KERNFORSCHUNGSZENTRUM

KARLSRUHE

Oktober 1970

KFK 1305

Datenverarbeitungszentrale

Eine virtuelle Speicherorganisation für Prozeßrechner mit mittlerer Wortlänge

J. Nehmer

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

Oktober 1970 KFK 1305

Datenverarbeitungszentrale

Eine virtuelle Speicherorganisation für Prozeßrechner mit mittlerer Wortlänge

von

J. Nehmer

Vorgetragen auf der NTG-Fachtagung über
"Rechnerstrukturen und Betriebsprogrammierung"
5. - 7. Okt. 1970, Erlangen

Gesellschaft für Kernforschung m.b.H., Karlsruhe

Inhalt:

Kurzfassung

Zukünftige hochintegrierte Prozeßrechensysteme sind durch eine ausgeprägte interaktive Komponente zwischen dem prozeßgekoppelten DV-System und aktiv handelnden Benutzern gekennzeichnet (z.B. Laborautomation, Verkehrsprozeß, medizinische Diagnostik). Dadurch werden neben der bestehenden Forderung nach hoher Reaktionsschnelligkeit wesentlich erweiterte Anforderungen an Betriebsverhalten und Betriebssicherheit zukünftiger Prozeßrechensysteme gestellt, die bei der Konzeption der Rechnerstruktur berücksichtigt werden müssen.

Der Bericht beschreibt ein virtuelles Speicherkonzept, das wesentliche Verbesserungen für die genannten Systemeigenschaften bringt. Dabei ist aus ökonomischen Gründen davon auszugehen, daß auch zukünftige hochintegrierte Prozeßsysteme mit mittleren Wortlängen (etwa 24 Bits) aufgebaut werden, und daß die im Befehlswort mitgeführten Adressenkomponenten vergleichsweise kurz gehalten werden müssen. (Op-Code, Modifier, Adressmode und Adresskomponente sollten in einem Speicherwort untergebracht werden.)

Das vorgelegte Konzept schlägt eine Gliederung der virtuellen Adresse in Bank-, Segment- und Seitenadresse derart vor, daß durch die Wahl der Bankadresse die Größe des Segmentes bestimmt wird (gleitende Segmentierung). Dadurch können eine große Zahl (beispielsweise über 1000) gegeneinander verriegelte Segmente sehr unterschiedlicher Größe im virtuellen Speicherraum untergebracht werden.

Abstract

Future high integrated computer systems for process control will have to take into account the interactive communication requirement between the data processing system and its users (laboratory automation, traffic control, medical diagnostics, etc.).

Apart from the time requirements in real time computer systems reliability and availibility must be improved essentially.

In this publication a new virtual memory concept is described which is characterized by considerable improvements for the mentioned system properties. It is supposed that future data processing systems for process control applications will also have medium word sizes for economical reasons (about 24 bits). Therefore, the address portion of instructions is relatively short.

In this concept a virtual address is devided into four parts - a bank-, segment-, page- and lineaddress. In each bank of virtual memory it is possible to choose another segment size and thereby one can obtain a great number of protected segments.

1. Einführung

Als Prozeßrechenanlagen werden gegenwärtig vorwiegend kleinere DV-Systeme mit einfacher Rechnerarchitektur, kurzen bis mittleren Wortlängen und wenig flexiblen "zugeschnittenen" Betriebsorganisationen und festen Programmausrüstungen im Anwenderbereich eingesetzt.

Die zunehmende Automation komplexer technisch-physikalischer Prozesse, in denen neben der Führung von Abläufen der laufenden Interaktion des prozeßgekoppelten DV-Systems mit aktiv handelnden Menschen eine zentrale Bedeutung zukommt (Laborautomation, Verkehrsprozeß, medizinische Diagnostik am lebenden Organismus usw.), stellt wesentlich erweiterte Anforderungen an Betriebsverhalten, Programmierbarkeit, Reaktionsschnelligkeit und Betriebssicherheit zukünftiger Prozeßrechensysteme.

Im Spektrum der DV-Anwendungen rücken Prozeßrechensysteme damit in die Nähe von Teilnehmersystemen, ohne dabei jedoch ihre spezielle operative Charakteristik zu verlieren. Man kann daher voraussetzen, daß die Organisation der Aufgaben in Form von Tasks auch in einem Prozeßrechner ein geeignetes Mittel darstellt, um das geforderte Betriebsverhalten zu realisieren [1].

Die Rechnerarchitektur muß diesen neuen betriebsorganisatorischen Gesichtspunkten natürlich angepaßt werden. Die dabei bezüglich einer Speicherorganisation zu lösenden Probleme sind allen multiprogrammierten DV-Systemen gemein und können im wesentlichen in vier Punkten zusammengefaßt werden, die zuerst von DENNIS [2] formuliert wurden:

a) Prozeduren und Daten müssen im Arbeitsspeicher der DV-Anlage voll verschieblich sein (relocatable).

b) Die gemeinsame, parallele Benutzung von Prozeduren und Datenfeldern durch viele Tasks (shared data and procedures) muß durch spezielle Einrichtungen des Rechners sichergestellt werden.

c) Datenbereiche sollten während eines Programmlaufs expandieren können, ohne daß eine Verschiebung der bereits existierenden Bestände im Speicher erforderlich wird.

---

* zum Druck eingereicht am 31.10.70

d) Ein Speicherschutzmechanismus muß nicht autorisierte Zugriffe auf fremde Prozeduren und Datenfelder unterbinden. Erlaubte Zugriffe sollten dabei ferner mit abgestuften Zugriffsprivilegien versehen werden können.

Diese grundsätzlichen Entwicklungskriterien bilden die Basis, auf der nun Lösungsansätze für eine Speicherorganisation von Prozeßrechnern diskutiert werden. Sie wurden unter dem Gesichtswinkel ausgewählt, daß sie die geforderte Reaktionsschnelligkeit nicht vermindern und außerdem den harten wirtschaftlichen Randbedingungen genügen, die an derartige Anlagen gestellt werden.

2. Relokation und Adressenaufweitung

Das Relokationsproblem im Arbeitsspeicher wird in neuen DV-Maschinen zunehmend durch virtuelle Adressierungsmechanismen gelöst. Der größte Teil der mit Hilfe des Betriebssystems vom Programmierer direkt oder indirekt abzuwickelnden Speicherverwaltungsaufgaben wird im virtuellen Raum durchgeführt. Virtuelle Speicher erlauben dabei aufgrund ihrer umfangreichen Kapazitätsreserven eine großzügige Belegung, ohne daß dort eine laufende Verschiebung von Programmen und Daten erforderlich wird.

Die Vergabe des Arbeitsspeichers dagegen wird unsichtbar für den Programmierer zentral in betriebssysteminternen Zuordnungstabellen registriert, durch die Teile des virtuellen Speichers auf den reellen Arbeitsspeicher der Maschine abgebildet werden. Die im virtuellen Bereich existierenden Adressbeziehungen und -belegungen bleiben von diesen Manipulationen jedoch unberührt.

KRÜGER [3] weist darauf hin, daß die Beschränkung auf einen systemweiten virtuellen Speicher im Gegensatz zur Verwendung mehrerer taskzugehöriger virtueller Speicher (IBM 360/67 [4,5], MULTICS [6,7]) für Prozeßautomationssysteme mit ihrer speziellen Betriebscharakteristik

zusätzliche Vorteile bringt, die in erster Linie zu einer erhöhten Umschalt- und Ablaufgeschwindigkeit von Programmen führen. Da die Reaktionsschnelligkeit ein entscheidender Faktor für die Beurteilung eines Prozeßrechensystems ist, beschränken wir uns deshalb bei den folgenden Untersuchungen auf die Existenz eines virtuellen Speichers, der allen Speicheranforderungen aus dem System- und Problembereich genügt.

Die Größe des virtuellen Systemadressenraumes sollte dabei in der Regel durch den über ein volles Speicherwort adressierbaren Bereich fixiert werden. Bei Prozeßrechnern mit sehr kurzer Wortlänge (8 - 16 Bit) müßten unter Umständen auch Mehrwortadressbildungen zugelassen werden.

Auf eine weitere vorteilhafte Eigenschaft virtueller Adressierungssysteme soll noch kurz hingewiesen werden. Durch erweiterten Operationscode, zusätzliche Adressfelder für Arbeits- und Indexregister sowie Adress-Modifier wurde der Adressteil eines Befehlswortes allmählich eingeengt, so daß eine vollständige Adressierung, selbst des Arbeitsspeichers, durch ihn in vielen Fällen nicht mehr möglich ist. Die erforderliche Aufweitung der Befehlswortadresse stellt somit insbesondere bei kleinen bis mittelgroßen DV-Maschinen eine unabhängig vom Relokationsproblem zu lösende Aufgabe dar, die von virtuellen Adressierungssystemen nebenbei mitgelöst wird.

3. Ablaufinvarianz von Prozeduren durch funktionelle Adressbasen

Legt man für die weiteren Untersuchungen die Existenz eines virtuellen Speichers zugrunde, dann stellt seine wirtschaftliche Nutzung eine der vorrangig zu lösenden Aufgaben dar.

Eine wirksame Methode zur Speicherplatzoptimierung besteht darin, durch geschickte Programmorganisation Kopien von öffentlichen Prozeduren im Speicher zu vermeiden, die gleichzeitig von mehreren Tasks als Betriebsmittel benutzt werden. Das setzt voraus, daß ein Programm in die taskspezifischen Variablen und den ablaufinvarianten Prozedurkörper getrennt

wird. Erst dann ist es möglich, eine Prozedur "reentrant" aufzurufen, in dem ihr mehrere Variablenbereiche unterschiedlicher Tasks quasi-parallel zugeordnet werden.

Die Umschaltung von einem Variablenbereich einer Task auf einen anderen bei völliger Invarianz der reentrant benutzten Prozedur erfordert die Zwischenschaltung einer Adressbasis, mit der alle im Prozedurkörper existierenden Adressen verknüpft werden. Sie wird im folgenden als V-Basis bezeichnet. Jede Task, die eine Prozedur als Betriebsmittel benutzt, hat ihre eigene, vom Betriebssystem im Rahmen der virtuellen Speicherverwaltung zugewiesene V-Basis. Ihre Funktion wird in Bild 1 verdeutlicht.

Der Prozedurkörper enthält den eigentlichen Programmcode und die Zahlenkonstanten eines Programmes. Über den Befehlszähler (Program counter), der eine "mitlaufende" Basis darstellt, können normalerweise alle Grössen der Prozedur angesprochen werden. Es hat sich jedoch für bestimmte Fälle als zweckmäßig erwiesen (z.B. für SWITCH-Felder), zusätzlich eine feststehende Prozedurbasis, die P-Basis einzuführen, die ständig auf den Beginn der aktiven Prozedur zeigt. In der Praxis wird man für die P-Basis und den Befehlszähler, hier als PC-Basis bezeichnet, nur ein Register im Rechnerkern ausbilden, da die P-Basis leicht aus der PC-Basis abgeleitet werden kann. Im Bild 2 sind diese beiden Adressbasen zusätzlich zur V-Basis aufgeführt.

Eine Art von Programmgrößen wurde bisher noch nicht erwähnt. Es handelt sich um die Adresskonstanten, die auf getrennt übersetzte (externe) Prozeduren sowie außerhalb des Programmes vereinbarte (externe) Datenfelder hinweisen. Sie werden ebenso wie die Variablen von dem Prozedurkörper abgespalten, um dessen Ablaufinvarianz vom Zeitpunkt der Compilierung zu erhalten. Diese externen Adressenbeziehungen sind zur Compilezeit nämlich nicht definiert und können erst bei der Übernahme einer Prozedur in den virtuellen Speicher oder zu einem späteren Zeitpunkt aufgelöst werden.

Die Ablage der externen Adressen einer Prozedur in den Variablenbereich der angeschlossenen Task, die prinzipiell möglich wäre, erscheint nicht

sehr sinnvoll, da bei einer Reentrantbenutzung der Prozedur überflüssigerweise in jedem der taskspezifischen Variablenbereiche eine Kopie dieser Externbezüge angelegt würde. Ein weiterer Nachteil entstünde dadurch, daß nach dem Ableben einer Task alle eingerichteten Externadressen einer Prozedur verloren gingen und mit jeder neu kreierten Task erst wieder aufgebaut werden müßten. Bei den weitgehend festen Programmausrüstungen von Prozeßrechensystemen erscheint es vorteilhafter, die Adresskonstanten nicht taskgebunden, sondern prozedurbezogen anzulegen, um diese unnötige Verwaltungsarbeit einzusparen.

Durch die Abtrennung der Externbezüge vom Prozedurkörper und ihre Ablage in einem eigenen, über die E-Basis adressierbaren Speicherbereich, wird einerseits die Ablaufinvarianz einer Prozedur vom Zeitpunkt ihrer Übersetzung an garantiert und die Reentrantfähigkeit auch auf die externen Adresskonstanten ausgedehnt.

Die Adressierungsumgebung einer aktiven Task wird dann durch vier Adressierungsbasen bestimmt, die jeweils eine vollständige virtuelle Adresse aufnehmen. Der Adress-Modifier im Befehlswort enthält die verschlüsselte Vorschrift, welche der 4 Basisadressen mit der Adresskomponente des Befehlswortes zu einer kompletten virtuellen Zieladresse verknüpft werden sollen.

Der mnemotechnische Code

E, LØAD 23

würde z.B. bedeuten, daß der Inhalt der 23. Zelle im E-Bereich in ein Arbeitsregister zu laden ist.

Dagegen bedeutet der Befehl

V, STØRE 123,

daß der Inhalt des Arbeitsregisters in die 123. Speicherzelle des V-Bereiches abzuspeichern ist.

Durch diesen ersten Adressierungszyklus, der in den beiden Beispielen demonstriert wurde, können nur Teilbereiche des gesamten virtuellen Speichers mit einem Umfang bis zum maximalen Inhalt der Befehlswortadresse (BA) erschlossen werden.

Der Zugriff auf beliebige Speicherzellen im virtuellen Raum ist nur über Ersetzungsoperationen möglich. Dabei werden die im ersten Adressierungszyklus gewonnenen virtuellen Adressen durch die Inhalte im E- und V-Bereich ersetzt, auf die sie zeigen. Der für diesen zweiten Adressierungszyklus notwendige Ersetzungsschritt wird dabei ebenfalls durch einen speziellen Code im Adress-Modifier-Feld eines Befehlswortes ausgelöst.

Die Befehlsfolge

```
PC, LØAD  53
VI, STØRE 27
```

lädt die 53. relativ zum Befehlszähler abgelegte Zahlenkonstante in ein Arbeitsregister und speichert sie an diejenige virtuelle Adresse, die in der 27. Zelle des V-Bereiches steht.

4. Segmentierung des virtuellen Speichers

Nach den bisherigen Ausführungen umfaßt eine Task mindestens drei getrennte Speicherbereiche (P,E,V-Bereich), die wegen der geforderten Reentrantfähigkeit der Prozeduren normalerweise keinen zusammenhängenden Abschnitt im virtuellen Speicher belegen können. Durch diese starke Untergliederung eines Programmes sowie durch die mögliche hohe Zahl parallel existierender Tasks entstehen unter Umständen sehr viele unterschiedliche Prozedur- und Datenbereiche, die möglichst einfach und raumsparend im virtuellen Speicher verwaltet werden müssen.

Ein häufig angewendetes Mittel zur Bildung definierter Abschnitte im virtuellen Speicher, die der Aufnahme logisch zusammenhängender Informationsmengen dienen, ist das Segmentverfahren. Der virtuelle Speicher wird dabei in Segmente - u.U. unterschiedlicher Länge - untergliedert,

die ausgezeichnete, feste Positionen in diesem Adressenraum einnehmen. Sie sind durch Segmentbasisadressen adressierbar, die als Teilkomponenten in jeder virtuellen Adresse enthalten sind. Das hat den Vorteil, daß die Segmente durch einfache Interpretation aus einer virtuellen Adresse identifiziert und in ihrem Gültigkeitsbereich definiert werden. Zusätzliche Angaben über Beginn und Länge der Segmente in Segment-Descriptor-Tabellen sind daher nicht mehr notwendig.

Bei der Wahl der Segmentgrößen ist zu beachten, daß durch expandierende Datenfelder, die in ihrem Gültigkeitsbereich nicht festliegen, ein Bedarf an sehr großen Segmenten besteht, damit Verschiebungen im virtuellen Speicher aufgrund von Segmentüberschreitungen vermieden werden. Bei dem bekannten Verfahren der starren Segmentierung (MULTICS, IBM 360/67) muß deshalb die feste Segmentgröße an der maximalen Informationslänge orientiert werden (in diesen beiden Systemen z.B. 256 K!).

A. BATSON u.a. [8] ermittelten dagegen aufgrund von Realzeitmessungen an einer B 5500, daß ca. 60 % aller im System geführten Segmente weniger als 40 Worte umfassen. Obwohl die Resultate stark anlagenabhängig sind, spiegeln sie doch die Tendenz im Bedarf an überwiegend kleinen Segmenten wieder, die auch für den Bereich der Prozeß- und Laborautomation bestätigt werden kann. DV-Systeme mit starr segmentiertem Speicher müssen deshalb den überwiegenden Teil des verfügbaren virtuellen Adressenraumes verschenken.

Die Festlegung auf einen virtuellen Speicher, der durch die Beschränkung auf virtuelle Adresslängen von 24 - 32 Bit zusätzlich eingeengt wird, legt eine bessere Nutzung des verfügbaren Raumes nahe. Eine Möglichkeit besteht in der Bereitstellung variabel langer Segmente, wobei eine Ausweitung der virtuellen Adresse durch eine Längenangabe vermieden werden soll.

In dem Vorschlag der Abbildung 3 wird eine virtuelle Adresse in vier Teilkomponenten, die Bank (Länge b)-, Segment (Länge s)-, Page (Länge p)- und Zeilenadresse (Länge z) untergliedert.

Durch die Bankadresse wird im virtuellen Speicher eine von 2^b Bänken adressiert und die relative Adresse innerhalb der Bank durch die verbleibenden Adresskomponenten (Segment-, Page-, Zeilenadresse) bestimmt.

Das Besondere dieses Verfahrens besteht nun darin, daß die Segmentlängen für jede Bank getrennt festgelegt werden können. Ein Segmententschlüssler speichert die Zuordnungen zwischen allen Bänken und den zugehörigen Segmentgrößen. Mit jeder neu gebildeten virtuellen Adresse wird über den Segmententschlüssler die gültige Segmentgröße in der adressierten Bank bestimmt. In der Abbildung wird dieser Vorgang durch die Stellung eines Zeigers symbolisch dargestellt, der auf die Grenze zwischen Segment- und Page-Adresse dieser Bank zeigt (Voraussetzung s + p = const.).

Geht man davon aus, daß der virtuelle Adressenraum durch einen Paging-Mechanismus auf den reellen Speicher abgebildet wird, dann sind kleinere Segmente als die Länge einer Seite (2^z) sinnlos. Nach oben wird der Umfang der Segmente durch die Bankgröße begrenzt (2^{w-b}). Die in einer Bank wählbare, konstante Segmentgröße kann dann alle 2-er Potenzen zwischen diesen beiden Extremwerten annehmen.

Die Vorteile des geschilderten Segmentierungskonzeptes sollen abschliessend noch einmal zusammengefaßt werden:

Durch die Einführung einer Bankadresse als zusätzliche Adressierungskomponente einer virtuellen Adresse gewinnt man einen Freiheitsgrad, der gegenüber den bekannten Lösungen der starren Segmentierung die Einrichtung von Segmenten gleitender Länge gestattet, ohne den benötigten Adressenumfang aufzuweiten. Durch die bessere Anpassungsfähigkeit der Segmentgrößen an die tatsächliche Informationsstruktur kann der Speicherverschnitt im virtuellen Raum erheblich reduziert und auch in kleinen DV-Maschinen mit ihrer begrenzten Wortlänge eine große Zahl variabel langer Segmente gebildet werden.

Dies soll an einem Beispiel kurz demonstriert werden:

Bei starrer Segmentierung des virtuellen Speichers und einer Segmentgröße von 256 K (18 Bit) entstehen durch eine 24-Bit-Adresse 64 Segmente. Mit dem Verfahren der gleitenden Segmentierung können dagegen bei gleicher Wortlänge (24 Bit) und einer Zeilenadresskomponente von 8 Bit bis zu 65536 Segmente kreiert werden, die dann eine Länge von 256 Worten haben.

Dabei wird vorausgesetzt, daß die Segmentverteilung in den Bänken dynamisch dem jeweiligen Anwendungsprofil angepaßt wird. Erste Simulationsergebnisse mit einem betriebsorganisatorisch einfach zu handhabenden Verfahren für eine adaptive Segmenteinstellung zeigen, daß der virtuelle Speicher im Mittel zu etwa 50 % ausgenutzt werden kann.

5. Kontrolle des Speicherzugriffs

Geht man von einem segmentierten virtuellen Speicher aus, dann erfolgt die Vergabe von zusammenhängenden Abschnitten im virtuellen Speicher segmentweise durch das Betriebssystem. Ein Segment nimmt dabei eine Prozedur oder einen geschlossenen Datenbereich auf und bildet die natürliche Einheit des Speicherschutzes. Die unter den Adressbasen angelegten Speicherbereiche (P-,E-,V-Bereiche) stellen dann unterschiedliche Segmente des virtuellen Speichers dar.

In multiprogrammierten DV-Systemen ist es erforderlich, den Zugriff von Tasks auf für sie zugelassene, d.h. vom Betriebssystem freigegebene Segmente zu beschränken, da andernfalls unbefugte Mitbenutzung oder gar Überschreiben taskfremder Prozeduren- und Datenbereiche möglich wäre.

Die gemeinsame, überlappte Benutzung derselben Prozedur- und Datensegmente durch verschiedene Tasks darf dadurch jedoch nicht ausgeschlossen werden, wobei die Möglichkeit bestehen muß, die Tasks mit abgestuften Zugriffsrechten für die Segmente ihres Wirkungsbereiches auszustatten.

Die Durchführung der dazu erforderlichen Kontrollen ist ohne ständige Interaktion mit dem Betriebssystem durch Dezentralisation des Speicherschutzes zu erreichen. Das bedeutet, daß im virtuellen Speicher taskspezifische Teilräume auf der Basis der vorhandenen Segmentstruktur aufgebaut werden müssen, die gegenüber allen anderen verriegelbar sind. Die überlappte Benutzung von "Common"-Segmenten kann dadurch realisiert werden, in dem sie gleichzeitig mehreren Task-Räumen zugeordnet werden.

Überträgt man diesen Lösungsansatz auf das diskutierte Adressierungsmodell, dann wird ein taskspezifischer Teilraum durch mindestens 3 Basis-Segmente (P-,E-,V-Segment) gebildet.

Dieser Mindest-Adressenraum einer Task kann jedoch durch eine unbestimmte Anzahl externer Segmente erweitert werden, die durch virtuelle Verweisadressen im E- oder V-Bereich mit den Basissegmenten verbunden sind. Die Abbildung 4 zeigt, wie dadurch eine Baumstruktur von miteinander verkoppelten Segmenten entsteht, die in ihrer Gesamtheit den Teilraum einer Task bilden.

Eine wirksame Überwachung von Zugriffsbeschränkung einer Task auf zugelassene Speichersegmente kann nur dann gewährleistet werden, wenn

a) eine Task nicht selbständig virtuelle Segment-Adressen in ihrem Teilraum produzieren kann und
b) zugelassene virtuelle Adressen nur innerhalb des Segmentes, auf das sie verweisen, gültig sind.

Die Einhaltung dieser Bedingungen kann per Hardware automatisch überprüft werden, wenn jede Speicherzelle des virtuellen Raumes eine Datentypkennung durch ein zusätzliches Bit erhält. Im gesetzten Zustand weist es den Inhalt als gültige virtuelle Adresse aus, über die Adressoperationen ausgeführt werden dürfen, andernfalls wird der Speicherinhalt als "Nicht-Adresse" interpretiert und darf dann nicht als Indexzelle benutzt werden. Für den Anwendungsprogrammierer ist dieses Bit unerreichbar und nur durch einen privilegierten Systembefehl modifizierbar.

Bei Transporten von Speicherinhalten in Arbeitsregister des Rechners wird das Adressbit als Typenkennung den jeweiligen Akkumulatoren vorangestellt und veranlaßt im gesetzten Zustand (Adresse) die Überwachung der Segmentadresse (Bild 5).

Zu diesem Zweck muß der in Abschnitt 4 eingeführte Segmententschlüssler als programmierbarer Hardware-Baustein ausgebildet werden. Aus jeder neu gebildeten virtuellen Adresse, die in irgendeinem Register des Rechnerkerns steht, wird über den Segmententschlüssler automatisch die gültige Segmentadresskomponente ausgeblendet und daraufhin überwacht, daß sie sich während der Ausführung einer Operation nicht verändert.

Für das Rechnen mit Adressen können verbotene, bedingt zulässige und generell zulässige Operationen unterschieden werden. Verbotene Operationen mit Adressen sowie Überschreiten der Zulässigkeitsbedingungen bei bedingt zulässigen Operationen haben einen Adressenalarm zur Folge und lösen außerdem die zwangsweise Umsetzung der Datentypkennung des betroffenen Registers auf den Typ "Nicht-Adresse" aus. Dies hat den Sinn, Ergebnisse von Adressrechnungen, die gegen die Schutzbestimmungen verstoßen, nicht aus den Registern in den Variablenbereich einer Task abzuspeichern.

Verbotene Operationen sind

a) Shift- und logische Operationen
b) Addition zweier Adressen
c) Multiplikation, Division,

da die Segmentadresskomponente des Resultates in der Regel dabei verändert wird.

Bedingt zulässige Operationen sind:

d) Addition einer Adresse und einer Zahl
   und
e) Subtraktion einer Zahl von einer Adresse.

Die Zulässigkeitsbedingung wird verletzt, wenn der Segmentadressteil der Ergebnisadresse durch die Operation geändert wird, d.h. die zulässige Segmentgrenze überschritten wird.

Eine generell erlaubte Operation stellt die

f) Subtraktion zweier Adressen dar.

Da das Ergebnis in jedem Fall ein Relativwert ist, wird dabei aber die Datentypkennung im Resultatregister gelöscht (Kennung "Nichtadresse").

Ein weiteres Problem stellt die Unterscheidbarkeit von taskspezifischen Zugriffsprivilegien für die Segmente eines Teilraumes dar. Dabei muß es auch möglich sein, an verschiedene Tasks unterschiedliche Zugriffsrechte für ein COMMON-Segment zu vergeben, das allen taskspezifischen Teilräumen überlappt angehört. Man erreicht dies durch eine Zugriffskennung (Z), die jeder virtuellen Adresse zugeordnet wird. Bei fester Wortlänge muß die virtuelle Adresse um den für die Zugriffskennung benötigten Platz verkleinert werden. Vor Ausführung jeder Operation wird dann geprüft, ob sie mit der zulässigen Zugriffsart in dem adressierten Segment verträglich ist. Bei positivem Ausgang der Kontrolle wird die Operation freigegeben, andernfalls erfolgt ein Fehleralarm.

Eine implizite Zugriffsfestlegung kann zusätzlich durch ein Schreibverbot im P-Segment sowie durch ein Schreib- und Execute-Verbot im E-Segment jeder Task getroffen werden.

Eine Bemerkung ist noch über die Verwaltung der Speichersegmente des virtuellen Adressenraumes erforderlich:

Bei dynamischer Vergabe von "leeren" Segmenten an aktive Tasks muß darauf geachtet werden, daß die Speicherinhalte vor Bereitstellung im Arbeitsspeicher den definierten Zustand "Nicht-Adresse" erhalten. Andernfalls ist eine Zuweisung undefinierter, d.h. von vorhergehenden Programmläufen existierenden Adressen an Adressvariable möglich, die unkontrollierte Speicherzugriffe zur Folge haben können.

Unter Beachtung dieser Randbedingung erlaubt das vorgestellte Speicherschutzkonzept den taskbezogenen Aufbau von sehr vielen gegeneinander verriegelbaren und sich beliebig überlappenden Teilräumen im virtuellen Speicher, die nur durch die maximal mögliche Zahl von Segmenten begrenzt wird.

6. Schlußbemerkung

In dem vorliegenden Bericht wurde lediglich die Basisstruktur für das Adressierungssystem eines Prozeßrechners diskutiert, dagegen wurde nicht auf spezielle Registerstrukturen, Befehlsformate und -klassen sowie andere Rechnerkomponenten wie z.B. das Interruptsystem eingegangen. Diesbezügliche Voraussetzungen (z.B. über die Wortstruktur bei Befehlen) wurden so allgemein wie möglich dargestellt und entsprechen dem derzeitigen Stand der Technik. Die aus den Lösungsansätzen abgeleiteten Hardware-Aufwendungen stellen demnach Minimalanforderungen dar, die für einen effizienten Programmablauf in einem Prozeßrechner unbedingt notwendig erscheinen.

Eine erschöpfende Behandlung aller Auswirkungen des diskutierten Speicherkonzeptes auf - insbesondere die Ausführung von Prozedurwechseln, Parameterübertragungen zwischen Prozeduren, die Realisierung problemorientierter Sprachen sowie die Behandlung von rekursiven Funktionen - war im Rahmen dieses Berichtes nicht möglich. Detailliertere Information über diese mehr sprachorientierten Probleme der Adressierung in Verbindung mit dem vorgestellten Adressierungssystem können jedoch bei [9,10] bezogen werden.

Literatur:

[1] W. Werum, J. Nehmer
Eine Taskorganisation für reaktionsschnelle Prozeßrechensysteme
NTG-Fachtagung über "Rechnerstrukturen und Betriebsprogrammierung", 5.-7. Okt. 1970, Erlangen

[2] Jack B. Dennis
Segmentation and the Design of Multiprogrammed Computer Systems
ACM, Vol. 12, No. 4 (Oct. 1965) pp. 589-602

[3] G. Krüger
Laborautomation und Prozeßrechensysteme
NTG-Fachtagung über "Rechnerstrukturen und Betriebsprogrammierung", 5.-7. Okt. 1970, Erlangen

[4] F.J. Corbato, V.A. Vyssotsky
Introduction and Overview of the MULTICS-System
Proc. Fall Joint Computer Conf., 1965, S. 185 p.p.

[5] Robert C. Daley, Jack B. Dennis
Virtual Memory, Processes and Sharing in MULTICS
ACM, Vol. 11, No. 5 (May 1968) S. 306 p.p.

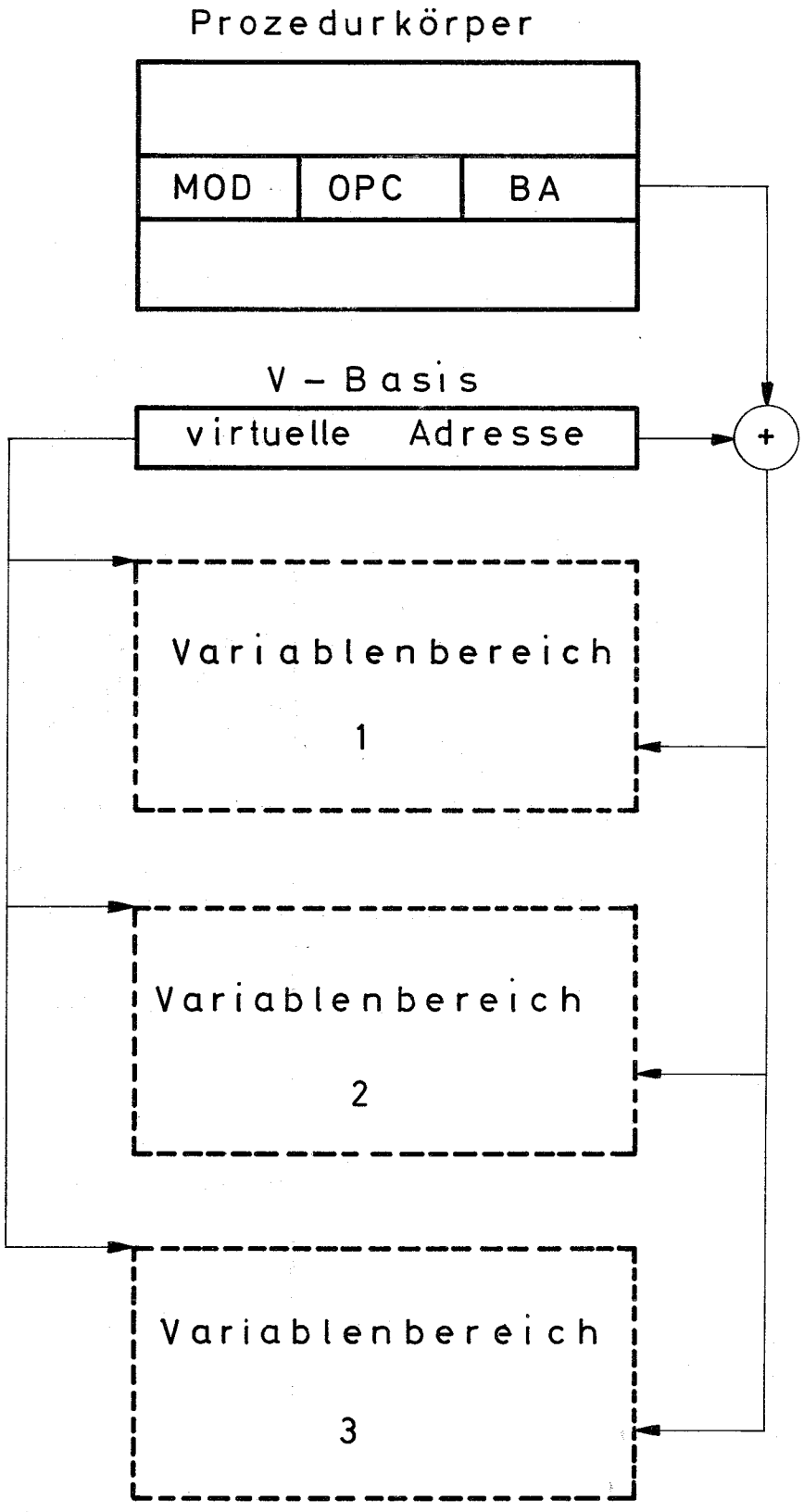
[6] Charles T. Gibson
Time-Sharing in the IBM System/360: Model 67
Proc. Spring Joint Computer Conf., 1966, S. 61 p.p.

[7] Alexander S. Lett, William L. Konigsford
TSS/360: A Timeshared Operating System
Proc. Fall Joint Computer Conf., 1968, S. 15 p.p.

[8] Alan Batson u.a.
Measurements of Segment Size
ACM, Vol. 13, No. 3 (March 1970)

[9] J. Nehmer
Die Abbildung von Blockstrukturen problemorientierter Sprachen auf ein spezielles Adressierungssystem
Kernforschungszentrum Karlsruhe Externer Bericht 19/70-1, November 1970

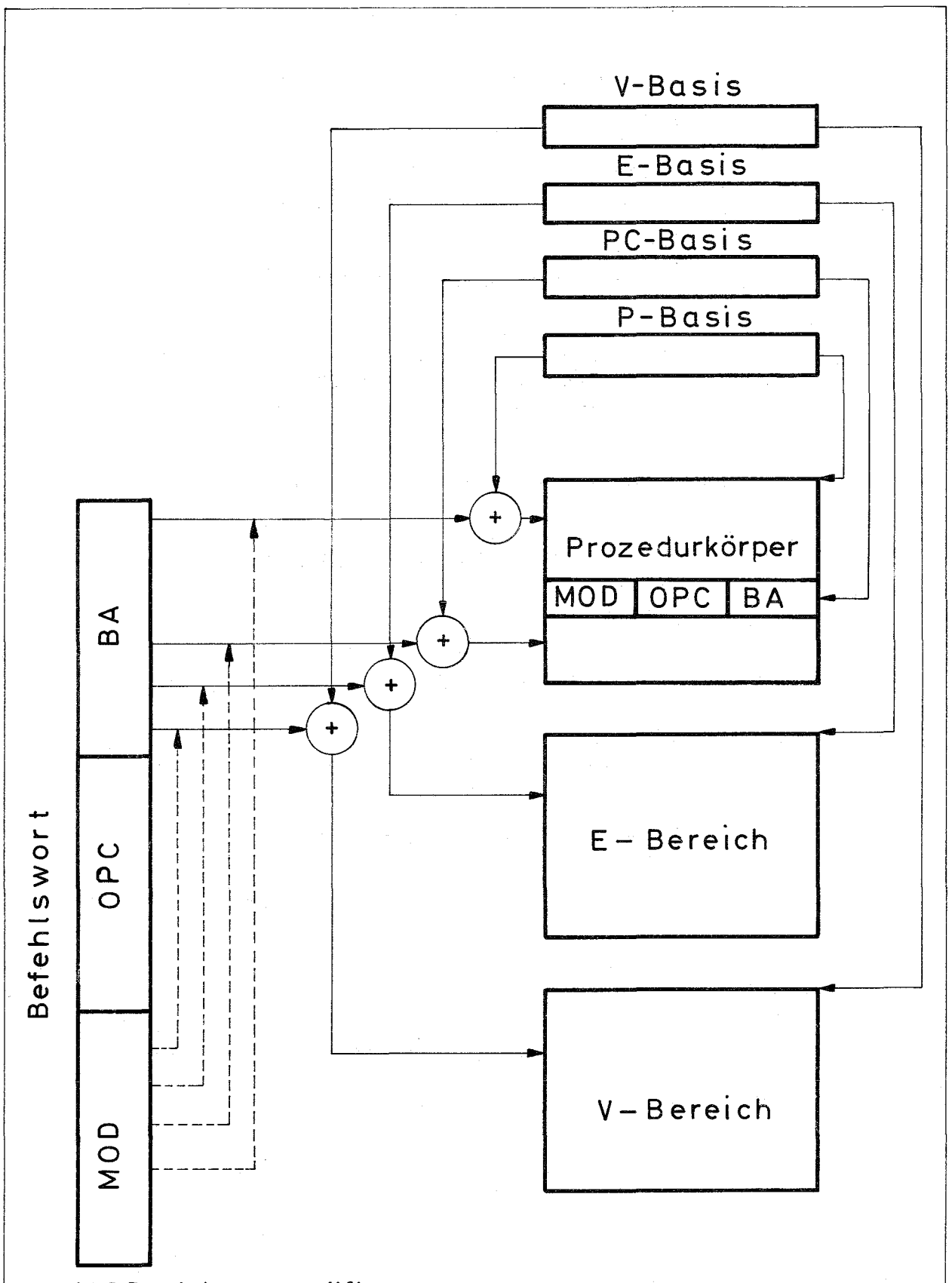
[10] J. Swoboda
Sprachorientierte Rechner-Probleme der Adressierung
Elektronische Rechenanlagen 12 (1970), H.1, S. 26-35



B A Befehlswortadresse
O P C Operationscode

BILD 1

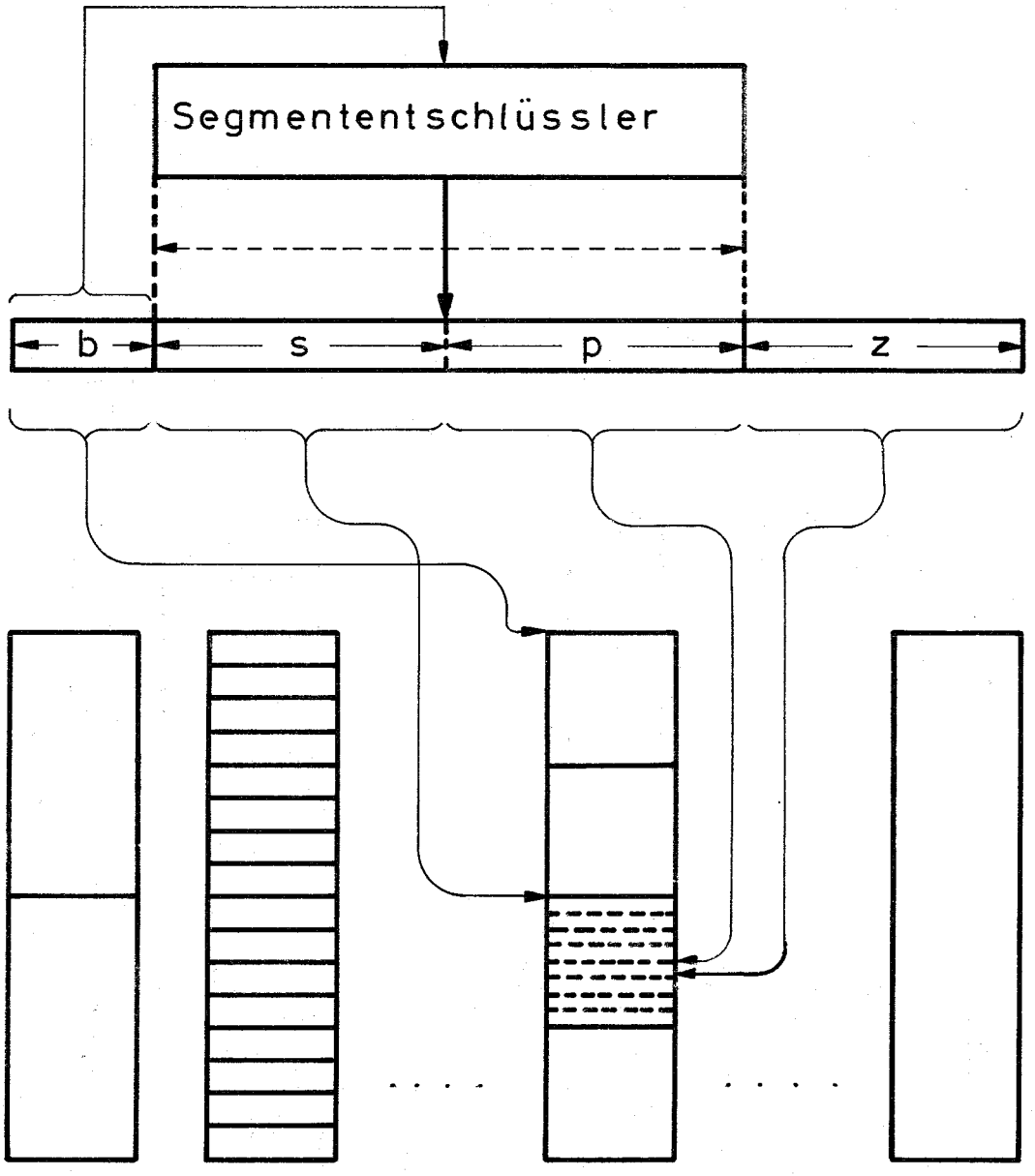
Realisierung von Reentrant-Prozeduren durch Einbeziehung einer Relocationsbasis in den Adressbildungsvorgang



MOD = Adressmodifier
OPC = Operationscode
BA = Befehlswortadresse

BILD 2

Adressierung des virtuellen Speichers in einem aktiven Progr. durch 4 funktionelle Adressbasen



$w = b + s + p + z$
(Wortlänge)

b	Bankadresslänge
s	Segmentadresslänge
p	Pageadresslänge
z	Zeilenadresslänge

BILD 3

Gliederung des virtuellen Speichers

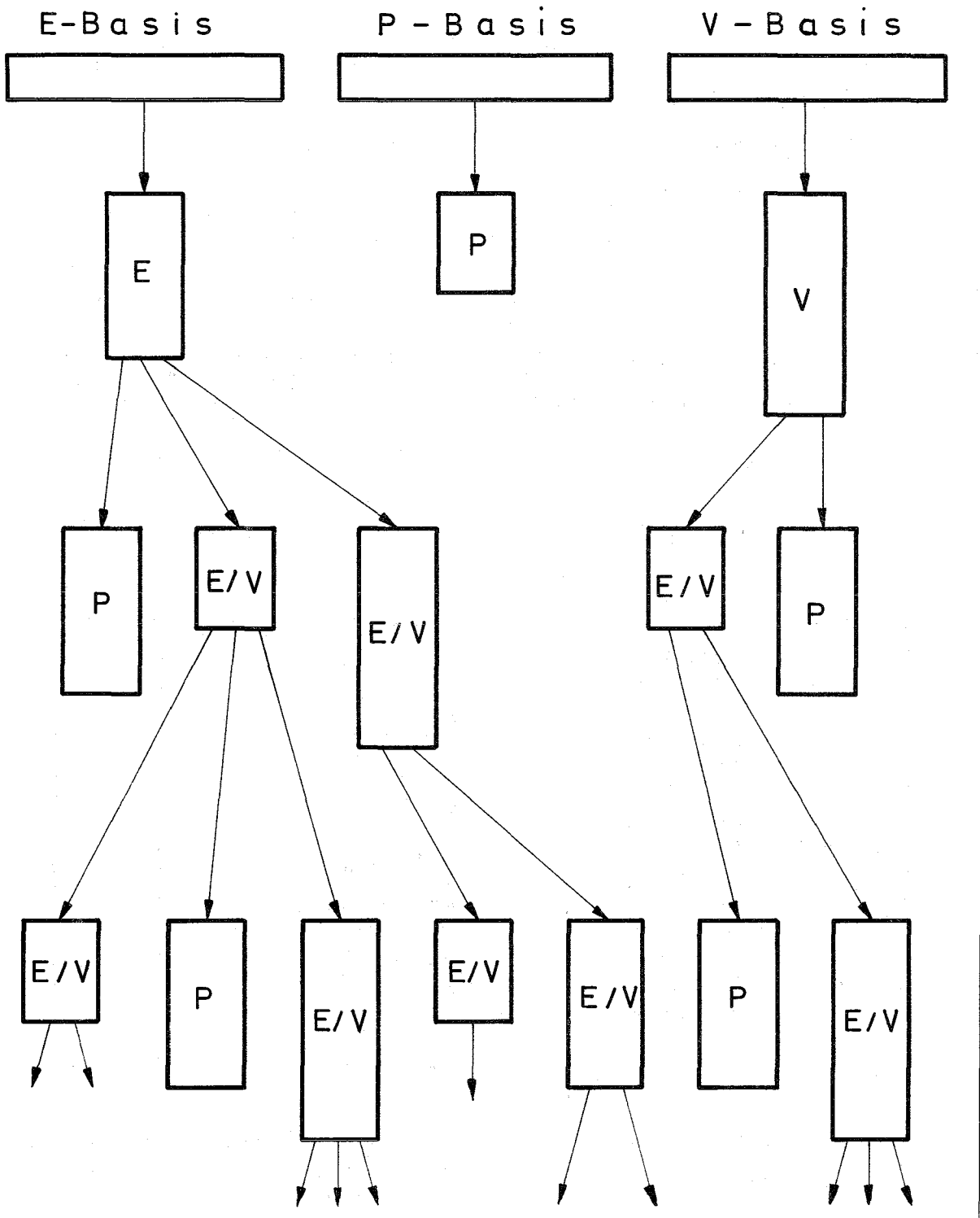


BILD 4

Zusammensetzung eines task-spezifischen Teilraumes durch Segmente des virtuellen Speichers

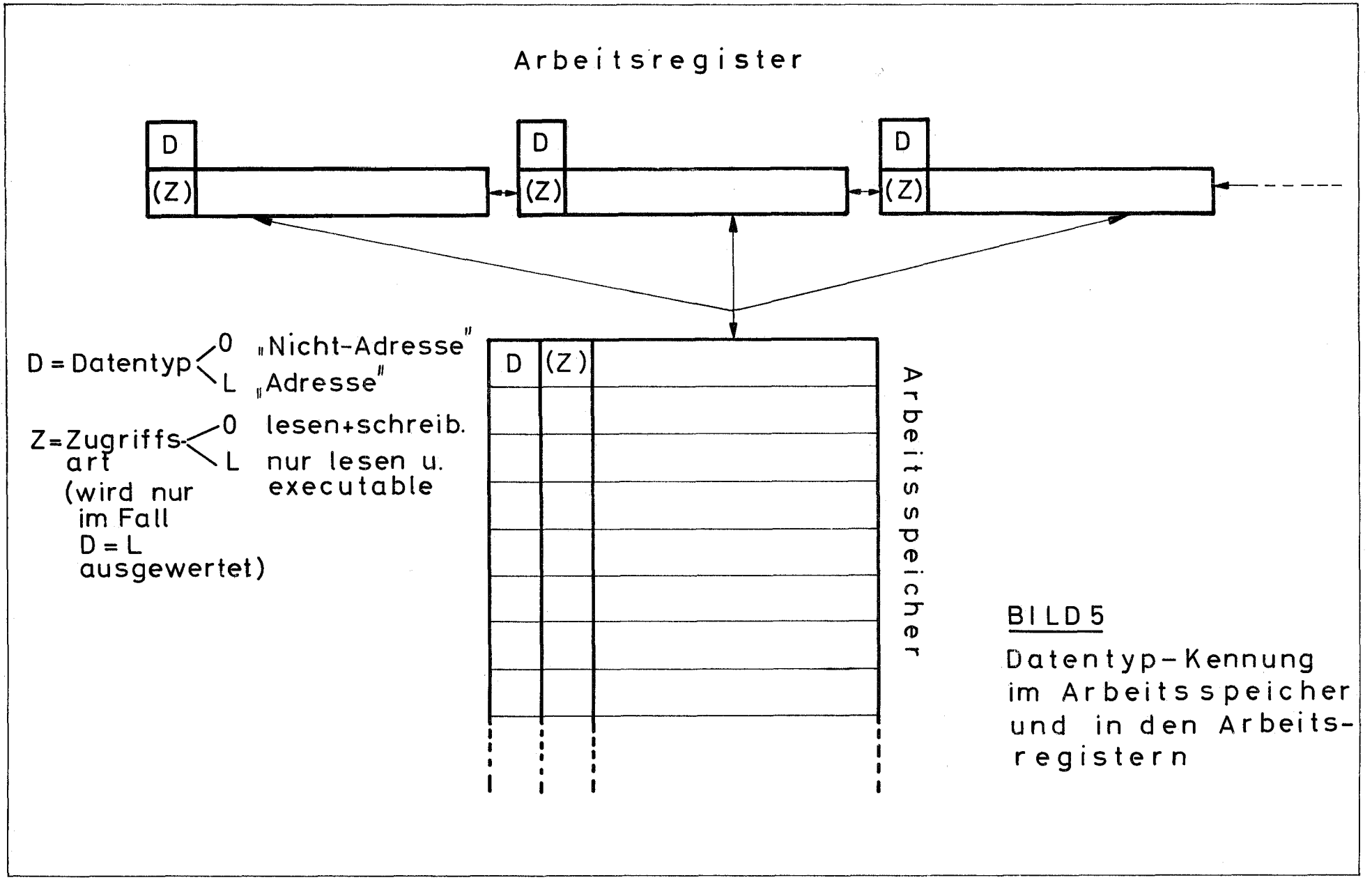


BILD 5

Datentyp-Kennung im Arbeitsspeicher und in den Arbeitsregistern