10 Virtueller 8086-Modus

Der 80486 unterstützt die Ausführung von 8086/88/186 oder 80188-Programmen in der 80486-„Protected Mode"-Umgebung. In dieser Umgebung läuft ein 8086-Programm als Teil einer sogenannten ***virtuellen 8086-Task*** (V86-Task). Dabei können

- mehrere V86-Tasks ihre individuellen 8086-Programme ausführen und
- zusammen mit anderen 80486-Tasks multiprogrammiert werden.

Mit einer V86-Task läßt sich eine sogenannte „virtuelle Maschine" implementieren, auf der 8086-Programme ausführbar sind.

Eine solche Maschine besteht aus 80486-Hardware und System-Software.

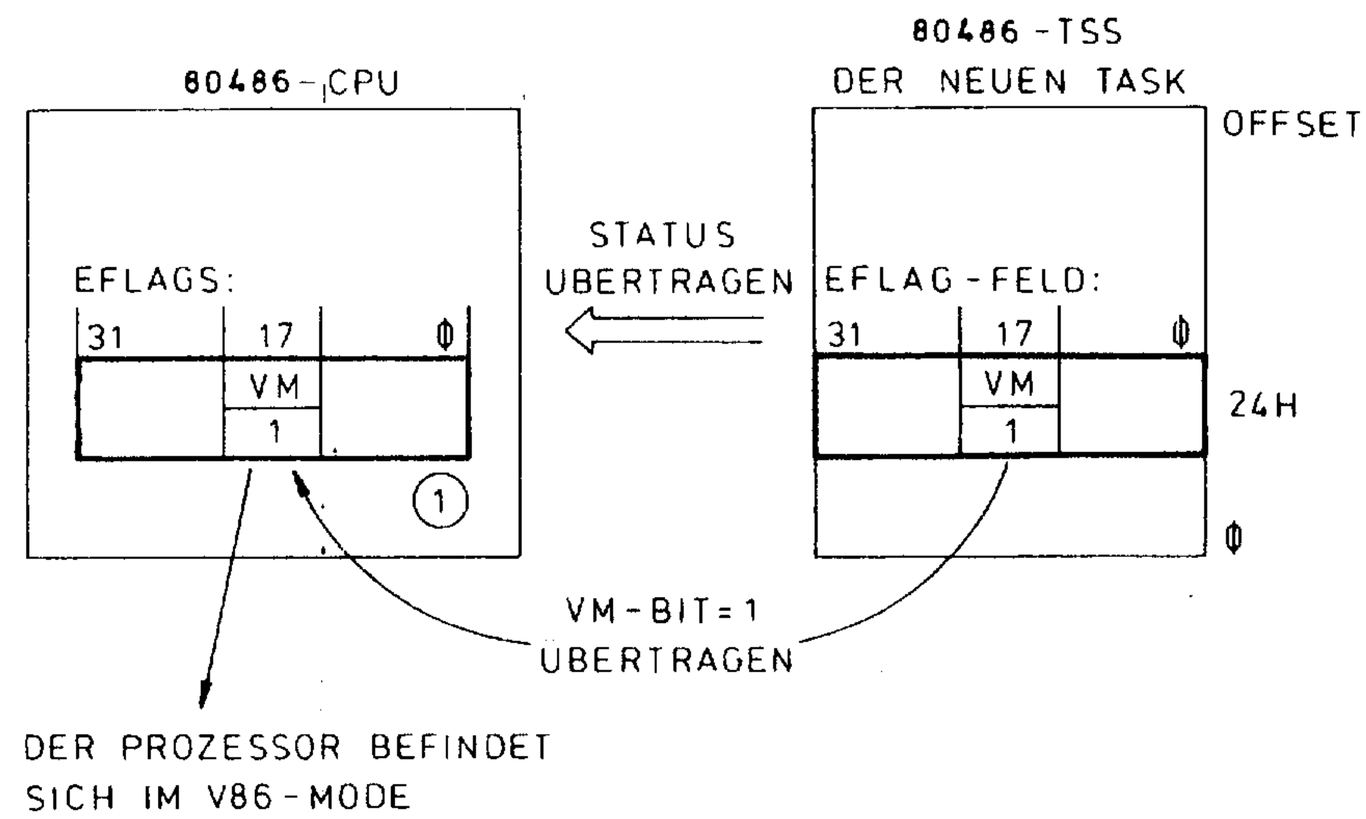
- Die ***Hardware*** steuert die internen Abläufe der virtuellen Maschine. Sie liefert über das Task-Status-Segment einen Satz von virtuellen Registern und einen virtuellen Speicherraum. Dieser Raum ist mit dem ersten Megabyte des linearen Adreßraums der Task identisch. Zusätzlich führt die Hardware alle Befehle aus, die mit diesen Registern und mit diesem Adreßraum arbeiten.
- Die ***Software*** steuert die externen Angelegenheiten der virtuellen Maschine wie I/O-Zugriffe, Interrupts und Ausnahme-Bedingungen.

10.1 Eintritt in den virtuellen 8086-Modus

Ein 8086-Programm läuft typischerweise in einer einzelnen ungeschützten Task. Wenn ein solches Programm in einer geschützten V86-Task laufen soll, muß der 80486 in den sogenannten virtuellen 8086-Modus (V86-Modus) umgeschaltet werden. In diesem Modus kann das Betriebssystem ein existierendes 8086-Programm in die geschützte Multitasking- und Paging-Umgebung des 80486 integrieren.

Dabei gibt es zwei Möglichkeiten, um den Prozessor in den V86-Modus zu bringen:

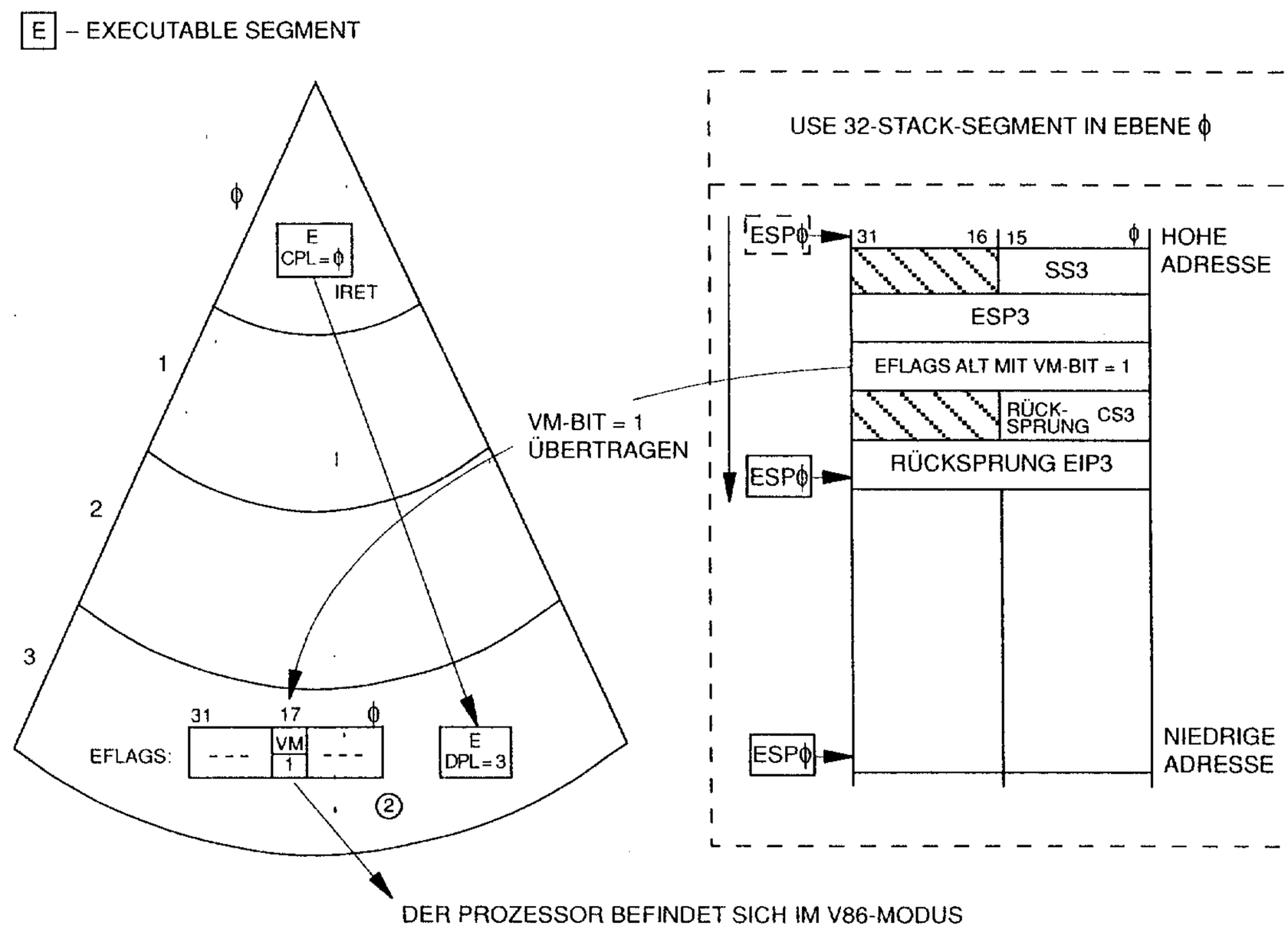
1) Durch einen ***Task-Wechsel*** über ein Task-Gate oder über einen IRET-Befehl, wenn das NT-Bit gesetzt ist (NT = 1). Bei dieser Aktion wird der Status der eintretenden Task vom zugehörigen Task-Status-Segment gelesen und in den Prozessor übertragen. Zu diesem Status gehört das EFLAG-Feld, das im Bit 17 das VM-Bit (Virtual 8086 Mode) enthält. Wenn dieses Bit gesetzt ist (VM = 1), befindet sich der Prozessor nach dem Task-Wechsel im V86-Modus ①.



Die neue Prozedur führt nun 8086-Befehle in einer V86-Task aus. Dies bedeutet, daß der Prozessor das Adressierungsschema des 8086 benützt. Die vom Task-Status-Segment gelesenen Segment-Selektoren haben daher die ***gleiche*** Bedeutung wie beim 8086. Der Prozessor schiebt nach dem Task-Wechsel die Selektoren um 4 Bits nach links und bildet so die üblichen 20-Bit physikalischen Basis-Adressen der individuellen 8086-Segmente.

HINWEIS:
Eine V86-Task läuft immer in der Privileg-Ebene 3 (CPL = 3).

2) Durch einen ***IRET-Befehl,*** wenn das NT-Bit gelöscht ist (NT = 0). Bei dieser Aktion erfolgt ein Privileg-Wechsel von Ebene 0 nach Ebene 3. Dabei werden unter anderem die EFLAGs vom Ebene-0-Stack in das EFLAG-Register der CPU übertragen. Ist bei dieser Übertragung das VM-Bit im EFLAG-Feld des Stack gesetzt (VM = 1), befindet sich der Prozessor nach IRET im V86-Modus ②.



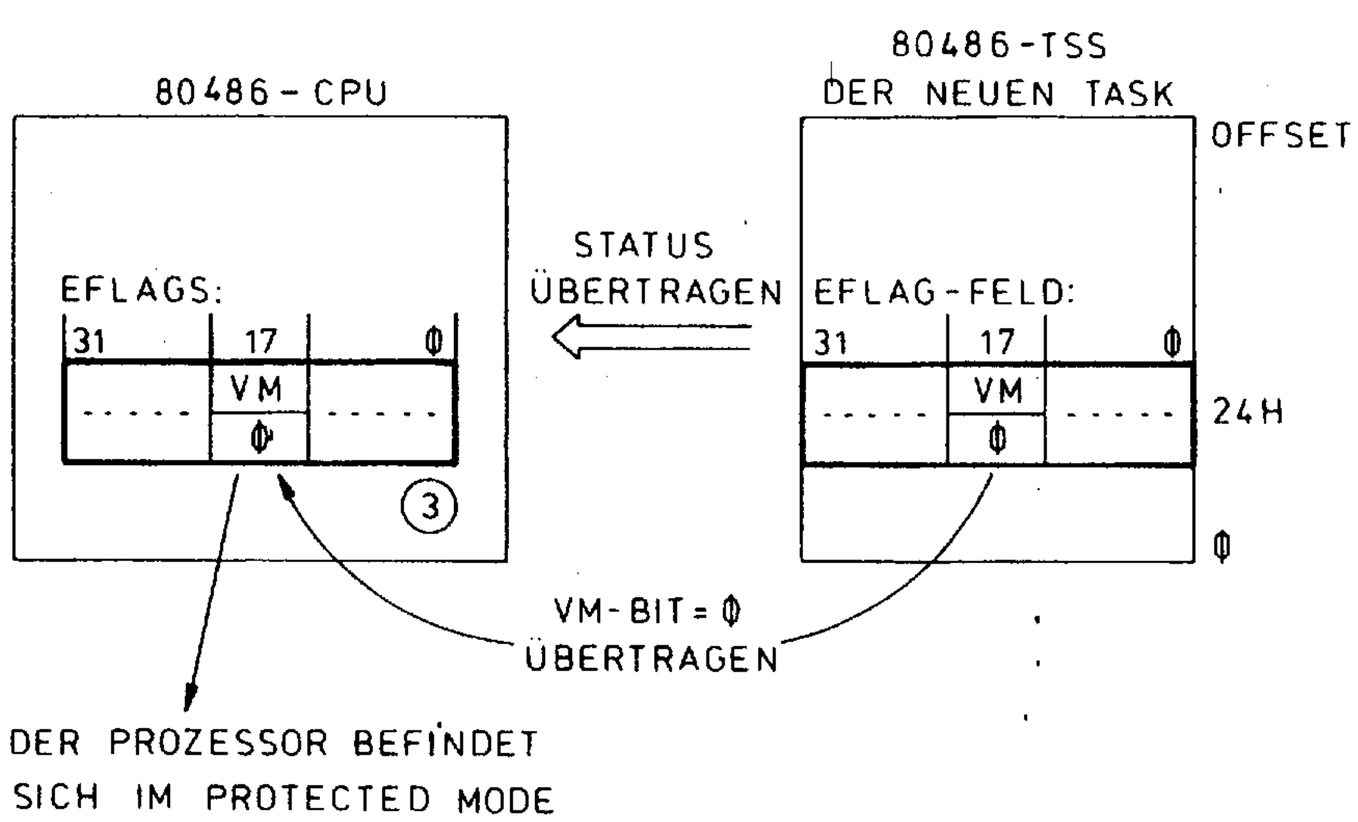
Der Prozessor führt nach dem Privileg-Wechsel 8086-Befehle in einer V86-Task aus und benützt dabei das Adressierungsschema des 8086.

HINWEIS:
Der angegebene Mechanismus funktioniert nur, wenn IRET in der ***Privileg-Ebene 0*** ausgeführt wird. Im anderen Fall bleibt das VM-Bit im EFLAG-Register der CPU unbeeinflußt.

10.2 Verlassen des virtuellen 8086-Modus

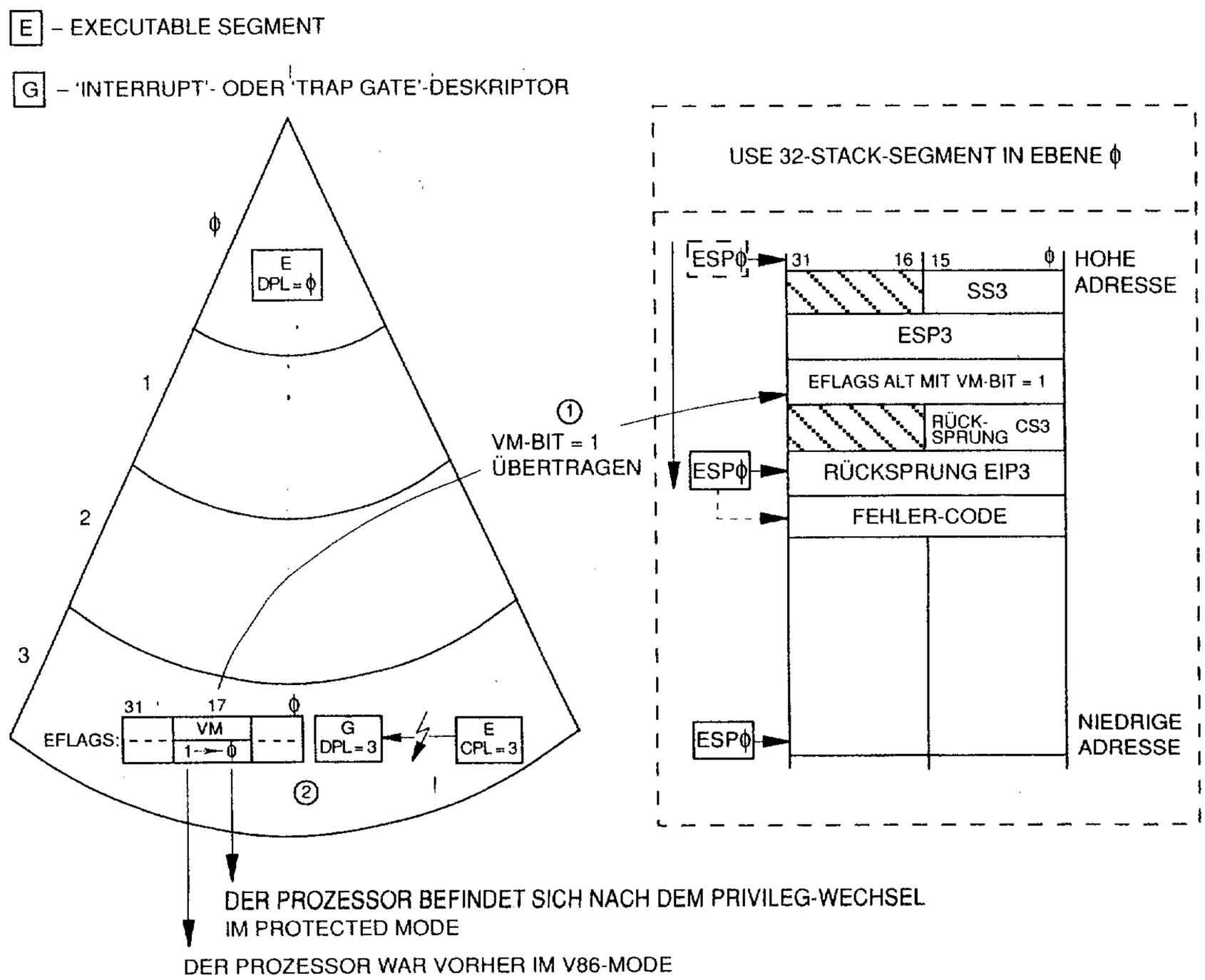
Der Prozessor verläßt den V86-Modus entweder durch einen ***Task-*** oder einen ***Privileg-Wechsel.***

1) Ein Task-Wechsel kann über ein Task-Gate entweder vom Scheduler des Betriebssystems oder durch einen Interrupt bzw. eine Ausnahme-Bedingung veranlaßt werden. Bei dieser Aktion wird der Status der eintretenden Task vom zugehörigen Task-Status-Segment gelesen und in den Prozessor übertragen. Ist die neue Task durch ein 80486-Task-Status-Segment repräsentiert und das VM-Bit im EFLAG-Feld gelöscht (VM = 0), befindet sich der Prozessor nach dem Task-Wechsel in seiner natürlichen „Protected Mode"-Umgebung ③.



Die neue Prozedur führt nun Befehle in einer 80486-Task aus. Dies bedeutet, daß der Prozessor das Adressierungsschema seines „Protected Mode" benützt.

2) Ein Privileg-Wechsel kann entweder durch einen Interrupt oder eine Ausnahme-Bedingung über ein Trap- bzw. Interrupt-Gate veranlaßt werden. Dabei ***muß*** sich die aufgerufene Prozedur in der ***Privileg-Ebene 0*** aufhalten. Der Prozessor speichert dabei die EFLAGs der CPU mit VM = 1 im EFLAG-Feld des Ebene-0-Stack ① und ***löscht*** anschließend in den EFLAGs der CPU das VM-Bit (VM = 0) ②.

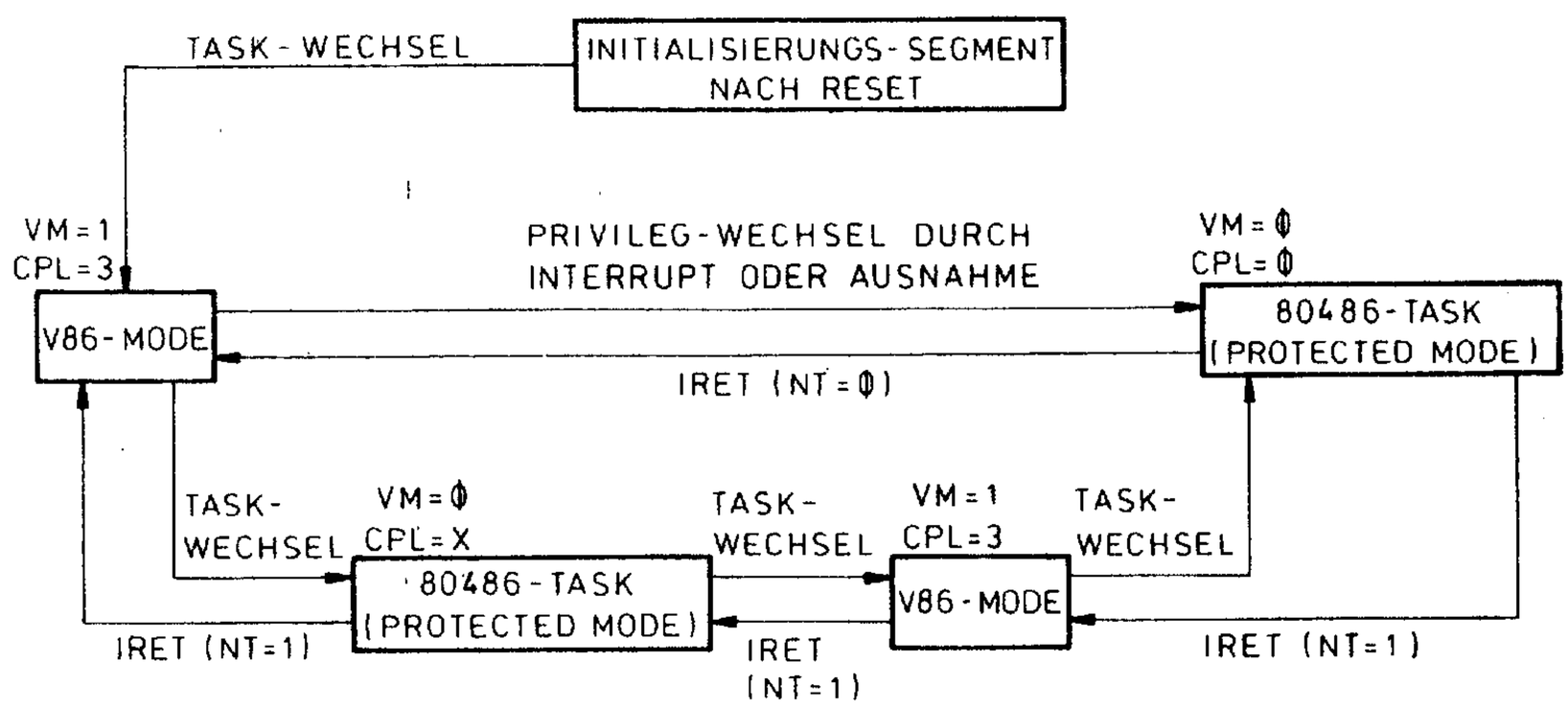


Der Interrupt- oder Ausnahme-Handler führt nun seine Befehle in der natürlichen 80486-„Protected Mode"-Umgebung aus.

HINWEIS:
Wenn der Interrupt- oder Ausnahme-Vektor auf ein Conforming-Code-Segment oder auf eine Privileg-Ebene anders als 0 zeigt, erkennt der Prozessor eine allgemeine Schutzverletzung. Der Fehler-Code ist dann mit dem Selektor des Ziel-Code-Segments identisch.

Wie bereits erwähnt, können mehrere V86-Tasks (VM = 1) zusammen mit 80486-„Protected Mode"-Tasks (VM = 0) gleichzeitig laufen.

Das folgende Mode-Übergangs-Diagramm illustriert hierfür einige Wege des Prozessors:



10.3 Task-Management

Damit eine Task im virtuellen 8086-Mode beginnen kann, muß das Betriebssystem folgende Initialisierungen im korrespondierenden 80486-Task-Status-Segment vornehmen:

1) Setze im EFLAG-Feld das VM-Bit auf 1.
2) Lade das CS-Selektor-Feld mit einem Wert, der nach der Multiplikation mit 16 (4 Bit-Positionen nach links schieben) die lineare Adresse des ersten Code-Segments der Task liefert.
3) Lade das IP-Feld mit einem Wert, der die Eintrittstelle der Task spezifiziert.
4) Setze IOPL im EFLAG-Feld auf 3, wenn die V86-Task das Interrupt-Enable-Flag IF durch Befehle wie CLI, STI, POPF oder IRET erreichen soll. Diese Befehle sind im V86-Mode ***IOPL-sensitiv*** und sind Mitglieder der folgenden Befehlsgruppe:

CLI	– Lösche Interrupt-Enable-Flag
STI	– Setze Interrupt-Enable-Flag
LOCK	– Multibus sperren
PUSHF	– Push Flags
POPF	– Pop Flags
INT n	– Software-Interrupt
IRET	– Interrupt Return

Wenn diese Befehle im V86-Mode (CPL = 3) bei ***IOPL < 3*** aufgerufen werden, verursachen sie eine allgemeine Schutzverletzung.

5) Lade das LDT-Selektor-Feld mit 0. LDT´s werden im V86-Mode ***nicht*** benützt. Wenn aber durch einen Interrupt oder eine Ausnahme-Bedingung ein Privileg-Wechsel erfolgt und die zugehörigen Prozeduren eine LDT benützen, muß das LDT-Feld in der Task mit einem Nicht-Null-Selektor initialisiert sein.
6) Initialisiere die I/O-Permission-Bit-Map mit einem Bit-String, der den Zugriff auf individuelle I/O-Ports erlaubt oder verhindert.

Ein 8086-Programm benützt für die Behandlung von Interrupts und Ausnahmen eine Interrupt-Tabelle, die bei der linearen Adresse 0 beginnt und alls 32-Bit-Zeiger zu den entsprechenden Interrupt- oder Ausnahme-Prozeduren enthält.

Wenn ein 8086-Programm in einer V86-Task läuft, benützt der 80486 diese Tabelle ***nicht.*** Bei allen Interrupts und Ausnahmen, die in einer V86-Task vorkommen, zeigt der Prozessor zur IDT. Diese muß folgende Gate-Deskriptoren enthalten:

- Task-Gates
- 80486-Trap- oder Interrupt-Gates.

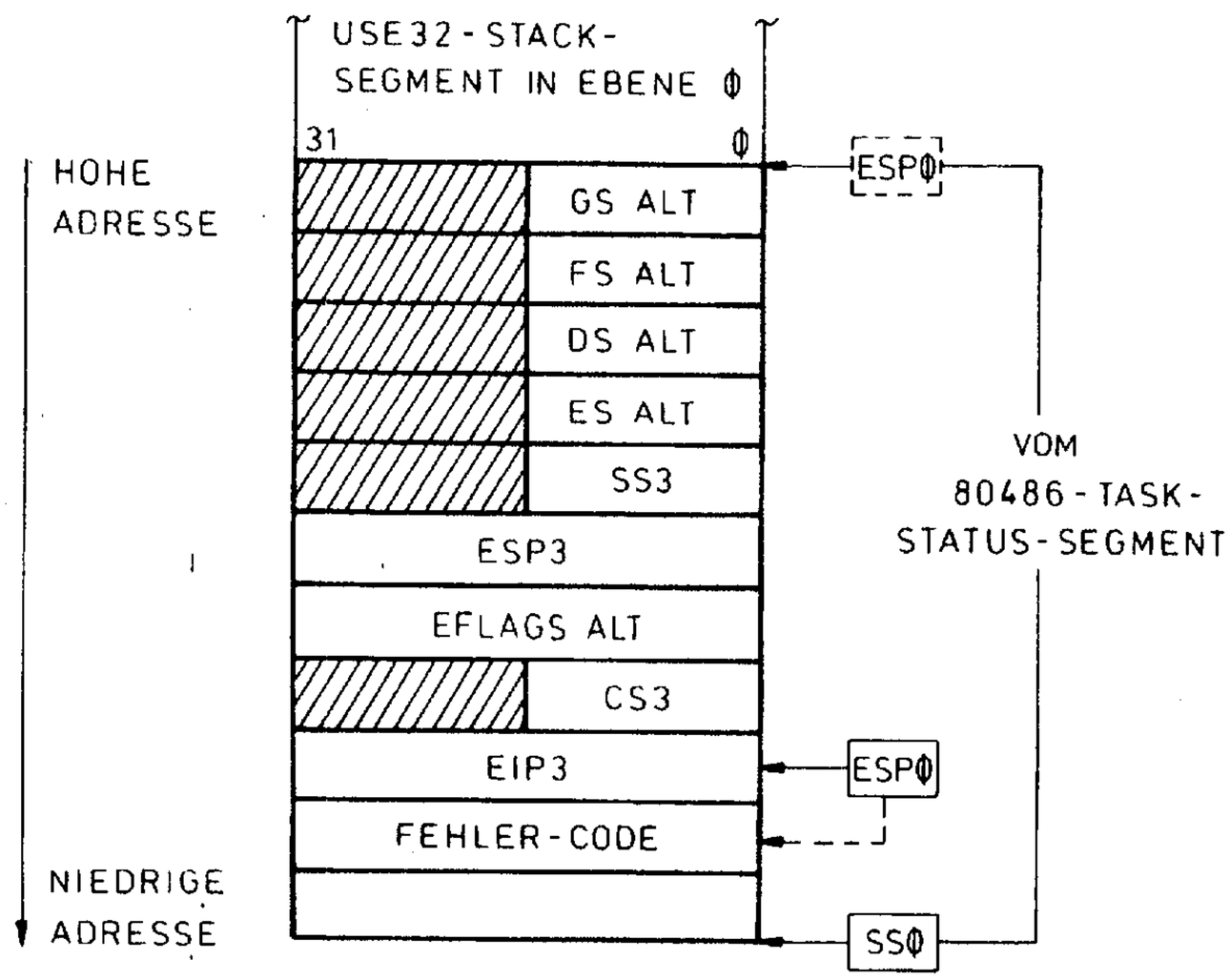
Wenn der Prozessor einen Task-Gate-Deskriptor benützt, kann die aufgerufene ***Handler-Task*** in jeder beliebigen Privileg-Ebene laufen.

Wenn der Prozessor einen Trap- oder Interrupt-Gate-Deskriptor benützt, liefern die CS:EIP-Felder in diesen Deskriptoren einen Zeiger (CSO:EIPO), der die Position einer geeigneten ***Handler-Prozedur*** in der Privileg-Ebene 0 spezifiziert. Der Privileg-Wechsel

bewirkt, daß automatisch auch das Stack-Segment gewechselt wird. Dabei liefern die SS:ESP-Felder im 80486-Task-Status-Segment den Zeiger (SSO:ESPO) dieses Stack.

Da sich der Prozessor nach dem Privileg-Wechsel im „Protected Mode" befindet, können die V86-Werte der Segment-Register ES ... GS ***nicht*** benützt werden. Damit sie aber nach dem Verlassen der Handler-Prozedur mit IRET im V86-Mode wieder zur Verfügung stehen, rettet sie der Prozessor vor dem Wechsel in die Privileg-Ebene 0 in den Privileg-0-Stack.

Das Layout dieses Stack zeigt das folgende Bild:



Wenn die Handler-Prozedur die Segment-Register ES... GS für eigene Aufgaben benötigt, müssen sie als 80486-Selektoren zur Verfügung stehen.

Der Prozessor unterstützt dies, indem er vor dem Aufruf des Prozedur-Handlers die CPU-Segment-Register ES ... GS mit ***Null*** initialisiert. Um die erforderliche Register-Manipulation zu realisieren, kann die Handler-Prozedur den Ebene-0-Stack benützen.

Zu diesem Zweck

- rettet er im Prolog die Segment-Register mit PUSH-Befehlen in den Stack,
- verändert sie in geeigneter Weise und
- lädt sie bei Bedarf in die CPU-Register.

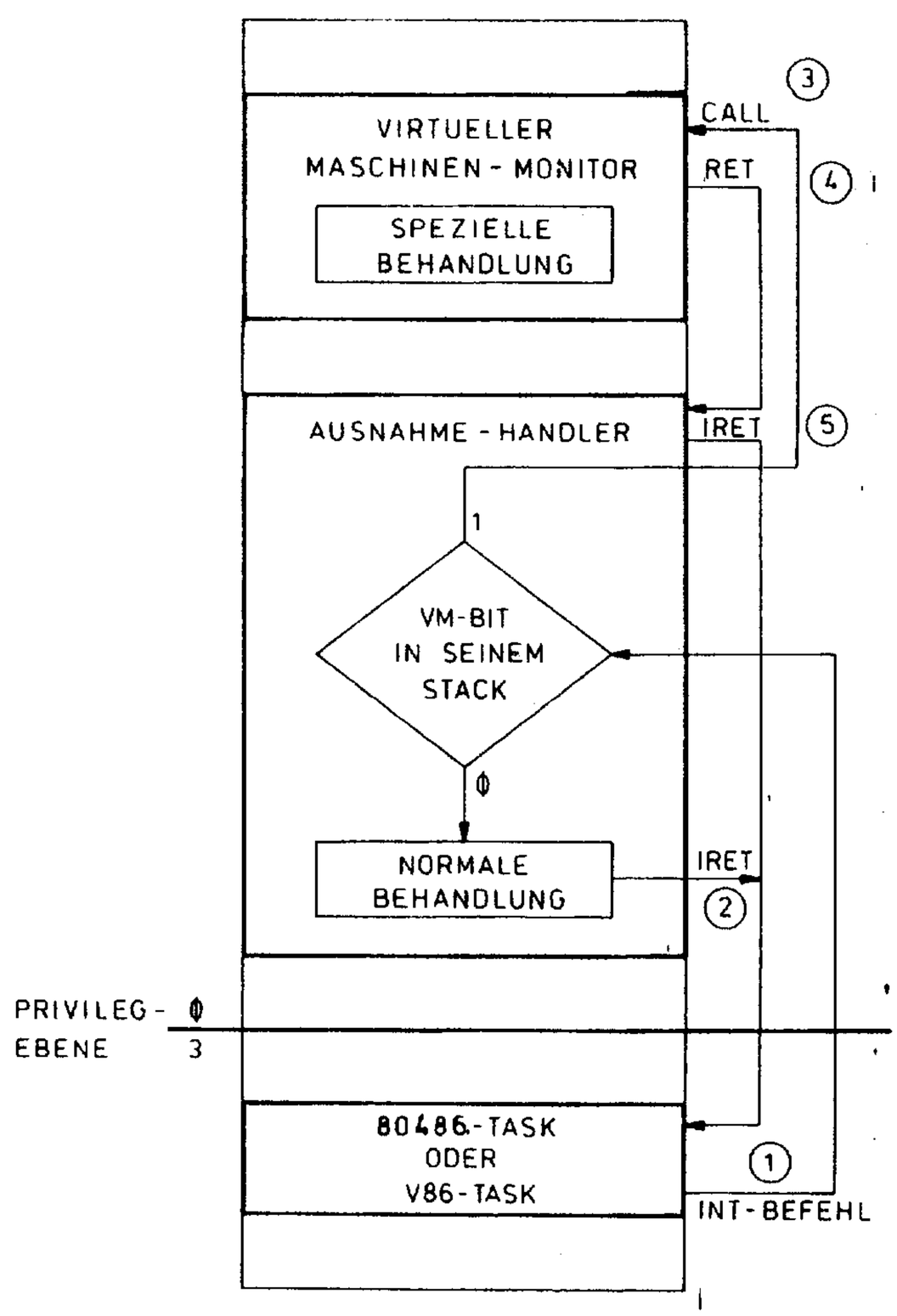
Bevor die Prozedur mit IRET verlassen wird, werden im Epilog die Selektoren mit POP-Befehlen wieder aus dem Stack entfernt. Wenn bei dieser Aktion Null-Selektoren in die CPU zurückgeschrieben werden, führt dies in der Handler-Prozedur zu keiner Ausnahme.

HINWEIS:
Die angegebene Methode kann auch benützt werden, wenn in einer ***80486-Task*** eine Handler-Prozedur über einen Interrupt oder eine Ausnahme-Bedingung aufgerufen worden ist.

10.4 Der virtuelle Maschinen-Monitor

Da Programme meistens für verschiedene Maschinen und verschiedene Betriebssysteme geschrieben sind, müssen Ausnahme-Bedingungen, die eine V86-Task verursacht, vom 80486-Betriebssystem in besonderer Weise behandelt werden. Wenn z. B. in einer 80486-„Protected Mode"-Task der Befehl INT 21H ausgeführt wird, kommt es wahrscheinlich zum Abbruch dieser Task. Der gleiche Befehl in einer V86-Task repräsentiert jedoch einen legitimen Aufruf des 8086-Betriebssystems. Es ist daher angebracht, den Code, der speziell auf eine V86-Ausnahme reagiert, in einer oder mehreren Prozeduren unterzubringen. Dieses Paket von Prozeduren wird als ***virtueller Maschinen-Monitor*** (VMM) bezeichnet. Er ***simuliert*** alle 8086-Befehle, die der 80486 im V86-Mode ***nicht*** ausführen kann.

Dabei ***muß*** sich der Code des virtuellen Maschinen-Monitors in einem USE32-Code-Segment in der Privileg-Ebene 0 aufhalten. Wann der virtuelle Maschinen-Monitor von einem Ausnahme-Handler aufgerufen wird, illustriert das folgende Bild:



Es ist erkennbar, daß in der Privileg-Ebene 3 entweder eine 80486-Task oder eine V86-Task läuft. Entdeckt der Prozessor eine Ausnahme-Bedingung wie z. B. einen INT-Befehl, wird über ein Trap- oder Interrupt-Gate der Ausnahme-Handler in der Privileg-Ebene 0 aufgerufen ①. Der Handler wird zunächst das VM-Bit im EFLAG-Feld seines Stack überprüfen. Bei VM = 0 ist die Ausnahme in einer 80486-Task vorgekommen und kann daher

normal behandelt und anschließend mit IRET verlassen werden ②. Bei VM = 1 ist die Ausnahme in einer V86-Task vorgekommen. In diesem Fall wird der Handler zur Simulation des Ausnahme-Befehls den virtuellen Maschinen-Monitor mit CALL aufrufen ③.

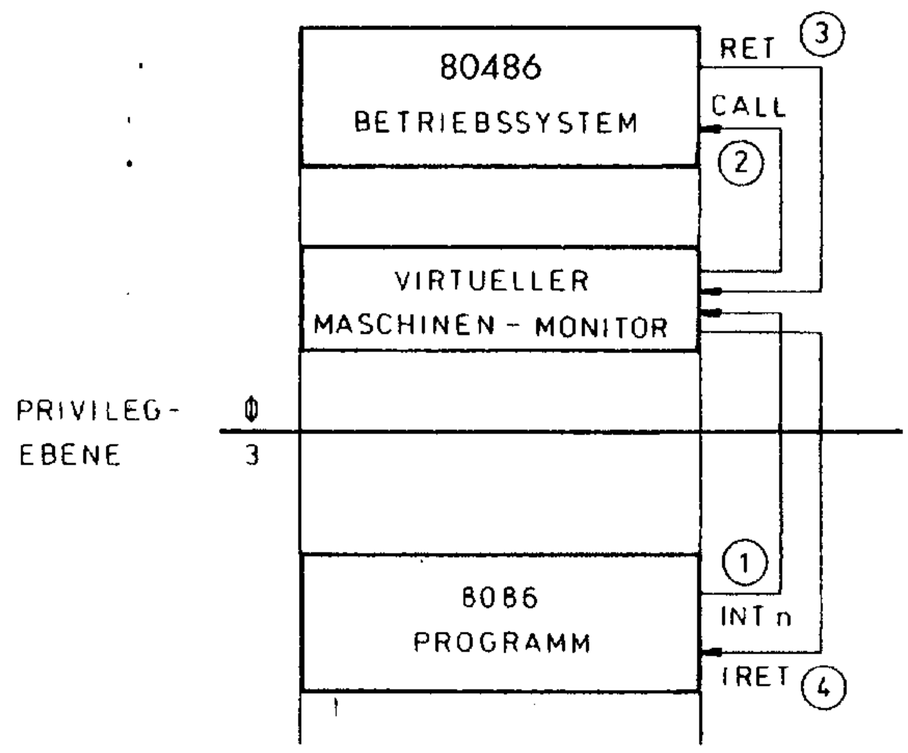
Am Ende des Monitors erfolgt mit RET die Rückkehr zum Ausnahme-Handler ④, der anschließend mit IRET die Fortsetzung der V86-Task veranlaßt ⑤.

10.5 System-Aufrufe

Viele 8086-Betriebssysteme benützen den Befehl ***INT n*** für einen System-Aufruf (System Call). Ob in einer V86-Task die Ausführung von INT n möglich ist, hängt vom IOPL-Wert der Task ab. Ist ***IOPL < 3,*** verursacht der Befehl INT n eine allgemeine Schutzverletzung. Die durch diese Ausnahme aktivierte Handler-Prozedur ruft den virtuellen Maschinen-Monitor auf, der nun den 8086-System-Aufruf simulieren muß. Eine solche Simulation ist aus folgenden Gründen vorteilhaft:

- Die Funktionen des 8086-Betriebssystems können zwischen mehreren V86-Tasks bequem koordiniert werden.
- Die Funktionen des 8086-Betriebssystem (z. B. MS-DOS) können über CALL-Befehle zum 80486-Betriebssystem (z. B. UNIX) bequem emuliert werden.

Welche Aktionen bei der Simulation von 8086-System-Aufrufen ablaufen, illustriert das folgende Bild:



HINWEIS:
Um die Graphik zu vereinfachen, wurde der Ausnahme-Handler weggelassen.

Nachdem mit INT n über den Ausnahme-Handler der virtuelle Maschinen-Monitor aufgerufen worden ist ①, übersetzt der Monitor den System-Aufruf zunächst in ein 80486-Betriebssystem-Äquivalent. Anschließend ruft er mit einem normalen CALL-Befehl das 80486-Betriebssystem auf ②. Hat das 80486-Betriebssystem seine Aktion beendet, erfolgt mit RET die Rückkehr zum virtuellen Maschinen-Monitor ③. Dieser muß nun die Ergebnisse in ein Format umwandeln, wie es von der V86-Task erwartet wird. Da bei einer allgemeinen Schutzverletzung der fehlerverursachende Befehl (INT n) wiederholt werden

kann (Typ Fault), muß im Ebene-0-Stack der EIP-Wert des unterbrochenen 8086-Programms inkrementiert werden. Anschließend kann nach der Ausführung von IRET die V86-Task fortgesetzt werden ④.

Wenn der IOPL-Wert einer V86-Task 3 ist, verursacht INT n ***keine*** allgemeine Schutzverletzung. In diesem Fall wird ein individueller Gate-Deskriptor n in der IDT den Ausnahme-Handler aktivieren. Dieser Handler kann nun ebenfalls den virtuellen Maschinen-Monitor aufrufen, um den System-Aufruf zu simulieren.

10.6 Schutz innerhalb einer V86-Task

Während der Ausführung von 8086-Programmen bezieht sich der 80486 auf keine Deskriptoren. Er kann daher die von den Deskriptoren gelieferten Schutzmechanismen ***nicht*** beanspruchen.

Eine System-Software, die in einer V86-Task läuft, kann mit folgenden Methoden vor 8086-Programmen geschützt werden:

1) Reserviere das erste Megabyte plus 64 KBytes des linearen Adreßraums der V86-Task für 8086-Programme (0 ... 10FFEFH).

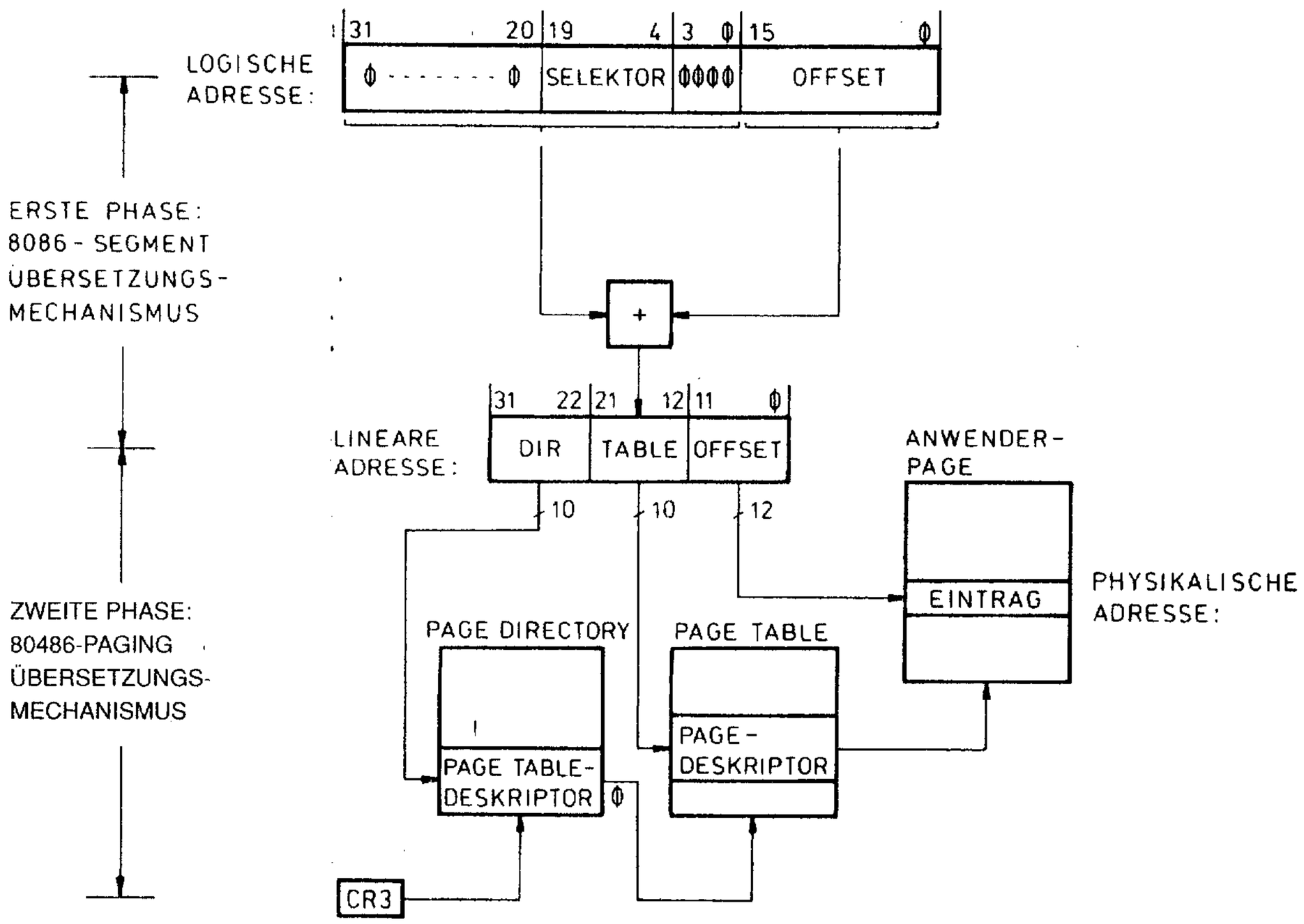
2) Bewahre den virtuellen Maschinen-Monitor und andere System-Software in Supervisor-Pages auf (U/S = 0). Da der Prozessor im V86-Mode in der Privileg-Ebene 3 läuft, halten sich 8086-Programme nur in User-Pages auf (U/S = 1). Daher können 8086-Programme nicht auf den Maschinen-Monitor zugreifen.

10.7 Paging für 8086-Tasks

Sind in einem 80486-System V86-Tasks präsent, kann der Paging-Mechanismus in der gewohnten Weise benützt werden. Das Paging ist immer dann erforderlich, wenn die physikalischen Adreßräume mehrerer V86-Tasks voneinander getrennt werden müssen.

Alle V86-Tasks laufen im ersten Megabyte des linearen Adreßraums und können auf verschiedene physikalische Adreßräume abgebildet werden.

Wie der Paging-Übersetzung-Mechanismus im V86-Mode arbeitet, illustriert das folgende Bild:



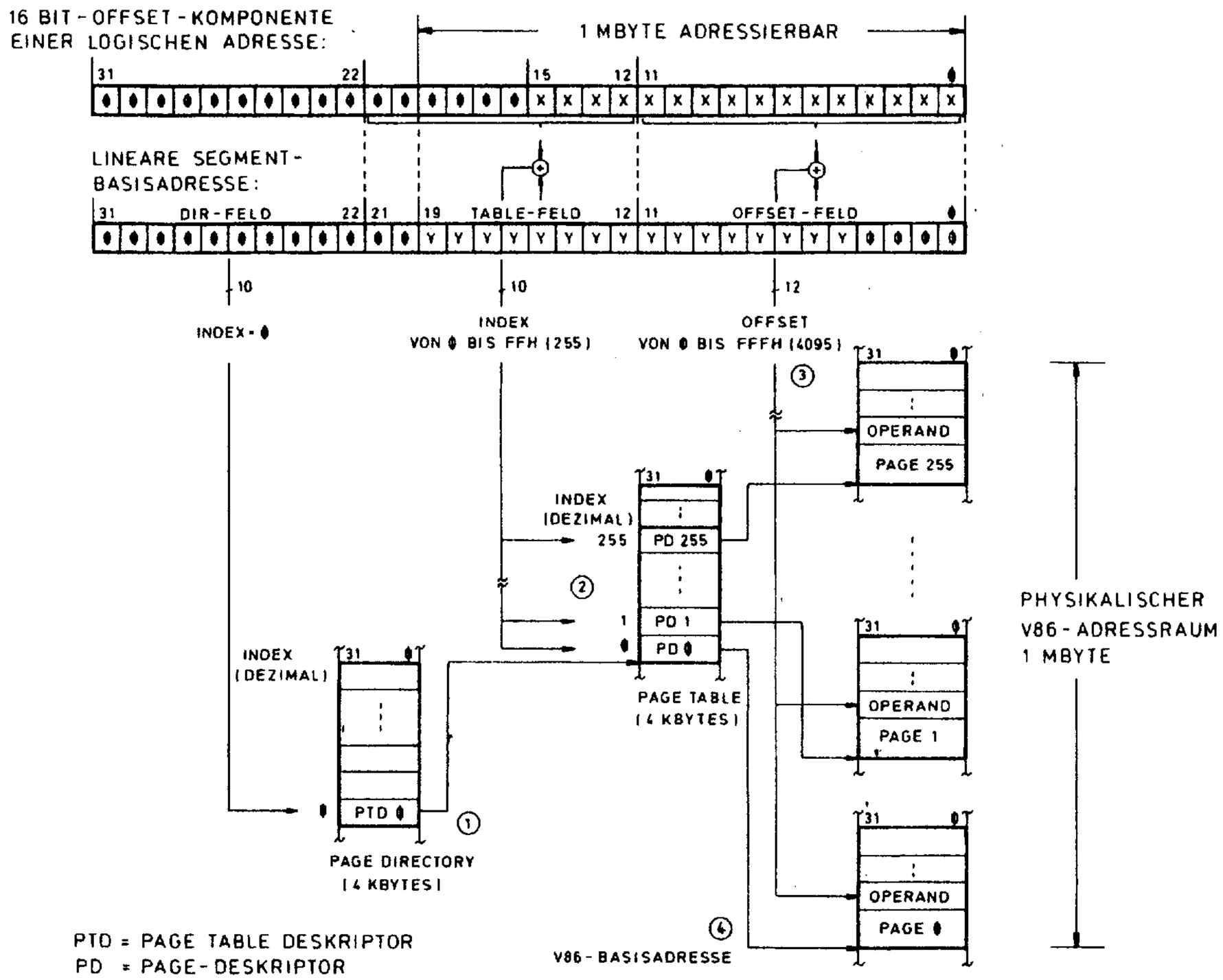
In der ***ersten*** Phase konvertiert der Segment-Übersetzungs-Mechanismus die logische Adresse eines Segment-Eintrags in eine lineare Adresse. Bei dieser Aktion benützt der Prozessor das 8086-Adressierungs-Verfahren. Er

- schiebt das 16-Bit-Segment-Register um 4-Bit-Positionen nach links,
- addiert zur resultierenden 20-Bit linearen Segment-Basis-Adresse den 16-Bit-Offset und
- besetzt die obersten 12 Bits (Bit 20 ... 31) mit 0-Bits.

Das Ergebnis ist die 32-Bit lineare Adresse eines Segment-Eintrags, die in der üblichen Weise in die Felder DIR, TABLE und OFFSET unterteilt ist.

In der ***zweiten*** Phase bildet der 80486-Paging-Übersetzungs-Mechanismus aus der linearen Adresse des Segment-Eintrags die physikalische Adresse.

Wie der 1 MByte physikalische Adreßraum einer V86-Task in Pages unterteilt wird, illustriert das folgende Bild:



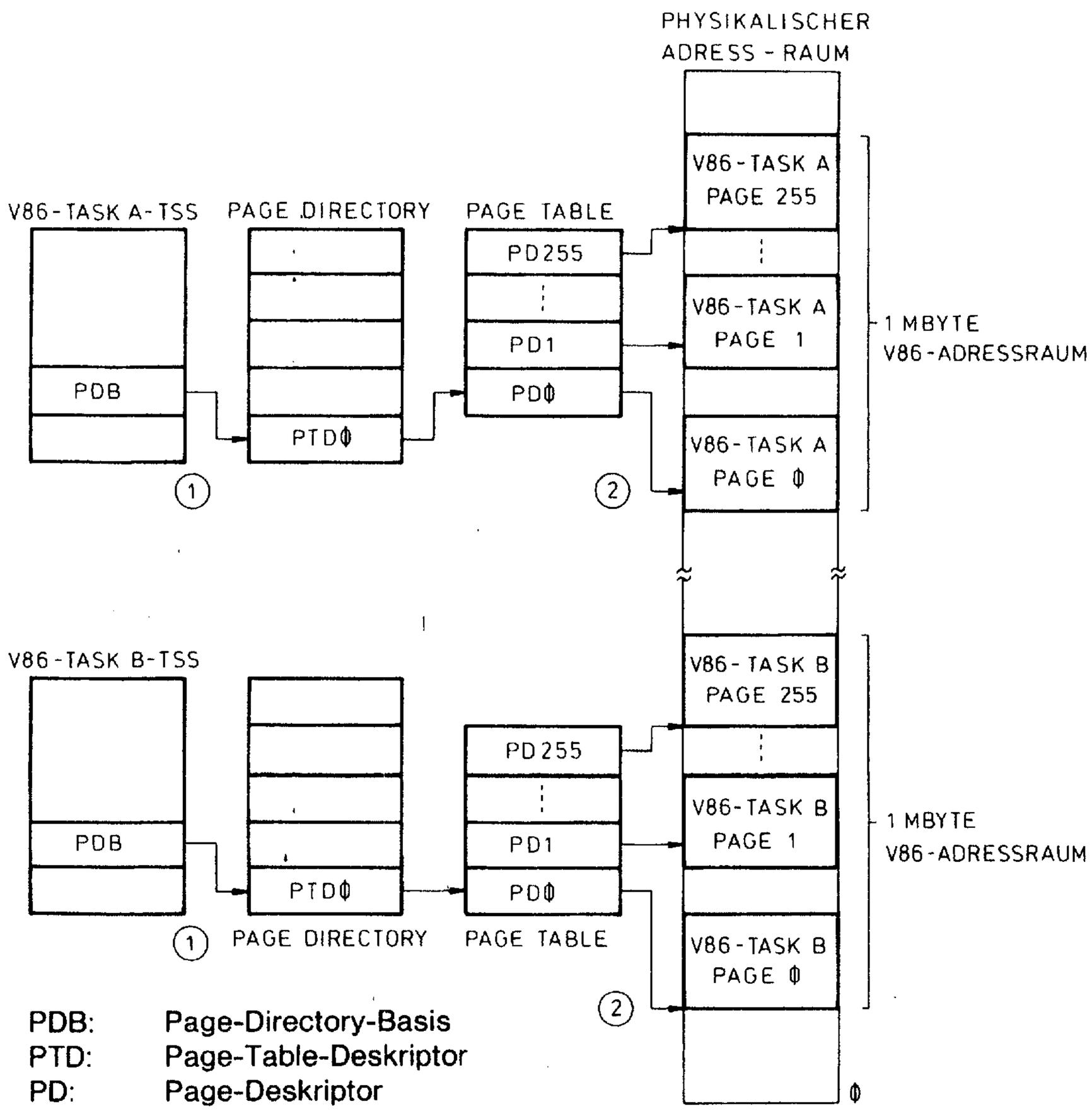
Der Prozessor addiert zur 20-Bit linearen Basis-Adresse des Segments (Segment-Wert x 16) den 16-Bit-Offset des Operanden. Dabei ergibt sich im DIR-Feld der linearen Operanden-Adresse immer der Index-Wert 0. Dies bedeutet, daß der Page-Table-Deskriptor (PTDO) dem ***ersten*** Eintrag in der Page-Directory entspricht ①.

Im TABLE-Feld ergeben sich Werte, die im Bereich 0000000000B ... 0011111111B = 00H ... FFH liegen. Damit können 256 verschiedene Page-Deskriptoren (PDO ... PD255) indiziert werden ②. Diese Deskriptoren liefern die Basis-Adresse von 256 individuellen Pages mit jeweils 4096 verschiedenen Eintragungen. Die Eintragungen in der ausgewählten Page werden über das OFFSET-Feld der linearen Operanden-Adresse mit Werten im Bereich von 000H ... FFFH erreicht ③.

Wenn in der V86-Task 16-Bit-Segment-Werte im Bereich von 0000H ... F000H benützt werden, liegen die 20-Bit linearen Segment-Basis-Adressen zwischen 00000H ... F0000H. Da der 16-Bit-Offset einer logischen Adresse den maximalen Wert FFFFH haben kann, liegt der physikalische V86-Adreßraum im Bereich von 00000H ... FFFFFH (1 MByte). Diesen Adreßraum kann die Paging-Einheit in exakt 256 verschiedene Pages unterteilen. Die Basis-Adresse des physikalischen V86-Adreßraums bestimmt der erste Page-Deskriptor (PDO) in der Page-Table ④. Da dieser Deskriptor variable Adressen liefern kann, ist es möglich, den V86-Adreßraum irgendwo im 4-Gigabyte physikalischen Adreßraum des 80486 zu plazieren.

Dieses Leistungs-Merkmal erlaubt es, die physikalischen Adreßräume mehrerer V86-Tasks voneinander zu trennen.

Wie ein V86-Multitasking-System arrangiert ist, illustriert das folgende Bild:



Jede V86-Task ist durch ein 80486-Task-Status-Segment repräsentiert. Weil die individuellen PDB-Felder unterschiedliche Page-Directory-Basis-Adressen enthalten ①, hat jede V86-Task ihre eigene Page-Directory und Page-Table. Dabei bestimmt der erste Page-Deskriptor in der jeweiligen Page-Table die physikalische Position des individuellen V86-Adreßraums ②.

Bei Mitgliedern der 8086-Familie ist es möglich, einen Adreßraum größer als 1 MByte zu spezifizieren. Dies kommt vor, wenn der Segment-Wert FFFFH ist. Mit einem Offset-Wert von FFFFH liegt dann die effektive Adresse bei 10FFEFH (1 MByte + 65519). Da der 8086 nur 20-Bit lange Adressen kennt, schneidet er das oberste Adreß-Bit ab und liefert die Adresse FFEFH.

Dieser sogenannte 1 MByte-„Wraparound" kann der 80486 mit Alias-Pages simulieren. Die Page-Table muß daher neben den 256 Page-Deskriptoren, die 1 MByte definieren, weitere 16 Page-Deskriptoren enthalten. Diese zusätzlichen Deskriptoren definieren die 65519 Bytes oberhalb des 1-MByte-Adreßraums einer V86-Task.

wenn er das VM-Bit in seinem Stack prüft. Ist es gesetzt (VM = 1), kann er den virtuellen Maschinen-Monitor aufrufen, der dann die V86-Task abbricht.

10.9 Eingabe/Ausgabe

Viele 8086-Programme, die für Single-Task-Systeme entwickelt worden sind, greifen direkt auf I/O-Ports zu. Wenn aber die gleichen 8086-Programme in einer Mulitasking-Umgebung laufen, können direkte Zugriffe auf I/O-Ports zu Ausnahme-Situationen führen.

Die Methode, I/O-Zugriffe zu kontrollieren, hängt davon ab, ob die Ports im isolierten 64-KByte-I/O-Raum (I/O Mapped I/O) oder im 4-GByte-Speicherraum (Memory Mapped I/O) plaziert sind.

- Wenn sich im V86-Mode die Ports im isolierten I/O-Raum aufhalten, konsultiert der Schutz-Mechanismus des 80486 bei der Ausführung der I/O-Befehle IN, INS, OUT und OUTS den IOPL-Wert ***nicht.***
  Der Schutz-Mechanismus verhält sich an dieser Stelle anders als im „Protected Mode". Dort wird bei I/O-Zugriffen zuerst die Bedingung Task CPL ≤ IOPL überprüft.
  Die Zugriffe auf Ports im isolierten I/O-Raum werden im V86-Mode ***ausschließlich*** über die „I/O Permission Bit Map" gesteuert. Sie bestimmt, welche I/O-Adressen erlaubt sind und welche nicht. Da jede V86-Task ihre eigene „I/O Permission Bit Map" haben kann, können I/O-Adressen in der einen Task gültig und der anderen Task ungültig sein.
- Wenn im V86-Mode die Ports im 4-GByte-Speicherraum aufbewahrt sind, können die I/O-Operationen nur von den Paging-Schutzmechanismen kontrolliert werden. Dabei ist es möglich, daß V86-Referenzen auf die Adressen dieser Ports physikalisch verlagert werden.

BEISPIEL:

In einem 80486-„Protected Mode"-System existiert ein Video-Refresh-Puffer ab einer physikalischen Adresse, die größer als 1 MByte ist. Ein solcher Puffer in einem V86-System muß notwendigerweise an einer physikalischen Adresse unterhalb von 1 MByte liegen.

Ein 80486-Betriebssystem kann nun die Page-Table einer V86-Task so initialisieren, daß Referenzen auf Adressen des V86-Puffers automatisch in Adressen des 80486-Puffers umgewandelt werden.

Wenn Zugriffe auf Ports im Speicherraum verhindert werden sollen, können die Pages, in denen sich die Ports aufhalten, als ***nicht-präsent*** und ***gesperrt für I/O*** markiert werden. Für diese Maßnahme stehen das Present-Bit und das AVAIL-Feld in den zugehörigen Page-Deskriptoren zur Verfügung.

Wenn die ersten 16 und die letzten 16 Tage-Deskriptoren ***identisch*** sind (Alias-Deskriptoren), werden die ersten und die letzten 64 KBytes des linearen Adreßraums auf die gleichen physikalischen Pages abgebildet.

Diese Technik erfordert aber zusätzliche Betriebssystem-Funktionen. Wenn z. B. eine Alias-Page ausgelagert worden ist, muß sie von beiden Page-Deskriptoren als ***nicht-präsent*** markiert werden.

10.8 Virtualisierung des Interrupt-Enable-Flags IF

Viele 8086-Programme, die für Single-Task-Systeme entwickelt worden sind, setzen und löschen IF, um Interrupts zu steuern. So wird z. B. IF gelöscht, um kritische Programm-Regionen vor Unterbrechungen zu schützen.

Wenn aber die gleichen 8086-Programme in einer Multitasking-Umgebung laufen, kann der Zugriff auf IF zu einer Ausnahme-Situation führen.

Da eine V86-Task in der Privileg-Ebene 3 läuft, bestimmt ***ihr*** IOPL-Wert, ob IF gesetzt oder gelöscht werden kann. Ist der IOPL-Wert der V86-Task ***kleiner*** als 3 (IOPL < 3), verursachen Befehle, die IF beeinflussen, eine allgemeine Schutzverletzung.

Zur Gruppe dieser sogenannten ***IOPL-sensitiven*** und mit IF verbundenen Befehle gehören PUSHF, POPF, INT n, IRET, STI und CLI. Wenn diese Befehle eine allgemeine Schutzverletzung verursachen, muß sie der virtuelle Maschinen-Monitor simulieren.

Ist der IOPL-Wert der V86-Task ***gleich*** 3 (IOPL = 3), führt der Prozessor alle mit IF verbundenen Befehle aus. In diesem Fall kann aber ein potentielles Risiko für den Rest des Systems bestehen. Daher sollte der IOPL-Wert einer V86-Task nur dann 3 sein, wenn die Leistungsanforderungen durch die Simulation von IF nicht erfüllt werden können.

Um IF zu simulieren, kann der virtuelle Maschinen-Monitor eine Variable im verfügbaren Status-Feld des Task-Status-Segments unterstützen. Diese Variable wird er dann mit einem entsprechenden Wert versorgen, wenn er Befehle simuliert, die IF im Stack speichern (z. B. PUSHF). Weiterhin kann der Monitor diese Variable aktualisieren, wenn er Befehle simuliert, die IF setzen (z. B. STI).

Eine V86-Task mit IOPL = 3 ist in der Lage, jeden der 256 verschiedenen INT-Befehle auszuführen. Um aber eine V86-Task daran zu hindern, einen Interrupt- oder Ausnahme-Handler über INT n aufzurufen, kann in der IDT das DPL-Feld im zugehörigen Gate-Deskriptor auf 0 gesetzt werden.

Der Versuch, einen Handler über einen Gate-Deskriptor aufzurufen, dessen DPL-Wert 0 ist, führt zur allgemeinen Schutzverletzung. Der virtuelle Maschinen-Monitor kann dann die V86-Task abbrechen.

Damit aber eine „Protected Mode"-Task in der Privileg-Ebene 3 ebenfalls einen Handler aufrufen kann, muß in der IDT das DPL-Feld im zugehörigen Gate-Deskriptor 3 sein. Der Handler kann bei einer solchen Situation eine fehlerhafte V86-Task nur identifizieren,

Immer dann, wenn ein „Memory Mapped"-Portzugriff auf eine nicht-präsente Page trifft, wird der „Page Fault Handler" aufgerufen. Dieser hat nun die Möglichkeit, das AVAIL-Feld zu prüfen. Stellt er fest, daß die Page für I/O-Zugriffe gesperrt ist, kann er für weitere Maßnahmen den virtuellen Maschinen-Monitor aufrufen.