6 Ausnahmen und Interrupts

Im Gegensatz zum Interrupt, der ***unabhängig*** vom augenblicklich ausgeführten Programm vorkommt, tritt eine Ausnahmesituation als direktes Ergebnis einer Programmausführung auf. Eine solche Situation kommt immer dann vor, wenn ein Befehl ***nicht*** komplett ausgeführt werden kann.

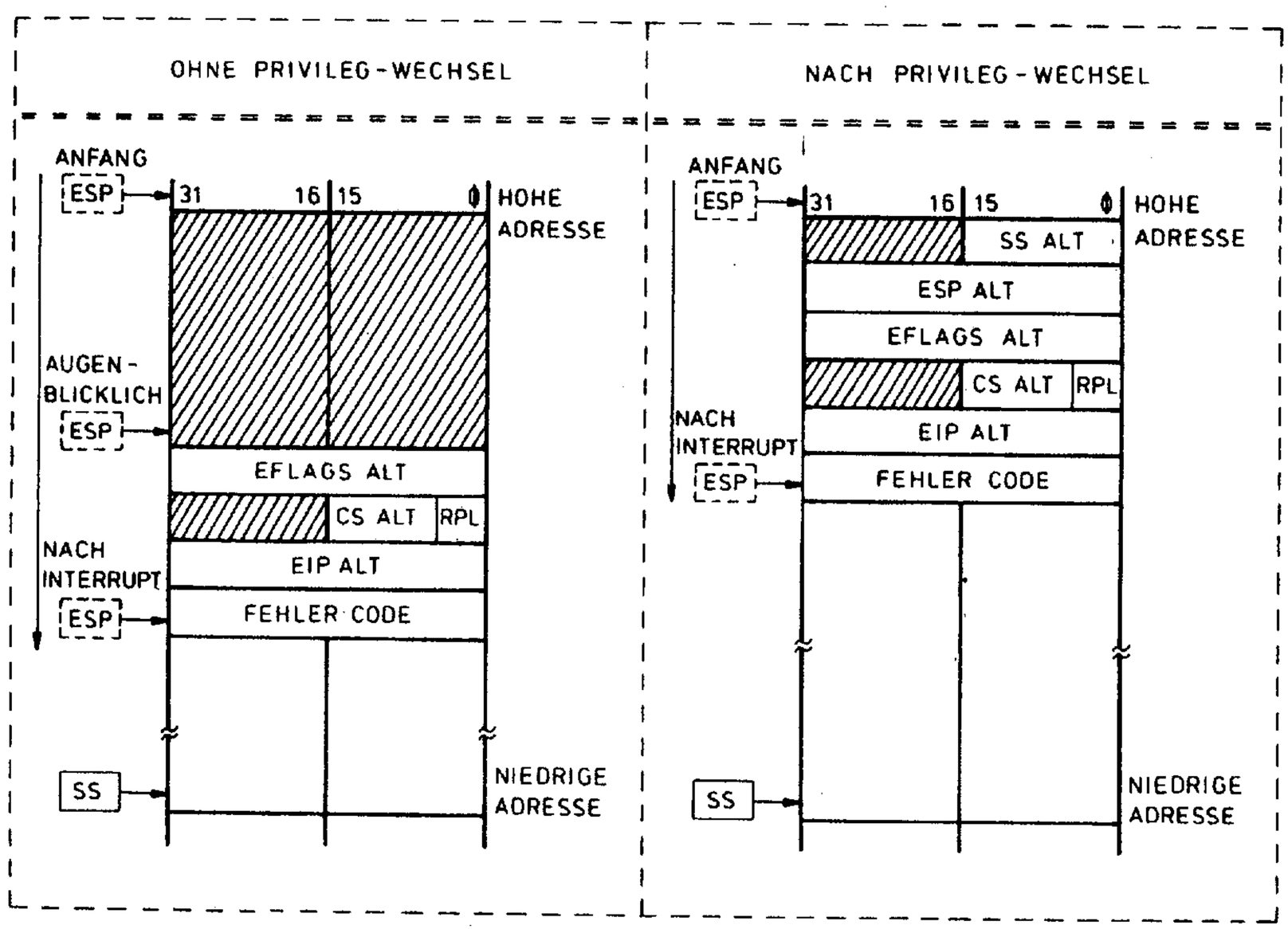
Obwohl Interrupts und Ausnahmen verschiedene Ursachen haben, benützen sie die gleichen Steuerablauf-Techniken und Privileg-Regeln. Daher gehören Ausnahmen zu einer bestimmten Klasse von Interrupts, die ausschließlich für die Erkennung von Defekten oder Programm-Fehlern benützt werden.

Interrupts und Ausnahmen haben jedoch zwei wesentliche Unterschiede:

- Ausnahmen sind ***nicht*** maskierbar
- bei bestimmten Ausnahme-Situationen legt der Prozessor einen ***Fehler-Code*** im Stack des Fehler-Handlers ab.

Benützt die Ausnahme-Bedingung einen „Interrupt"- oder „Trap Gate"-Deskriptor, befindet sich der Fehler-Handler in einer Interrupt-Prozedur. Wenn aber die Ausnahme-Bedingung einen „Task Gate"-Deskriptor benützt, erfolgt ein Wechsel in die zugehörige Interrupt-Task.

Das folgende Bild zeigt die Stack-Layouts von Fehler-Handlern, die über Interrupt- oder Trap-Gate-Deskriptoren aufgerufen worden sind. Sie halten sich entweder in der gleichen oder in einer höheren Privileg-Ebene auf wie die Prozeduren, in der die Ausnahme-Bedingung erkannt worden ist.



Für den Fall, daß die Ausnahme-Situation einen Fehler-Code liefert, legt ihn der Prozessor nach der Rücksprung-Adresse CS:EIP im neuen Stack ab.

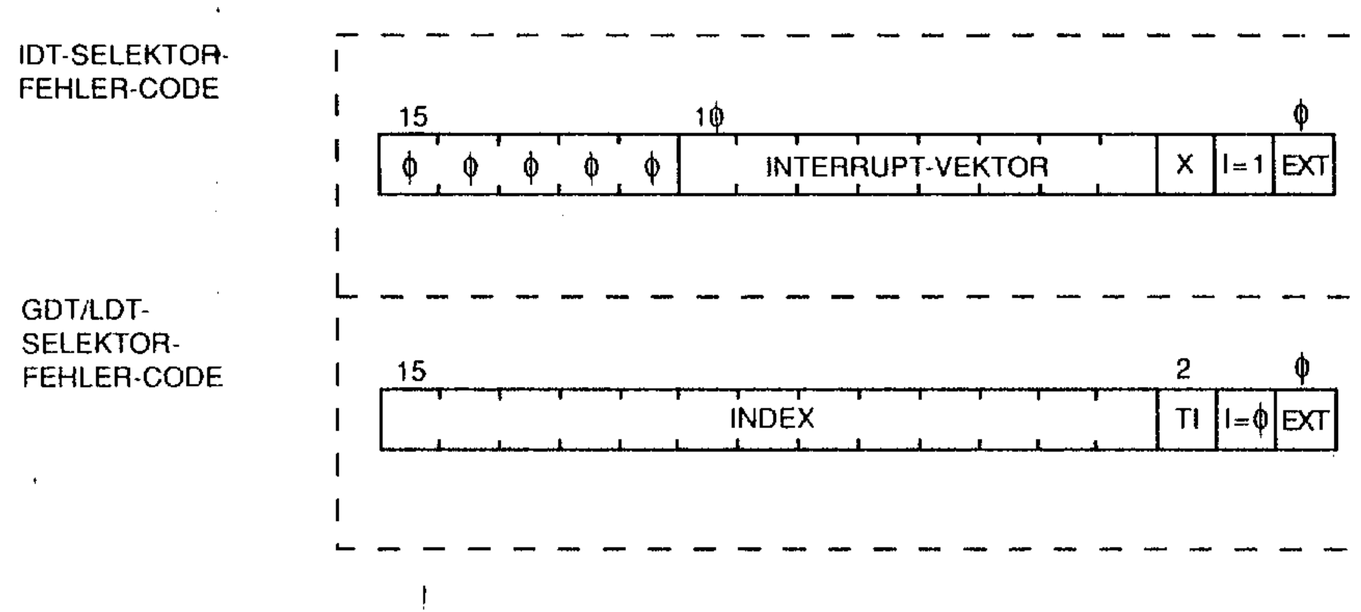
6.1 Interaktionen zwischen Interrupt-Anforderungen und „Gate"-Deskriptoren

Bevor das Format des Fehler-Codes betrachtet wird, sollen zunächst anhand der folgenden Tabellen die Interaktionen zwischen dem Interrupt-Typ und dem benützten Gate-Deskriptor illustriert werden.

Interrupt-Typ NMI	Gate-Deskriptor-Typ	weitere NMI´s?	weitere INTR´s?	weitere Aus-nahmen?	weitere Software-Interrupts?
NMI	Trap	Nein	Ja	Ja	Ja
NMI	Interrupt	Nein	Nein	Ja	Ja
INTR	Trap	Ja	Ja	Ja	Ja
INTR	Interrupt	Ja	Nein	Ja	Ja
Software-Interrupts	Trap	Ja	Ja	Ja	Ja
Software-Interrupts	Interrupt	Ja	Nein	Ja	Ja
Ausnahme	Trap	Ja	Ja	Ja	Ja
Ausnahme	Interrupt	Ja	Nein	Ja	Ja

6.2 Der Fehler-Code

Das Format des Fehler-Codes, den der Prozessor bei bestimmten Ausnahme-Situationen im Stack des Fehler-Handlers ablegt, ähnelt dem eines Selektors. Wie aber das folgende Bild zeigt, enthält der Fehler-Code statt des RPL-Feldes zwei andere Ein-Bit-Eintragungen:



- Der Prozessor setzt das EXT-Bit (External), wenn sich ein Fehler ***nicht*** direkt durch eine Aktion der Task ergibt. Ein solches Ereignis kennzeichnet im allgemeinen ein ***System***-Problem und kein Anwender-Software-Problem. So wird EXT

gesetzt, wenn ein externes Ereignis (z. B. externer Interrupt) den Fehler verursacht hat.

- Der Prozessor setzt das I-Bit (IDT), wenn sich der Index-Teil des Fehler-Codes auf einen Gate-Deskriptor in der IDT bezieht. In diesem Fall enthält dann das Index-Feld eine Interrupt-Vektor-Nummer.
  - Ist das I-Bit gesetzt (I = 1), kann der Fehler-Handler das TI-Bit ignorieren.
  - Ist das I-Bit gelöscht (I = 0), identifiziert das TI-Bit entweder die GDT (TI = 0) oder eine LDT (TI = 1).

Hat der Prozessor eine Ausnahme erkannt, dann identifiziert das Index-Feld abhängig vom Typ der Ausnahme entweder einen Gate-Deskriptor in der IDT oder in der GDT bzw. einer LDT den Deskriptor des fehlerhaften Segments.

6.3 Ausnahmen und reservierte Vektoren

Wie bereits früher erwähnt, existiert in der IDT ein bestimmter Bereich, der für die Behandlung von Ausnahme-Bedingungen reserviert ist.

Grundsätzlich gibt es zwei verschiedene Ursachen, die den Prozessor in eine Ausnahme-Situation bringen können:

1. Solche, die der Prozessor selbst entdeckt und
2. solche, die von den Befehlen INTO, INT 3, INT n und BOUND getriggert werden. Obwohl diese Befehle Software-Interrupts sind, behandelt sie der Prozessor als Ausnahmen.

Alle Ausnahmen lassen sich als ***FAULTS, TRAPS*** und ***ABORTS*** klassifizieren. Diese sogenannten Ausnahme-Typen teilen dem System-Programmierer mit, ob die Prozedur, in der eine Ausnahme vorgekommen ist, wiederholt werden kann:

- ***FAULTS:*** Wenn ein Fehler entdeckt worden ist, zeigen die in den Stacks geretteten CS- und EIP-Werte auf den fehlerverursachenden Befehl. Die Prozedur kann daher nochmals gestartet werden.
- ***TRAPS:*** Wenn ein Fehler entdeckt worden ist, zeigen die in den Stack geretteten CS- und EIP-Werte auf den Befehl, der im Programm-Ablauf dynamisch nach dem fehlerverursachenden Befehl folgt. Dies bedeutet folgendes: Ist der Fehler in einem JMP-Befehl aufgetreten, zeigen die CS- und EIP-Werte auf das ***Ziel*** von JMP und nicht auf den Befehl unmittelbar nach JMP.
- ***ABORTS:*** Dieser Ausnahme-Typ erlaubt weder die Lokalisierung des fehlerverursachenden Befehls noch den Neustart der Prozedur, in der der Fehler vorgekommen ist. ABORT wird daher benutzt, um fatale Fehler zu melden. Dies sind z. B. Hardware-Fehler und unbeständige oder illegale Werte in den System-Tabellen.

Die folgende Tabelle enthält alle Ausnahme-Bedingungen und zeigt, ob der fehlerhafte Befehl wiederholt werden kann (Ausnahme-Typ) und ob ein Fehler-Code im Stack des Fehler-Handlers abgelegt wird:

Vektor-Nummer	Ausnahme-Bedingung	Rücksprung-Adresse (im Stack) zeigt zum fehlerhaften Befehl	Wiederholbar? (Ausnahme-Typ)	Fehler-Code? (im Stack)
0	Divisions-Fehler	Ja	Ja (FAULT)	Nein
1	Debug	*	*	Nein
3	Breakpoint-Interrupt (INT 3-Befehl)	Nein	Nein (TRAP)	Nein
4	Overflow (INTO-Befehl)	Nein	Nein (TRAP)	Nein
5	Feldgrenze überschritten (BOUND-Befehl)	Ja	Ja (FAULT)	Nein
6	Undefinierter Operations-Code	Ja	Ja (FAULT)	Nein
7	Prozessor-Erweiterung nicht vorhanden	Ja	Ja (FAULT)	Nein
8	Doppel-Fehler	Ja	Nein (ABORT)	Ja (Immer 0000)
10	Ungültiges Task-Status-Segment	Ja	Ja** (FAULT)	Ja
11	Segment nicht vorhanden	Ja	Ja (FAULT)	Ja
12	Stack-Segment-Unter-/ Überlauf oder nicht vorhanden	Ja	Ja (FAULT)	Ja
13	Allgemeine Schutzverletzung	Ja	Ja (FAULT/TRAP)	Ja
14	Page-Fehler	Ja	Ja (FAULT)	Ja
16	Fehler der Prozessor-Erweiterung	Ja	Ja (FAULT)	Nein
17	Alignment Check	Ja	Ja (FAULT)	Ja (immer 0000)
18–31	Reserviert für Zwei-Byte-Software-Interrupts	Nein	Nein (TRAP)	Nein

* Bei Debug-Ausnahmen kann die Rücksprung-Adresse zum fehlerhaften oder zum nächsten Befehl zeigen. Der Fehler-Handler kann jedoch den Ausnahme-Typ feststellen, wenn er die Debug-Register DR6 und DR7 überprüft.

** Ein ungültiger TSS-Fehler ist nicht wiederholbar, wenn er während der Bearbeitung eines externen Interrupts vorkommt.

6.4 Ausnahme-Bedingungen und Fehlerbehandlung

6.4.1 Interrupt 0: Divisions-Fehler

Diese Ausnahme kann während der Ausführung eines DIV- (vorzeichenlose Division) oder eines IDIV-Befehls (Integer-Division) vorkommen, wenn z. B. durch Null dividiert wird oder sich ein Quotienten-Überlauf ergibt.

Der Fehler-Handler kann bei dieser Ausnahme den Quotienten durch einen vordefinierten Wert ersetzen und anschließend der fehlerhaften Task die Fortsetzung erlauben. Da der EIP-Wert auf das erste Byte des Divisions-Befehls zeigt, muß der Fehler-Handler vor der Rückkehr in die unterbrochene Prozedur den Rücksprung-Zeiger inkrementieren.

6.4.2 Interrupt 1: Debug

Der Prozessor verursacht eine Debug-Ausnahme nach jedem Befehl, wenn das TF-Bit (Trap Flag) gesetzt ist. Wenn er nun eine Interrupt-Prozedur aufruft, werden die EFLAGS in den Stack gerettet und das TF-Bit im CPU-Flag-Register zurückgesetzt, so daß es in der Handler-Prozedur zu keiner Debug-Ausnahme kommen kann. Erst wenn der Prozessor mit IRET den Handler verläßt, werden die EFLAGS und damit TF = 1 vom Stack in die CPU zurückgeschrieben. Die Handler-Prozedur für diese Ausnahme gehört typischerweise zu einem Debug-System, das den schrittweisen Test eines Programms erlaubt.

6.4.3 Interrupt 3: Breakpoint

Diese Ausnahme wird vom INT 3-Befehl verursacht und bietet die Möglichkeit, irgendwo in einem ausführbaren Segment einen Abbruchpunkt zu definieren. Dabei kann das Betriebssystem oder ein Debug-System für ein ausführbares Segment ein Alias-Daten-Segment benützen, um darin einen INT 3-Befehl zu plazieren. Dieser Befehl verhindert dann die normale Ausführung des Segments, so daß der Debugger bei kritischen Punkten in einer Task deren Register-Inhalte, Variablen usw. anzeigen kann.

Der EIP-Wert zeigt auf den nächsten Befehl. Hat der Debugger den INT 3-Befehl durch einen gültigen OpCode ersetzt, muß er vor der Rückkehr EIP mit 1 dekrementieren.

6.4.4 Interrupt 4: Overflow

Diese Ausnahme kommt vor, wenn das OF-Bit (Overflow Flag) gesetzt ist und der Prozessor einen INTO-Befehl ausführt. Da sowohl die vorzeichenbehaftete als auch die vorzeichenlose Arithmetik die gleichen Arithmetik-Befehle benützen, kann der Prozessor die beabsichtigte Arithmetik nicht erkennen. Daher generiert er keine Overflow-Ausnahme, sondern setzt nur das OF-Bit, wenn Ergebnisse als vorzeichenbehaftete Zahlen ihren Darstellungsbereich überschreiten.

Wenn eine Arithmetik vorzeichenbehaftete Operanden benützt, sollte das OF-Bit direkt getestet oder der INTO-Befehl benützt werden. Tritt die Ausnahme auf, wird der Ausnahme-Handler normalerweise die Task abbrechen. In manchen Anwendungen genügt es aber, das Ergebnis durch einen Maximal-Wert zu ersetzen, so daß anschließend die Task weitermachen kann. Der EIP-Wert zeigt auf den nächsten Befehl.

6.4.5 Interrupt 5: Feldgrenze überschritten

Diese Ausnahme kommt immer dann vor, wenn der Prozessor den BOUND-Befehl ausführt und dabei einen Operanden jenseits der spezifizierten Feldgrenze findet. Diese Bedingung kennzeichnet einen Programm-Fehler. Die sicherste Aktion für den Fehler-Handler ist dann der Abbruch der Task. In manchen Anwendungen genügt es aber, den fehlerhaften Operanden zu korrigieren. Der EIP-Wert zeigt auf das erste Byte des fehlerhaften Befehls.

6.4.6 Interrupt 6: Undefinierter Operations-Code

Diese Ausnahme wird verursacht, wenn der Prozessor im Befehls-Strom einen undefinierten OpCode entdeckt. So kann es z. B. vorkommen, daß ein JMP-Befehl-Read-Only-Daten in einem ausführbaren Segment benützt.

Die undefinierte Op-Code-Ausnahme hat mehrere Varianten:

- Das erste Byte eines Befehls ist vollständig ungültig; z. B. 64H.
- Das erste Byte eines Befehls kennzeichnet einen Zwei-Byte-OpCode, wobei das zweite Byte ungültig ist; z. B. OFH gefolgt von OFFH.
- Einer der Operanden des Befehls kann nicht zusammen mit dem OpCode benützt werden.
- Die OpCode-Erweiterung im zweiten Byte des Befehls enthält einen Wert, der nicht zusammen mit dem OpCode benützt werden kann; z. B. OpCode OF6H mit XX001XXXB im zweiten Byte.
- Der OpCode fordert einen Speicher-Operanden, bekommt aber ein Register; z. B. LGDT AX.

Da der gelieferte Operand ungültig ist, sollte der Ausnahme-Handler den Befehl nicht wiederholen.

Der Anwender kann diese Ausnahmen benützen, um eine Erweiterung des 80486-Befehlsatzes zu implementieren. Der Ausnahme-Handler wird dann diesen Befehl interpretieren und vor der Rückkehr den EIP-Wert auf eine Position jenseits des erweiterten Befehls stellen.

6.4.7 Interrupt 7: Prozessor-Erweiterung nicht vorhanden

Diese Ausnahme kommt bei einer von zwei Bedingungen vor:

- Der Prozessor trifft auf einen ESC-Befehl (Escape), und das EM-Bit (Emulate) im CR0-Register ist gesetzt (EM = 1).
- Der Prozessor trifft auf einen WAIT-Befehl oder einen ESC-Befehl, und sowohl das MP-Bit (Math present) als auch das TS-Bit (Task Switched) im CR0-Register sind gesetzt.

6.4.8 Interrupt 8: Doppelfehler

Wenn der Prozessor eine Ausnahme entdeckt und beim Aufruf des Fehler-Handlers eine weitere Ausnahme hinzukommt, kann der Prozessor normalerweise die beiden Ausnahmen seriell behandeln. Sollte dies aber nicht möglich sein, kommt es statt dessen zur Doppelfehler-Ausnahme. Damit der 80486 bestimmen kann, wann er einen Doppelfehler melden muß, unterteilt er die Ausnahmen in drei Klassen:

- die umgänglichen Ausnahmen,
- die mitverursachenden Ausnahmen und
- die Page-Fehler.

Die Bedeutung dieser Klassifizierung ist in der folgenden Tabelle zu sehen.

Klasse	Vektor-Nummer	Ausnahme
Umgängliche Ausnahmen	1	Debug
	2	NMI
	3	Breakpoint
	4	Overflow
	5	Feldgrenze überschritten
	6	Undefinierter Operations-Code
	7	Prozessor-Erweiterung nicht vorhanden
	16	Fehler der Prozessor-Erweiterung
Mitverursachende Ausnahmen	0	Divisions-Fehler
	9	Segment-Überlauf der Prozessor-Erweiterung
	10	Ungültiges Task-Status-Segment
	11	Segment nicht vorhanden
	12	Stack-Segment-Unter-/Überlauf oder nicht vorhanden
	13	Allgemeine Schutzverletzung
Page-Fehler	14	Page-Fehler

Welche Ausnahme-Kombinationen zu einem Doppelfehler führen und welche seriell behandelt werden, zeigt die folgende Tabelle:

		Zweite Ausnahme		
		Umgängliche Ausnahme	Mitverursachende Ausnahme	Page-Fehler
Erste Ausnahme	Umgängliche Ausnahme	seriell	seriell	seriell
	Mitverursachende Ausnahme	seriell	Doppelfehler	seriell
	Page-Fehler	seriell	Doppelfehler	Doppelfehler

Beispiel:

- Das Code-Segment, das den Ausnahme-Handler enthält, ist als „nicht präsent" (P = 0) markiert (mitverursachende Ausnahme; Vektor-Nummer = 11)
- Beim Aufruf dieses Handlers wird ein Stack-Überlauf verursacht (mitverursachende Ausnahme; Vektor-Nummer = 12).

Bei einem Doppelfehler speichert der Prozessor den Null-Fehler-Code im Stack. Kommt beim Aufruf des Doppel-Fehler-Handlers noch ein weiterer Fehler hinzu, schaltet sich der Prozessor ab (Shut-down-Zustand).
Um eine solche „Katastrophe" zu vermeiden, muß sich der Doppel-Fehler-Handler in einer separaten Task aufhalten. Dadurch kann die Abspeicherung des Fehler-Codes keinen Stack-Fehler verursachen. Zusätzlich muß der Handler immer präsent sein (P = 1). Die Ausnahme – Segment nicht vorhanden – kann dann nicht vorkommen.

6.4.9 Interrupt 10: Ungültiges Task-Status-Segment

Diese Ausnahme kommt vor, wenn der Prozessor während eines Task-Wechsels eine der folgenden Regelwidrigkeiten im neuen Task-Status-Segment entdeckt. Dabei wird im Stack ein Fehler-Code abgelegt. Das EXT-Bit zeigt an, ob die Ausnahme von einem externen Ereignis getriggert worden ist. So könnte z. B. ein externer Interrupt über einen Task-Gate-Deskriptor einen Task-Wechsel verursacht haben, wobei das neue Task-Status-Segment als fehlerhaft erkannt worden ist.

Der Handler für diese Ausnahme muß sich in einer separaten Task aufhalten, die über einen Task-Gate-Deskriptor in der IDT aufgerufen wird. Der Fehler-Code identifiziert entweder die fehlerhafte TSS oder enthält einen fehlerhaften Selektor aus der TSS. Der Befehl, der den Fehler verursacht hat, kann nach IRET wiederholt werden.

Die folgende Tabelle zeigt die möglichen Ursachen für Interrupt 10 und die daraus resultierenden Fehler-Codes:

Ursache	Fehler-Code
Der Limit-Wert im TSS-Deskriptor ist kleiner als 103	TSS-Selektor +EXT
Ungültiger LDT-Selektor oder LDT nicht vorhanden	LDT-Selektor +EXT
Stack-Segment-Selektor ist außerhalb der Tabellen-Grenze	SS-Selektor +EXT
Stack-Segment ist kein beschreibbares Segment	SS-Selektor +EXT
Stack-Segment DPL erreicht nicht den neuen CPL-Wert	SS-Selektor +EXT
Stack-Segment-Selektor RPL ≠ CPL	SS-Selektor +EXT
Code-Segment-Selektor ist außerhalb der Tabellen-Grenze	CS-Selektor +EXT
Code-Segment-Selektor bezieht sich nicht auf das Code-Segment	CS-Selektor +EXT
„Non-Conforming"-Code-Segment DPL ≠ neuer CPL	CS-Selektor +EXT
„Conforming"-Code Segment DPL > neuer CPL	CS-Selektor +EXT
DS-, ES-, FS- oder GS-Segment-Selektoren sind außerhalb der Tabellen-Grenzen	DS/ES/FS/GS-Selektoren +EXT
DS, ES, FS oder GS kennzeichnen keine lesbaren Segmente	DS/ES/FS/GS-Selektoren +EXT

Interrupt 10 kommt ***nicht*** vor, wenn ein Segment-Register-Feld oder das „BACK LINK"-Feld als „nicht präsent" markiert, aber sonst gültig ist. In diesem Fall wird die Ausnahme „Segment nicht vorhanden" (Interrupt 11) geliefert.

Zeigt während eines Interlevel-Call-Befehls ein privilegierter Stack-Selektor im TSS auf einen „nicht präsenten" Deskriptor, wird die Ausnahme „Stack-Segment-Überlauf" oder „nicht vorhanden" (Interrupt 12) ausgegeben.

Einige Bedingungen, die zu Interrupts führen, können zurückgenommen werden. Enthält z. B. ein System einen virtuellen Speicher, so kann eine „nicht vorhandene" LDT vom Hintergrund-Speicher nachgeladen werden.

6.4.10 Interrupt 11: Segment nicht vorhanden

Diese Ausnahme kommt vor, wenn die CPU entdeckt, daß das P-Bit in einem Deskriptor gelöscht ist. Dies kann sich in einem der folgende Fälle ereignen:

- Während des Ladens der CS-, DS-, ES-, FS- oder GS-Register, nicht aber während des Ladens von SS. Im letzteren Fall wird die Ausnahme Stack-Segment-Unter-/Überlauf oder „nicht vorhanden" ausgegeben.
- Während des Ladens der LDT-Register mit einem LLDT-Befehl, nicht aber während des Ladens von LDT durch eine Task-Wechsel-Operation. Im letzteren Fall wird die Ausnahme „Ungültiges Task-Status-Segment" ausgegeben.
- Während des Versuchs, einen Gate-Deskriptor zu benützen, der als „nicht vorhanden" markiert ist.

Ein Betriebssystem wird typischerweise diese Ausnahme benützen, um ein virtuelles Speichersystem zu implementieren. Der Befehl, der Interrupt 11 verursacht hat, ist nach IRET wiederholbar. Davon ausgenommen ist der Fall eines Task-Wechsels.

Wird während eines Task-Wechsels CS, DS, ES, FS oder GS geladen und dabei eine „nicht vorhanden"-Ausnahme entdeckt, kommt diese Ausnahme in der ***neuen*** Task vor. In diesem Fall zeigt der Rücksprung-Wert auf den ersten Befehl in der neuen Task.

6.4.11 Interrupt 12: Stack-Segment-Unter-/Überlauf oder „nicht vorhanden"

Diese Ausnahme kommt in einer von zwei allgemeinen Bedingungen vor:

- Wenn durch eine Operation, die sich auf das SS-Register bezieht, die Segment-Grenze verletzt wird. Diese Operation enthält sowohl stackorientierte Befehle wie POP, PUSH, ENTER und LEAVE als auch andere speicher-bezogene Befehle, die SS implizit benützen; z. B. MOV AX, [EBP + 6].
  ENTER verursacht diese Ausnahme, wenn der Stack für den lokalen Arbeitsbereich zu klein ist. Ein Interlevel-CALL bezieht sich auf zwei Stacks. Daher kann eine Stack-Limit-Verletzung von einem der beiden Stacks verursacht werden.
- Wenn versucht wird, das SS-Register mit einem als „nicht vorhanden" markierten Deskriptor zu laden. Dies kann bei einem Task-Wechsel, einem Interlevel-CALL, einem Interlevel-Rücksprung, einem LSS-Befehl oder einem MOV- oder POP-Befehl mit SS vorkommen.

Wenn die Ausnahme auf ein „nicht vorhandenes" Stack-Segment oder auf einen Überlauf des neuen Stack während eines Interlevel-CALL zurückzuführen ist, enthält der Fehler-Code den Selektor des entsprechenden Segments. Der Ausnahme-Handler kann dann das Present-Bit im Deskriptor prüfen, um festzustellen, welche der beiden Bedingungen zur Ausnahme geführt hat.

Der Befehl, der Interrupt 12 verursacht hat, kann normalerweise nach IRET wiederholt werden.

Wird aber während eines Task-Wechsels beim Laden von SS ein Stack-Fehler erkannt, dann wird diese Ausnahme in der neuen Task geliefert. In diesem Fall zeigt der Rücksprung-Wert auf den ersten Befehl in der neuen Task.

6.4.12 Interrupt 13: Allgemeine Schutzverletzung

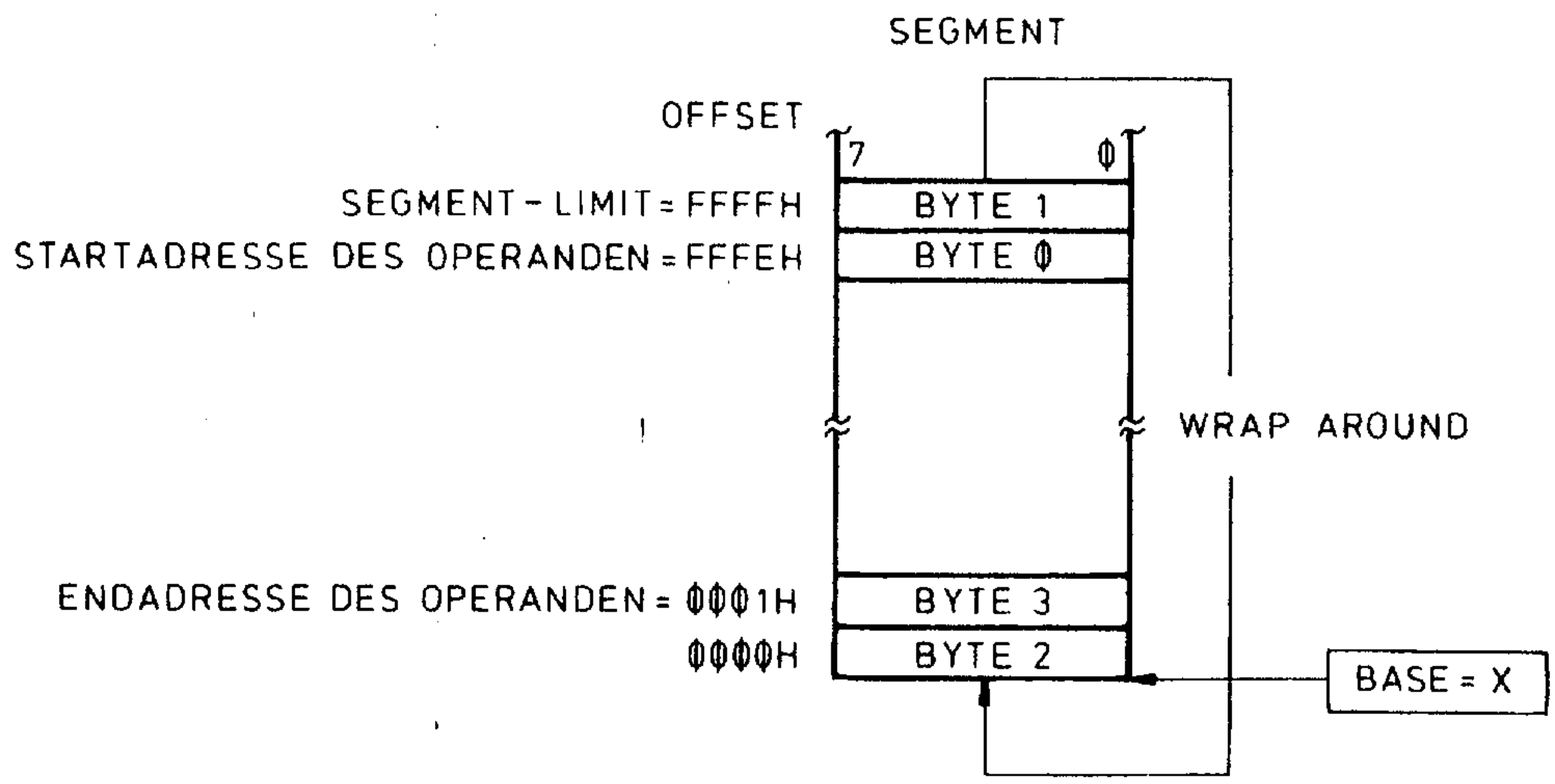
Alle Schutzverletzungen, die nicht durch eine spezielle Ausnahme gekennzeichnet sind, verursachen eine allgemeine Schutz-Ausnahme. Diese enthält:

- Überschreitung der Segment-Obergrenze bei der Benützung von CS, DS, ES, FS oder GS
- Überschreitung der Segment-Obergrenze beim Zugriff auf eine Deskriptor-Tabelle

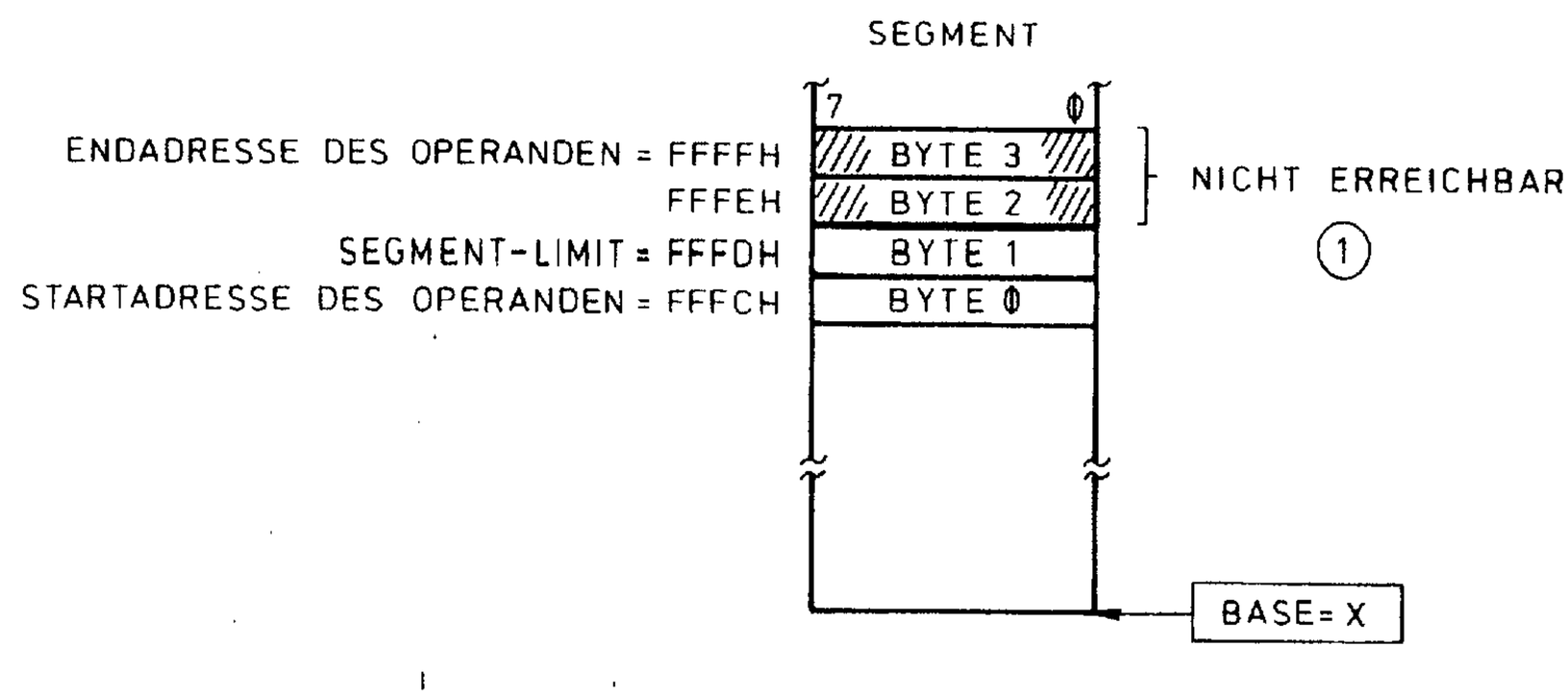
- Sprung in ein Segment, das nicht ausführbar ist.
- Schreiben in ein Read-Only-Segment oder in ein Executable-Segment
- Lesen von einem Execute-Only-Segment
- Lesen des SS-Registers mit einem Read-Only-Deskriptor. Sollte der Selektor während eines Task-Wechsels von einem Task-Status-Segment geliefert werden, führt dies zur Ausnahme „Ungültiges Task-Status-Segment".
- Laden von SS, DS, ES, FS oder GS mit dem Deskriptor eines System-Segments.
- Laden von DS, ES, FS oder GS mit dem Deskriptor eines Executable-Segments, das nicht lesbar ist.
- Laden von SS mit dem Deskriptor eines Executable-Segments
- Laden von CS (z. B. durch CALL, JMP oder Interrupt) mit dem Deskriptor eines Daten-Segments
- Zugriff auf den Speicher über DS, ES, FS oder GS, wenn die Register einen Null-Selektor enthalten
- Wechsel in eine aktive (busy) Task
- Verletzung der Privileg-Regeln
- Laden von CR0 mit PG = 1 (Paging einschalten) und PE = 0 (Protected Mode ausschalten)
- Mit einem Interrupt oder eine Ausnahme-Bedingung über ein Trap- oder Interrupt-Gate vom virtuellen 86-Modus (V86-Mode) in eine „Protected Mode"-Privileg-Ebene unterhalb von 0 (1, 2 oder 3) umschalten.
- Überschreiten der maximalen Befehlslänge von 15 Bytes (dies kann nur vorkommen, wenn redundante Prefix-Bytes vor den Befehl plaziert werden).
- Zugriff auf Port-Adressen, die in der „I/O Permission Bit Map" durch „1"-Bits gekennzeichnet sind.
- Segment-Überlauf der Prozessor-Erweiterung

Diese Ausnahme kommt vor, wenn der 80486 während eines Coprozessor-Operanden-Transfers eine Page- oder Segment-Verletzung entdeckt.
Solche Verletzungen werden durch die folgenden Bedingungen verursacht:

1. Ein Coprozessor-Befehl plaziert einen Operanden an eine Adresse nahe der Segment-Grenze (FFFFH bei USE16-Segmenten, FFFFFFFFH bei USE32-Segmenten, 0 bei Expand-Down-Segmenten). Dabei kann es vorkommen, daß die Start- und Endadressen des Coprozessor-Operanden an den ***gegenüberliegenden*** Enden des Segments liegen (wrap around). Eine solche Bedingung illustriert am Beispiel eines 4-Byte-Operanden das folgende Bild:

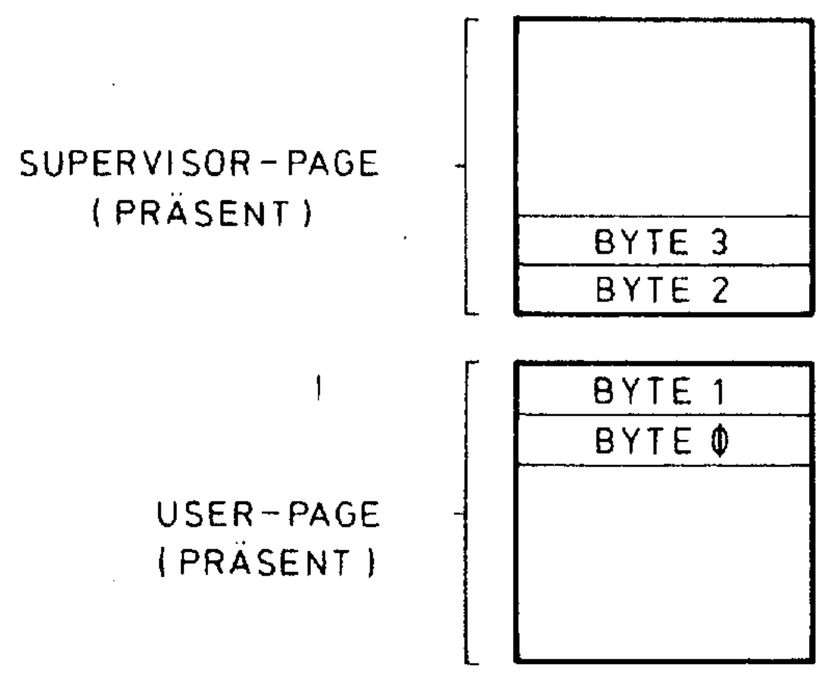


2. Ein Coprozessor-Operand beansprucht Adressen, die ***nicht erreichbar*** sind. Um dies zu verdeutlichen, soll die folgende Bedingung vorliegen:



Ein 4-Byte-Operand umfaßt die Adressen FFFCH-FFFFH (4 Bytes). Das Segment, das den Operanden aufnehmen soll, hat den Limit-Wert FFFDH. Dies bedeutet, daß die Adressen FFFEH und FFFFH nicht erreichbar sind ①.

3. Ein Coprozessor-Operand beginnt in einer präsenten und erreichbaren Page und endet in einer nicht-präsenten oder nicht-erreichbaren Page. Das folgende Bild illustriert eine solche Bedingung. Dabei ist vorausgesetzt, daß sich der Prozessor in der User-Ebene 3 aufhält.



Die Fehler im Fall 2 und Fall 3 können vermieden werden, wenn am oberen Ende des Segments oder der Page ein Puffer von 108 Bytes freigehalten wird. Dies entspricht der maximalen Größe eines Coprozessor-Operanden. Die Segment-Überlauf-Ausnahme sollte vom FNINIT-Befehl (No-Wait-Form) behandelt werden. Dieser Befehl bringt den Coprozessor in einen definierten Anfangszustand. Der Ausnahme-Handler sollte zusätzlich die Task abbrechen.

Der Prozessor schreibt einen Fehler-Code in den Stack des Ausnahme-Handlers. Wenn der Deskriptor, der die Ausnahme verursacht, geladen wird, enthält der Fehler-Code den Selektor dieses Deskriptors.

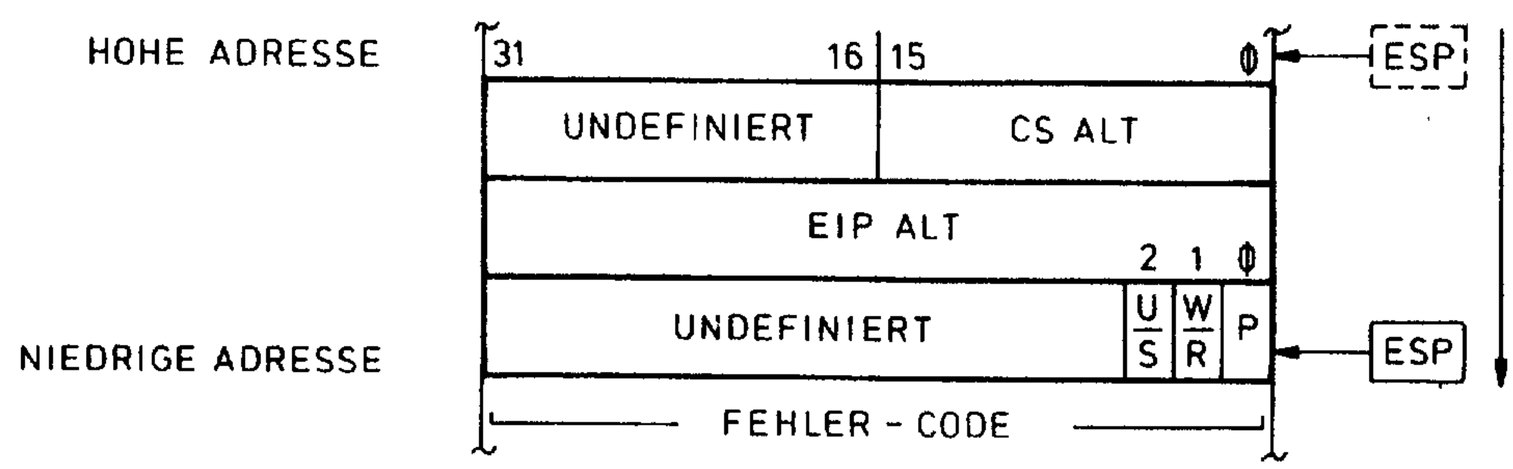
Der Rücksprung-Wert zeigt auf den Anfang des fehlerhaften Befehls. Da eine allgemeine Schutz-Ausnahme meistens auf einen Programmier-Fehler zurückzuführen ist, sollte sie durch einen Neustart des fehlerhaften Befehls nicht zurückgewonnen werden. Denn es ist zu mühsam, den Fehler auf diese Weise zu identifizieren.

6.4.13 Interrupt 14: Page-Fehler

Diese Ausnahme kommt vor, wenn der Paging-Mechanismus eingeschaltet ist (PE = 1) und der Prozessor bei der Übersetzung einer linearen Adresse in eine physikalische Adresse eine der folgenden Bedingungen entdeckt:

- Der an der Adreß-Übersetzung beteiligte Page-Table- oder Page-Deskriptor hat in seinem Present-Bit eine Null.
- Die augenblickliche Prozedur ist nicht genügend privilegiert, um auf die gewünschte Page zuzugreifen.

Damit der „Page Fault Handler" den Typ des Fehlers identifizieren kann, speichert der Prozessor im Stack des Handlers einige Diagnose-Daten.
Das Layout dieses Stack zeigt das folgende Bild:

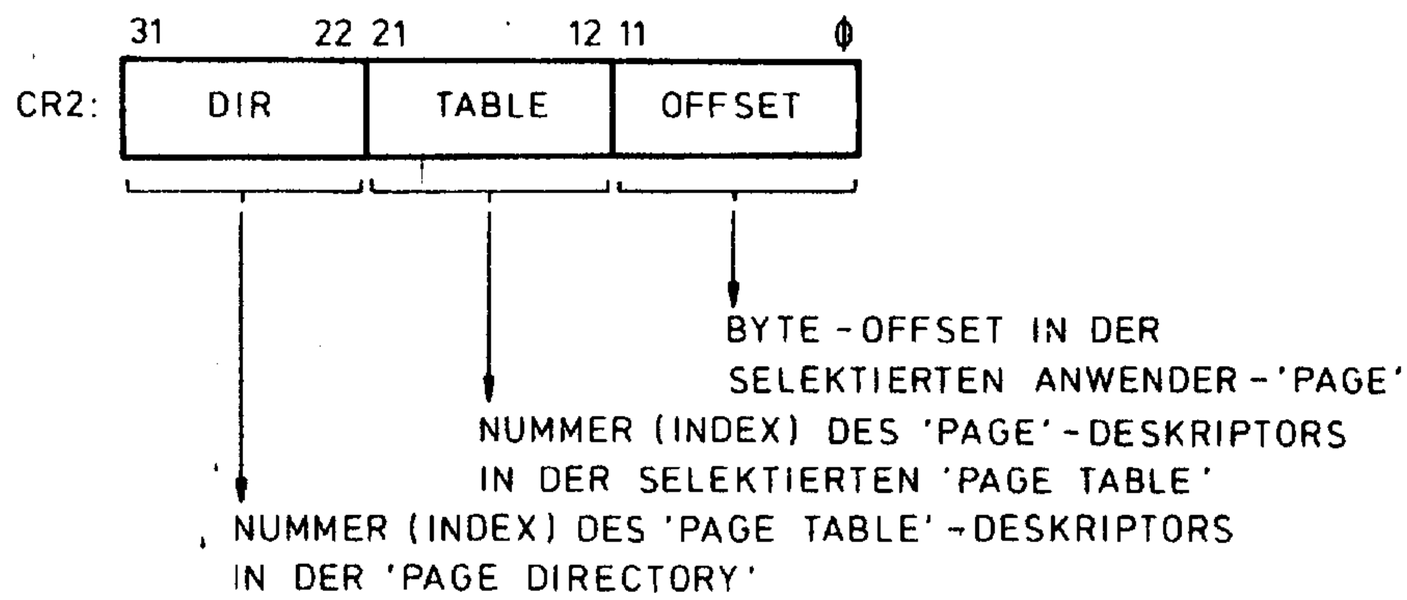


Es ist zu erkennen, daß ab der augenblicklichen ESP-Position (Top of Stack) die ***logische*** Adresse CS:EIP des Befehls enthalten ist, der den Fehler verursacht hat. Danach folgt ein ***Fehler-Code,*** der den Typ des Fehlers beschreibt. Dabei haben die Felder in diesem Code folgende Bedeutung:

U/S:	**Prozessor-Modus** 0 = Der Fehler hat sich bei einem Zugriff ereignet, als der Prozessor im Supervisor-Modus aktiv war. 1 = Der Fehler hat sich bei einem Zugriff ereignet, als der Prozessor im User-Modus war.
W/R:	**Zugriffs-Typ** 0 = Der Fehler hat sich bei einem Lesezugriff ereignet. 1 = Der Fehler hat sich bei einem Schreibzugriff ereignet.
P:	**Page-Fehler-Interpretation** 0 = Der Fehler wurde durch eine nicht-vorhandene „Page Table" oder Anwender-Page verursacht. 1= Der Fehler wurde durch eine Page-Schutz-Verletzung verursacht.

Damit der „Page Fault Handler" die am Page-Fehler beteiligten Page-Table- und Page-Deskriptoren lokalisieren kann, stellt der 80486 das Spezial-Register ***CR2*** zur Verfügung. In diesem Register speichert der Prozessor die ***lineare*** Adresse, die beim fehlerverursachenden Zugriff benutzt wurde.

Das Format des CR2-Registers zeigt das folgende Bild:



Der Page-Fault-Handler kann die im CR2-Register gespeicherte lineare Adresse benutzen, um die korrespondierenden Page-Table- und Page-Deskriptoren zu lokalisieren (siehe Programmierbeispiel 3.19.9 – Zugriff auf Page-Table- und Page-Deskriptoren und ihre Manipulation bei Page-Fehlern).

Wenn während der Ausführung des Page-Fault-Handlers ein anderer Page-Fehler auftreten kann, sollte der Handler mit

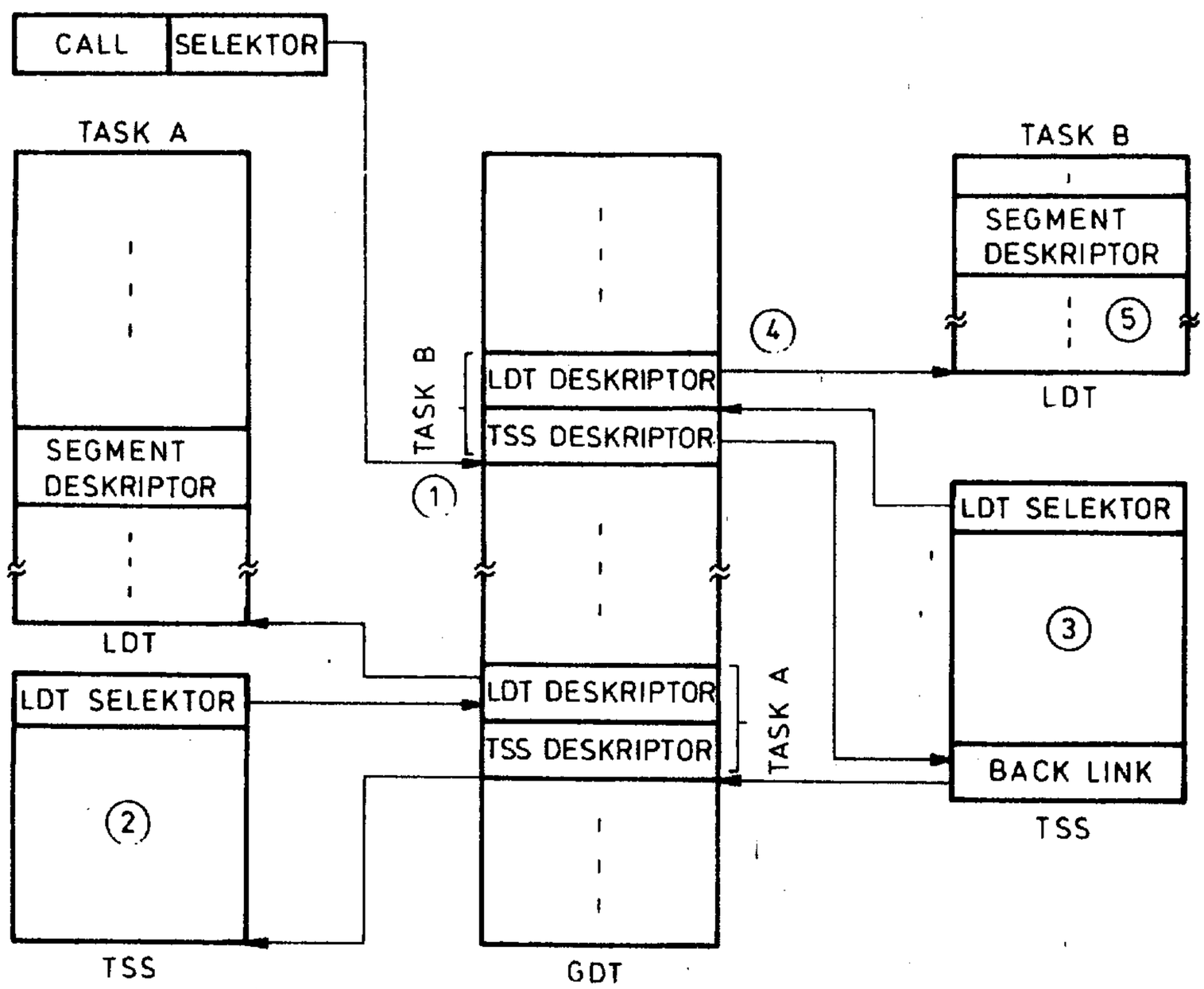
- MOV r32,CR2 ;Lade CR2 in ein allgemeines 32-Bit-Register
- PUSH r32 ;Rette CR2 in den Stack

das CR2-Register in den Stack retten.

6.4.13.1 Page-Fehler bei Task-Wechsel

Wenn der Prozessor einen Task-Wechsel ausführt, aktiviert er in Übereinstimmung mit der angegebenen Hardware-Struktur folgende Mechanismen:

- Er liest in der GDT den TSS-Deskriptor der eintretenden Task B ①.
- Er schreibt den Status der ausscheidenden Task A in das Task-Status-Segment dieser Task ②.
- Er liest den Status der eintretenden Task B vom Task-Status-Segment dieser Task ③.
- Er liest in der GDT den LDT-Deskriptor der eintretenden Task B ④.
- Er liest die Segment-Deskriptoren in der LDT der eintretenden Task B ⑤.



Es ist erkennbar, daß der Prozessor bei diesen Aktionen auf vier System-Segmente zugreift:

- die GDT
- das Task-Status-Segment der ausscheidenden Task A
- das Task-Status-Segment der eintretenden Task B
- die LDT der eintretenden Task B.

Wenn diese System-Segmente in Pages untergebracht sind, kann es vorkommen, daß es während des Task-Wechsels beim Zugriff auf eines dieser Segmente zu einem Page-Fehler kommt.
Dabei ereignet sich der Fehler im Context der neuen Task, wenn der Prozessor auf das Task-Status-Segment, den LDT-Deskriptor oder die LDT der eintretenden Task B zugreift.

Der „Instruction Pointer" EIP zeigt in diesem Fall ***nicht*** auf den Befehl, der den Task-Wechsel verursacht hat, sondern zum nächsten Befehl in der neuen Task. Wenn das Betriebssystem Page-Fehler während der Task-Wechsel zuläßt, sollte der Page-Fault-Handler über einen Task-Gate-Deskriptor aufgerufen werden.

6.4.13.2 Page-Fehler mit inkonsistentem Stack-Pointer

Wenn in einem 80486-Programm ein neues Stack-Segment gebraucht wird, kann z. B. die folgende Befehls-Sequenz benutzt werden:

- MOV SS,AX
- MOV ESP,TOP_OF_STACK

Nachdem das SS-Register umgeladen worden ist, muß der Prozessor beim zweiten Befehl auf den Speicher zugreifen. Dabei kann es vorkommen, daß in einem page-orientierten System während der ESP-Initialisierung ein Page-Fehler auftritt. Dies bedeutet, daß die zwei Teile des Stack-Pointers SS:ESP nicht zusammenpassen, also inkonsistent sind.

Der Prozessor benutzt einen inkonsistenten Stack-Pointer ***nicht,*** wenn der Fehler-Handler eine Task ist oder über ein Trap- oder Interrupt-Gate in einer privilegierten Ebene aufgerufen wird.
Tritt aber der Page-Fehler in der gleichen Ebene auf, in der sich auch der Fehler-Handler befindet, versucht der Prozessor, den augenblicklichen (ungültigen) Stack-Pointer zu benutzen.
Für diesen Fall sollte die Software zur Initialisierung eines neuen Stack ***nicht*** das obige Befehls-Paar, sondern den neuen LSS-Befehl benützen. Wird der Page-Fault-Handler typischerweise in der Privileg-Ebene 0 ausgeführt, kann die Reichweite des Problems auf den Kern des Betriebssystems beschränkt werden.

6.4.14 Interrupt 16: Fehler der Prozessor-Erweiterung

Die Floating-Point-Einheit (FPU) des 80486 kann insgesamt sechs verschiedene Ausnahmebedingungen während der Befehlsausführung erkennen:

- Invalid Operation
- Overflow
- Division durch Null
- Underflow
- Denormalized Operand
- Precision (unexaktes Resultat)

Interrupt 16 wird dabei immer dann generiert, wenn diese Ausnahmen ***nicht*** maskiert sind und das NE-Bit (Numerik Exception) im CR0-Register gesetzt ist (NE = 1). Für mehr Informationen siehe Abschnitt 8.1.3.

6.4.15 Interrupt 17: Alignment-Überprüfung

Durch die Alignment-Überprüfung können Programmstellen entdeckt werden, bei denen Zugriffe auf nicht-ausgerichtete Operanden erfolgen. Beispiele hierfür sind Word-Zugriffe auf ***ungerade*** Byte-Adressen oder Doubleword-Zugriffe auf Adressen, die nicht durch 4 teilbar sind usw.
Die Alignment-Überprüfung kann nur Programme in der Privileg-Ebene 3 (CPL = 3) erfolgen und zwar wenn folgende Bedingungen erfüllt sind:

- Das AM-Bit (Alignment Mask) im CR0-Register muß gesetzt sein (AM = 1) und
- das AC-Bit (Alignment Check) in den EFLAGs muß ebenfalls gesetzt sein (AC = 1).

Für mehr Informationen siehe Abschnitt 7.1.4.