***G***1-***E***3 sind ***gültig,*** da hier die oben angegebenen Regeln befolgt werden.

Wird ein Interrupt-Befehl benützt, um über einen Task-Gate-Deskriptor eine Interrupt Task aufzurufen, prüft der Prozessor ***zwei*** verschiedene Privileg-Ebenen-Felder.



Wie das angegebene Bild zeigt, sind dies:

- die augenblickliche Privilegebene (CPL) des augenblicklich ausgeführten Code-Segments und
- die Privilegebene (DPL) des Task-Gate-Deskriptors.

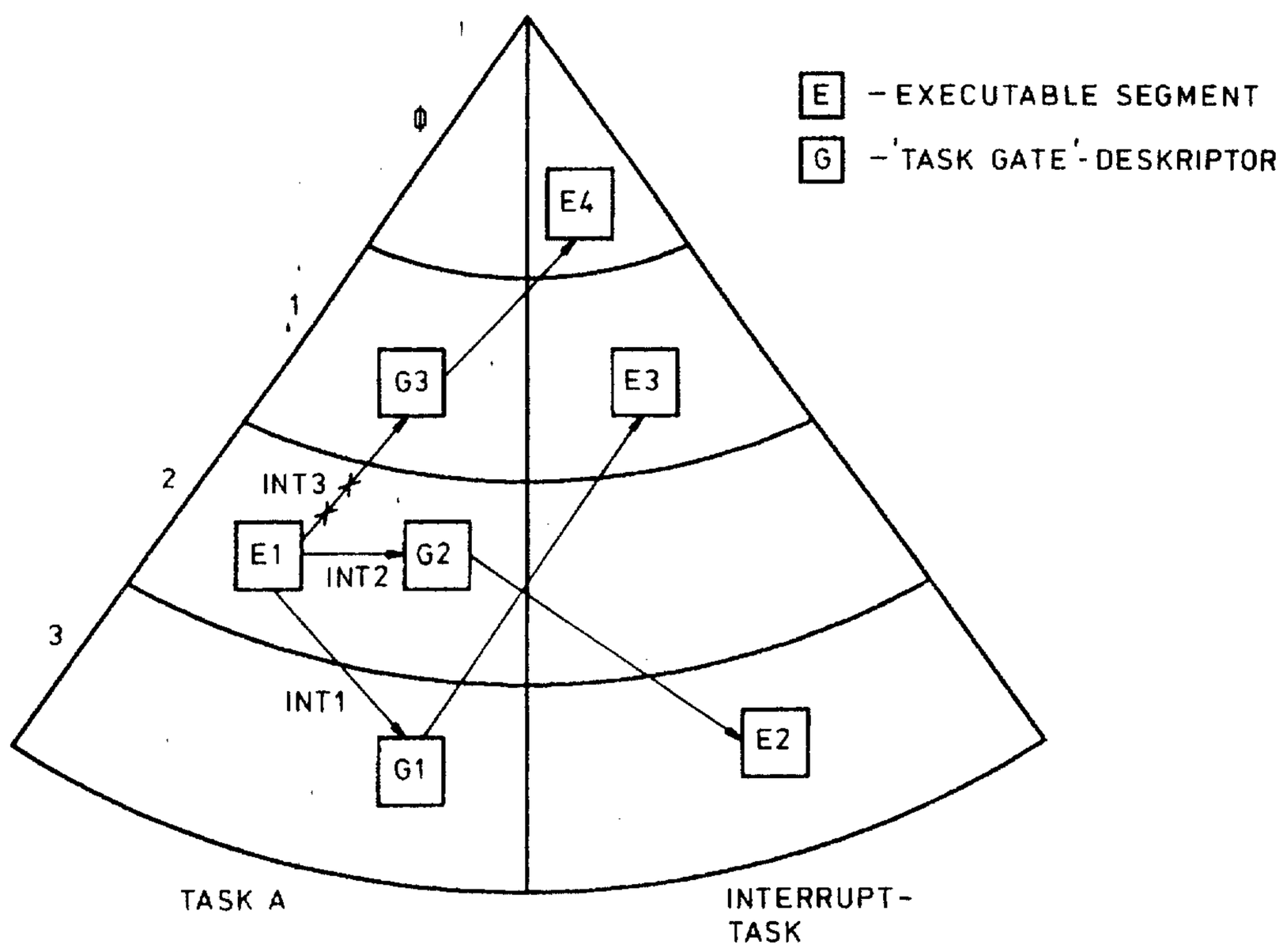
Eine Interrupt-Task kann nur dann erfolgreich aufgerufen werden, wenn die folgende Beziehung zwischen den Privileg-Kennziffern erfüllt ist:

„Task" CPL ≤ „Task Gate" DPL

Die angegebene Regel besagt, daß ein Interrupt-Befehl einen Task-Gate-Deskriptor immer dann problemlos benutzen kann, wenn sich

- das durch den Interrupt-Befehl unterbrochene Code-Segment auf einer numerisch ***kleineren*** oder auf der ***gleichen*** Privileg-Ebene befindet wie der Task-Gate-Deskriptor.

Das folgende Bild illustriert gültige und ungültige Versuche mit INT n-Befehlen einen Task-Wechsel zu realisieren:

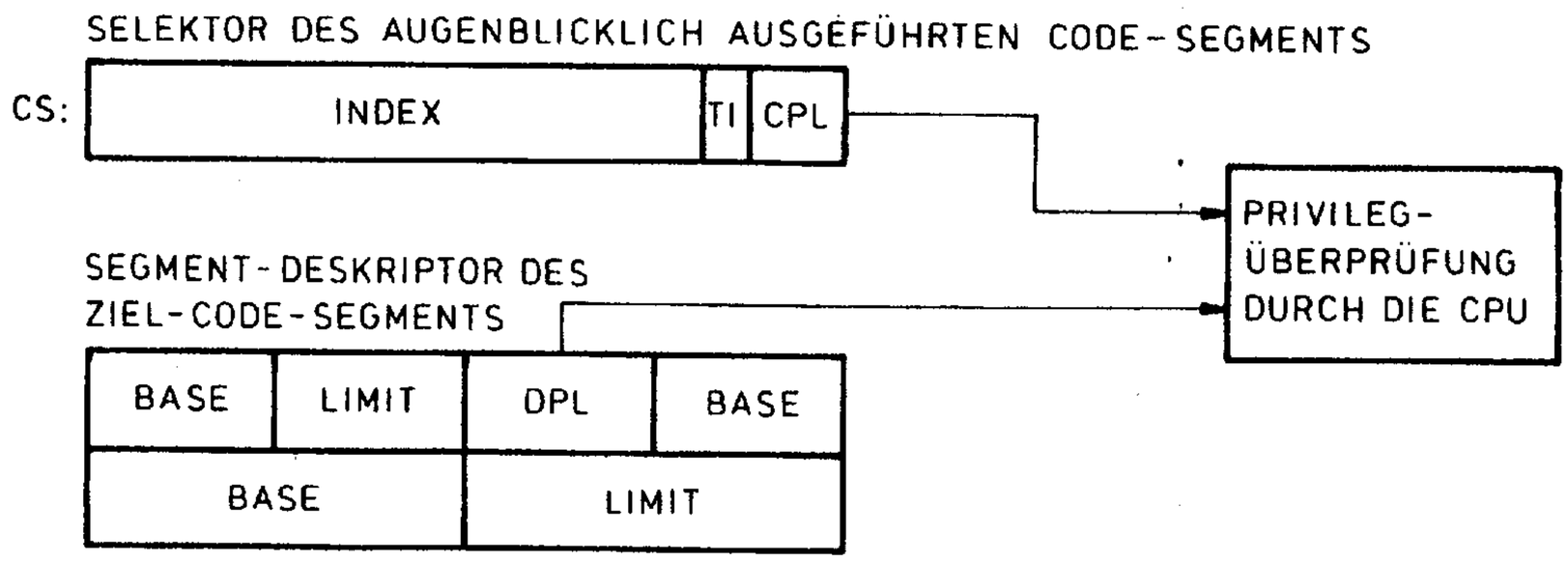


Die von den Befehlen INT 1 und INT 2 initiierten Task-Wechsel können realisiert werden, da die Pfade ***E***1-***G***1-***E***3 und ***E***1-***G***2-***E***2 gültig sind.

Der Versuch, mit INT 3 die „Interrupt Task" aufzurufen, ist ungültig, da für den Pfad ***E***1-***G***3-***E***4 die oben angegebene Bedingung nicht erfüllt ist.

Wenn ein Hardware-Interrupt eine Task unterbricht, wird die augenblickliche Privileg-Ebene CPL der Task ignoriert. Das ist deswegen so, weil sich eine externe Interrupt-Quelle naturgemäß, im Gegensatz zum Interrupt-Befehl, in keiner Task aufhält. Daher kann der Vergleich ***„Task" CPL*** $\leqslant$ ***„Gate" DPL*** nicht durchgeführt werden.

Benützt ein Hardware-Interrupt einen Interrupt-Gate- oder Trap-Gate-Deskriptor, prüft der Prozessor ***zwei*** andere Privileg-Ebenen-Felder.



Wie das angegebene Bild zeigt, sind dies:

- die augenblickliche Privileg-Ebene (CPL) des augenblicklich ausgeführten Code-Segments und
- die Privilegebene (DPL) im Segment-Deskriptor des auszuführenden Ziel-Code-Segments.

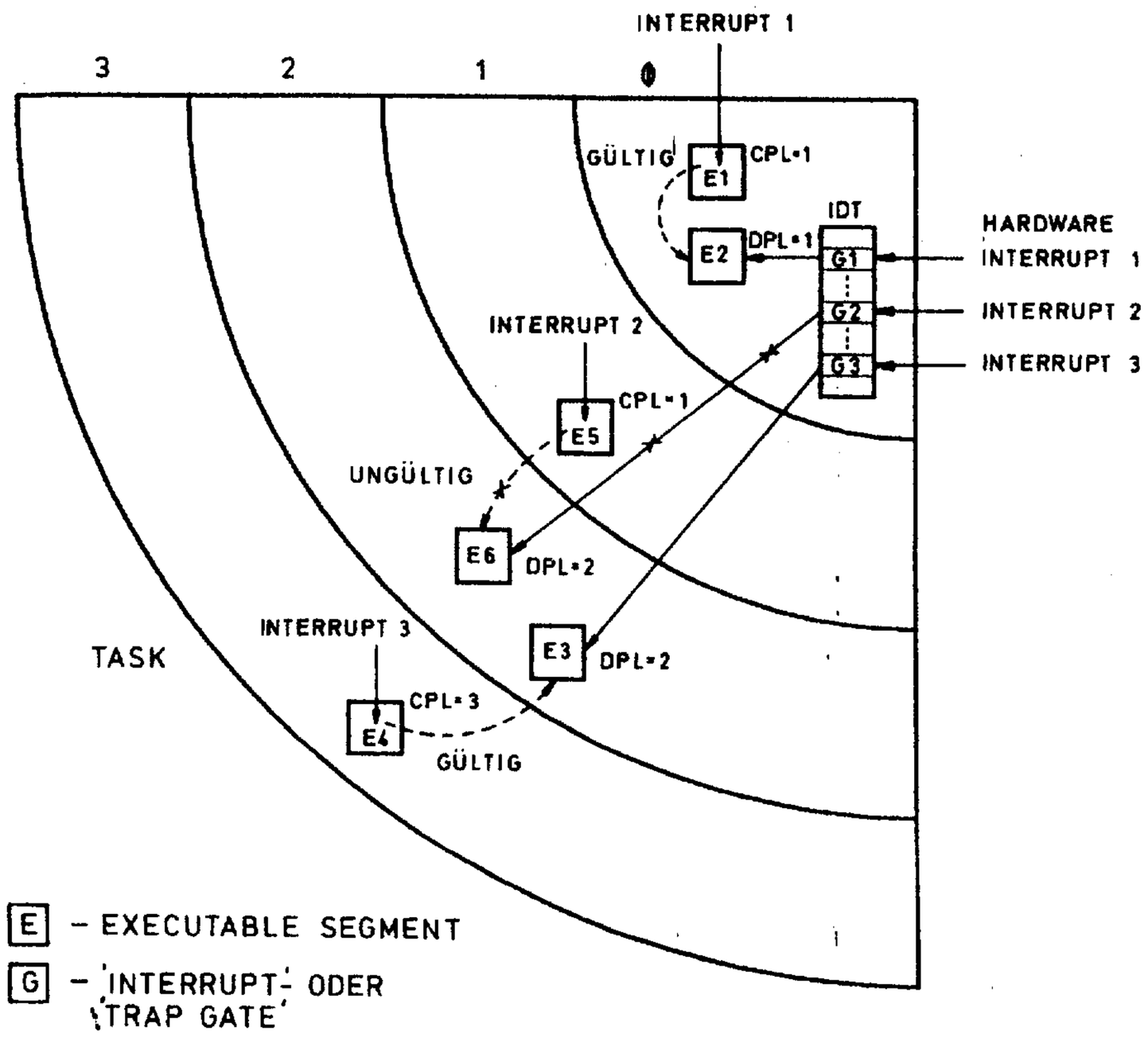
Eine Interrupt-Prozedur kann von einem Hardware-Interrupt nur dann erfolgreich aufgerufen werden, wenn folgende Regel erfüllt ist:

Zielsegment DPL ⩽ Task CPL

Sie besagt, daß sich

- die aufgerufene Interrupt-Prozedur in einem Segment aufhalten muß, dessen Privileg-Ebene numerisch ***kleiner*** oder ***gleich*** CPL der unterbrochenen Task ist.

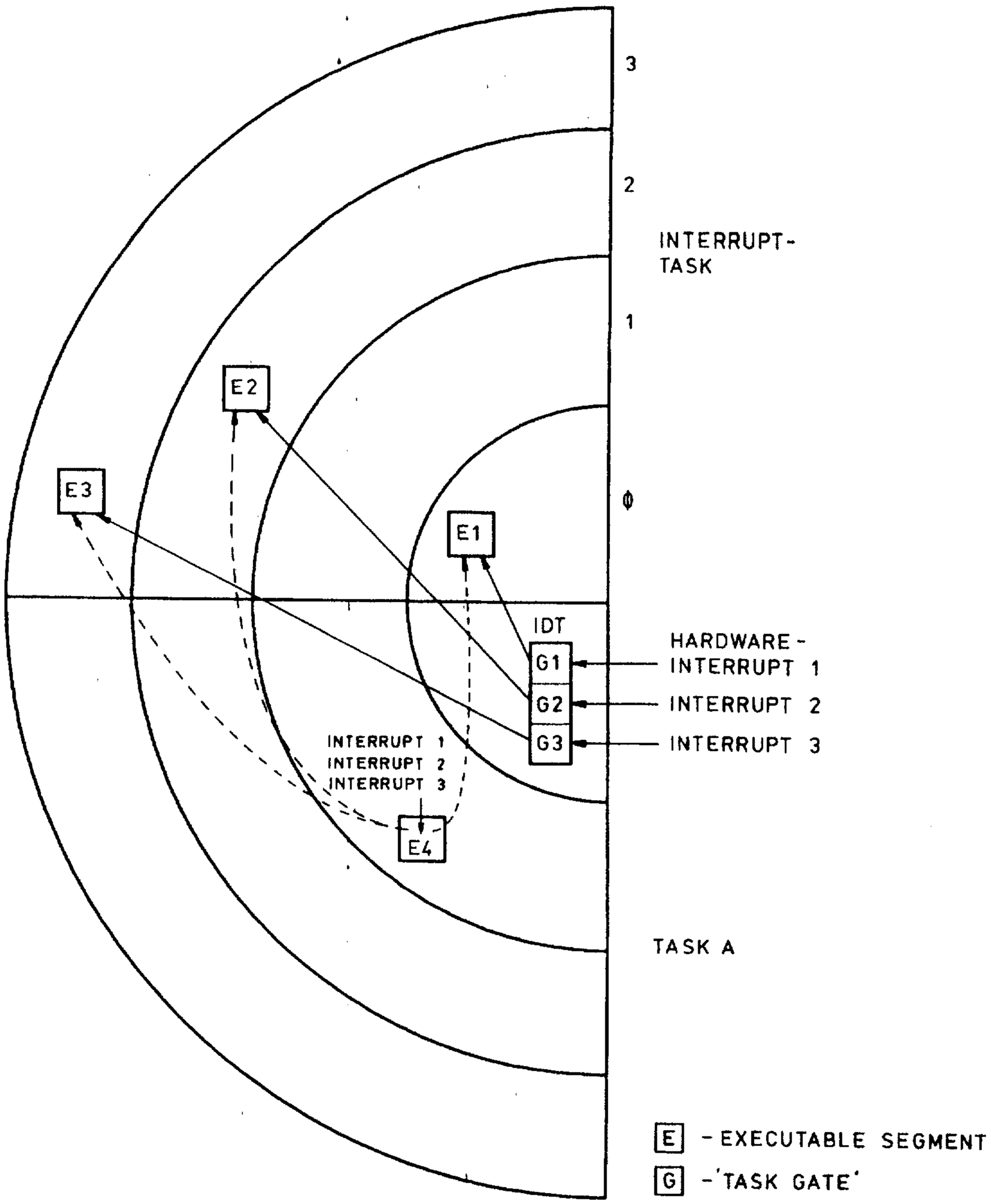
Das folgende Bild zeigt gültige und ungültige Versuche, durch externe Unterbrechungen Interrupt-Prozeduren über Interrupt- und Trap-Gate-Deskriptoren aufzurufen:



Dabei unterbricht INTERRUPT 1 das Code-Segment E1 in der Privileg-Ebene 0 und benützt G1, um den „Interrupt Handler“ in E2 aufzurufen. Diese Unterbrechung ist realisierbar. INTERRUPT 2 versucht E5 in der Privileg-Ebene 1 zu unterbrechen, um über G2 den „Interrupt Handler“ in E6 zu erreichen. Dieser Versuch ist ungültig.
Dagegen kann E4 von INTERRUPT 3 unterbrochen und der „Interrupt Handler“ in E3 aufgerufen werden.

Wenn ein ***Hardware-Interrupt*** einen ***Task-Gate***-Deskriptor benützt, werden keinerlei Privileg-Prüfungen durchgeführt. Die aktive Task scheidet aus, und es erfolgt ein Wechsel in die „Interrupt Task“. Dabei darf sich der „Interrupt Handler“ in der „Interrupt Task“ in jeder Privileg-Ebene aufhalten.

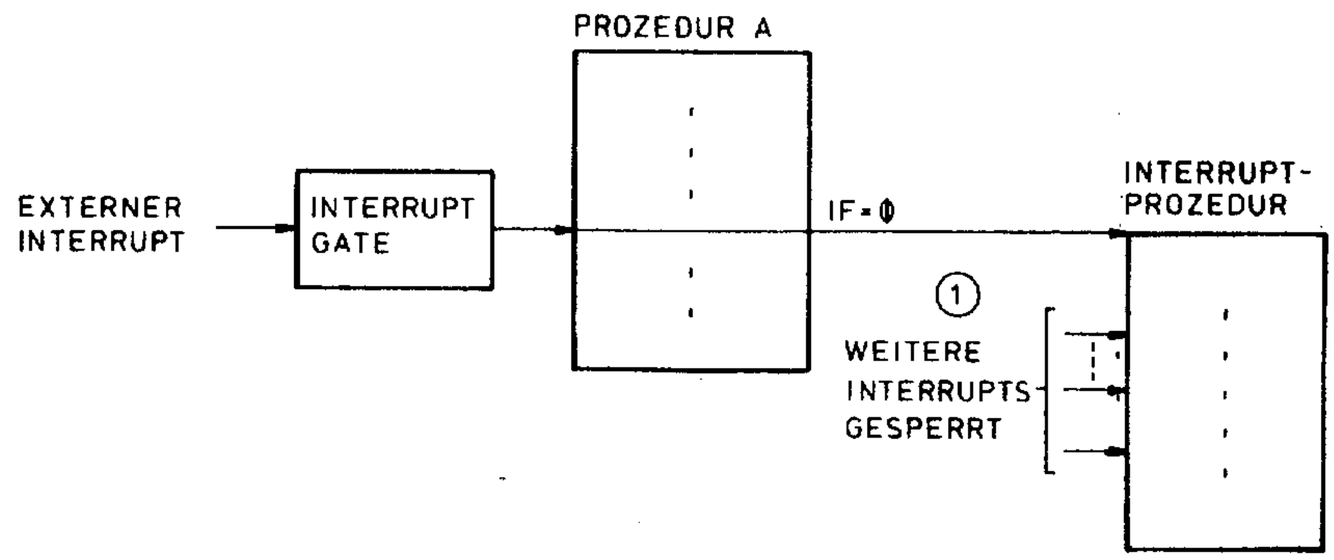
Das folgende Bild zeigt hierfür ein Beispiel:



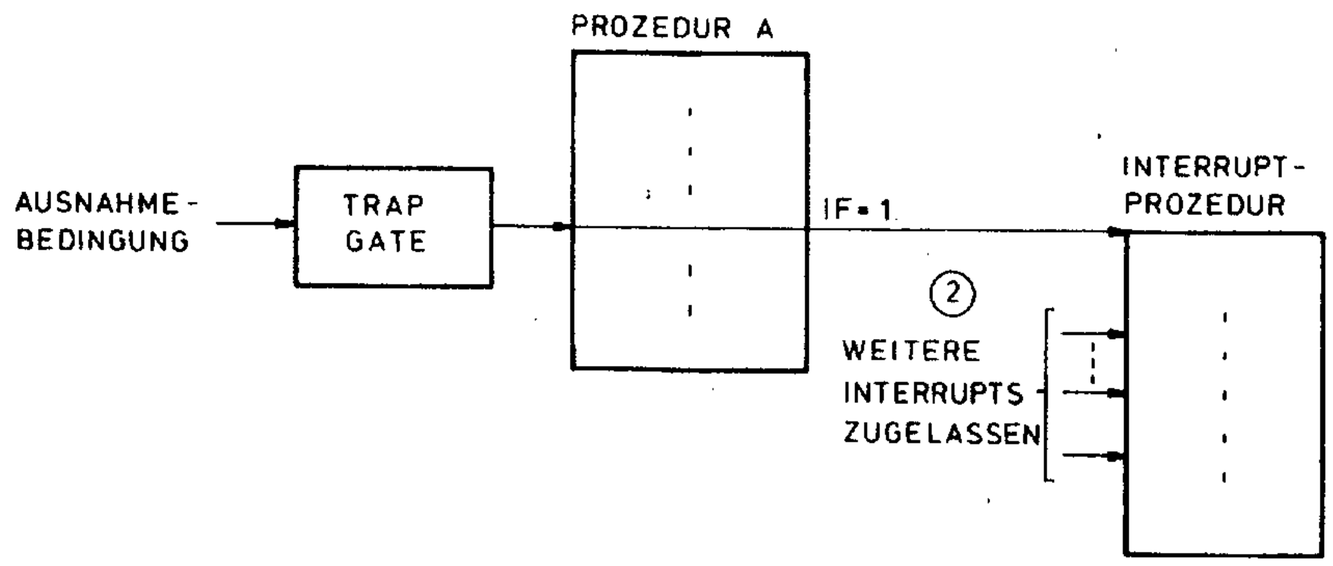
Es ist ersichtlich, daß im Augenblick die Task A das Code-Segment E4 in der Privileg-Ebene 1 ausführt. Die Hardware-Interrupts 1, 2 oder 3 unterbrechen E4 und benützen hierfür jeweils einen Task-Gate-Deskriptor, um in der „Interrupt Task" die „Interrupt Handler" in E1, E2 oder E3 in den verschiedensten Privileg-Ebenen auszuführen.

5.6 Unterschied zwischen Interrupt-Deskriptoren und Trap-Gate-Deskriptoren

Benützt eine Interrupt-Quelle einen ***Interrupt-Gate***-Deskriptor, setzt der Prozessor automatisch vor dem Wechsel in die Interrupt-Prozedur das Interrupt-Flag IF in den EFLAGS zurück (IF = 0). Dies hat zur Folge, daß die aufgerufene Interrupt-Prozedur durch weitere Interrupt-Anforderungen ***nicht*** unterbrochen werden kann ①. Daher werden Interrupt-Gate-Deskriptoren normalerweise von ***externen*** Interrupts benützt.



Benützt eine Interrupt-Quelle einen ***Trap-Gate***-Deskriptor, setzt der Prozessor das Interrupt-Flag IF in den EFLAGS ***nicht*** zurück (IF = 1). Dies bedeutet, daß die aufgerufene Interrupt-Prozedur jederzeit durch weitere Interrupt-Anforderungen unterbrochen werden kann ②. Daher werden Trap-Gate-Deskriptoren normalerweise im Zusammenhang mit erkannten Ausnahme-Bedingungen benützt.



5.7 Task-Wechsel-Effekt bei Interrupts

Welchen Einfluß Task-Wechsel-Operationen, die durch ***externe*** Interrupts oder Interrupt-Befehle ausgelöst worden sind, auf

- das Busy-Bit im TSS-Deskriptor
- das NT-Bit im CPU-EFLAG-Register und
- das „BACK LINK WORD“ im Task-Status-Segment

der unterbrochenen Task und der eintretenden Interrupt-Task haben, zeigt die folgende Tabelle:

Beeinflußtes Feld	Effekt des Interrupt-Befehls und des externen Interrupts
BUSY-Bit im TSS-Deskriptor der eintretenden „Interrupt Task“	(Muß vorher 0 sein) Wird gesetzt
BUSY-Bit im TSS-Deskriptor der unterbrochenen Task	Bleibt unverändert
NT-Bit im „CPU-EFLAG-Register“ der eintretenden „Interrupt Task“	Wird gesetzt
NT-Bit im „CPU-EFLAG-Register“ der unterbrochenen Task	Bleibt unverändert
„BACK LINK“-Feld im TSS der eintretenden „Interrupt Task“	Es wird ein Selektor zum Task-Status-Segment der unterbrochenen „Task“ gesetzt
„BACK LINK“-Feld im TSS der unterbrochenen Task	Bleibt unverändert

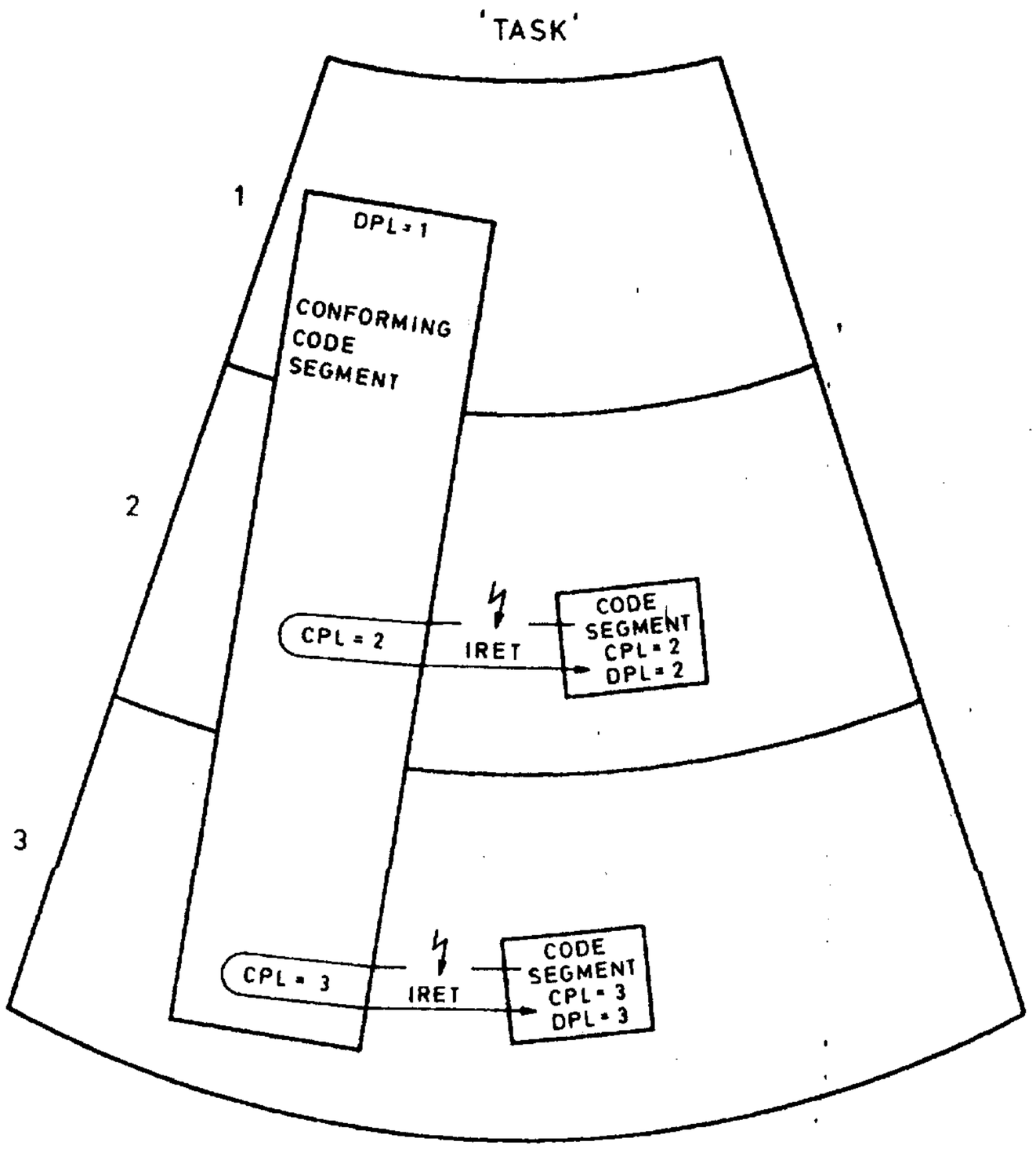
5.8 „Conforming“-Interrupt-Prozeduren

Manchmal ist es wünschenswert, daß sich bestimmte Interrupt-Prozeduren in der gleichen Privileg-Ebene aufhalten wie die unterbrochene Prozedur.

Sollte z. B. in einer Prozedur ein Divisionsfehler (Division durch Null) auftreten, wird sie unterbrochen und der „Interrupt Handler“ liefert für den Quotienten einen bestimmten Binär-Code (z. B. ∞).

Da der „Interrupt Handler“ dieses Ergebnis unabhängig von der Privileg-Ebene der unterbrochenen Prozedur immer liefert, kann er in einem Conforming-Segment plaziert werden.

Für Interrupt-Prozeduren in Conforming-Segmenten setzt der Prozessor automatisch CPL auf DPL des Segments, der die unterbrochene Prozedur enthält. Das folgende Bild soll diese Situation verdeutlichen:



Es ist ersichtlich, daß sich der „Interrupt Handler“ in einem Conforming-Segment auf der Privileg-Ebene 1 (DPL = 1) aufhält. Immer dann, wenn die Code-Segmente in den Ebenen 2 (CPL = 2) oder 3 (CPL = 3) z. B. durch Ausnahme-Situationen unterbrochen werden, führt die „Task“ die „Interrupt Handler“ in diesen Ebenen aus.

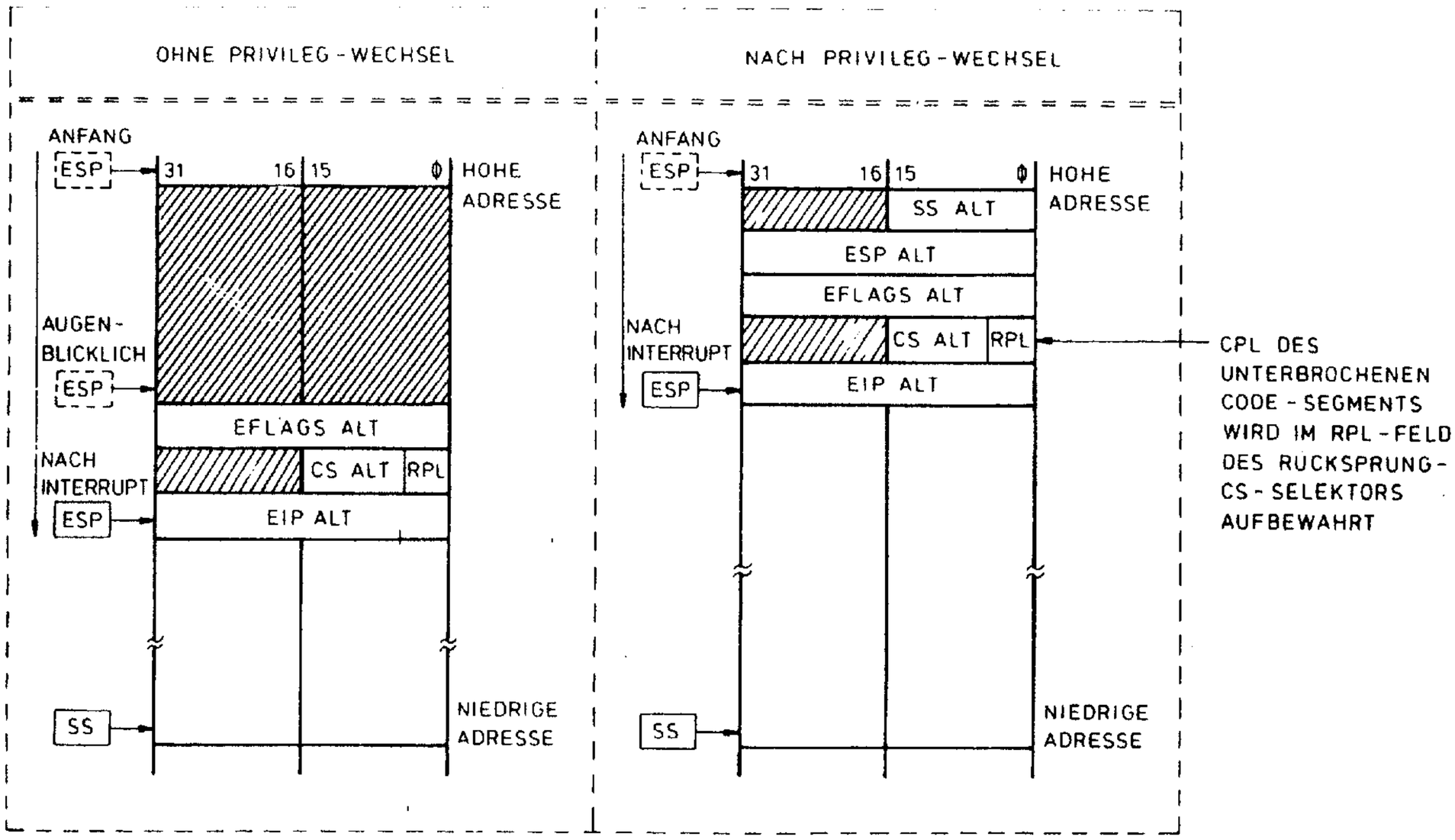
Es sei darauf hingewiesen, daß diese Technik nur angewendet werden kann, wenn die Privileg-Ebene des Conforming-Code-Segmentes (DPL) numerisch ***kleiner*** oder ***gleich*** CPL der unterbrochenen Prozedur ist ***(DPL ≤ CPL)***.

5.9 Stack-Eintragungen nach Interrupt

Benützen Ausnahme-Situationen, externe Interrupts oder Interrupt-Befehle Interrupt- oder Trap-Gate-Deskriptoren, werden sie in ähnlicher Weise behandelt wie CALL-Befehle, die CALL-Gate-Deskriptoren benützen. Der Hauptunterschied ist der, daß der Prozessor automatisch

- den EFLAG-Status der unterbrochenen Prozedur in den Stack der aufgerufenen Prozedur für die Rücksprung-Adresse CS:EIP schreibt.,

Dieser Mechanismus ist unabhängig davon, ob für die Bearbeitung des Interrupts ein Privileg-Wechsel erforderlich ist oder nicht. Das folgende Bild illustriert Stack-Layouts von Interrupt-Prozeduren, die sich entweder in der gleichen oder in einer höheren Privileg-Ebene aufhalten wie die unterbrochene Prozedur:

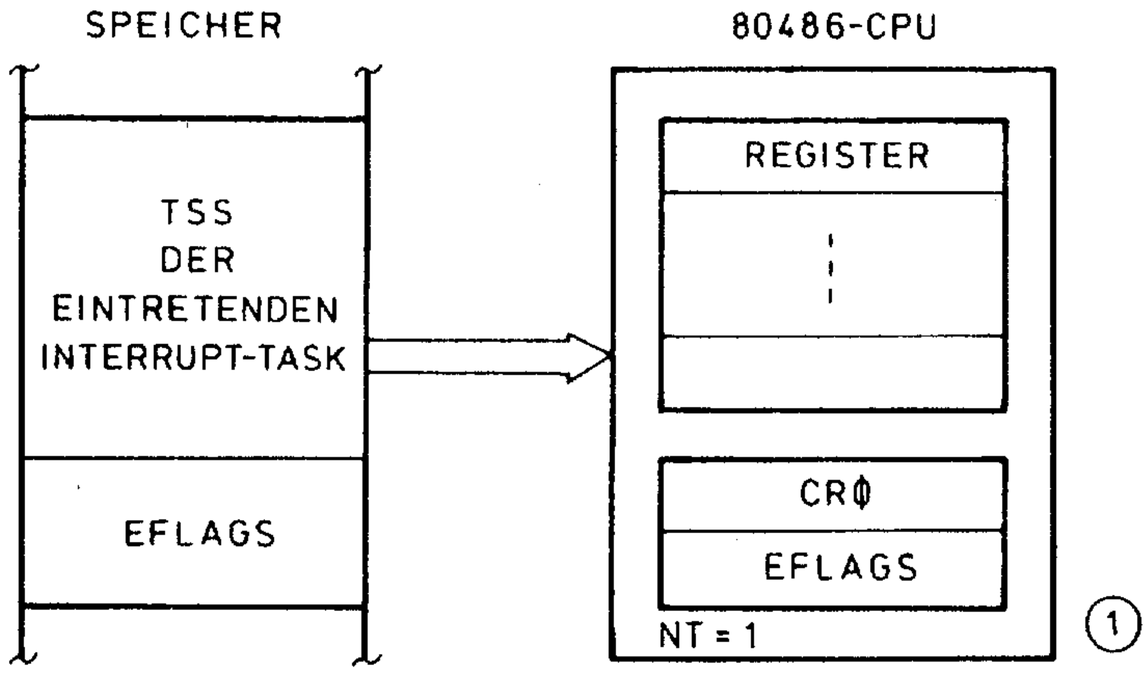


5.10 Verlassen des „Interrupt Handlers“

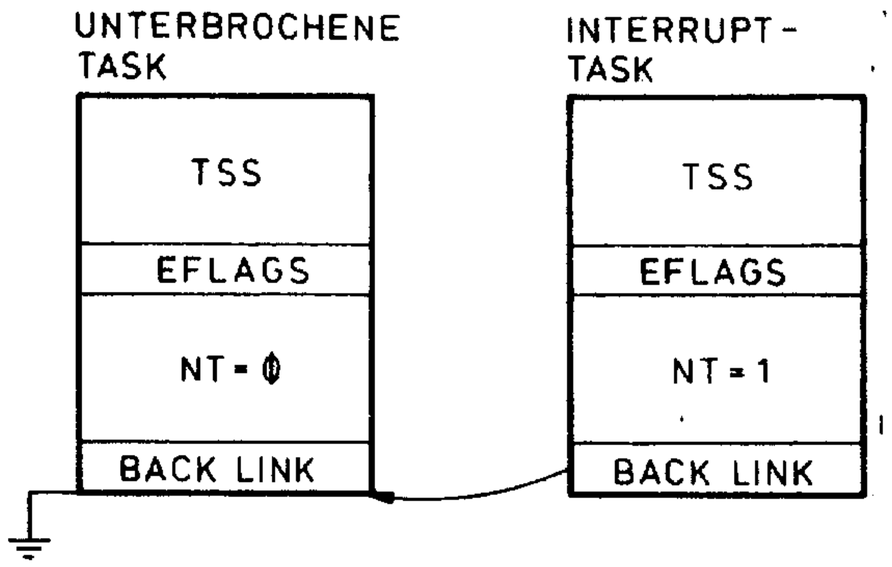
Ein „Interrupt Handler“ wird ***unabhängig*** davon, ob er über einen Task-Gate-, einen Interrupt-Gate- oder einen Trap-Gate-Deskriptor aufgerufen worden ist, ***immer*** mit dem IRET-Befehl verlassen.

Da der EFLAG-Status der unterbrochenen Prozedur im Stack der Interrupt-Prozedur gespeichert ist, steht er nach der Rückkehr in die unterbrochene Prozedur wieder zur Verfügung. Dies bedeutet, daß das Interrupt-Flag IF wieder den gleichen Wert hat, wie vor dem Aufruf der Interrupt-Prozedur.

Jeder Task-Wechsel, der von einem externen Interrupt oder einem Interrupt-Befehl über einen Task-Gate-Deskriptor initiiert worden ist, setzt automatisch das NT-Bit (***N***ested ***T***ask) in den CPU-EFLAGS der eintretenden Interrupt-Task ①.



Damit wird einem Programm mitgeteilt, daß im „BACK LINK“-Feld des Interrupt-Task-Status-Segments eine gültige Verbindung (Selektor) zum Status-Segment der unterbrochenen Task gespeichert ist. Das folgende Bild zeigt eine solche Struktur:



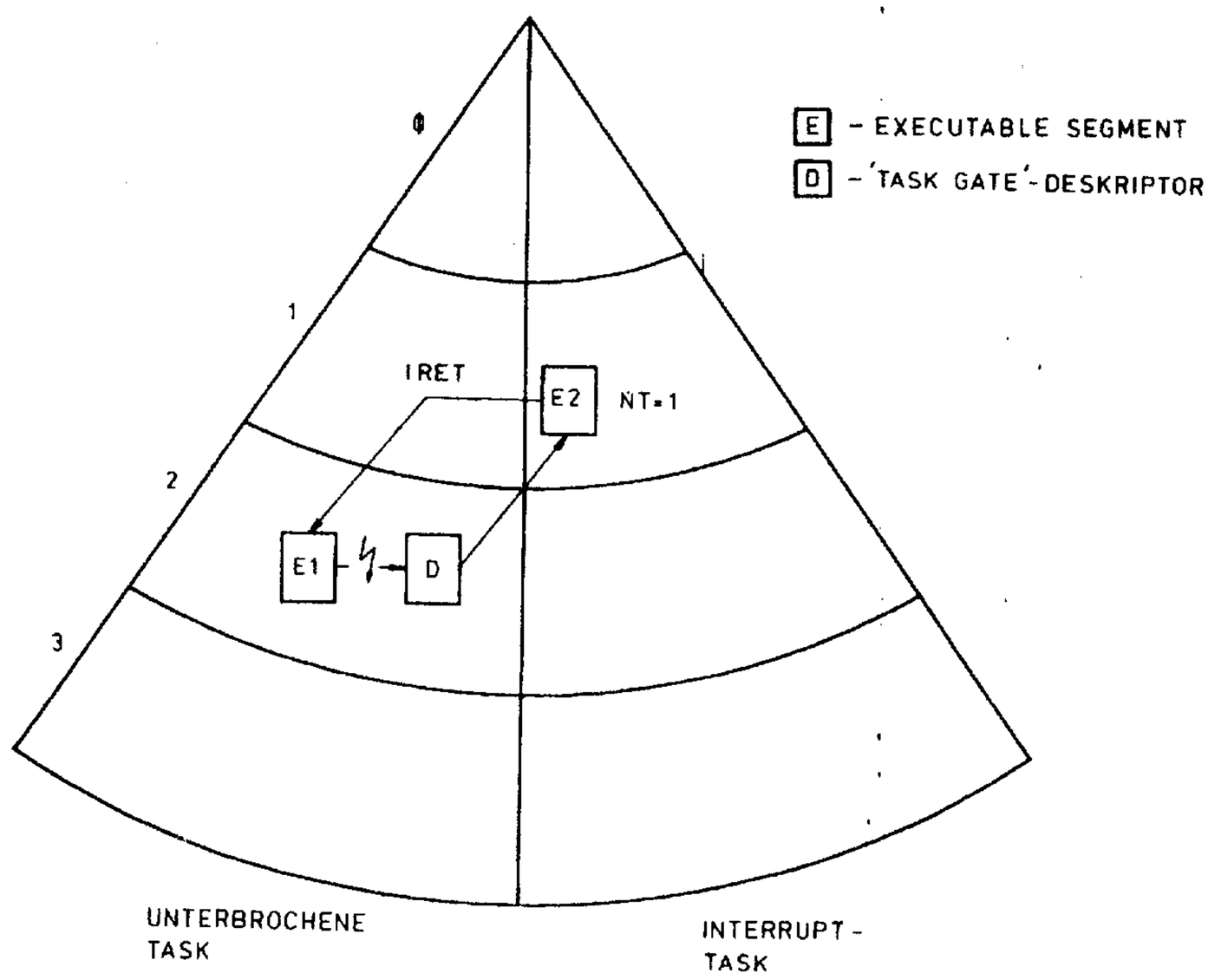
Jeder IRET-Befehl, der bei einem gesetzten NT-Bit (NT = 1) ausgeführt wird, löst einen Task-Wechsel aus. Dabei benützt IRET das „BACK LINK“-Feld im augenblicklich aktiven Task-Status-Segment. Durch dieses Feld ist die Ziel-Task eindeutig definiert.

Dies bedeutet, daß IRET

- sowohl für die Rückkehr von einer Interrupt-Task in die unterbrochene Task
- als auch für die Rückkehr von einer durch CALL aufgerufenen normalen Task in die aufrufende Task

benützt werden kann.

Das folgende Bild illustriert hierfür ein Beispiel:



Es zeigt die Unterbrechung des Code-Segments E1 durch einen Interrupt, der den Task-Gate-Deskriptor D benützt, um einen „Interrupt Handler“ im Code-Segment E2 in der Interrupt-Task aufzurufen.

Da nach dem Task-Wechsel das NT-Bit gesetzt ist, kann nun mit IRET in die unterbrochene Task zurückgekehrt werden.

5.11 Task-Wechsel-Effekt bei IRET

Welchen Einfluß Task-Wechsel-Operationen, die durch IRET-Befehle ausgelöst worden sind, auf

- das BUSY-Bit im TSS-Deskriptor
- das NT-Bit im CPU-EFLAG-Register und
- das „BACK LINK WORD“ im Task-Status-Segment

der eintretenden und der ausscheidenden Task haben, zeigt die folgende Tabelle:

Beeinflußtes Feld	Effekt des IRET-Befehls
BUSY-Bit im TSS-Deskriptor der eintretenden Task	Bleibt unverändert (Muß gesetzt sein)
BUSY-Bit im TSS-Deskriptor der ausscheidenden Task	Wird gelöscht
NT-Bit im CPU-EFLAG-Register der eintretenden Task	Bleibt unverändert
NT-Bit im CPU-EFLAG-Register der ausscheidenden Task	Wird gelöscht
„BACK LINK“-Feld im TSS der eintretenden Task	Bleibt unverändert
„BACK LINK“-Feld im TSS der ausscheidenden Task	Bleibt unverändert

Hinweis:
Die angegebene Tabelle gilt, wenn mit IRET

- eine „Interrupt Task“ oder
- eine mit CALL aufgerufene normale Task

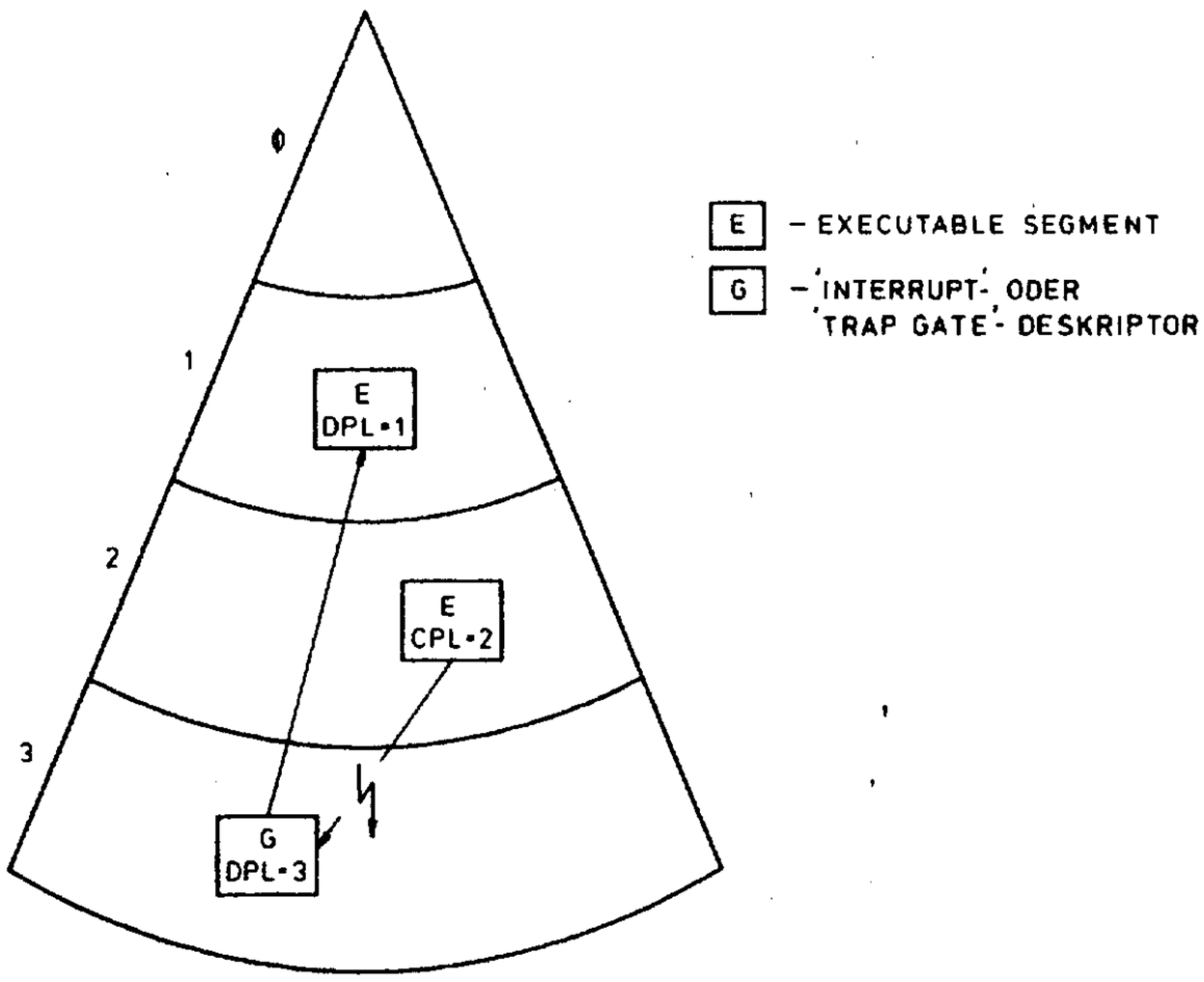
verlassen wird.

5.12 Interrupt-Rückkehr innerhalb der gleichen Task

Benützt ein Hardware-Interrupt oder ein Interrupt-Befehl einen Interrupt-Gate- oder Trap-Gate-Deskriptor, kommt es zu ***keinem*** Task-Wechsel. In beiden Fällen bleibt das NT-Bit gelöscht (NT = 0).

Der aufzurufende „Interrupt Handler" darf sich innerhalb der augenblicklich aktiven Task entweder auf der gleichen oder einer höheren (numerisch kleineren) Privileg-Ebene wie die unterbrochene Prozedur aufhalten.

Um den „Interrupt Handler" zu verlassen, muß auch hier ein IRET-Befehl ausgeführt werden. Da das NT-Bit ***gelöscht*** ist, bewirkt IRET das gleiche wie eine normale 80486-Interrupt-Rückkehr-Funktion. Dies soll das folgende Beispiel verdeutlichen:

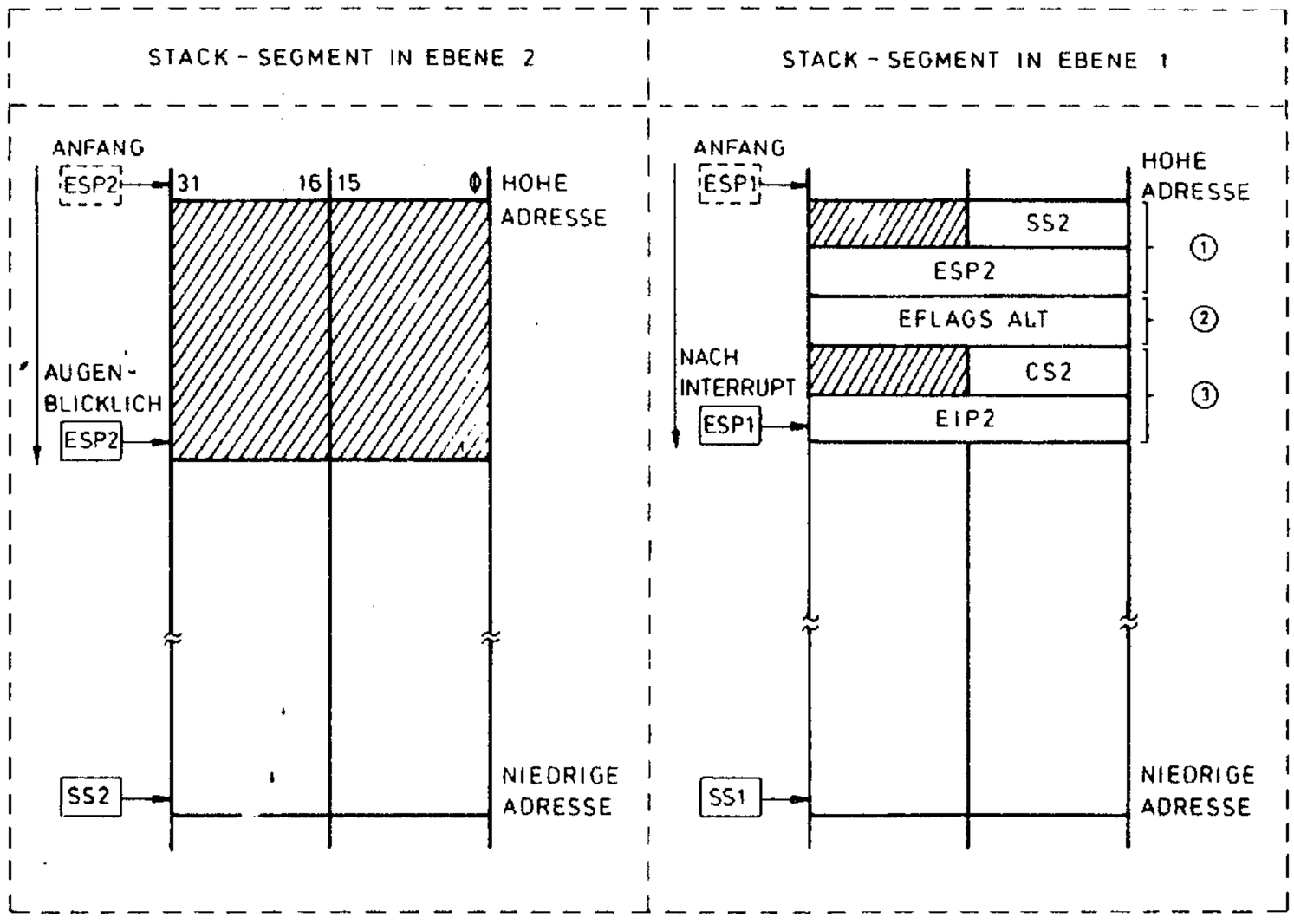


Es sei vorausgesetzt, daß ein Code-Segment in der Privileg-Ebene 2 (CPL = 2) unterbrochen wird. Dabei benützt die Interrupt-Quelle einen Interrupt-Gate- oder Trap-Gate-Deskriptor in der Privileg-Ebene 3 (DPL = 3), um einen „Interrupt Handler" in der Privileg-Ebene 1 (DPL = 1) aufzurufen.

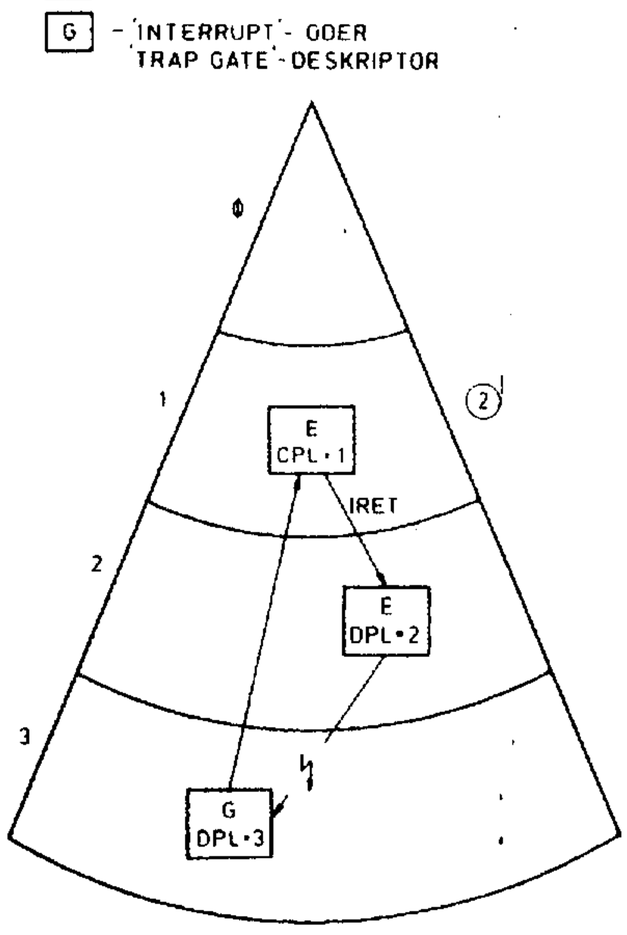
Die Unterbrechung des Code-Segments in der Privileg-Ebene 2 (CPL = 2) bewirkt, daß zunächst der momentane 48-Bit-Zeiger SS2:ESP2 zum Stack-Segment in Ebene 2 an das Stack-Segment in Ebene 1 übergeben wird ①.

Dabei ist das Ebene-1-Stack-Segment durch einen eigenen, im Task-Status-Segment aufbewahrten 48-Bit-Zeiger SS1:ESP1 gekennzeichnet.

Der EFLAG-Status zum Zeitpunkt der Unterbrechung wird anschließend in den Ebene-1-Stack übertragen ②. Danach folgt die Rücksprung-Adresse CS2:EIP2 zum unterbrochenen Code-Segment ③, wobei EIP2 zum nächsten Befehl nach der Unterbrechung zeigt.



In Analogie zum CALL-Befehl wird auch hier im RPL-Feld des Rücksprung-CS-Selektors die augenblickliche Privileg-Ebene (CPL) des unterbrochenen Code-Segments gespeichert. Im Beispiel ist dies die Ebene 2 ①.



Wenn die Task den „Interrupt Handler“ ausführt, befindet sie sich in der Privileg-Ebene 1 (CPL = 1). Der IRET-Befehl in diesem Programm ② schreibt die im Ebene-1-Stack gespeicherten Informationen wie SS2:ESP2, EFLAGS und CS2:EIP2 in die CPU zurück. Dabei hat der Rücksprung-Selektor CS2 die Privileg-Ebene 2 (RPL = 2).

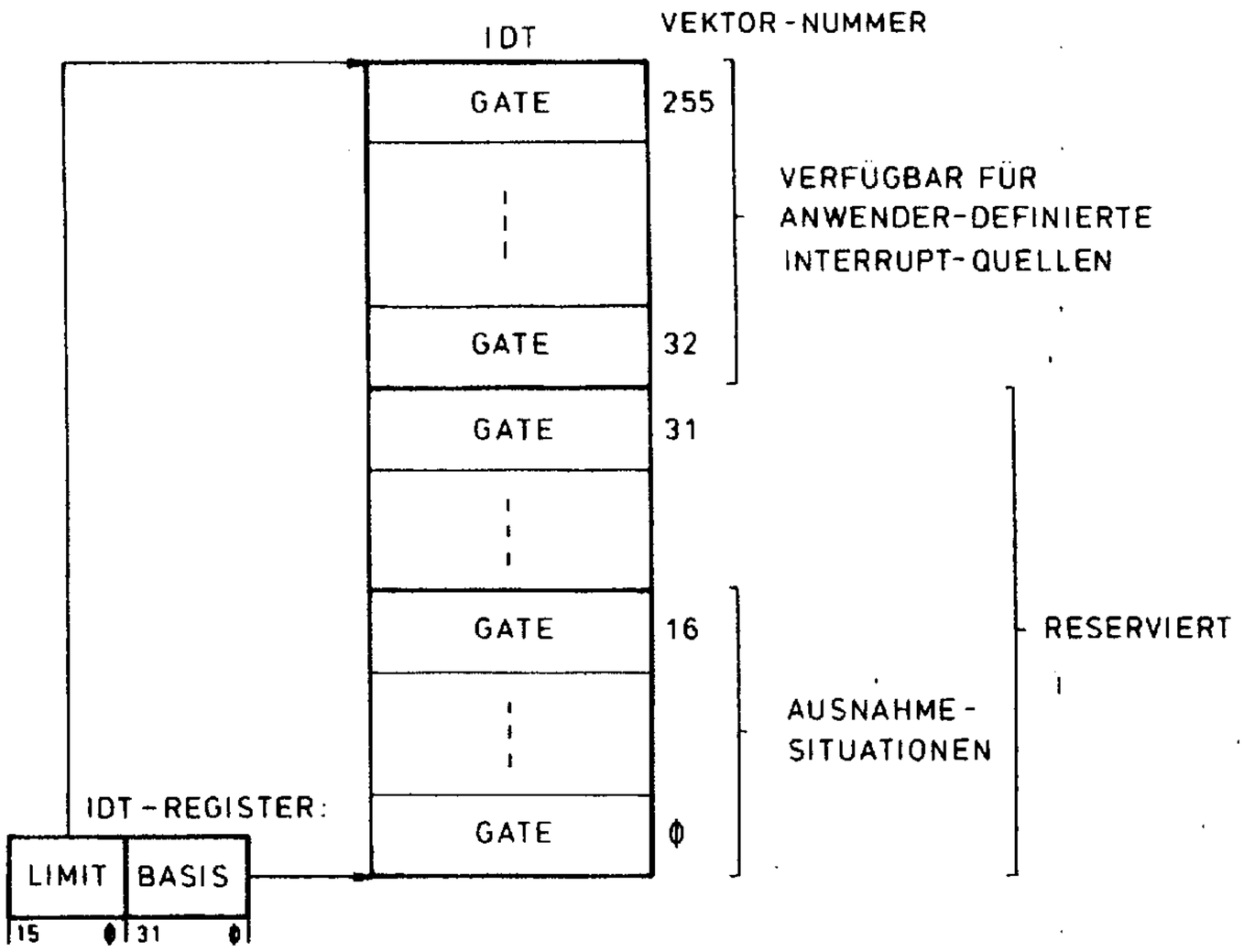
Immer dann, wenn der IRET-Befehl auf einen CS-Wert trifft, dessen RPL > CPL ist, wird eine Interlevel-Rückkehr eingeleitet. Dies ist im angegebenen Beispiel mit RPL = 2 und CPL = 1 der Fall.

Generell gilt, daß eine Interrupt-Rückkehr innerhalb der gleichen Task mit IRET nur dann möglich ist, wenn die augenblickliche Privileg-Ebene CPL, in der sich der IRET-Befehl befindet, numerisch ***kleiner*** oder ***gleich*** DPL des unterbrochenen Code-Segments ist, zu dem die Rückkehr erfolgen soll.

5.13 Externe Interrupts

Wie bereits früher erwähnt, enthält die Interrupt-Deskriptor-Tabelle IDT maximal 256 Eintragungen zu je 8 Bytes. Dabei ist jeder Eintrag („Gate"-Deskriptor) über eine bestimmte Vektor-Nummer im Bereich 0 ... 255 erreichbar.

Die Vektor-Nummern 0 ... 31 sind reserviert, so daß der Anwender noch weitere 224 verschiedene Interrupt-Quellen definieren kann. Alle diese Interrupts werden vom 8259-Interrupt-Controller bedient, der seinerseits den maskierbaren INTR-Eingang (Interrupt Request) der 80486-CPU benützt.

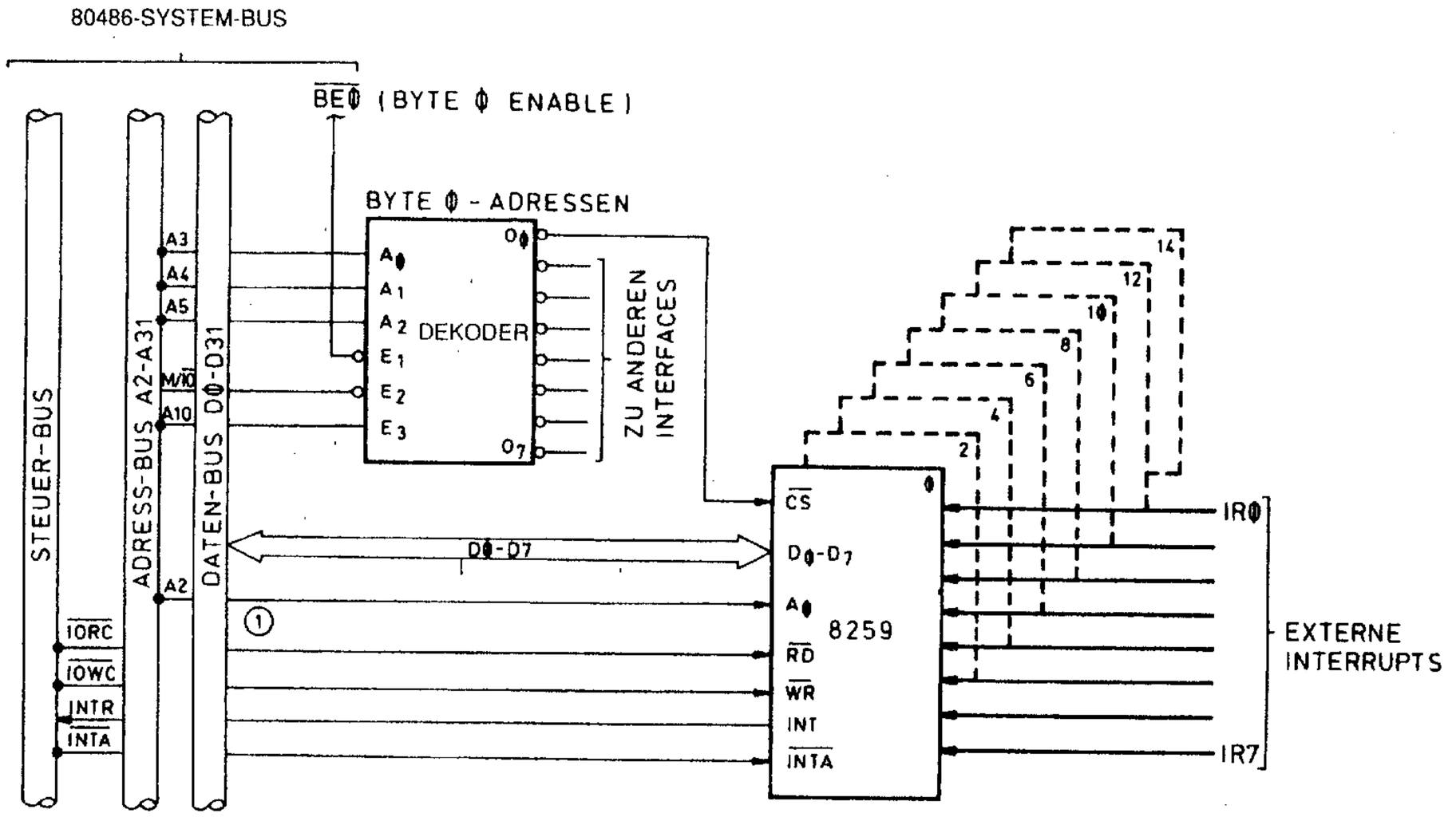


Das angegebene Bild zeigt das Layout einer komplett belegten Interrupt-Deskriptor-Tabelle. Es ist ersichtlich, daß in einem Teil des reservierten Bereichs (Vektor-Nummern 0 ... 31) Gate-Deskriptoren aufbewahrt sind, die für die Behandlung von intern aufgetretenen Ausnahme-Situationen benützt werden.

Wenn der Anwender externe Interrupt-Quellen benützt, ist es notwendig, daß für jede dieser Quellen eine bestimmte Vektor-Nummer im Bereich von 32 ... 255 vereinbart wird.

Zur Definition dieser Nummern steht das ***ICW2***-Register im 8259-Interrupt-Controller zur Verfügung.

Am Beispiel der folgenden 80486-System-Konfiguration soll nun der Funktionsablauf einer externen Interrupt-Anforderung gezeigt werden:



Es ist ersichtlich, daß der Binär-Dekoder als sogenannter Chip-Select-Generator für maximal 8 Byte-Interfaces benützt wird. Er ist immer dann freigegeben, wenn die CPU die Systembus-Signalkombination

$\overline{BE0}$	A10
L	H

ausgibt, so daß er dann abhängig von den Adreß-Informationen A5, A4 und A3 einen seiner Ausgänge $0_0 - 0_7$ aktivieren kann. Jeder dieser Ausgänge ist mit dem Chip-Select-Eingang $\overline{CS}$ eines Byte-Interfaces verbunden, die alle wegen $\overline{BE0}$ = „Low“ über die Datenbus-Leitungen D0 ... D7 zu erreichen sind (siehe Kapitel Eingabe/Ausgabe). Da der Prozessor nur die Adreß-Signale A2 ... A31, ***nicht*** aber A0 und A1 liefert, wird in der Schaltung A2 benützt, um über A1 des Interrupt-Controllers die internen Register dieses Bausteins zu selektieren ①.

Wenn der Interrupt-Controller eine externe Unterbrechungs-Anforderung akzeptiert hat, leitet er sie über INT zum INTR-Eingang der 80486-CPU weiter. Ist das IF-Flag (Interrupt Enable) gesetzt (IF = 1), nimmt der Prozessor den Interrupt an und quittiert ihn über die Buszyklus-Signale $M/\overline{IO}$, $D/\overline{C}$, $W/\overline{R}$ = 000 (Transfertyp: Interrupt Acknowledge) mit ***zwei*** $\overline{INTA}$-Impulsen.

Dabei wird der ***erste*** INTA-Impuls zur Steuerung des 8259 benützt:

- Er setzt das flankengesteuerte Eingangs-FlipFlop zurück.
- Er „friert“ die Anforderung am IRn-Eingang im zugehörigen Anforderungs-FlipFlop IRRn (IRR = Interrupt Request Register) ein und
- er setzt das ISRn-FlipFlop (ISR = Interrupt Service Register) der zu bearbeitenden Interrupt-Ebene.

Mit dem ***zweiten*** $\overline{INTA}$-Impuls liest der 80486 die zum Anforderungs-Eingang gehörige Vektor-Nummer vom 8259-Interrupt-Controller. Maximal kann der Controller 8 Vektoren

liefern. Dabei ist die Vektor-Nummer ***nur*** für Anforderungen an IR0 im ICW2-Register programmierbar. Die restlichen Nummern setzt der 8259 selbständig ein. Dies soll die folgende Tabelle verdeutlichen:

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	A15	A14	A13	A12	A11	1	1	1
IR6	A15	A14	A13	A12	A11	1	1	0
IR5	A15	A14	A13	A12	A11	1	0	1
IR4	A15	A14	A13	A12	A11	1	0	0
IR3	A15	A14	A13	A12	A11	0	1	1
IR2	A15	A14	A13	A12	A11	0	1	0
IR1	A15	A14	A13	A12	A11	0	0	1
IR0	A15	A14	A13	A12	A11	0	0	0

5.13.1 **Programmierbeispiel:** Interrupt-Vektor-Nummern

In einem 80486-System soll ein einzelner 8259-Interrupt-Controller existieren und Unterbrechungs-Anforderungen an IR5 und IR6 bedienen. Seine Basis-Adresse sei 0400H. Die vom Interrupt-Controller gelieferten Vektor-Nummern sollen 45 (2DH) für IR5 und 46 (2EH) für IR6 sein.

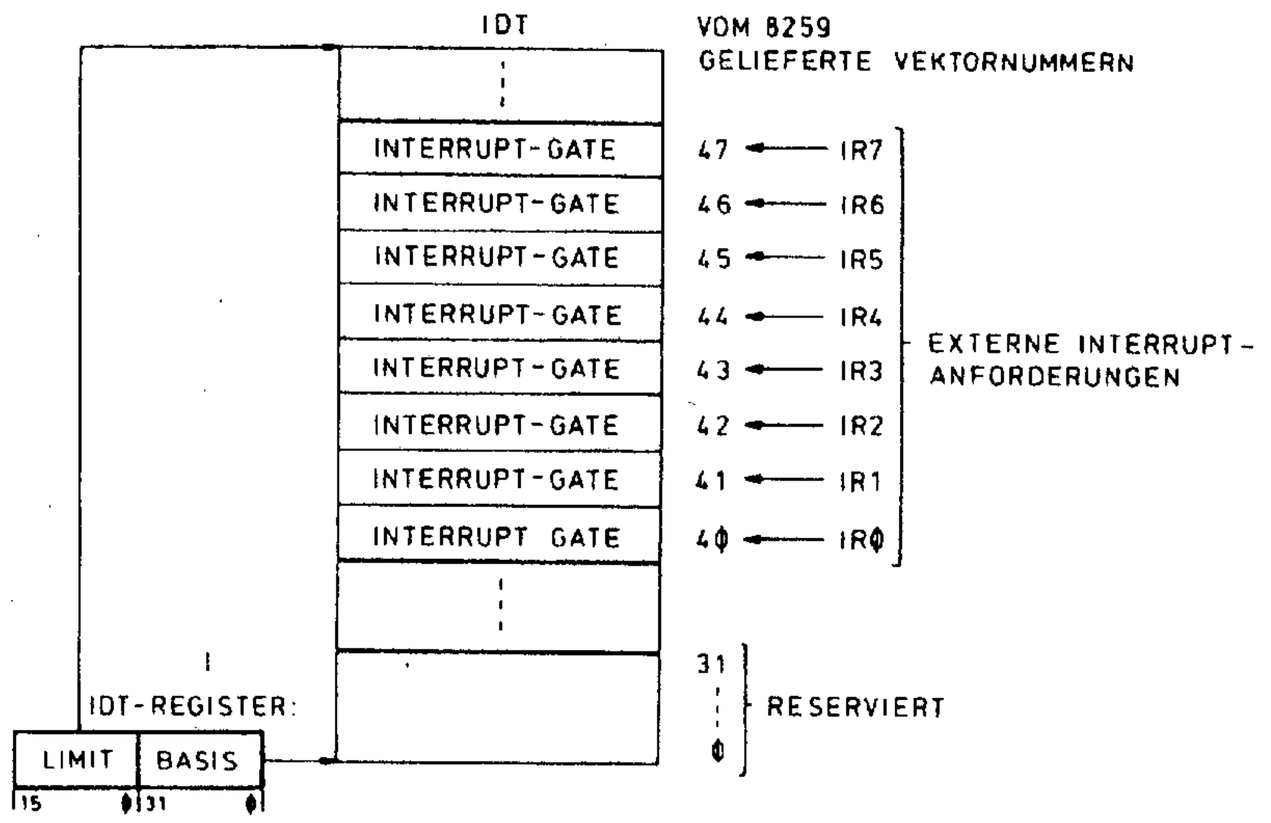
Damit diese Nummern vom 8259 geliefert werden können, muß das ICW2-Register mit dem Wert 40 (28H) initialisiert worden sein ①.

	D7	D6	D5	D4	D3	D2	D1	D0	
									VEKTOR-NUMMERN
IR7									
IR6	0	0	1	0	1	1	1	0	≙ 46 (2EH)
IR5	0	0	1	0	1	1	0	1	≙ 45 (2DH)
IR4									
IR3									
IR2									
IR1									
① IR0	0	0	1	0	1	0	0	0	≙ 40 (28H); IM ICW2 PROGRAMMIERBAR
	↓ PROGRAMMIERBAR					↓ MUSS NULL SEIN			

Befehlsfolge:

```
MOV   DX,0401H      ; (DX) = Adresse des ICW2-Registers
MOV   AL,00101000B  ; (AL) ← ICW2
OUT   DX,AL         ; ICW2 laden
```

Für das angegebene Programmierbeispiel ergibt sich dann das folgende IDT-Layout:



5.14 Nicht-maskierbare externe Interrupts

Nicht-maskierbare externe Interrupts werden über den NMI-Eingang (Non-maskable Interrupt) der 80486-CPU angefordert. Sie haben höhere Priorität als die maskierbaren Interrupts. Dies bedeutet, daß im Fall von gleichzeitigen Anforderungen der nicht-maskierbare Interrupt zuerst bedient wird.

Ein nicht-maskierbarer Interrupt hat die feste Vektor-Nummer 2, so daß eine Interrupt-Quittungs-Sequenz ($\overline{\text{INTA}}$-Impulse) ***nicht*** erforderlich ist.

Einen NMI wird man typischerweise immer dann benützen, wenn z. B. bei einem Spannungsausfall die hierfür notwendige Handler-Prozedur aufgerufen werden muß.

Eine Prozedur, die einen NMI bearbeitet, kann so lange nicht unterbrochen werden, bis ein IRET-Befehl ausgeführt worden ist. Allerdings „erinnert“ sich die Hardware an weitere NMI-Anforderungen, so daß sie nach dem ersten IRET-Befehl bedient werden können.

Um zu verhindern, daß ein Interrupt (INTR) einen NMI-Interrupt-Handler unterbricht, muß das IF-Flag gelöscht sein. Dies ist z. B. realisierbar, wenn für NMI kein Trap-Gate- sondern ein Interrupt-Gate-Deskriptor benützt wird ①. Selbstverständlich kann auch IF in den EFLAGS der Task gelöscht werden.