RUHR-UNIVERSITÄT BOCHUM

Arbeitsberichte

des

Rechenzentrums

Direktor: o.Prof. Dr. Hartmut Ehlich

Nr. 7710 ISSN 0341-0358

BO&PAGING

Der virtuelle Kernspeicher für den TR440

von

Rainard Buchmann

Bochum im August 1977

ISSN 0341-0358

Rechenzentrum der Ruhr-Universität Bochum
Universitätsstraße 150, Gebäude NA
Postfach 102148

4630 Bochum 1

INHALTSVERZEICHNIS

0. Einleitung

Virtueller Kernspeicher bedeutet: alles, was zum Programm gehört (also alle Befehle, Konstanten und Variablen), wird automatisch dann in den Kernspeicher gebracht, wenn man es benötigt, und wird, falls erforderlich, wieder aus dem Kernspeicher entfernt, wenn es nicht mehr gebraucht wird.

Auf diese Weise wird für andere Programmteile Platz gemacht.

Diejenigen Teile des Programms, die gerade nicht im Kernspeicher liegen, müssen solange an einem anderen Ort "aufbewahrt" werden, um sie, wenn sie benötigt werden, in den Kernspeicher laden zu können. Solche Programmteile liegen auf der Platte. Bei allen Lade- und Verdrängungsvorgängen werden Einheiten von einer Seite (= 1024 Worte) transportiert, um den Kernspeicher optimal auszunutzen und überflüssige Transporte zu vermeiden.

Ein solches Vorgehen - Demand Paging genannt - ist aus mehreren Gründen sinnvoll und wünschenswert:

a) Man möchte mit sehr großen Feldern arbeiten (z. B. großen Matrizen), die nicht mehr in den physikalisch zur Verfügung stehenden Kernspeicher passen.

b) In Systemen mit Multiprogramming-Betrieb ist es unumgänglich, dem einzelnen Benutzer Beschränkungen hinsichtlich seiner Betriebsmittelforderungen - insbesondere beim Kernspeicherbedarf - aufzuerlegen. Dadurch kann es unmöglich werden, ohne Demand Paging große Programme zu benutzen.

c) Es hat sich gezeigt, daß ein Auftrag in einem Timesharing-Rechner um so schneller abgewickelt werden kann, je weniger Kernspeicher er benötigt, da bei einer großen Zahl konkurrierender Aufträge der Kernspeicher einen wesentlichen Betriebsmittel-engpaß darstellt. (Die "Liegezeit" wird kürzer).

d) Ein wesentlicher Gesichtspunkt, der oft vergessen wird, ist der Gesamtdurchsatz einer Rechenanlage.

   Darunter versteht man die Rechenleistung, die pro Zeiteinheit an die Benutzer abgegeben werden kann, also etwa die Anzahl der Aufträge, die pro Tag "geschafft" werden.

   Dieser Durchsatz ist im Timesharing-Betrieb stark abhängig davon, wieviele Aufträge gleichzeitig im Kernspeicher liegen können, die sich um den oder die Rechnerkern(e) bewerben. Je größer die Anzahl solcher Aufträge ist, desto höher ist die Wahrscheinlichkeit, daß welche dabei sind, die gerade nicht auf die Beendigung eines EA-Transportes warten und somit einen Rechnerkern nutzen können.

   Die Anzahl der Aufträge, die gleichzeitig Platz im Kernspeicher haben, läßt sich nun mit Hilfe des Demand Pagings durch Verringerung ihres Kernspeicherbedarfes vergrößern.

1. Randbedingungen des TR440-Programmiersystems

Voraussetzung für ein vernünftiges Demand Paging ist es, daß der Benutzer gar nichts von seiner Existenz merkt, außer seinem geringeren Kernspeicherbedarf. Er muß genauso programmieren können, als stünde ihm der gesamte virtuelle Kernspeicher (ca. 1920 K Worte ≈ 11 Megabytes am TR440) real zur Verfügung.

1.1. Testhilfen

Insbesondere müssen natürlich sämtliche Testhilfen, die das System bietet, voll benutzbar sein, z. B.: dynamische Kontrollen (DYNKON), Rückverfolgung der Unterprogramm-Verschachtelung, quellsprachenbezogene Dumps, Kontrollereignisse, Überwachung des Programmlaufes (TRACE) etc. .

Aus diesen Gründen wurde das Demand Paging auf Unterprogrammebene realisiert.

Dadurch ist es außerdem möglich, allein durch ein BIBANMELDE-Kommando vor dem MONTIEREN des Programmes die Leistungen des Demand Pagings nutzen zu können.

1.2. Dynamische Freispeicherverwaltungen

Um sinnvoll zu sein, muß eine Demand Paging-Realisierung natürlich Programme beliebiger Sprachen bearbeiten können.

Dazu ist es erforderlich, die dynamischen Speicher, die es z. B. in ALGOL 60, BCPL, ALGOL 68 u.a. gibt, in das Demand Paging einzubeziehen.

Dazu wurde in BO&PAGING eine globale Schnittstelle für beliebige Freispeicherverwaltungsprozeduren geschaffen (siehe 6.4.4.). Außerdem wurden die entsprechenden Prozeduren des Systems durch geringfügige Modifikationen an diese Schnittstelle angepaßt. BO&PAGING "verträgt" jedoch auch nicht angepaßte Freispeicherprozeduren - die entsprechenden Freispeicher sind dann resident und werden nicht weggeladen.

1.3. Overlay-Programme

Damit das Demand Paging auch dort einsetzbar ist, wo es am notwendigsten ist, nämlich bei extrem umfangreichen Programmen (wie z. B. SPSS), wurde besonderer Wert darauf gelegt, Programme, die Overlay erfordern, ebenfalls bearbeiten zu können (siehe auch (4.)).

1.4. Segmentierte Programme

Um der Forderung gerecht zu werden, daß ein Benutzer sein Programm für das Demand Paging nicht ändern muß, wurden die Standard-Schnittstellen für Programm-Segmentierung (LOAD/UNLOAD) unverändert übernommen und darüber hinaus zur benutzergesteuerten Algorithmus-Unterstützung verwendet (siehe (4.)).

2. Funktionsbeschreibung

2.1. Übersicht

Das Prinzip des virtuellen Kernspeichers beruht darauf, daß nur der Teil eines Programmes im Kernspeicher liegt, der aktuell benötigt wird. Der Rest lagert auf der Platte.

Bei Programmbeginn wird zunächst mit Hilfe einer Liste eine eindeutige Zuordnung zwischen dem Kernspeicher und den zugehörigen Plattenbereichen (Gebiete genannt) hergestellt und das gesamte Programm vom Kernspeicher auf die Platte verlagert. Wird im Programmlauf ein Wort angesprochen (d. h. benötigt), das gerade nicht im Kernspeicher liegt, so erfährt das Unterprogramm BO&PAGING dies durch einen Hardware- oder Software-Interrupt (Speicherschutzalarm bzw. Ereignisalarm (siehe (6.2.1.)).

Dann wird auf der Platte eine passende 1 K Worte große Umgebung dieses Wortes gesucht und als ein Stück in den Kernspeicher transportiert.

Vorher wird dafür gesorgt, daß genügend Platz im Kernspeicher frei ist, indem evtl. eine andere Seite verdrängt, d. h. vom Kernspeicher auf die Platte transportiert wird.

Die Entscheidung, welche Seite verdrängt wird, wird anhand des in (3.) beschriebenen Verdrängungsalgorithmus' getroffen.

2.2. Seitengebiete

Ein montiertes Programm besteht am TR440 aus mehreren einzelnen "Stücken", den sogenannten Gebieten.

Diese Gebiete haben einen (operatorlaufspezifischen) Gebietsnamen, über den sie identifiziert werden. (Die Gebiete werden vom Montierer einschließlich ihrer Namen protokolliert, wenn beim MONTIERE-Kommando die PROTOKOLL-Spezifikation besetzt ist).

Gebiete sind für ein Programm die einzigen ansprechbaren Speichereinheiten (außer Dateien) - man kann also nicht direkt mit Seiten des Kernspeichers bzw. der Platte arbeiten.

Da das Demand Paging aber seitenweise Laden und Verdrängen soll, muß es sich sogenannte Seitengebiete erzeugen, also Gebiete, die nur 1 K groß sind und jeweils die Kopie eines entsprechenden Ausschnittes eines größeren Gebietes enthalten.

Lediglich wenn ein vom Montierer erzeugtes Gebiet nur 1 K groß ist, wird dieses Gebiet direkt als Original-Seitengebiet genommen und nicht kopiert.

2.3. Laden einer Seite

Beim Laden einer Seite sind 3 Fälle zu unterscheiden:

a) Die Seite liegt in einem Gebiet, das schon vom Montierer in der Länge 1 K angelegt war:

Dieses Gebiet wird als Original-Seitengebiet in den Kernspeicher verlagert.

b) Für die Seite existiert bereits ein Kopie-Seitengebiet:

Das Seitengebiet wird in den Kernspeicher verlagert.

c) Es existiert noch kein Seitengebiet:

Ein Seitengebiet wird im Kernspeicher kreiert und aus dem Originalgebiet (auf der Platte) gefüllt.

2.4. Verdrängen einer Seite

Für das Verdrängen einer Seite aus dem Kernspeicher gibt es 2 verschiedene Fälle:

a) Es handelt sich um ein Original- oder Kopie-Seitengebiet, das nicht gelöscht werden muß:

Das Seitengebiet wird auf den Hintergrund verlagert und damit aus dem Kernspeicher entfernt.

b) Es handelt sich um ein Kopie-Seitengebiet, das gelöscht werden muß (die Anzahl von Gebieten ist auf maximal 107 pro Operatorlauf beschränkt):

Wenn die Möglichkeit besteht, daß das Seitengebiet gegenüber dem Original verändert wurde, da es sich um Variable handelt, wird der Inhalt des Kopie-Seitengebietes zunächst in das Originalgebiet zurückgeschrieben.

Dann wird das Seitengebiet gelöscht und belegt somit den Kernspeicher nicht mehr.

3. Der Verdrängungsalgorithmus

Soll ein Seitengebiet in den Kernspeicher geladen werden, das nicht mehr hineinpaßt, so steht ein Demand Paging-Programm vor der schwierigen Entscheidung, welche Seite aus dem Kernspeicher entfernt werden soll, um Platz zu schaffen.

3.1. Wünschenswerte Algorithmen

Theoretische Untersuchungen haben gezeigt, daß der optimale Algorithmus darin besteht, diejenige Seite auszulagern, die von diesem Zeitpunkt an am längsten nicht mehr benötigt wird.

Da bisher jedoch noch kein Algorithmus bekannt ist, der ein Programm mit hellseherischen Fähigkeiten ausstattet, muß die Entscheidung mit Hilfe von Informationen getroffen werden, die aus dem bisherigen Programmablauf bekannt sind.

Dazu bieten sich zwei Verfahren an:

1. Die Seite wird ausgelagert, auf die bisher am wenigsten zugegriffen wurde.
2. Die Seite wird ausgelagert, auf die am längsten nicht mehr zugegriffen wurde.

Leider sind auch diese beiden Verfahren am TR440 nicht realisierbar, da keine Hardware-Einrichtung existiert, die über Speicherzugriffe Buch führt.

3.2. Der realisierte Algorithmus

Einen Pagefault nennen wir den Zugriff auf eine Seite, die gerade nicht im Kernspeicher liegt. Die Seite, deren Fehlen im Kernspeicher den aktuellen Pagefault ausgelöst hat, heiße Pagefault-Seite.

Welche Ursachen ein Pagefault haben kann, ist unter (6.2.1) beschrieben.

Eine Seite, die im unmittelbar folgenden Programmlauf im Kernspeicher liegen muß, nennen wir benötigte Seite.

Natürlich ist eine Pagefault-Seite immer eine benötigte Seite - sonst hätte sie keinen Pagefault ausgelöst.

Die übrigen Kriterien, anhand derer das Demand Paging-Programm BO&PAGING eine Seite als benötigt erkennt, sind in (6.2.2.) beschrieben. Die Informationen über den bisherigen Programmablauf werden von BO&PAGING immer dann aktualisiert, wenn es aktiviert wird.

Der Verdrängungsalgorithmus arbeitet mit einem Kurzzeit- und einem Langzeit-"Gedächtnis".

Das "Kurzzeit-Gedächtnis" ist als Ringpuffer mit einstellbarer Länge (siehe 4.5.7.)) konstruiert, in dem zyklisch die jeweils letzten Pagefault-Seiten vermerkt werden; es hat Vorrang gegenüber dem "Langzeit-Gedächtnis". Das bedeutet, daß zunächst versucht wird, eine Seite zu verdrängen, die nicht in diesem Ringpuffer für die letzten Pagefault-Seiten steht.

Das "Langzeit-Gedächtnis" besteht aus einem Zähler für jede mögliche Seite des Programms, den wir Verdrängungszähler nennen wollen.

Jedesmal, wenn eine Seite als benötigt erkennbar ist, wird ihr Verdrängungszähler um 1 erhöht.

Es wird dann bei Bedarf diejenige Seite verdrängt, die den kleinsten Verdrängungszähler hat.

Um zu verhindern, daß eine Seite, die einmal einen großen Verdrängungszähler bekommen hat und dann nie mehr benötigt wird, beliebig lange unnötigerweise im Kernspeicher liegen bleibt, löscht BO&PAGING nach einer einstellbaren Anzahl (siehe (4.5.7.)) von Pagefaults sein "Langzeit-Gedächtnis".

Da der Benutzer des Demand Paging i.a. die meisten Informationen über den Ablauf seines Programms besitzt, wurde eine Reihe von Möglichkeiten geschaffen, die es ihm erlauben, durch Unterprogrammaufrufe steuernd in den Verdrängungsalgorithmus einzugreifen. Diese Möglichkeiten werden im folgenden beschrieben.

4. Unterprogramme zur Steuerung

Alle Unterprogramme zur Steuerung des Demand Paging können von TAS, BCPL, COBOL, FORTRAN und ALGOL60 aus aufgerufen werden.

4.1. Liste der Unterprogramme

PAGING	Initialisierungsaufruf
LOAD	Zuladen (nur bei Overlay relevant)
	BL.LOAD dto. für BCPL
	S&LOAD und S&LOLA dto. für TAS
UNLOAD	Wegladen
	BL.UNLOAD dto. für BCPL
	S&UNLOAD und S&LOVE dto. für TAS
LDNORM	Normieren aller Verdrängungszähler
	BL.LDNORM dto. für BCPL
LDPEIN	Einschalten des Ladeprotokolls
	BL.LDPEIN dto. für BCPL
LDPAUS	Ausschalten des Ladeprotokolls
	BL.LDPAUS dto. für BCPL
LDSTEU	allgemeine Steuerungen des Demand Paging
	BL.LDSTEU dto. für BCPL
LDPROT	Ausdrucken der Anzahl Pagefaults und Zuladungen
	BL.LDPROT dto. für BCPL
S&BEECALARM, GSF	nur aus Kompatibilitätsgründen gegenüber Standard-Segmentierung vorhanden - Aufruf bedeutungslos

4.2. Deklaration der Unterprogramme:

a) TAS:

```
EXTERN S&LOAD(S&LOLA,S&LOVE,...),
                oder
EXTERN BO&PAGING(S&LOLA,S&LOVE,...),
```

b) BCPL:

```
EXTERNAL BO.PAGING:BL.LOAD,BL.UNLOAD,...
```

c) COBOL: nicht nötig

d) FORTRAN: nicht nötig

e) ALGOL60: bei LOAD und UNLOAD nicht nötig, aber erlaubt - sonst:

```
'PROCEDURE' LDSTEU(X); 'CODE';
                :
```

4.3. Prozedur-Parameter:

Alle Unterprogramme können ohne Parameter oder mit einer variablen Anzahl von Parametern aufgerufen werden.

Als Parameter sind nur ganze Zahlen erlaubt. In den verschiedenen Sprachen sind folgende Parametertypen zulässig:

a) TAS: Festkommazahlen

b) BCPL: ganze Zahlen als Werte

c) COBOL: PIC 9(13) COMP.
(mit oder ohne Vorzeichen)

d) FORTRAN: INTERGER*4

e) ALGOL60: 'INTEGER'

4.4. Aufrufkonventionen

a) TAS:

S&LOAD,S&UNLOAD,S&LOLA und S&LOVE werden genauso versorgt wie die entsprechenden Unterprogramme der Standard-Segmentierung (siehe [2]), wobei die dort mögliche Fehleradresse nicht ausgewertet wird: es gibt keinen Fehlerausgang.

Beispiel: BA(2), SFBE(S&LOVE/A),
(Segment mit VNR 2 wird entladen)

Bei allen anderen Unterprogrammen muß in RA die Anfangsadresse eines Versorgungsblockes bereitgestellt werden, der folgenden Aufbau hat:

TK

1	FA (24)	SS (4)	O (12)	P
1				P_1
1				P_2

FA = Fehleradresse (irrelevant)

SS = Sprachschlüssel (= +0)

P = Parameteranzahl

P_i = i-ter Parameter als Festkommazahl

Der Aufruf muß mit SFBE (nicht mit SFB) erfolgen, da man sonst in der falschen Großseite landen kann.

Beispiel: BA(2,10,11), SFBE(UNLOAD/A),
(Die Vorrangnummern 10 und 11 werden entladen.)

b) BCPL:

Die Unterprogramme genügen den Bedingungen der rekursiven Aufruftechnik.

Der erste Parameter muß die Anzahl der noch folgenden Parameter enthalten.

Beispiele: BL.LDNORM(0)

(parameterloser Aufruf von LDNORM)

BL.UNLOAD(2,10,11)

(Die Vorrangnummern 10 und 11 werden entladen)

c) COBOL:

ENTER TAS UNLOAD USING Z10,Z11

(Das Programm muß mit VARIANTE=GR übersetzt werden)

d) FORTRAN:

CALL UNLOAD(10,11)

(Das Programm muß mit VARIANTE=GR übersetzt werden)

e) ALGOL60:

UNLOAD(10,11);

4.5. Einzelbeschreibung der Unterprogramme

4.5.1. PAGING (parameterlos)

Der Aufruf von PAGING dient zur Initialisierung des Demand Paging. Alle anderen Aufrufe werden ignoriert, wenn der Initialisierungsaufruf noch nicht stattgefunden hat. Jeder weitere Aufruf von PAGING ist wirkungslos.

Der Initialisierungsaufruf wird in BCPL-, COBOL-, FORTRAN-, ALGOL60- und ALGOL 68 -Programmen sowie in TAS-Programmen, die S&CC benutzen (z. B. bei SPRACHE(UEBERSETZE)=TASR) automatisch bei Programmbeginn von S&CC ausgeführt (siehe auch (7.1.)), muß also nicht vom Benutzerprogramm aus erfolgen.

4.5.2. LOAD, S&LOAD, S&LOLA, BL.LOAD

Der Aufruf dieser Unterprogramme, deren Parameter Vorrangnummern sind, ist normalerweise wirkungslos und daher überflüssig (aber immer erlaubt).

Lediglich bei extrem großen Programmen, die beim MONTIEREN OVERLAY erfordern, sind LOAD- und UNLOAD-Aufrufe erforderlich, um dem Demand Paging mitzuteilen, welche der Overlay-Vorrangnummern aktuell zuladbar sind.

Die Benutzung von LOAD und UNLOAD ist dabei dieselbe wie beim Zu- und Wegladen mit Hilfe der Standard-Segmentierung. (Im Hinblick auf eventuelle zukünftige Montierer-Verbesserungen wurde die Möglichkeit berücksichtigt, daß mehrere Overlay-Bereiche in Großseite 0 und 1 existieren).

4.5.3. UNLOAD, S&UNLOAD, S&LOVE, BL.UNLOAD

Der Aufruf dieser Unterprogramme ist eigentlich nicht nötig, da das Demand Paging automatisch durch Verdrängung Platz im Kernspeicher schafft, falls nötig.

Bei Programmen mit OVERLAY ist das explizite UNLOAD für Overlay-Vorrangnummern jedoch erforderlich (wie bei der Standard-Segmentierung).

Die Wirkung ist folgende:

Alle Seitengebiete, die zu einer der als Parameter angegebenen Vorrangnummern gehören, werden sofort aus dem Kernspeicher auf die Platte verlagert, und unter der Annahme, daß sie längere Zeit nicht mehr benötigt werden, wird ihr Verdrängungszähler (siehe (3.2.)) auf 0 gesetzt.

Ist eine Vorrangnummer negativ angegeben, so wird angenommen, daß die dazuge-

hörenden Seitengebiete nie mehr oder sehr lange nicht mehr benötigt werden. Die Kopie-Seitengebiete werden dann - evtl. nach Zurückkopieren in die Originalgebiete - gelöscht (siehe auch (2.4.)).

Mit Hilfe des expliziten UNLOAD kann der Benutzer u.U. sein Programm beschleunigen.

4.5.4. LDNORM, BL.LDNORM

Diese parameterlosen Prozeduren bewirken, daß das "Langzeit-Gedächtnis" von BO&PAGING normiert wird, daß also alle Verdrängungszähler auf 0 gesetzt werden.

Ihr Aufruf dient zur Beschleunigung des Verdrängungsalgorithmus' und ist nützlich, wenn ein Programm von einer Phase in eine andere wechselt, in der ganz andere Unterprogramme und Variable benutzt werden.

Dieses Unterprogramm sollte man also aufrufen, wenn ein Einschnitt im Programmablauf eintritt, der es nicht sinnvoll erscheinen läßt, weiterhin den bisherigen Programmablauf im Verdrängungsalgorithmus zu berücksichtigten. Als Beispiel stelle man sich einen Compiler vor, der in der 1. Phase die Quelle liest und lexikalische Analysen macht, in der 2. Phase irgendwelche Listen aufbaut und in der 3. Phase den eigentlichen Code erzeugt. Dort ist es vernünftig, zwischen den einzelnen Phasen das Unterprogramm LDNORM aufzurufen.

4.5.5. LDPEIN, BL.LDPEIN, LDPAUS, BL.LDPAUS

Diese Unterprogramme können parameterlos oder mit einem Parameter aufgerufen werden. Sie dienen zum Ein- und Ausschalten von Testprotokollierungen.

Der parameterlose Aufruf ist identisch zum Aufruf mit einem Parameter, der die Zahl 1 enthält. Er ist der für den normalen Benutzer interessanteste Fall.

Er dient dazu, ein Standard-Ladeprotokoll zu erzeugen, das Informationen liefert, welche Seiten wann zu- und weggeladen werden. Diesen Informationen kann der Benutzer Hinweise zum Optimieren seines Programmes entnehmen (siehe (5)).

Es gibt folgende Ladeprotokollierungen:

a) Aufruf mit Parameter = 1 oder parameterlos:

```
***verdraengt: LO5.3.4 *zugeladen: DO2.7*KSB=i
```

Dabei sind LO5 und DO2 die Namen der Original-Plattengebiete, aus denen die Seitengebiete stammen - also die Gebietsnamen, die der Montierer protokolliert (siehe (2.1)).

3 bzw. 7 sind die relativen Seitennummern innerhalb des Plattengebietes (die Seitennummern werden von 0 an durchnumeriert).

Die Zahl 4 sagt aus, daß das Gebiet zu der Vorrangnummer 4 gehört.

Fehlt im Protokoll die Vorrangnummer, so handelt es sich um ein Gebiet, das zu keiner Vorrangnummer gehört (z.B. D02.7).

Fehlt die Seitennummer, so war das Plattengebiet im Original nur 1 K groß, es handelt sich also um ein Original-Seitengebiet und nicht um ein Kopie-Seitengebiet (z.B.: D08..10). Wird statt der Zeichenfolge "verdraengt" "entladen" protokolliert, so handelt es sich um einen expliziten UNLOAD-Aufruf. Die Zahl i ist der Kernspeicherbedarf in K, den der Programmlauf zu diesem Zeitpunkt belegt.

Wird bei einem LOAD-Aufruf als Parameter eine Vorrangnummer angegeben, die zu einem nichtzugeladenen Overlay-Segment gehört, so wird protokolliert:

***LOAD(i)

wobei i die Vorrangnummer ist.

b) Aufruf mit Parameter = 2:

Alle im Programm auftretenden SSR-Fehler werden protokolliert, und zwar in der Form:

***Fehler bei SSR i j, A/Q/D/H:

Dahinter werden die Register RA,RQ,RD und RH nach dem SSR-Fehler ausgegeben.

Ist das Ablaufprotokoll eingeschaltet, so wird danach noch der Versorgungsblock des fehlerhaften SSR's ins Ablaufprotokoll gedumpt.

Die Zahlen i und j sind die beiden Adreßteile des SSR's.

c) Aufruf mit Parameter = 4:

Alle Pagefaults werden protokolliert, und zwar in der Form:

*****alarmausloesende Adresse: i,BC/STB2/STB1 = j .

Dabei ist i die (sedezimale) Adresse der Kernspeicherzelle, auf die zugegriffen wurde, obwohl die Seite, zu der diese Zelle gehörte, nicht im Kernspeicher lag.

Die Zeichenfolge j besteht aus 12 sedezimalen Ziffern (Tetraden), deren erste beide den Befehlscode (BC) des Befehles darstellen, der den Pagefault auslöste.

Die 3. bis 6. Tetrade sind die Steuerbits 2 (STB2) und 7. bis 12. die Steuerbits 1 (STB1), die gewisse Hardwarezustände bei dem Pagefault anzeigen (siehe Beschreibung des SSR 4 8 ([1])).

d) Aufruf mit Parameter = 8:

Es werden in BO&PAGING intern auftretende Engpässe protokolliert, und zwar Kernspeicherengpässe (beim Verlängern von Gebietslisten im Abwickler bzw. beim Anlegen eines großen Leitblockes und Gebietslistenengpässe (beim Überlauf von Gebietslisten im Abwickler). Dahinter wird noch die aktuell existierende Anzahl von Kopie-Seitengebieten ausgegeben.

Beispiel: ***KSP-Engpass, 42 Seitengebiete

e) Aufruf mit Parameter = 16:

Es werden Kreation und Löschung von Kopie-Seitengebieten protokolliert.

Diese Protokollierungen haben folgende Form:

```
***kreiert:    i
***geloescht:  i j .
```

Dabei ist i die operatorrelative (sedezimale) Seitennummer des Kopie-Seitengebietes (von 0 an durchnumeriert).

Für die Zeichenfolge j gibt es folgende Möglichkeiten:

1. (SSP)
   Es handelt sich um ein Kopie-Seitengebiet, das zu einem schreibgeschützten (unveränderlichen) Gebiet gehörte und deswegen nicht zurückgeschrieben werden mußte.

2. (1/1)
   Das Kopie-Seitengebiet mußte zurückgeschrieben werden. Das konnte jedoch mit einem Transport von der Länge 1 K geschehen, weil entweder das Kopie-Seitengebiet noch im Kernspeicher lag oder weil Kernspeicherreserve für den Transport Hintergrund-Hintergrund vorhanden war.

3. (8/8)
   Das Kopie-Seitengebiet mußte mit 8 Transporten von der Länge 1/8 K zurückgeschrieben werden.

4. (LOE)
   Das Kopie-Seitengebiet gehört zu einem dynamischen Freispeichergebiet, das inzwischen soweit gekürzt wurde, daß das Kopie-Seitegenbiet bereits außerhalb lag und deshalb einfach gelöscht werden konnte.

f) Aufruf mit Parameter = 32:

Alle im Programm auftretenden Alarmkeller werden im Ablaufprotokoll gedumpt, falls dieses eingeschaltet ist (siehe Beschreibung des SSR 4 8 [1]).

g) Aufruf mit Parameter = 64:

Der gesamte Ablauf des Demand Pagings wird zeitlich überwacht (durch Einschalten von Meßroutinen). Dabei werden Rechnerkern-Zeit und E/A-Transportindex gemessen, und zwar vom Demand Paging insgesamt und von jedem im Demand Paging durchgeführten SSR einzeln.

Beim LDPROT-Aufruf wird dann eine Statistik der Zeiten und SSR's, die das Demand Paging selbst benötigt hat, ausgegeben (siehe (4.5.6.)).

Alle dieses Protokollierungen können kombiniert werden durch Addition der entsprechenden Parameterwerte (die einzelnen Typen sind bitweise von rechts nach links in dem Parameter verschlüsselt).

Wird LDPEIN bzw. BL.LDPEIN mit einem Wert aufgerufen, so werden die entsprechenden Testprotokollierungen eingeschaltet - bei Aufruf von LDPAUS bzw. BL.LDPAUS ausgeschaltet. Die Testprotokollierungen, deren zugeordnete Bits im Parameter nicht gesetzt sind, werden von einem solchen Aufruf nicht verändert. Alle Testprotokollierungen werden im Konsol- und Ablaufprotokoll (falls eingeschaltet) ausgeführt, sofern nicht ausdrücklich anders beschrieben.

4.5.6. LDPROT, BL.LDPROT

Der Aufruf dieser parameterlosen Unterprogramme bewirkt sofortiges Ausdrucken der Anzahlen Pagefaults und Zuladungen seit Operatorlaufbeginn.

LDPROT wird einmal von S&CC kurz vor Ende des Programmlaufes aufgerufen (siehe auch (7.1.)). Sind die Meßroutinen des Demand Paging eingeschaltet (siehe (4.5.5. f)), so wird zusätzlich ausgegeben, was das Demand Paging selbst "gekostet" hat, und zwar:

a) die gesamte Rechnerkern-Zeit,

b) der gesamte Transportindex,

c) alle SSR's im Demand Paging mit ihrer jeweiligen Anzahl, ihrer gesamten benötigten Rechnerkernzeit, ihrem Transportindex und ihrem mittleren Rechnerkern-Zeit-Bedarf pro SSR,

d) die Rechnerkernzeit- und Transportindex-Summe aller SSR's und der Rechnerkernzeit-Bedarf gemittelt über alle SSR's.

Beispiel:

```
Anzahl Pagefaults: 132
Anzahl Zuladungen: 136
RK- Zeit Paging  :       3.46330
EA- Zeit Paging  :      13.79600

Statistik der SSR's in BO&PAGING :

   SSR Anzahl    RK- Zeit     EA- Zeit   RK- Mittel

  4 32   1452     1.23420                  0.00085
  4  8    127
  0 32    127     0.02522                  0.00019
  0 34    127
  4 28    128     0.03257                  0.00025
  3 56    210     1.14240     9.37600      0.00544
  0 30    124
  3  0     31     0.28081     1.17600      0.00905
  3 68     32     0.04553     1.52900      0.00142
  3 24      1     0.00037                  0.00037
  4  0      1     0.00033                  0.00033
  0 22      2     0.00021                  0.00010
253 79      1     0.00154                  0.00154
253 11      1     0.00086                  0.00086
  6 16     17     0.03306     0.04200      0.00194
GESAMT   2381     2.79710    12.12300      0.00117
```

Die SSR's 4 32 sind dabei die zur Überwachung benutzten SSR's (T=5).

4.5.7. LDSTEU, BL.LDSTEU

Diese Unterprogramme können mit beliebiger Parameterzahl aufgerufen werden und dienen zur Einstellung gewisser Steuergrößen für das Demand Paging.

Alle nicht als Parameter aufgeführten Steuergrößen bleiben unverändert.

Bedeutung der Parameter:

1. MAXKSB

   Mit Hilfe dieser Größe kann angegeben werden, wieviel K Kernspeicher maximal benutzt werden darf, auch wenn mehr zur Verfügung steht.

   Man kann somit den maximal benutzten Kernspeicher freiwillig reduzieren, während das Demand Paging den Kernspeicherbedarf normalerweise an den zur Verfügung stehenden Kernspeicher anpaßt.
   (Voreinstellung: ∞)

2. KSBRES

   gibt an, wieviel K Kernspeicher auf jeden Fall physikalisch frei bleiben sollen. (Voreinstellung: 0)

   Der Kernspeicherbedarf eines Programmes ergibt sich dann als maximal

   Min(KSB-KSBRES,MAXKSB).

   KSB ist dabei der dem Auftrag maximal zur Verfügung stehende Kernspeicher.

3. INORM

   ist die Anzahl der Pagefaults, nach der das "Langzeit-Gedächtnis" des Verdrängungsalgorithmus' normiert werden soll (siehe (3.2.)).
   (Voreinstellung: 128)

4. LNGPU

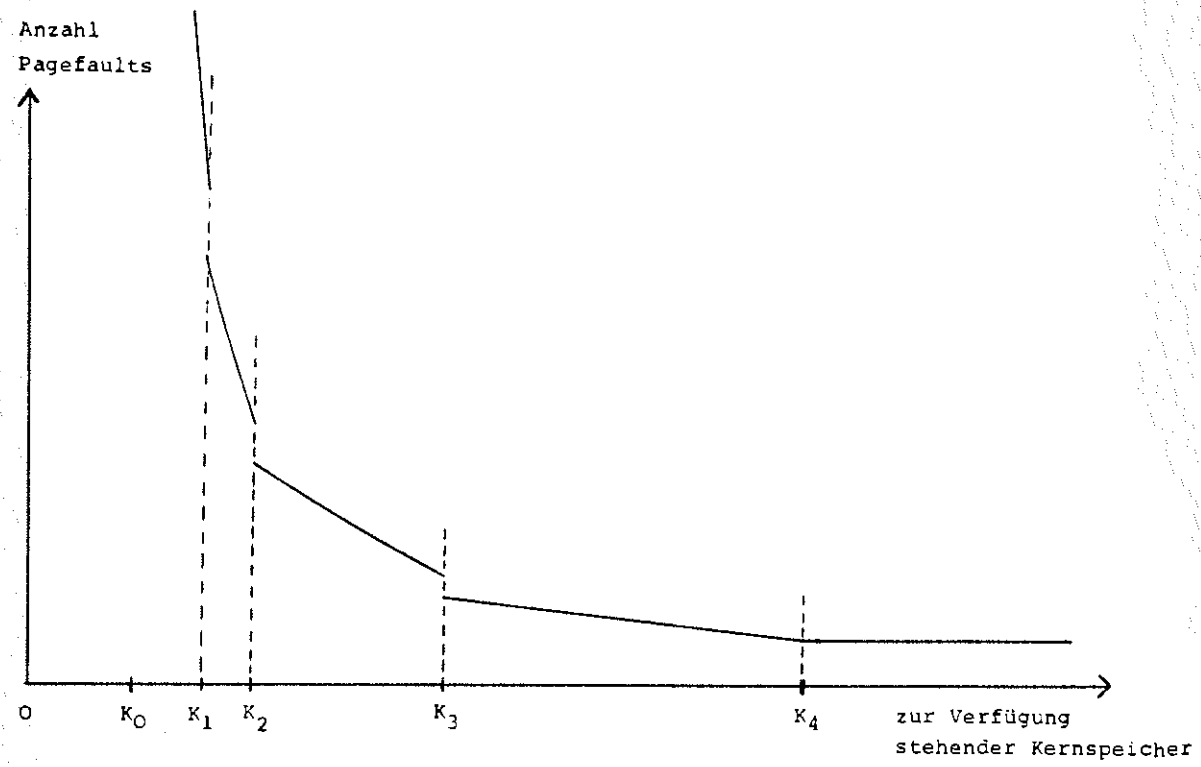
   stellt die Länge des Ringpuffers für die letzten Pagefault-Seiten ein, LNGPU ist also ein Maß für die Größe des "Kurzzeit-Gedächtnisses" des Verdrängungsalgorithmus' (siehe (3.2.)).
   (Voreinstellung: 3) .

5. Optimierungshinweise für den Benutzer

Messungen haben gezeigt, daß ein Pagefault zusammen mit dem Laden einer Seite und dem Verdrängen einer anderen im Durchschnitt ca. 20 m sec Rechnerkern-Zeit kostet. (Dazu kommt natürlich auch noch E/A-Wartezeit.)

Daraus ergibt sich, daß man Wert darauf legen muß, die Anzahl Pagefaults im Programm möglichst gering zu halten.

Zunächst soll anhand eines Diagramms gezeigt werden, wie in einem "normalen" Programm die Pagefault-Anzahl von dem zur Verfügung stehenden Kernspeicher abhängt.



Man kann 6 Bereiche auf der Achse des Kernspeichers unterscheiden, die folgende Bedeutung haben (unter KSP wird der dem Programm zur Verfügung stehende Kernspeicher verstanden).

a) $0 \leq KSP < KO$

Es ist zuwenig Kernspeicher vorhanden, um das Programm überhaupt ans Laufen zu bringen.

Als Faustregel für KO gilt:

$$KO = \max(RES+11, LGMAX+4)$$

Dabei ist RES die Größe des residenten Programmteils in K Worten und LGMAX die Länge des größten Laufzeitgebietes in K. Diese Größen werden vom Montierer ausgegeben, wenn die Spezifikation PROTOKOLL beim MONTIERE-Kommando besetzt ist.

(Vorsicht: der Montierer protokolliert die Größen als sedezimale Zahlen.)

KO muß übrigens nicht kleiner als K1 sein - in Extremfällen kann der Wert sogar größer als K4 sein, z.B. wenn keine zuladbaren Programmteile vorhanden sind und kaum dynamische Speicher gebraucht werden.

b) $KSP \leq K1$

Es steht so wenig Kernspeicher zur Verfügung, daß der Programmlauf im wesentlichen nur aus Zuladungen und Verdrängungen besteht und die Rechenzeit deshalb gegen Unendlich strebt.

In diesem Bereich ist ein Programmlauf nicht sinnvoll.

c) $K1 < KSP \leq K2$

In diesem Bereich steigt die Anzahl der Pagefaults und damit die Rechenzeit erheblich, wenn der Kernspeicher geringfügig verringert wird. Ein Programmlauf ist nur "in Notfällen" sinnvoll, also höchstens dann, wenn der physikalisch vorhandene Kernspeicher kleiner als K2 ist.

d) $K2 < KSP \leq K3$

Bei großen Programmen ist dies der günstigste Arbeitsbereich als Kompromiß zwischen der zusätzlich benötigten Rechenzeit und der Systembelastung durch den Kernspeicherbedarf.

Der Sprung bei K2 entsteht dadurch, daß bei geringerer Kernspeichergröße in Schleifen ständig mehr verschiedene Seiten benötigt werden, als gleichzeitig in den Kernspeicher passen. Diese Seiten werden dann dauernd verdrängt, um gleich darauf wieder benötigt und damit zugeladen zu werden.

Den Punkt K2 wird man im Normalfall dort lokalisieren können, wo ein Programm ca. 3- bis 5-mal soviel Rechenzeit benötigt wie bei $KSP > K4$.

e) $K3 < KSP \leq K4$

In diesem Bereich, in dem die Rechenzeit bei Verringerung des zur Verfügung stehenden Kernspeichers nur unwesentlich steigt, sollte man nur arbeiten, wenn K4 sehr klein und die Systembelastung durch die Kernspeicherforderung vernachlässigbar ist, wenn also z.B. bei einer 256-K-Anlage $K4 \leq 40$ ist.

K3 ist im allgemeinen die Kernspeichergröße, bei der ein Programm ca. 25 % mehr Rechenzeit aufnimmt als bei $KSP > K4$.

f) $K4 < KSP$

Oberhalb von K4 hat der zur Verfügung stehende Kernspeicher keinen Einfluß mehr auf die Rechenzeit, da alle Seiten, die das Programm benötigt, gleichzeitig im Kernspeicher Platz haben und nur einmal zugeladen werden.

Im Sinne der Gesamtauslastung des Systems ist es völlig unrationell, in diesem Kernspeicherbereich zu arbeiten.

Was kann nun der Benutzer tun, um sein Programm so zu optimieren, daß es bei geringem Kernspeicherbedarf möglichst wenig Pagefaults erzeugt?

Zunächst muß dafür gesorgt werden, daß KO so klein wird, daß das Programm überhaupt gestartet werden kann.

Das erreicht man dadurch, daß möglichst viele Programmteile als zuladbar erklärt werden (Spezifikation TRANSFER im UEBERSETZE- bzw. MONTIERE-Kommando).

Es dürfen alle Unterprogramme als zuladbar erklärt werden außer:

BO&PAGING

S&CC

BCPL&R

ALG68_RAHMEN

eigenes Hauptprogramm (bei BCPL und ALGOL68 darf dieses auch zuladbar sein).

Zur eigentlichen Minimierung der Pagefault-Anzahl sind folgende Punkte zu berücksichtigen:

5.1. Maßnahmen im Programm

a) Bei mehrdimensionalen Feldern sollten in Schleifen die linken Indizes schneller laufen als die rechten.

   Allgemein gesagt:
   Beim Abarbeiten von größeren Datenmengen sorge man für ein möglichst lineares Vorgehen und vermeide tunlichst "wildes" Zugreifen auf Feldelemente.

b) In BCPL sollte man Vektoren in BL.LISTE weder verlängern noch verkürzen, da hierbei immer alle Vektoren mit größeren Nummern verschoben werden müssen. Das Verschieben führt dazu, daß alle betroffenen Seiten zugeladen werden müssen.

c) In ALGOL60 sorge man dafür, daß die Deklarationen von Prozeduren, die häufig aufgerufen werden und/oder sich gegenseitig aufrufen, direkt hintereinander im Programm stehen (gemeint sind nicht vorübersetzte Prozeduren).

d) Werden zuladbare Programmteile längere Zeit nicht mehr benötigt, so entlade man sie durch einen expliziten UNLOAD-Aufruf mit der entsprechenden Vorrangnummer.

   Damit unterstützt man den Verdrängungsalgorithmus.

e) Werden zuladbare Programmteile sehr lange oder gar nicht mehr benötigt, so rufe man UNLOAD mit negativer Vorrangnummer auf - das spart Plattenraum und bei Programmen mit mehr als 100 K virtuellem Kernspeicher auch Rechenzeit.

f) Sind im Programmlauf einzelne Phasen trennbar, in denen jeweils andere Programmteile und/oder Felder benutzt werden, so rufe man vor Beginn jeder solchen Phase die Prozedur LDNORM auf, um dem Verdrängungsalgorithmus zu helfen.

g) In der Testphase lohnt es sich manchmal, das Programm mit verschiedenen Größen von Kurzzeit- und Langzeit-"Gedächtnis" des Verdrängungsalgorithmus' laufen zu lassen, um eine optimale Einstellung herauszufinden (siehe Beschreibung der Prozedur LDSTEU).

Leider lassen sich hierfür keine allgemeingültigen "Rezeptchen" angeben, da sich die Vorgänge im Programm kaum algorithmisieren lassen.

5.2. Maßnahmen beim Übersetzen/Montieren

a) Alle Programmteile, die häufig benutzt werden, versehe man mit derselben Vorrangnummer, damit sie vom Montierer "nahe beieinander" montiert werden und nicht in vielen verschiedenen Seiten liegen.

b) Man verwende möglichst wenige Vorrangnummer und möglichst nur solche < 50, damit nicht soviel Verschnitt am Ende der einzelnen Gebiete entsteht.

Dabei muß man jedoch darauf achten, daß die zugehörigen Laufzeitgebiete nicht zu groß zum Laden des Programmes werden.

c) Programmteile, die sich häufig gegenseitig benutzen, sollten in der TRANSFER-Angabe beim Montieren dicht nebeneinander angegeben werden, da die Reihenfolge, in der die Objekte unter TRANSFER aufgeführt sind, für die Reihenfolge der Montage maßgeblich ist.

d) Wann immer es geht, sollte man die Programmteile in der TRANSFER-Spezifikation des MONTIERE-Kommandos statt des UEBERSETZE-Kommandos aufführen, da man dann bessere Steuerungsmöglichkeiten hat, wie die Programmteile abgelegt werden.

e) Wenn irgend möglich, verzichte man auf die ZUSATZ-Spezifikation SSI des MONTIERE-Kommandos, da ZUSATZ = SSI beim Montieren die im Programmlauf durchschnittlich erforderliche Rechenzeit pro Pagefault erhöht.

5.3. Informationen zur Optimierungshilfe

Einen groben Anhaltspunkt über den Erfolg eines Optimierungsversuches gibt die am Programmende ausgedruckte Anzahl von Pagefaults und Zuladungen. Detaillierte Informationen über die Vorgänge im Demand Paging erhält man durch Einschalten der verschiedenen Ladeprotokollierungen (siehe Beschreibung der Prozedur LDPEIN).

Hat man sein Programm gesprächsfähig übersetzt, so kann man die Ladeprotokollierung auch mit Hilfe der Kontrollereignisverwaltung steuern, die die Kontrollanweisungen LDPEIN und LDPAUS versteht.

Beispiel:

```
STDHP *KE= START␣:LDPEIN␣.
STDHP *KE=**HALT␣:LDPAUS(127)␣.
```

Die Anweisungen können also auch parametrisiert werden.

6. Konstruktionsbeschreibung von BO&PAGING

Das Montageobjekt BO&PAGING ist in acht Teile gegliedert:

a) die Initialisierung,

b) der Rahmen für die Alarmbehandlung,

c) Unterprogramme für die SSR-Fehler-Behandlung,

d) Unterprogramme zum Laden und Entladen,

e) Protokoll- und Test-Unterprogramme,

f) die Abarbeitung der Eingänge zur Steuerung,

g) Konstanten-Bereich,

h) Variablen-Bereich .

6.1. Die Initialisierung (Eingang PAGING)

Zunächst werden zwei dynamische Gebiete kreiert, die zur späteren Aufnahme der Befehle, Konstanten und Variablen dienen. Das Gebiet für Befehle und Konstanten wird sofort in der richtigen Länge (2 K), das Gebiet für Variable in der maximal nötigen (5 K) kreiert.

Die Zone Z&LTB wird daraufhin untersucht, ob Teile des Adreßraumes doppelt belegt sind.

Falls Doppelbelegungen vorhanden sind, werden diese als Overlay-Bereiche interpretiert, und es werden die Overlaytabelle und die Overlaybitleiste angelegt (siehe (6.4.2.)).

Dann wird die Seitentabelle aufgebaut, die für jede belegte Seite des Adreßraumes 2 Ganzworte enthält (siehe (6.4.1.)).

Die Adreßraumbelegung wird aus den Gebieten ermittelt, die in der Zone Z&LTB aufgeführt sind.

Der Montierer wird zum Generieren der Zone Z&LTB (Ladeteilbeschreibung) gezwungen, indem das Montageobjekt BO&TRANSFER von BO&PAGING verlangt wird. Dieses MO enthält weder Befehle noch Konstante noch Variable, sondern besteht gewissermaßen nur aus dem Eingang BO&TRANSFER und der Vorrangnummer 99. Zusätzlich zu den statischen Gebieten (aus Z&LTB) werden aus der Common-Zone DYN&GEBIETE (siehe (6.4.4.)) die angeschlossenen Freispeicherprozeduren ermittelt und die von ihnen belegten Adreßraum-Seiten in die Seitentabelle aufgenommen.

Die statischen, vorbesetzten Variablen werden dann in das dynamische Variablengebiet kopiert, und das Variablengebiet wird auf seine benötigte Länge verkürzt.

Alle Befehle (und Konstanten) werden bis auf die nicht mehr benötigten Teile in das Befehls- und Konstanten-Gebiet kopiert, und dieses wird schreibgeschützt. Die Alarmadresse wird mit Ummeldungssperre in das dynamische Befehlsgebiet gelegt (SSR O 20, S=L) .

Der Rest der Initialisierung läuft bereits in den verschobenen Befehlen im dynamischen Befehls-Gebiet ab. Dabei werden alle in der Seitentabelle berücksichtigten Gebiete, sowohl die statischen als auch die dynamischen der Freispeicherprozeduren, vom Kernspeicher auf den Hintergrund verlagert. Die Indexbasis und die Seite, in der die Rücksprungadresse aus PAGING liegt, werden zugeladen.

Mit SSR O 34 (Modus=1) wird die Umleitung von SSR-Fehlern auf Ereignisalarme eingeschaltet. Nach dem Wiedereinstellen der Aufruf-Indexbasis erfolgt der Rücksprung aus PAGING, und die Initialisierung ist beendet.

6.2. Der Rahmen für die Alarmbehandlung

Das dynamische Befehlsgebiet ist logisch in zwei Bereiche aufgeteilt:

Der erste Bereich enthält die Befehle der eigentlichen Alarmbehandlung, der zweite Protokoll- und Test-Unterprogramme, die Befehle für die Eingangsbehandlung sowie die Konstanten.

Nach dem üblichen Alarmbehandlungsvorspann wie eigene Indexbasis und UPZ einstellen, Register sichern und Alarmkeller besorgen wird anhand dieser Unterteilung verzweigt, je nachdem ob der Alarm in der eigentlichen Alarmbehandlung oder außerhalb aufgetreten ist.

6.2.1. Alarmarten

Es gibt 4 Sorten von Alarmen:

1. Zuladungsalarme
2. Engpaßalarme
3. Ummeldungsalarme
4. nicht behandelte Alarme .

6.2.1.1. Zuladungslarme

Zu den Zuladungsalarmen zählen alle Alarme, deren Ursache ein Pagefault ist, bei denen also eine Seite zugeladen werden muß:

a) Hardware-Alarme, bei denen das BEEC-Bit in den Steuerbits gesetzt ist.

b) Versorgungs-Alarme, bei denen Bit 1 im Register Q gesetzt ist.

c) Auf Ereignisalarme umgeleitete SSR-Fehler mit Fehlerschlüssel '2'.

Alle benötigten Seiten werden zugeladen. Danach wird der Operatorlauf fortgesetzt.

6.2.1.2. Engpaßalarme

Engpaßalarme sind diejenigen Alarme, deren Ursache ein Kernspeicherengpaß oder ein Gebietslistenüberlauf im Abwickler ist. Hierbei kommen nur auf Ereignislarme umgeleitete SSR-Fehler in Frage.

BO&PAGING behandelt die Fehlerschlüssel:

'5C' mit RQ = 0 (KSP-Mangel)
'23' (kein freier Kernspeicher für Achtelseitentransport)
'5D', '5E', '62' } Gebietslisten-Überlauf

Die Reaktion auf Engpaßalarme besteht aus Verdrängen von Seitengebieten (bei KSP-Mangel) bzw. Löschen von Kopie-Seitengebieten (bei Gebietslistenüberlauf).

6.2.1.3. Ummeldungsalarme

Da BO&PAGING seine Alarmadresse mit Ummeldungssperre anmeldet, wird ihm ein Ereignisalarm mit Bit 13 = L in RQ zugestellt, wenn ein anderes Unterprogramm versucht, die Alarmadresse umzumelden. Die Reaktion darauf besteht in einer kompletten Simulation des SSR 0 20, der zu dem Ummeldungsalarm geführt hat, einschließlich einer Simulation der Ummeldungssperre.

Ist die simulierte Ummeldungssperre gesetzt, so wird, falls ein Ummeldungsalarm auftritt, dieser leicht modifiziert und an die nächste Alarmbehandlung weitergereicht. Dadurch bleibt gewährleistet, daß jede beliebige Alarmbehandlung mit Demand Paging genauso funktioniert wie ohne.

6.2.1.4. Nicht behandelte Alarme

Alle anderen Alarme werden als "echte Alarme" klassifiziert.

Ein solcher Alarmkeller wird aufgehoben und

a) an die Standard-Alarmbehandlung weitergereicht,

b) bei jedem behandelten Alarm mittels SSR O 28 im Abwickler "restauriert".

Dadurch ist es möglich, im Demand Paging die Standard-Alarmbehandlung des Operators genauso wie alle anderen Programmteile zu behandeln, also zu- und wegzuladen.

6.2.1.5. Alarme in der Alarmbehandlung

Bei Engpaßalarmen wird nach Beseitigung der Ursache (d.h. nach Verdrängen bzw. Löschen eines Seitengebietes) und Restaurieren aller Register die Alarmbehandlung bei dem auslösenden SSR aufgesetzt.

Bei anderen umgeleiteten SSR-Fehlern wird mit dem SSR O 26 auf der SSR-Fehleradresse fortgestartet. Alle anderen Softwarealarme werden gespeichert: Die Alarmbehandlung wird an der Unterbrechungsstelle fortgesetzt und der gespeicherte Softwarealarm beim Verlassen der Alarmbehandlung an die Standard-Alarmbehandlung des Operators durchgestellt.

Alle Hardware- und Versorgungs-Alarme in der Alarmbehandlung führen mit der Meldung "+++++ Alarm im Alarm" zu sofortigem Operatorlaufabbruch.

6.2.1.6. Funktionsfähigkeit anderer Alarmbehandlungen unter Demand Paging

Da BO&PAGING so konstruiert ist, daß jeder beliebige Operator mit Demand Paging genauso abläuft wie ohne, ist natürlich auch gewährleistet, daß jede Alarmbehandlung des Operators genauso funktioniert wie ohne Demand Paging.

Da es jedoch in BS3 keine Möglichkeit gibt, sich dagegen zu schützen, daß irgendein Unterprogramm die Ereignisalarm-Zustellungssperre setzt oder die Umleitung von SSR-Fehlern auf Ereignisalarme abschaltet, muß eine Alarmbehandlung, die unter Demand Paging laufen soll, zwei Bedingungen genügen:

a) Es darf niemals die Ereignisalarm-Zustellungssperre gesetzt werden.

b) Die Umleitung von SSR-Fehlern auf Ereignisalarme darf niemals ausgeschaltet werden.

6.2.2. Erkennen von benötigten Seiten

Da jeder Pagefault Rechenzeit kostet, wurde besondere Mühe darauf verwendet, bei einem Pagefault möglichst alle benötigten Seiten zu erkennen und dafür zu sorgen, daß

a) alle benötigten Seiten sofort zugeladen werden
und
b) keine benötigte Seite verdrängt wird.

Das Weglassen von a) würde dazu führen, daß manchmal mehrere Pagefaults nötig wären, um einen einzigen Befehl bzw. SSR korrekt auszuführen.

Das Weglassen von b) hätte den katastrophalen Effekt zur Folge, daß im schönen Wechsel immer eine andere benötigte Seite verdrängt wird, bis die Verdrängungszähler aller benötigten Seiten so hochgeschaukelt sind, daß zufällig mal alle benötigten Seiten zugeladen sind.

6.2.2.1. Erkennen der Pagefault-Seite

In den meisten Fällen kann man aus der alarmauslösenden Adresse direkt die Pagefault-Seite ermitteln.

Bei einigen Pagefault-Arten bekommt man aber nur die Anfangsadresse eines Speicherbereiches ausgeliefert und muß anhand der Länge dieses Bereiches untersuchen, welche der evtl. an dem Speicherbereich beteiligten Seiten die Pagefault-Seite ist.

Hat man die Pagefault-Seite erkannt, und diese Seite ist bereits zugeladen, so handelt es sich bei dem behandelten Speicherschutzalarm um einen Schreibschutzalarm statt um einen Zuladungsalarm.

Schreibschutzalarme müssen als "echte Alarme" an die Standard-Alarmbehandlung durchgereicht werden

Die Pagefault-Seite wird von BO&PAGING folgendermaßen ermittelt:

(siehe nächste Seite)

ALARM	Register im Alarmkeller, das die alarmauslösende Adresse enthält	Register im Alarmkeller, das die Anfangsadresse des alarmauslösenden Speicherbereiches enthält	Länge des alarmauslösenden Speicherbereiches
BEEC-Alarm mit Befehlscode "BL"	RS, linkes Halbwort		
BEIC-Alarm		RS, linkes Halbwort	128 Ganzworte
BEEC-Alarm mit Befehlscode '0'	BF		
Sonstige BEEC-Alarme	BA		
Versorgungsalarm (Bit 1 in RQ=L)		BB	aus der SSR-Tabelle (siehe (6.4.3.)) entnommen
umgeleitete SSR-Fehler mit Fehlerschlüssel '2'	RQ, rechtes Halbwort		

6.2.2.2. Erkennen sonstiger benötigter Seiten

a) Grundsätzlich werden die Seiten benötigt, die an der aktuellen Indexbasis beteiligt sind.

b) Die Seite wird benötigt, auf die der aktuelle Befehlsfolgezähler zeigt.

Ausnahme: Der anstehende Befehl ist ein Sprungbefehl (Steuerbit BEMP im Alarmkeller gesetzt).

Bei den Befehlen "WTV" und "WTR" steht der Befehlsfolgezähler im Register BB des Alarmkellers, bei SSR-Fehlern im linken Halbwort des Registers RA und sonst im Register BF des Alarmkellers.

c) Befehle "SE", "SUE" und "SFBE":

Wird der Pagefault dadurch ausgelöst, daß der Speicheroperand nicht zugeladen ist, wird außer der Seite, in der die alarmlösende Adresse liegt, auch die Seite benötigt, in der das Sprungziel liegt.

Diesen Fall erkennt man daran, daß im Alarmkeller das Steuerbit BEAB gesetzt ist.

d) Befehle "ZI" und "BCI":

Tritt bei diesen Befehlen ein BEEC-Alarm auf ohne BEIC-Bit, so ist der Speicheroperand, der die Anfangsadresse der neuen Indexbasis enthält, nicht zugeladen. Der Inhalt dieser Zelle erlaubt es, die für die Bildung der neuen Indexbasis benötigten Seiten zu erkennen.

e) Versorgungsalarm oder ein SSR-Fehler mit einem der folgenden Fehlerschlüssel:

'2'
'23'
'5C' (RQ='0')
'5D'
'5E'
'62'

Diese SSR-Fehler werden von BO&PAGING speziell behandelt.

Nach Zuladung, Kernspeicherfreigabe oder Gebietslöschung werden die entsprechenden SSR's wiederholt.

Dabei klassifiziert BO&PAGING alle Seiten als benötigt, die entweder am Versorgungsblock oder an irgendeinem der im Versorgungsblock angegebenen Puffer beteiligt sind.

Das wird durch komplette Interpretation des jeweiligen Versorgungsblockes in Abhängigkeit vom auszuführenden SSR erreicht (siehe (6.4.3.)).

f) SSR-Fehler, die nicht speziell behandelt werden:

Die Seite wird als benötigt erkannt, in der die Fehleradresse für den SSR liegt.

6.3. Adreßraumaufteilung in BO&PAGING

BO&PAGING benötigt 33 Ganzworte 16-Bit-Konstanten und 71 Ganzworte 16-Bit-Variable.

Dazu kommt die vom Montierer angelegte 16-Bit-adressierbare Zone Z&LTB.
Die Befehle, Konstanten und Variablen der Initialisierung liegen in einer nicht schreibgeschützten freien Befehlszone, die 290 Ganzworte lang ist.

Die restlichen Befehle und Konstanten liegen in einer 2 K langen freien Befehlszone, die restlichen Variablen in einer 1 K langen 22-Bit-adressierbaren Datenzone. Ab Adresse '20800' beginnt eine 11 K lange Lücke, die zur Aufnahme der verschobenen Befehle, Konstanten und Vaiablen dient.

Nach der Verschiebung liegen die Befehle und Konstanten in dem 2 K langen dynamisch kreierten Gebiet mit dem OGNM "DMPBEF" ab Adresse '21000'.

Die Variablen werden in das dynamisch kreierte Gebiet mit dem OGNM "DMPVAR" verschoben.

Dieses Gebiet beginnt auf Adresse '23000' und ist 1 bis 5 K lang je nach Länge der Overlay- und der Seiten-Tabelle. In BO&PAGING wird nur die Lücke (mit dem Namen DMPLUE) explizit angeordnet. Explizit ist also nur der Adreßraum von '20800' bis '25FFF' belegt.

Wird die Prozedur FTNFSP benutzt (siehe (8.)), so liegt der FORTRAN-"Freispeicher" von FTNFSP direkt hinter der Lücke DMPLUE, also auf Adresse '26000', explizit angeordnet.

6.4. Aufbau der Listen von BO&PAGING

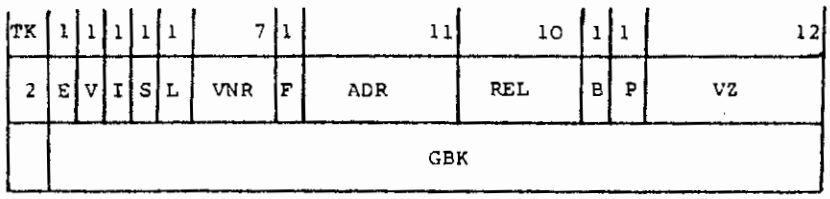
6.4.1. Die Seitentabelle

In der Seitentabelle werden für jede belegte Seite des Adreßraumes 2 Ganzworte freigehalten. Als belegt gelten alle Seiten, die (laut Zone Z&LTB) zu einem statischen Gebiet gehören.

Ausgenommen sind die Seiten, die die ursprünglichen Befehle, Konstanten und Variablen von BO&PAGING enthalten.

Hinzu kommen noch die Seiten, die zu der Freihaltezone einer an die Schnittstelle für Freispeicherprozeduren angepaßten Prozedur gehören.

Aufbau der Seitentabelle:



Dabei bedeuten:

E Existenzbit
- = L Kopie-Seitengebiet existiert
- = O es existiert kein Kopie-Seitengebiet

V Verdrängungsbit
- = L Seite ist verdrängt
- = O Seite ist zugeladen

I Identitätsbit
- = L Seitengebiet ist ein Original-Seitengebiet
- = O Seitengebiet ist ein Kopie-Seitengebiet

S Schreibschutzbit
- = L Seite gehört zu schreibgeschütztem Originalgebiet
- = O Seite gehört zu schreibfreiem Originalgebiet

L Löschbit
- = L Seitengebiet kann nicht gelöscht werden, weil es nicht existiert oder ein Original-Seitengebiet ist.
- = O Seitengebiet kann gelöscht werden

VNR = Vorrangnummer, zu der das Originalgebiet gehört.
Gehört das Gebiet zu mehr als einer Vorrangnummer, so ist VNR die erste gefundene

F Freispeicherbit
- = L Seite gehört zu einem Freispeicher
- = O Seite gehört zu einem statischen Gebiet.

ADR = Adressierung
ADR ist die operatorlaufrelative Seitenummer der Seite.

REL = Relativadressierung
REL ist die Originalgebiet-relative Seitenummer der Seite.

B = benötigt-Bit
= L es handelt sich um eine aktuell benötigte Seite
= O Seite wird aktuell nicht benötigt

P = Pagefault-Bit
= L Seite ist in dem Ringpuffer für die letzten Pagefault-Seiten vermerkt ("Kurzzeit-Gedächtnis")
= O Seite ist nicht in dem Ringpuffer vermerkt

VZ = Verdrängungszähler der Seite ("Langzeit-Gedächtnis")

GBK = Gebietskennzeichen des Originalgebietes aus der Zone Z&LTB bzw. aus der Freispeichertabelle.
(OGNM oder PGNM)

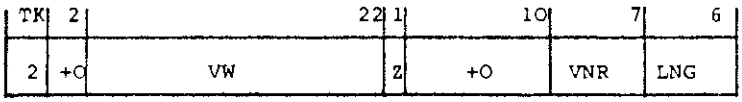
Die Redundanz in der Seitentabelle (L,B,P) dient dazu, Suchvorgänge optimal mit dem Befehl TMIN durchführen zu können:

Muß eine Seite verdrängt werden, so wird mit der Maske 'BFFFFF FFC000' ein TMIN durchgeführt, soll ein Kopie-Seitengebiet gelöscht werden, mit der Maske 'F7FFFF FFC000'.

6.4.2. Die Overlaytabelle

In der Overlaytabelle werden alle Gebiete vermerkt, die an einem Overlaybereich beteiligt sind.

Sie hat folgenden Aufbau:



VW = Verweis auf das Seitentabellenelement der Seite O des Overlaygebietes

Z = Zuladungsbit
= L Overlaygebiet gehört zu einer aktuell zugeladenen Vorrangnummer
= O Overlaygebiet gehört nicht zu einer zugeladenen Vorrangnummer, darf also nicht zum Zuladen einer Seite benutzt werden.

VNR = Vorrangnummer, zu der das Overlaygebiet gehört
Gehört das Overlaygebiet zu mehr als einer Vorrangnummer so werden mehrere Elemente in der Overlaytabelle angelegt.

LNG = Länge des Overlaygebietes in K.

Soll in einem Programm mit Overlay eine Seite zugeladen oder verdrängt werden, und ist die Seite in der Overlaybitleiste als Overlayseite vermerkt, so wird anhand der Overlaytabelle das zugehörige Overlaygebiet bestimmt.

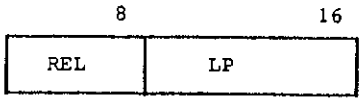
6.4.3. Die SSR-Tabelle

Die SSR-Tabelle wird zum Interpretieren der Versorgungsblöcke von SSR's benötigt. Sie enthält für jeden existierenden SSR ein Ganzwort und hat folgenden Aufbau:

ADR = Adreßteile des SSR's

LNG = 0: SPEZ ist eine Sprungadresse.
Dort findet dann eine Sonderbehandlung des SSR's statt.

LNG > 0: LNG ist die Länge des Versorgungsblockes.
SPEZ hat folgenden Aufbau:



LP = 0: REL ist die versorgungsblockrelative Adresse eines Ganzwortes, in dem eine Angabe der Form

steht.

LP > 0: REL ist die versorgungsblockrelative Adresse eines Halbwortes, in dem die Pufferadresse steht.

LP ist die Länge des Puffers in Ganzworten.

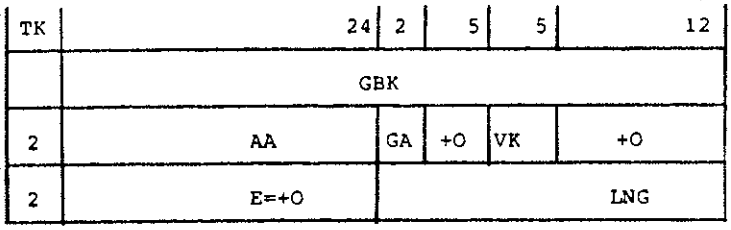
6.4.4. Die Freispeicherliste

Die Freispeicherliste ist die Schnittstelle für Freispeicherverwaltungsprozeduren. Sie liegt in der Common-Zone DYN&GEBIETE, die von BO&PAGING nicht explizit vorbesetzt wird.

Das erste Ganzwort darf nicht explizit vorbesetzt werden, da es zur Überprüfung auf die Operator-Vorbesetzung dient.

Ab dem zweiten Ganzwort enthält die Common-Zone 14 Elemente für jeweils eine Freispeicherverwaltungsprozedur.

Jedes Element ist 3 Ganzworte lang. Eine Freispeicherprozedur, die an diese Schnittstelle angepaßt werden soll, muß ein Element der Freispeicherliste nach folgenden Konventionen vorbesetzen:



GBK ist das Gebietskennzeichen des (dynamisch kreierten) Freispeichergebietes (OGNM oder PGNM)

AA ist die operatorlaufrelative Anfangsadresse des Gebietes

GA = Gebietsart
= 0 Laufzeitgebiet
= 2 Dauergebiet

VK = Verarbeitungsklasse des Gebietes (sollte = Kernspeicher sein).

E = 0, wenn das Gebiet noch nicht existiert (Vorbesetzung)

E:= belegte Länge des Gebietes (>0!), wenn das Gebiet kreiert wurde.

LNG = max. mögliche Länge des Freispeichergebietes (= Länge der Freihaltezone)

Die Schnittstelle ist so konstruiert, daß die Freispeicherverwaltungsprozedur auf jeden Fall funktionsfähig ist, gleichgültig ob der Operator mit Demand Paging arbeitet oder nicht, und gleichgültig, ob der Initialisierungsaufruf "PAGING" vor oder nach der Kreation des Freispeichergebietes erfolgt.

Dazu müssen folgende Bedingungen erfüllt sein:

a) Vorbesetzung: E = 0, VK = "Kernspeicher"

b) vor der Kreation des Freispeichergebietes wird das Halbwort mit GA und VK in das 14. Halbwort des Versorgungsblockes für den SSR 3 0 eingespeichert. (BO&PAGING ändert bei der Initialisierung VK in "Platte" um).

c) nach der Kreation wird E mit der Länge des Freispeichergebietes (> 0) besetzt. (BO&PAGING verlagert bei der Initialisierung das Gebiet auf die Platte, falls E > 0).

Folgende Elemente werden bereits benutzt, falls die entsprechenden Freispeicherverwaltungsprozeduren anmontiert sind:

1.	A&FSP -	Freispeichergebiet	(OGNM="A&FSP")
2.	A&FSP -	E/A-Freispeichergebiet	(OGNM="A&FSPE")
3.	BO&FSP		(OGNM="BO&FSP")
4.	BCPL&CLIST	(Kontakt BL&LISTE)	(OGNM="BL&LIS")
5.	BCPL&R	(BCPL-Stack)	(OGNM="BL&FSP")
6.	ALGOL68:	statischer Stack	(OGNM="A_STAT")
7.	ALGOL68:	dynamischer Stack	(OGNM="A_DYNA")
8.	ALGOL68:	Heap	(OGNM="A_HEAP")
9. - 13.		frei	
14.	FTNFSP -	Freispeicher	(OGNM="FTNFSP")

Ein Element der Freispeicherliste wird als unbelegt erkannt, wenn das 1. Ganzwort (GBK) des Elementes gleich +0 oder gleich dem 1. Ganzwort der Common-Zone DYN&GEBIETE ist.

6.5. Realsierung der Unterprogramme

Für die Realiserung der Eingänge in BO&PAGING (siehe (4.)) mußte ein unkonventioneller Weg beschritten werden.

Die Schwierigkeit lag darin, daß die Eingänge auf die ursprünglichen, noch nicht verschobenen Befehle zeigen.

Diese Befehlsbereiche existieren aber nach der Initialisierung nicht mehr, so daß jeder Aufruf eines Einganges in BO&PAGING zu einem Speicherschutz-alarm führt.

Damit Transporte eingespart werden können, wird jeder BEEC-Alarm daraufhin untersucht, ob die alarmauslösende Adresse zu einem Eingang gehört.

In diesem Fall wird anhand einer Tabelle die dem Eingang zugeordnete UP-Adresse im verschobenen Befehlsbereich identifiziert und direkt angesprungen.

Die Register für die Unterprogramm-Versorgungen werden dann dem Alarm-keller entnommen.

7. Geänderte Standard-Unterprogramme

7.1. S&CC

In S&CC wurde der Initialisierungsaufruf für das Demand Paging eingebaut, und zwar direkt hinter CP1 nach dem Umstellen der Indexbasis:

```
BA(O/2), SFBE(PAGING/A),
```

Damit alle Testhilfen benutzbar bleiben, wird vor dem Start des Rückverfolgers oder eines Sprach-Dump-Operators der interne Kontaktname GEB&LOE in BO&PAGING) angesprungen:

```
BA(O/2), SFBE(GEB&LOE/A),
```

Dadurch werden vor einem solchen Operatorstart alle Kopie-Seitengebiete gelöscht (evtl. nach Zurückschreiben ins Original-Plattengebiet), damit keine Doppelbelegungen von Adressen existieren, die den Rückverfolger und die Dumpoperatoren verwirren.

Vor Beendigung des Operatorlaufes (hinter CPA1) wurde ein Aufruf von LDPROT eingefügt:

```
BA(O/2),SFBE(LDPROT/A),
```

7.2. S&KEP

In S&KEP mußte das Setzen der Ereignisalarm-Zustellungssperre verhindert werden, da sonst SSR-Fehler nicht auf Ereignisalarme umgeleitet werden.

Das wurde durch Ändern des Versorgungsblockes VBO22S erreicht (S:=O).

Zur leichteren Handhabung der Optimierungshilfen wurden die Kontrollanweisungen

```
LDPEIN  und
LDPAUS
```

eingebaut, die mit "Parameter", also einer in Klammern eingeschlossenen Zahl, oder ohne gegeben werden können.

7.3. A&FSP und MEMORY&A

A&FSP wurde die Benutzung der Schnittstelle für Freispeicherverwaltungsprozeduren eingebaut. Damit die ALGOL6O-Prozedur MEMORY beim Demand-Paging sinngemäß wie sonst funktioniert, wurde sie so geändert, daß sie nicht mehr den noch zur Verfügung stehenden Kernspeicher berücksichtigt, sondern die Länge der Freihaltezone in A&FSP (256 K Worte).

7.4. BCPL&CLIST und BCPL&R

BCPL&CLIST

Eingang BL&LISTE wurde desgleichen an die Schnittstelle für Freispeicherverwaltungsprozeduren angepaßt.

Aus Effektivitätsgründen wurde auf die Vorbesetzung der Listen verzichtet.
Die Verwaltung des BCPL-Stacks in BCPL&R wurde ebenfalls an die Schnittstelle für Freispeicherverwaltungsprozeduren angepaßt. (Länge der Freihaltezonen jeweils 256 K Worte)

7.5. BO&FSP

BO&FSP (siehe [3], [4]) wurde an die Schnittstelle für Freispeicherverwaltungsprozeduren angepaßt und seine Freihaltezone auf 256 K Worte verlängert.

8. Beschaffung großer Felder

Das Betriebssystem ist aus zwei Gründen leider nicht in der Lage, Programme mit sehr großen statisch deklarierten Feldern (z.B. in FORTRAN: INTEGER A(200000)) zu bearbeiten:

1. Der Montierer kann keine Gebiete erzeugen, die größer als 256 K sind.

2. Der Abwickler muß beim Start eines Programmes jedes Laufzeitgebiet zum Vorbesetzen komplett in den zur Verfügung stehenden Kernspeicher laden können.

   Alle Variablen-Felder, die zur selben Vorrangnummer (bzw. dem residenten Teil) gehören, liegen normalerweise in einem einzigen Laufzeitgebiet und müssen deshalb beim Programmstart in den Kernspeicher passen.

   Programmen, die mit dem Demand Paging arbeiten, stehen folgende Möglichkeiten zur Verfügung, sehr große Felder zu benutzen:

   a) TAS: TAS-Programme können entweder die Freispeicherprozeduren der höheren Programmiersprachen oder direkt die in (6.) beschriebene Schnittstelle für Freispeicherverwaltungen benutzen.

   b) BCPL: Es können 256 K Ganzworte (!) an dynamischen Variablen benutzt werden.

   Darüber hinaus können noch einmal 256 K Worte bei der Prozedur BL.LISTE angefordert werden.

   c) COBOL: Die einzige Möglichkeit besteht darin, ein FORTRAN- oder ALGOL60-Rahmenprogramm zu schreiben, das das COBOL-Programm aufruft und dabei ein großes Feld als Parameter übergibt.

   d) FORTRAN: In FORTRAN-Programmen kann man die COMMON-Zone FTNFSP benutzen, die verlängerbar ist.

   Beispiele:

   1. COMMON/FTNFSP/F(10)

      Hierbei ist auch EQUIVALENCE möglich:

      LOGICAL*1B(10)

      EQUIVALENCE(F,B)

      Verlängert wird die COMMON-Zone durch:

      CALL FTNFSP(200000)

      Dadurch wird sie 200000 Worte lang

2. COMMON/FTNFSP/X(100,50,1)

   CALL FTNFSP(250000)

   Wirkung wie

   DIMENSION X(100,50,50)

Die COMMON-Zone FTNFSP darf maximal 256 K Worte lang sein. Zusätzlich kann man sich mit Hilfe der in [3] beschriebenen Speicherprozeduren 256 K Worte "besorgen".

Einschränkung:
Da das Feld in der COMMON-Zone FTNFSP nicht in seiner tatsächlich benötigten Länge deklariert wird, kann man nicht mit DYNKON arbeiten!

Soll mit DYNKON gearbeitet werden, so empfiehlt sich ein ALGOL60-Hauptprogramm, in dem das Feld kreiert und als Parameter an ein FORTRAN-Unterprogramm durchgereicht wird.

e) ALGOL60: In ALGOL60 kann man wie üblich schreiben (z.B.):

'INTEGER' ARRAY' F[1:250000];

Dabei dürfen alle Felder zusammen nicht mehr als 250 K groß sein. Zusätzlich kann man sich mit Hilfe der in [4] beschriebenen Prozeduren Speicherplatz besorgen.

f) ALGOL68: Sowohl für Variable im dynamischen Stack als auch für solche im Heap stehen jeweils 256 K Worte zur Verfügung.

Für den statischen Stack stehen 64 K Worte zur Verfügung.

Alle Freispeicherverwaltungsprozeduren können beliebig in einem Programm gemischt verwendet werden, jedoch ist darauf zu achten, daß die Summe aller Maximallängen die Größe von ca. 1700 K nicht überschreitet - anderenfalls kann das Programm wegen Adreßraum-Überschreitung nicht montiert werden.

Die Maximalgröße der einzelnen Freispeicher von jeweils 256 K ist durch die Größe der dazugehörigen Freihaltezone bestimmt. Bei Bedarf können Spezialversionen der entsprechenden Prozeduren zur Verfügung gestellt werden, die eine andere Größe erlauben.

Dabei ist für einen einzelnen Freispeicher eine Maximalgröße von 1024 K Worten möglich.

9. Beachtenswertes

9.1. Inkompatibilitäten

a) Alle beteiligten FORTRAN- und COBOL-Montageobjekte müssen mit VARIANTE=GR übersetzt werden.

b) Die Eingänge LOAD, UNLOAD, S&LOAD, S&UNLOAD, LOAD&A, UNLOAD&A, S&LOLA und S&LOVE liegen nicht wie im Standard-Montageobjekt S&LOAD in einer Standard-Befehlszone und müssen daher mit SFBE angesprungen werden.

c) Als zuladbar deklarierte Montageobjekte (TRANSFER beim UEBERSETZE- bzw. MONTIERE-Kommando) werden nie im Originalzustand sondern immer im zuletzt verlassenen Zustand geladen - auch wenn ihre Vorrangnummer größer oder gleich 50 ist.

d) Die Leistungen von S&UEBERWACHE (also Überwachung des Programmlaufes, dynamische Kontrollereignisse etc.) können zwar in Anspruch genommen werden, dabei sind jedoch zwei Dinge zu beachten:

1. BO&PAGING selbst wird nicht mit überwacht, da BO&PAGING bei Alarmen die Regie behält.

   Dadurch kann S&UEBERWACHE selbst mit zu- und weggeladen werden.

2. Wird eines der in (4.) beschriebenen Unterprogramme aufgerufen, so wird die Überwachung automatisch ausgeschaltet, da S&UEBERWACHE nicht mehr an die Regie gelangt.

9.2. Zusätzliche Testhilfen durch das Demand Paging

a) Durch Einschalten der entsprechenden Testprotokollierung (siehe Prozedur LDPEIN (4.5.5.)) kann man alle SSR-Fehler, die im Programmlauf vorkommen (einschließlich eines Dumps ihrer Versorgungsblöcke) protokollieren lassen.

b) Wird auf eine Variable aus einem Freispeicher zugegriffen, und der Freispeicher ist zu kurz, so wird vor dem Durchreichen des (auch sonst üblichen) Speicherschutzalarmes eine Meldung ausgegeben:

```
+++++ Zugriff auf Seite i von j (LNG=k)
```

Dabei bedeutet:

- i die freispeicherrelative Seitenummer (von 0 an gezählt), in der die Variable läge, wenn der Freispeicher lang genug wäre,
- j der Name des Freispeichergebietes (z.B. A&FSP) und
- k die aktuelle Länge des Freispeichers in K Worten.

c) Tritt ein Fehler im Zusammenspiel zwischen LOAD- und UNLOAD-Aufrufen bei Programmen mit Overlay auf, so wird nicht schlicht "++++AO11 Fehler bei Systemkontakt" gemeldet, sondern es wird die Meldung ausgegeben:

+++++ fehlerhafte LOAD/UNLOAD-Reihenfolge bei Overlay

und anschließend ein Speicherschutzalarm weitergereicht.

Außerdem ist beim Demand Paging die Wahrscheinlichkeit wesentlich größer, daß solche Fehler, die normalerweise nur zu falschen Ergebnissen führen, überhaupt erkannt werden.

9.3. Fehlermeldungen des Demand Paging

a) +++++ Alarm im Alarm

Diese Meldung bedeutet (falls kein Programmierfehler in BO&PAGING vorliegt), daß der Variablenbereich von BO&PAGING überschrieben wurde. BO&PAGING dumpt dann alle seine Variablen und bricht den Operatorlauf ab.

b) +++++ Abbruch wegen Kernspeichermangels

Das bedeutet, daß aktuell mehr Seiten benötigt werden, als gleichzeitig in den zur Verfügung stehenden Kernspeicher passen.

c) +++++ Abbruch wegen Trommelspeichermangels und
+++++ Abbruch wegen Plattenspeichermangels sind selbsterklärend.

Bei b) und c) wird der Operatorlauf sofort abgebrochen, da diese Fälle nur auftreten, wenn nicht einmal mehr genug Speicher vorhanden ist, um die Standard-Fehlerbehandlung des Programmes zuzuladen.

d) +++++ Abbruch wegen verzögerter Alarmzustellung

führt ebenfalls zum sofortigen Operatorlaufabbruch.

Dieser Fehler kann nur auftreten, wenn irgendwelche Unterprogramme zeitweilig die Ereignisalarm-Zustellungssperre gesetzt haben, und ein SSR-Fehler auftritt, der verspätet (nach Lösen der Ereignisalarm-Zustellungssperre) zugestellt wird. Ein Fortsetzen des Programms ist dann nicht mehr sinnvoll.

e) +++++ AO5C ...

vor der Start-Meldung des Programms bedeutet, daß nicht genügend Speicher für die Initialisierung von BO&PAGING zur Verfügung steht.

LITERATURVERZEICHNIS

[1] Computer Gesellschaft Konstanz mbH
Systemdienste
SYSTEM TR440

[2] TR440 Standard-Unterprogramme
Teil 1
Organisatorische Unterprogramme

[3] Rechenzentrumsbericht Nr. 7503 der Ruhr-Universität Bochum
BOTRAN, eine FORTRAN-Spracherweiterung durch Code-Prozeduren

[4] Rechenzentrumsbericht Nr. 7603 der Ruhr-Universität Bochum
BOGOL-TAS, eine Spracherweiterung von ALGOL 60 durch Code-Prozeduren
zur Systemprogrammierung

Bisher erschienene Arbeitsberichte des Rechenzentrums

der Ruhr-Universität Bochum

Nr. 7101: K.-H. Mohn, M. Rosendahl, H. Zoller
AIDA; eine Dialogsprache für den TR 440 (vergriffen)

Nr. 7102: K.-H. Mohn, M. Rosendahl, H. Zoller
AIDA, ein Dialogsystem und seine Implementierung in ALGOL (vergriffen)

Nr. 7103: K.-H. Mohn, M. Rosendahl, H. Zoller
AIDA, Manual für den Benutzer (vergriffen)

Nr. 7104: 4. Jahresbericht des Rechenzentrums (Juni 1970 bis Juni 1971)

Nr. 7105: H. Wupper
WR 1802 - Ein einfaches Band-Betriebssystem für einen mittleren Rechner

Nr. 7201: H. Windauer
Existenzsätze zur (0,1,..,R-2,R) - Interpolation

Nr. 7202: W. Schelongowski
DIATRACE - Ein System zur interaktiven Assemblerprogrammierung

Nr. 7203: M. Jäger, M. Rosendahl, R. Staake
Einführung in die Listenverarbeitung anhand der Dialogsprache AIDA

Nr. 7204: R. Mannshardt, P. Pottinger
Einführung in die Benutzung des Teilnehmer-Rechensystems TR 440 in der RUB (vergriffen)

Nr. 7205: 5. Jahresbericht des Rechenzentrums (1.7.1971 bis 30.6.1972)

Nr. 7206: M. Rosendahl
BOGOL-TAS, ein Weg zur systemnahen Programmierung in ALGOL am TR 440

Nr. 7207: W. Stark
ILW, Programmsystem zur Berechnung des Instationären Ladungswechsels von Verbrennungskraftmaschinen (Modulbeschreibung und Eingabekonventionen)

Nr. 7208: W. Stark
ILW, Programmsystem zur Berechnung des Instationären Ladungswechsels von Verbrennungskraftmaschinen (Regelmechanismus und Berechnung der Rohrströmung)

Nr. 7209: H. Ehlich
Anregung und Kritik zum Betriebs— und Programmiersystem der TR 440

Nr. 7210: M. Rosendahl
BOGOL-STRING, eine flexible Zeichenkettenverarbeitung in ALGOL 60

Nr. 7211: H. Camici, H. Claus, H. Ehlich, D. Kipp
Arbeitsbericht über ein Programm zur Haushaltsführung

Nr. 7301: R. Mannshardt, K.-H. Mohn, H. Münch, P. Pottinger
Einführung in die Benutzung des Teilnehmer Rechensystems TR 440
2. geänderte Auflage (vergriffen)

Nr. 7302: K.-H. Mohn
Über einige Anwendungen des Computers in der Medizin

Nr. 7303: R. Buchmann
BODAT, ein schnelles und platzsparendes System zur Datenmanipulation und -speicherung in ALGOL 60 und FORTRAN

Nr. 7304: M. Hauenschild
Ansätze zur komplexen Kreisarithmetik

Nr. 7305: R. Buchmann
RB&QUELLHALT, ein TR440-Datenbanksystem zur platzsparenden Quellhaltung auf Datenträgern mit direktem Zugriff (LFD, WSP)

Nr. 7306: 6. Jahresbericht des Rechenzentrums (1.7.1972 bis 31.12.1973)

Nr. 7401: R. Buchmann
Der Systemoperator BO&BS3OP
Messungen und Steuerungen des Betriebssystems auf Operatorebene

Nr. 7402: R. Mannshardt
Herleitung und Prüfung spezieller Runge-Kutta-Verfahren mit einem impliziten Rechenschritt

Nr. 7403: R. Buchmann, H. Wupper
Unzulänglichkeiten des TR 440 Programmiersystems und ihre Umgehung

Nr. 7404: R. Green, K.-H. Mohn
Quellbezogene FORTRAN Optimierungen für den Compiler des TR 440

Nr. 7405: R. Buchmann
BODAT, ein schnelles und platzsparendes System zur Datenmanipulation und -speicherung in ALGOL 60 und FORTRAN (2..ergänzte Auflage)

Nr. 7501: R. Buchmann
Zur Theorie der Montage von Programmoduln

Nr. 7502: 7. Jahresbericht des Rechenzentrums (1.1. bis 31.12.1974)

Nr. 7503: H.-D. Sander
BOTRAN, eine Fortran Spracherweiterung durch Code-Prozeduren

Nr. 7504: W. Schelongowski
DIATRACE – Ein System zur interaktiven Assemblerprogrammierung

Nr. 7505: Camici, Prof. Dr. Ehlich, Schürmann
Über ein Programm zur Material- und Vervielfältigungs-Abrechnung

Nr. 7506: Camici, Prof. Dr. Ehlich, Herrmannies
Über ein Programm für die Telefonabrechnung einer Nebenstellenanlage

Nr. 7507: Camici, Prof. Dr. Ehlich, Kipp
Über ein Programm zur Haushaltsführung

Nr. 7508: Camici, Cipa, Prof. Dr. Ehlich
Bericht über ein Programm zu Verwaltung der Studentendaten

Nr. 7509: J. Riege
Zur mehrdimensionalen Spline-Interpolation bezüglich beliebiger linearer Funktionale

Nr. 7601: H. Ehlich, J. Riege u. K-H. Schloßer
Ein Programmsystem zur Ausleihverbuchung und interaktiven Rechnerunterstützung in der allgemeinen Buchbestandsverwaltung
Teil 1: Offline-System

Nr. 7602: M. Rosendahl
BOGOL-STRING, eine flexible Zeichenkettenverarbeitung in ALGOL60
2. erweiterte und geänderte Version

Nr. 7603: R. Buchmann, M. Rosendahl
BOGOL-TAS, eine Spracherweiterung von ALGOL60 durch Codeprozeduren zur Systemprogrammierung

Nr. 7604: R. Buchmann
AUFBEREITE, ein universell einsetzbarer Dateiänderungsoperator für verschiedene Datei- und Dialoggerättypen und/oder Betriebsarten

Nr. 7605: 8. Jahresbericht des Rechenzentrums (1. Januar bis 31. Dezember 1975)

Nr. 7606: R. Buchmann, H. Wupper
BO&ZEICHNE, Vorschlag zur geräteneutralen Graphik

Nr. 7701: Camici, Ehlich, Kipp, Wiedemann
Inventarisierungsprogramm

Nr. 7702: K.A. Görg, W. Stark, R. Wojcieszynski
Verfahren zur Berechnung der Strömung durch eine Drosselstelle ohne Speicherwirkung

Nr. 7703: H. Wupper
ALGOL68 Eine Einführung in die Programmierung für Anfänger

Nr. 7704: Camici, Prof. Dr. Ehlich, Volmer, Wiedemann
Aufbau und Verwendung der Personaldatei der Ruhr-Universität Bochum

Nr. 7705: M. Rosendahl
AIDA, Handbuch für den Benutzer

Nr. 7706: V. Riedel, K.-H. Schloßer
Ein Programmsystem zur Ausleihverbuchung und interaktiven Rechnerunterstützung in der allgemeinen Buchbestandsverwaltung
Teil 2: On-line-System

Nr. 7707: M. Peuser, H. Wupper
Ein geräteneutrales und rechnerneutrales System zur graphischen Ausgabe

Nr. 7708: H. Wupper
ERZEUGE, eine Erweiterung der Kommandosprache

Nr. 7709: A. Heidt, G. May
ADWAND ein System zur Bearbeitung von Analogdaten an TR86 und TR440

Nr. 7710: R. Buchmann
BO&PAGING Der virtuelle Kernspeicher für den TR440