KAPITEL 1

Einführung in die Speicherorganisation

1.1 Grundsätzliches

Die wichtigsten Charakteristiken einer Computer-Architektur sind

- die Organisation des Speichers
- die Methode des Zugriffs auf Informationen im Speicher.

Der Hauptspeicher eines Computers ist als ein Satz von Speicherplätzen organisiert, die, bei Null beginnend, fortlaufend durchnumeriert sind. Die Nummer, die einem dieser physikalischen Speicherplätze zugeordnet ist, wird ***physikalische Adresse*** und der Satz aller physikalischen Adressen ***physikalischer Adressraum*** genannt.

Dagegen ist eine ***logische Adresse*** eine Adresse, wie sie in einem Befehl vom Programmierer benützt wird.

Die Organisation des physikalischen Adressraums ist durch die Speichertechnologie und ihre Kosten bestimmt. Im Gegensatz dazu sollte die Organisation des logischen Adressraums durch die Struktur des Programms, das im Speicher abläuft, bestimmt sein.

Neuere Speicherarchitekturen zeigen eine Entwicklung in diese Richtung. Anfänglich mußte der logische Adressraum identisch mit dem physikalischen Adressraum sein. Beim 80286 dagegen hat sich die Speicherorganisation den Bedürfnissen der Programmorganisation wesentlich genähert.

In diesem Kapitel sollen einige dieser Entwicklungsschritte erläutert werden.

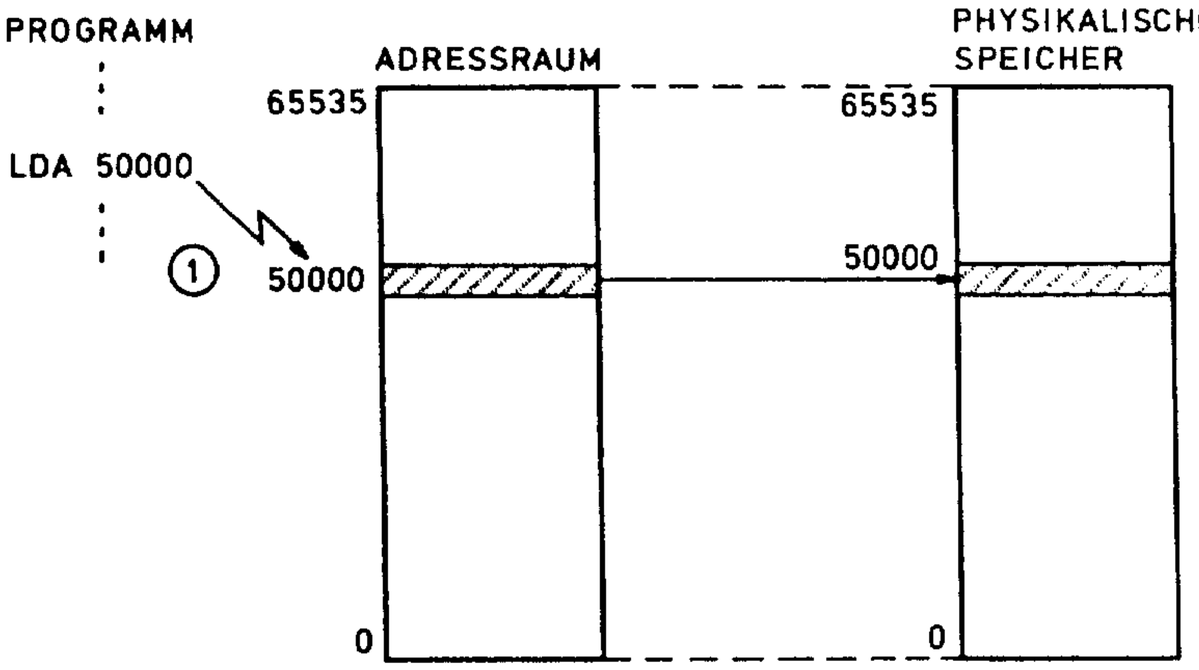
1.2 Linearer Speicher

In ihrer Grundform bildet die Architektur eines Speichers einen zusammenhängenden Adressraum. Bei dieser Art von Architektur beginnen die Adressen bei Null und schreiten in linearer Weise ohne Zwischenräume bis zu einer oberen Grenze fort.

Die Teile eines Programms, das im Normalfall aus mehreren Prozeduren und ihren Daten besteht, sind gewöhnlich in einem zusammenhängenden Adressraum untergebracht. In diesem Fall hat der logische Adressraum die gleiche lineare Organisation wie der physikalische Speicher.

Die einfachste Realisierung eines linearen Adressraums zeigt beispielsweise der 8085. 8085-Programme können 2^{16} (65536) unterschiedliche Adressen generieren. Diese Adressen werden direkt von der Speicher-Hardware benützt, um Daten zu positionieren.

Wie das folgende Bild für einen linearen Speicher zeigt, bezieht sich zum Beispiel der Befehl LDA 50000 auf die Adresse 50000 und liest ein Datum von dieser physikalischen Speicherposition ①.

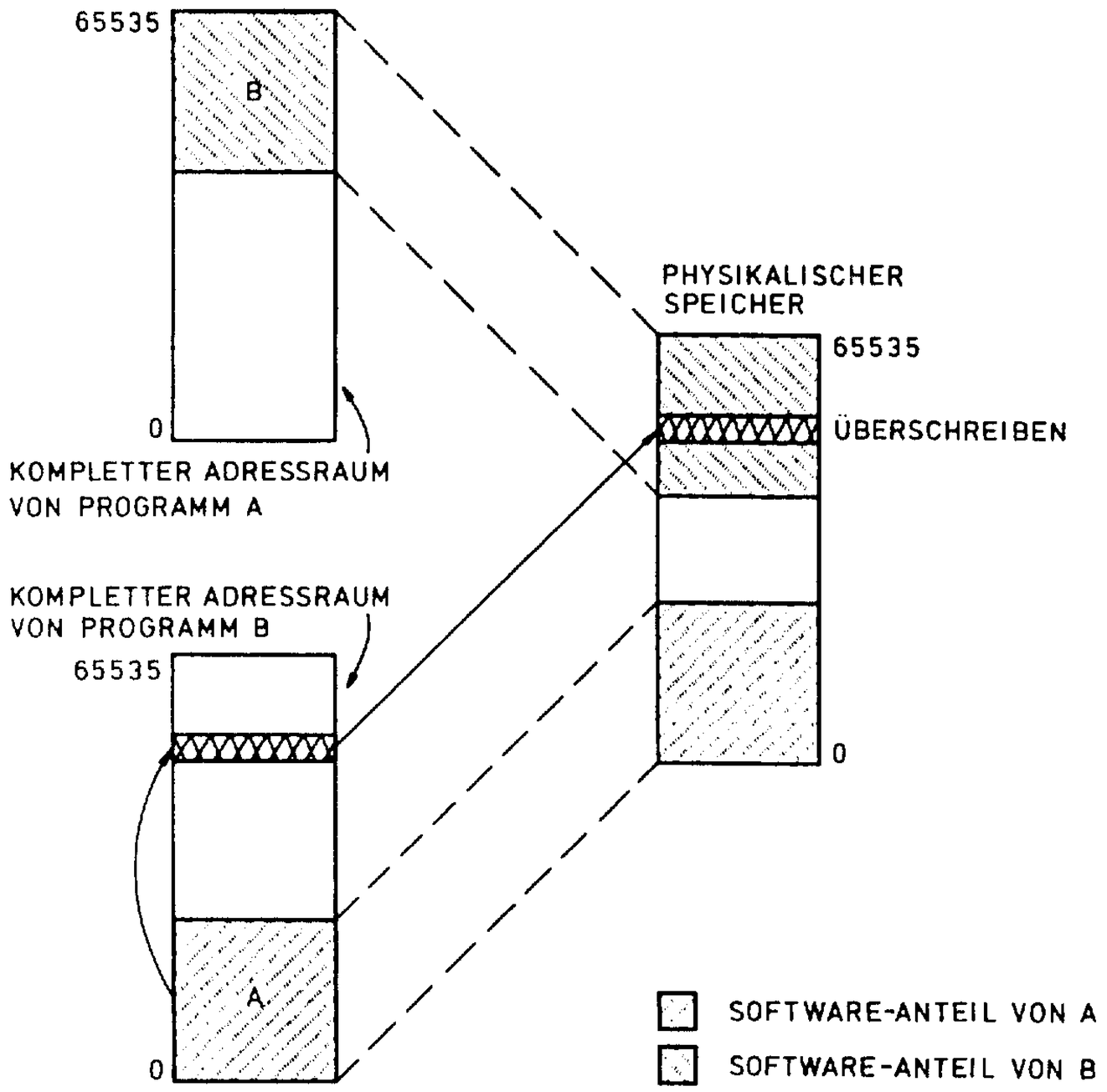


Die Beschränktheit der angegebenen Speicherorganisation wird bereits sichtbar, wenn sich verschiedene Programme den gleichen Prozessor teilen müssen, um die System-Ressourcen wirksamer zu nutzen.

Solche Multi-Programmsysteme sind schwer auf Prozessor-Konfigurationen zu implementieren, die eine so einfache Speicherorganisation wie der 8085 haben.

Vor allem hindert niemand ein Programm daran, auf Daten eines anderen Programms zuzugreifen oder diese sogar zu zerstören. Da alle Programme den gleichen logischen Adressraum haben, können sie auf beliebige Speicherpositionen zugreifen. Das bedeutet, daß ***kein*** Programm vor einem unberechtigten Zugriff eines anderen Programms geschützt ist. Ein weiterer Nachteil ist, daß sich alle Programme innerhalb des relativ eingeschränkten logischen Adressraums des 8085 befinden.

Das folgende Bild soll das Problem des unkontrollierten Zugriffs und damit die Empfindlichkeit einer einfachen Speicherorganisation verdeutlichen.



Da sich ein Programm auf jede Position im Speicher beziehen kann, kann nicht verhindert werden, daß in der angegebenen Weise Programm A Teile der Code- oder Datenregion von Programm B überschreibt. Nur ein einfacher Programmfehler in A kann das Programm B völlig zerstören.

Auch wenn beim Entwurf von Multi-Programmsystemen auf die Adressbelegung mit größter Sorgfalt geachtet wurde, ist trotzdem nicht garantiert, daß bei einer einfachen Speicherorganisation das System zuverlässig und gefahrlos arbeitet.

Um die Empfindlichkeit solcher Systeme zu überwinden, ***ohne*** aber auf einen linearen Adressraum zu verzichten, ist es notwendig, neue Mechanismen für das Speicher-Management zur Verfügung zu stellen.

Diese Mechanismen sind:

- Abbildung logischer Adressen auf physikalische Adressen.
  Das Verfahren wird auch als ''Mapping'' bezeichnet.
- Überwachte Speicherzugriffe.
- Virtuelle Speicher.

In den folgenden Abschnitten sollen diese Mechanismen erläutert werden.

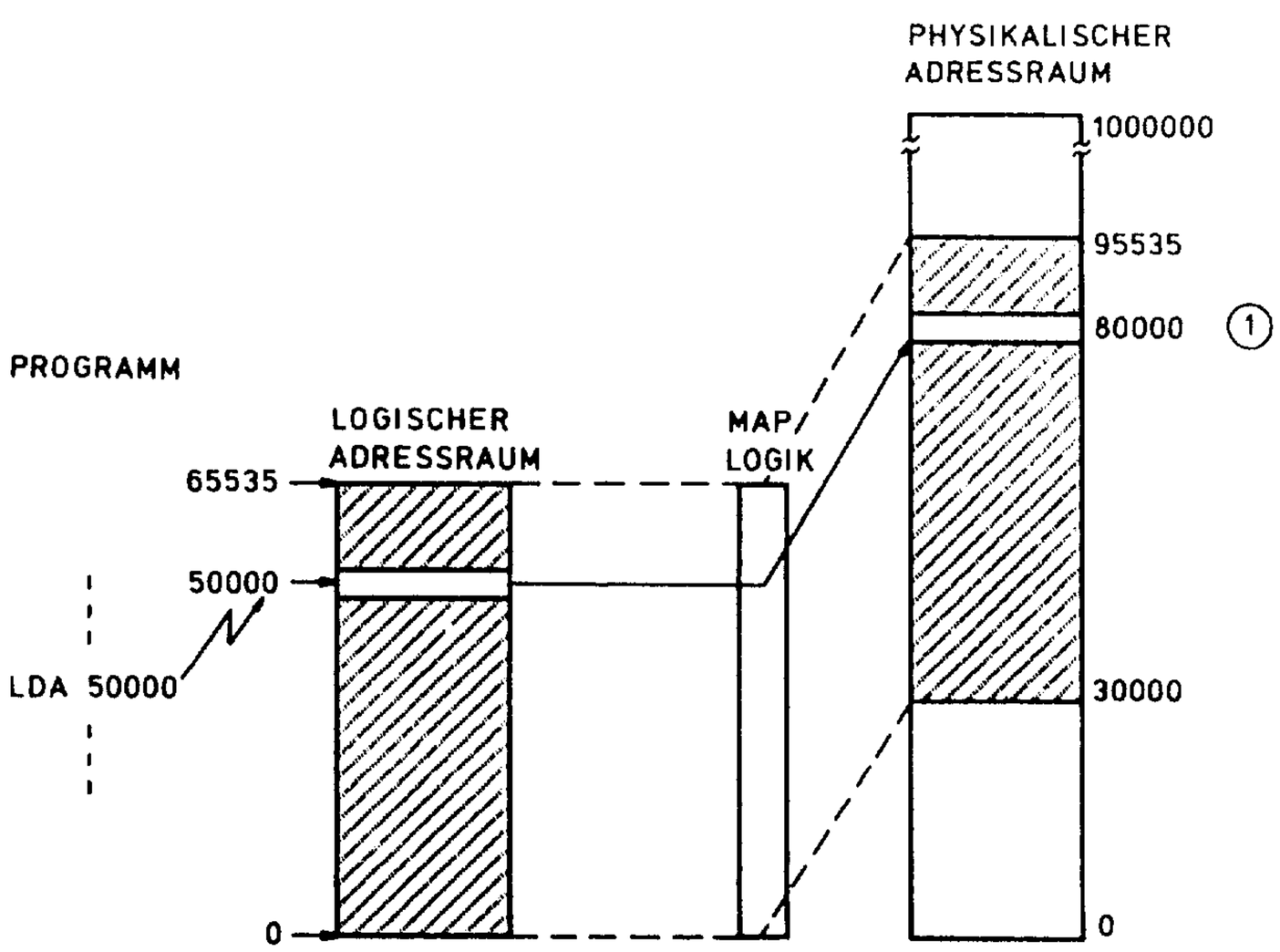
1.3 Mapping des linearen Speichers

Mapping ist der Prozess, logische Adressen auf physikalische Adressen abzubilden. In 8085-Systemen, die oben genannt wurden, sind die logischen Adressen mit den physikalischen Adressen identisch. Wird allerdings eine Mapping-Logik benützt, kann eine logische Adresse auf jede beliebige physikalische Adresse abgebildet werden.

Mapping erlaubt also, einen logischen Adressraum in einen physikalischen Adressraum zu übertragen.

Das folgende Bild zeigt eine einfache Mapping-Operation. Der vollständige logische Adressraum eines Programms (im Beispiel sind dies die Adressen 0–65535) ist an die physikalischen Positionen 30000–95535 übertragen worden.

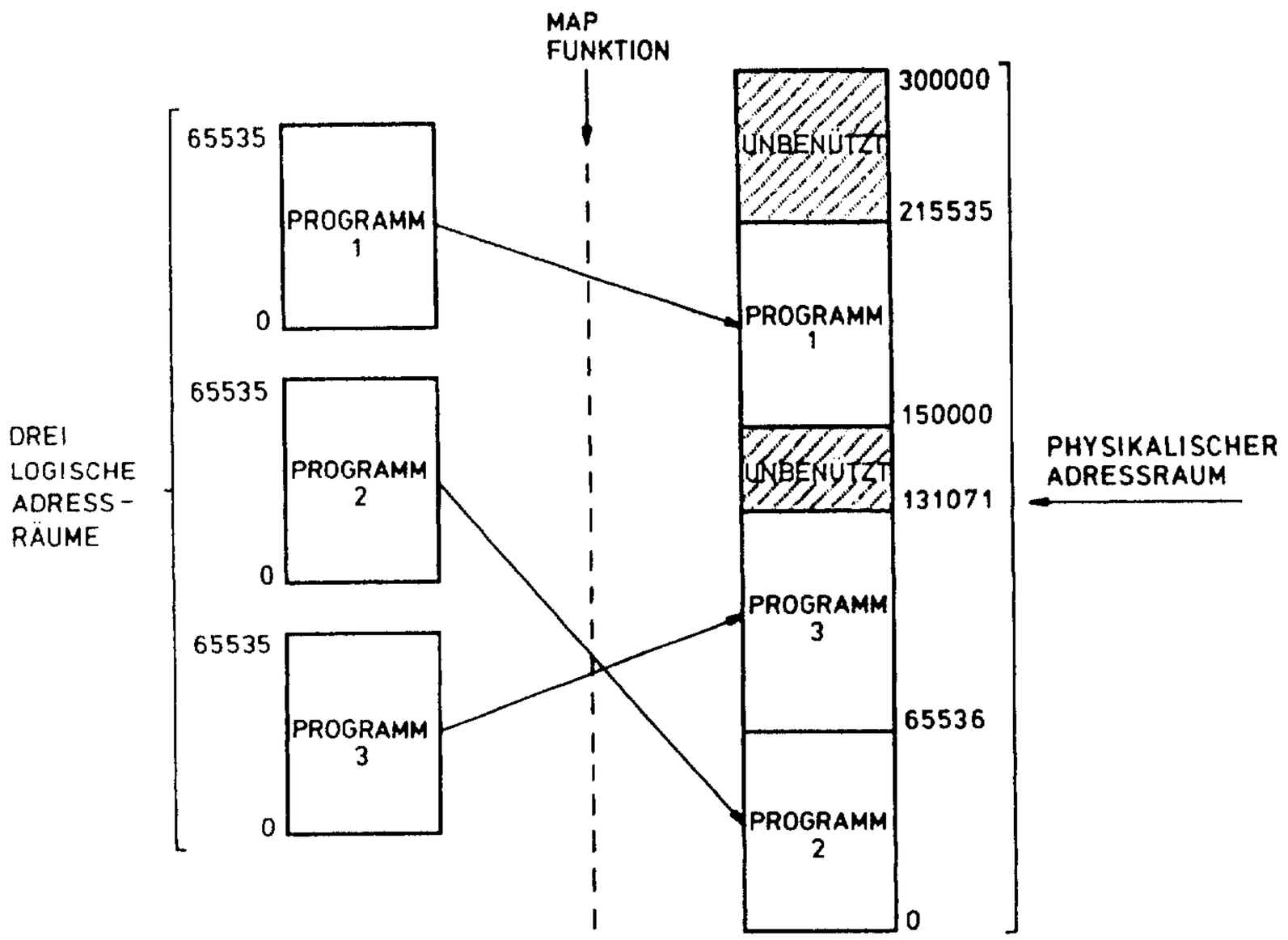
Dies bedeutet, daß ein Programm, das sich auf die Position 50000 bezieht, in Wirklichkeit Daten von der physikalischen Position 80000 (50000 + 30000) holt ①.



Mit dieser Art von Mechanismus, der für die meisten Minicomputer und einige fortschrittliche Mikrocomputer typisch ist, hat jedes Programm seinen eigenen logischen Adressraum und ist vollständig unabhängig von anderen Programmen.

Das bedeutet, daß sich viele Programme den gemeinsamen physikalischen Speicher teilen können, ohne sich gegenseitig zu beeinträchtigen. Die Mapping-Logik bildet also viele logische Adressräume auf den physikalischen Speicher ab.

Das angegebene Bild zeigt beispielhaft, wie die Mapping-Logik in einer Multiprogramm-Umgebung drei unabhängige logische Adressräume in den physikalischen Speicher überträgt.



1.4 Einteilung des logischen und physikalischen Adressraums in Abschnitte

Wie im vorigen Bild gezeigt, wurde der gesamte logische Adressraum eines Programms zusammenhängend auf den physikalischen Speicher abgebildet.

Fortschrittliche Adress-Übersetzungsmechanismen teilen stattdessen einen logischen Adressraum in Abschnitte (sogenannte "Pages") ein, wobei jeder Abschnitt durch eine starre Länge gekennzeichnet ist.

Die Folge ist, daß große Programme ***nicht*** mehr in einem zusammenhängenden physikalischen Speicherblock stehen müssen, sondern abschnittsweise auf mehrere kleine Adressbereiche verteilt werden können.

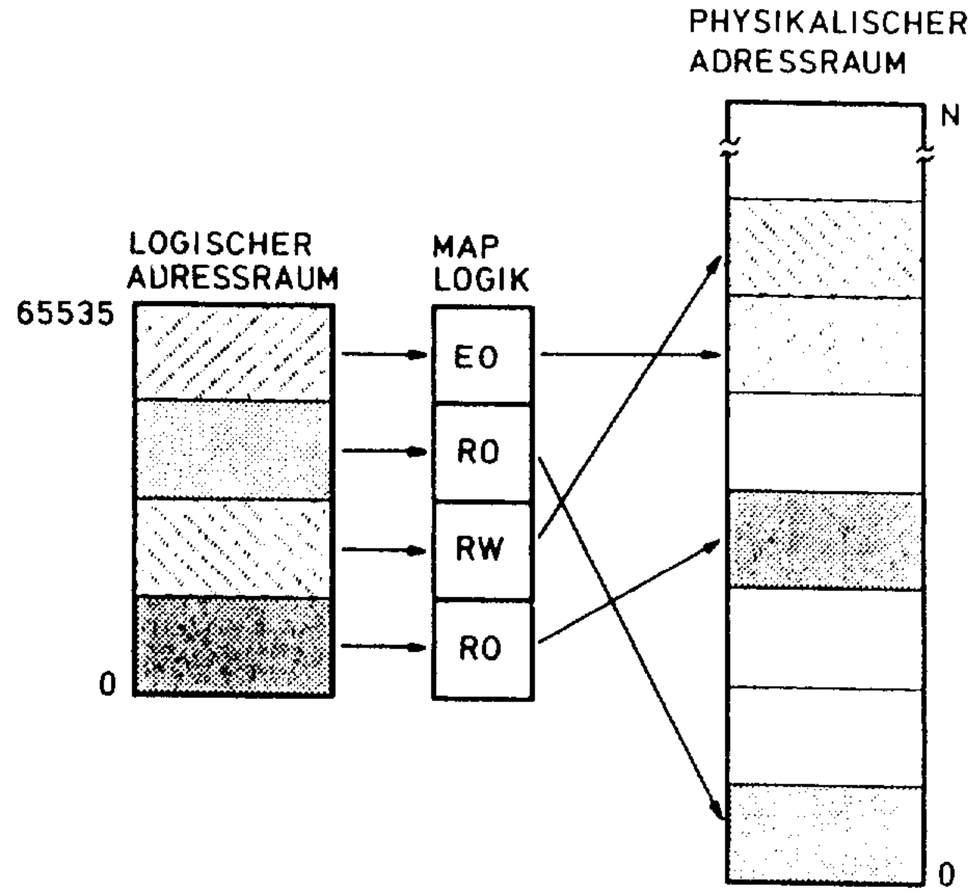
Der Vorteil dieser Technik liegt darin, daß kleinere Programmabschnitte gebildet werden können, die im Speicher leichter zu finden und zu testen sind.

1.5 Zugriffsrechte auf die Programmabschnitte

Der "Page"-Mechanismus kann auch die Grundlage für den Speicherschutz im logischen Adressraum liefern. So ist es möglich, jeden Programmabschnitt mit bestimmten Attributen zu versehen, die exakt angeben, ***wie*** auf einen Abschnitt zugegriffen werden darf. Das heißt, diese Attribute legen die Zugriffsrechte ("Access Rights") auf die Programmabschnitte fest.

So können die Attribute bestimmen, daß ein Programmabschnitt nur gelesen (Read Only), gelesen und beschrieben (Read-Write), nur ausgeführt (Execute Only) oder ausgeführt und gelesen (Execute-Read) werden darf.

Basierend auf der Einteilung eines Programms in Abschnitte, zeigt das folgende Bild ein Beispiel für einen "Page"-Mechanismus, der auch die Zugriffsrechte regelt.



EO-Execute Only (nur ausführbarer Abschnitt)
RO-Read Only (nur lesbarer Abschnitt)
RW-Read/Write (les-und beschreibbarer Abschnitt)

1.6 Virtuelle Speicher

In vielen Computer-Systemen ist der logische Adressraum wesentlich größer als der physikalische. ***Virtuelle Speicher*** sind ein Konzept, mit dem die Begrenzungen auf dem physikalischen Adressraum umgangen werden können.

Unter einem virtuellen Speichersystem erscheint es dem Anwender so, als würde der gesamte logische Adressraum für die Speicherung verfügbar sein. In Wirklichkeit aber werden nur zu gegebener Zeit einige Abschnitte des logischen Adressraums im physikalischen Adressraum abgebildet.

Die verbleibenden Abschnitte sind im Hauptspeicher momentan ***nicht*** gegenwärtig; sie stehen stattdessen in einer sekundären Speichereinheit, wie z. B. einer Diskette, abrufbereit zur Verfügung. Speichermedien dieser Art haben den Vorteil, daß die Kosten pro Bit geringer sind als bei üblichen Halbleiterspeichern.

Jedesmal, wenn auf einen fehlenden Programmabschnitt im Hauptspeicher zugegriffen wird, lädt das Betriebssystem diesen fehlenden Abschnitt von der Diskette. Umgekehrt legt das Betriebssystem auf der Diskette einen Abschnitt ab, auf den bis vor kurzem kein Zugriff erfolgt ist.

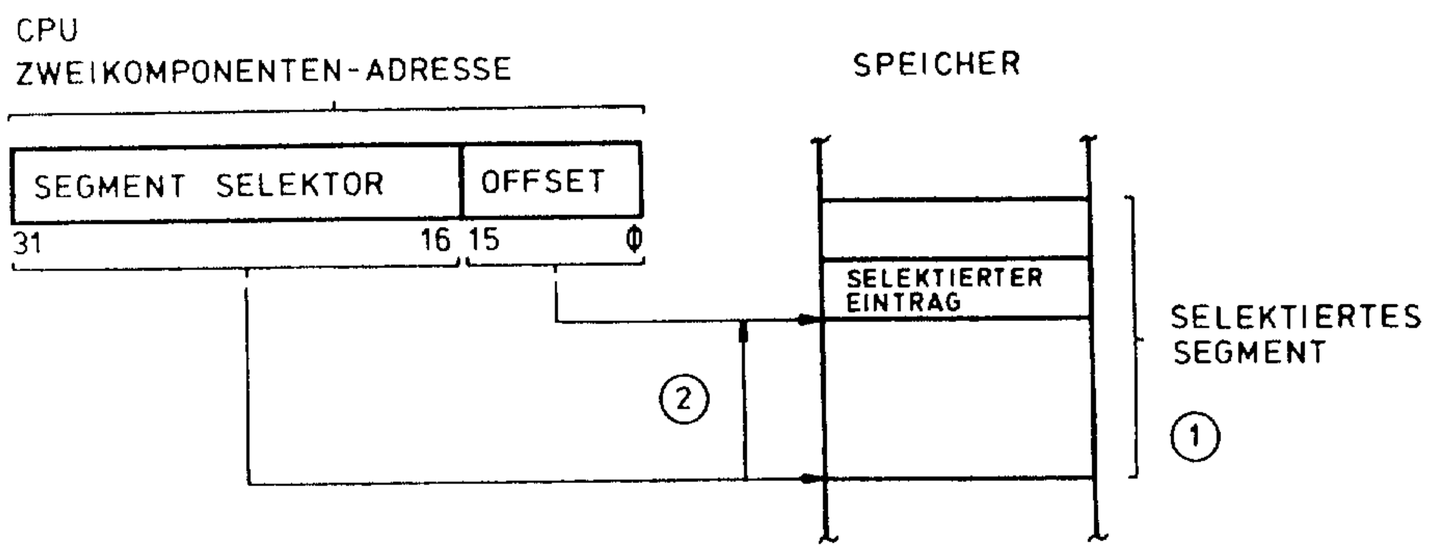
Der Anwender hat den Eindruck, mit einem gigantischen, wenn auch langsamen, physikalischen Speicher zu arbeiten.

1.7 Segmentierte Speicher

Es gibt eine Reihe neuer Prozessoren wie Intel 8086, 80186 und 80286, die auf die lineare Speicherarchitektur verzichten und stattdessen eine logische Speicherorganisation benutzen, die als ***segmentierter Speicher*** bezeichnet wird.

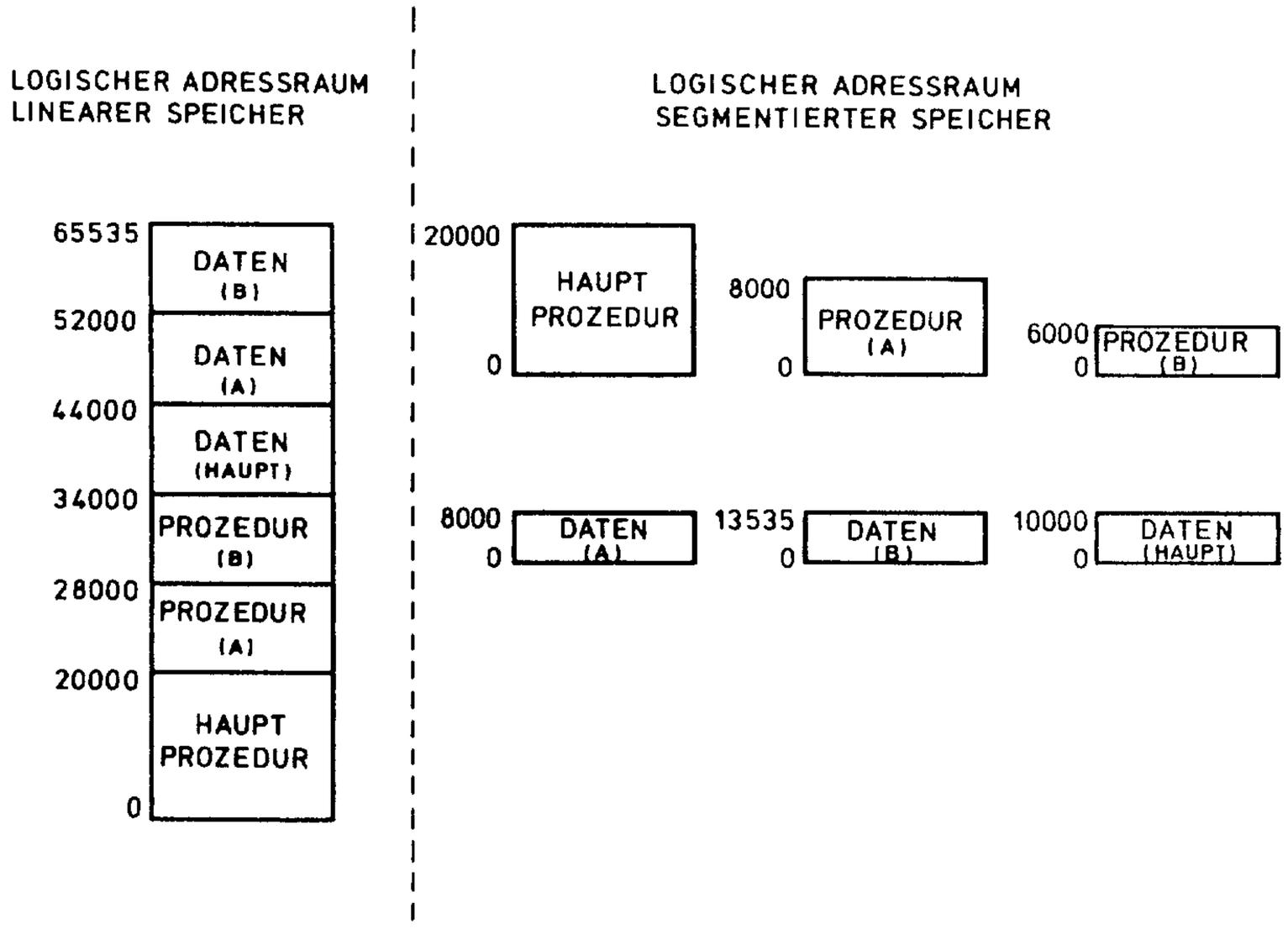
Segmentierte Speicher wurden eingeführt, weil Programme ***nicht*** als eine lineare Sequenz von Befehlen und Daten geschrieben werden, sondern aus mehreren Code- und Datenabschnitten bestehen, beispielsweise aus einer Haupt-Coderegion (Hauptprozedur) und vielen separaten Prozeduren. Die Datenregionen können als Felder, als Felder von Feldern oder als komplexe Datenstrukturen (sogenannte Records) organisiert sein. Um diese logische Struktur zu unterstützen, wird ein logischer Adressraum in viele lineare Adressräume mit individuell spezifizierten Längen zerlegt.

Jeder dieser linearen Adressräume wird als ***Segment*** bezeichnet. Um einen Eintrag in einem Segment zu erreichen, steht eine Zweikomponenten-Adresse zur Verfügung. Die erste Komponente (der Segment-Selektor) spezifiziert das Segment ①, und die zweite Komponente (das Displacement) spezifiziert den Offset, also den relativen Abstand zur Basis des Segments ②.



Das folgende Bild zeigt einen segmentierten und einen linearen Speicher im Vergleich. Es ist ersichtlich, daß mit der segmentierten Speicherarchitektur die logische Organisation des Programms durch die Struktur des logischen Adressraumes wiedergegeben werden kann.

Im Gegensatz dazu ist der lineare Adressraum strukturlos. Wie bereits erläutert wurde, bezieht sich der Schutzmechanismus für lineare Speicher auf Abschnitte starrer Länge. Diese Längen sind durch Hardware-Kriterien bestimmt und haben ***keine*** Beziehung zur logischen Struktur des Programms.



Wird ein logischer Adressraum mit seiner variierenden Größe in starre Abschnitte zerlegt, dann treten Probleme mit dem Schutzmechanismus auf. Der Schutzbereich ist entweder zu klein oder zu groß, weil die aus Hardware-Gründen starren Abschnitte nicht der logischen Programmstruktur folgen können.

Im Gegensatz dazu stehen Schutzmechanismen für Segmente. Da jedes Segment eine individuelle Länge haben kann, ist es einfach, sogar kleinen Programmen ein eigenes Segment zu geben und sie vor anderen Segmenten zu schützen.

Da Programme aus Hunderten oder sogar Tausenden von Modulen bestehen können, ist es wichtig, daß eine große Anzahl von Segmenten in der Architektur zur Verfügung steht.

Der virtuelle Speichermechanismus kann auch auf segmentierte Architekturen übertragen werden. In diesem Fall werden die Inhalte von Segmenten auf einem sekundären Speichermedium zwischengelagert.

1.8 Mapping des segmentierten Speichers

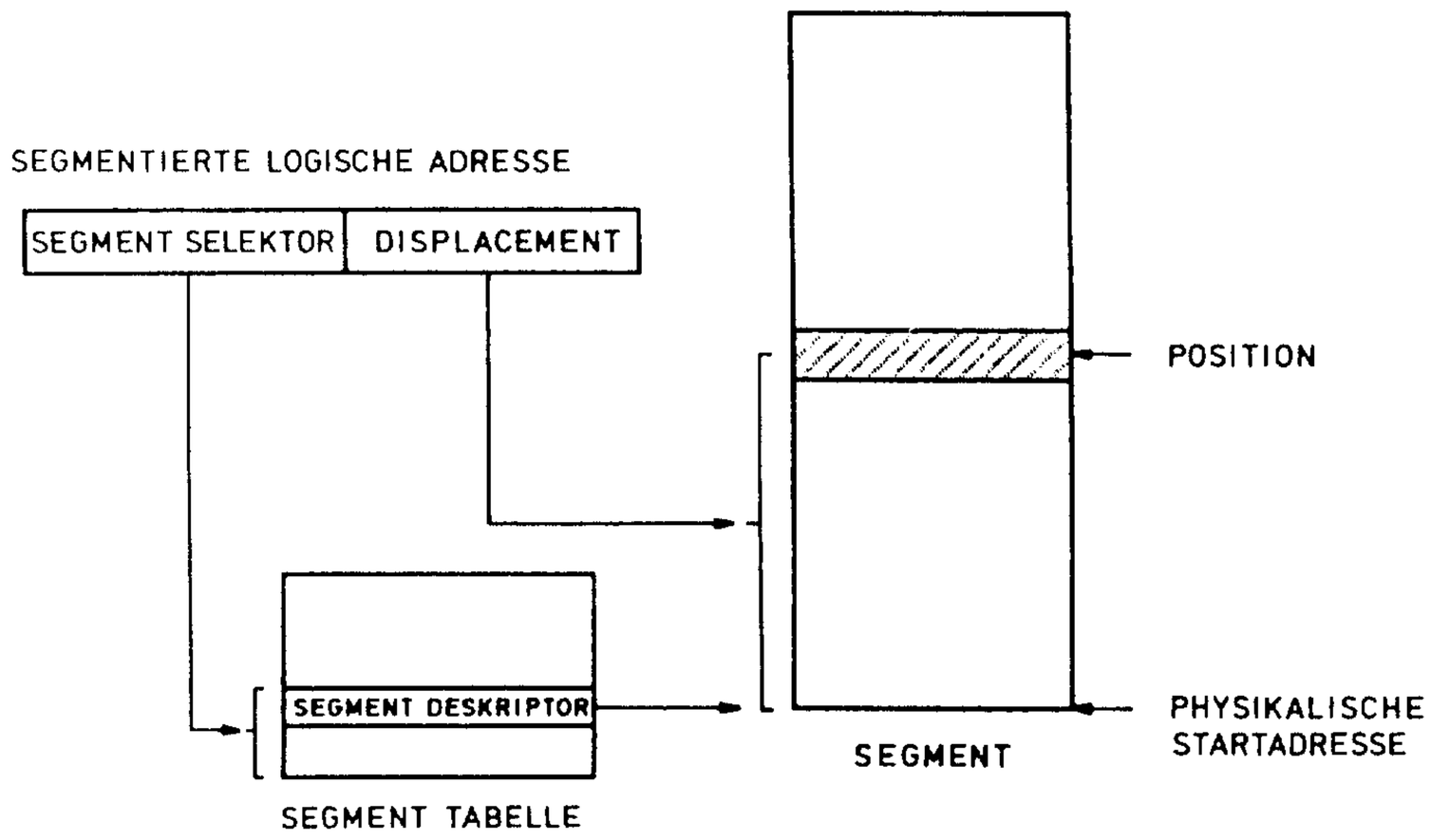
Eine segmentierte Architektur ist auch mit der Abbildung der logischen Adressbereiche auf physikalische Adressbereiche kompatibel. Allerdings wird nicht mehr der gesamte logische Adressraum oder eine Einheit mit starrer Länge in den physikalischen Adressraum übertragen. Jetzt ist das ***Segment*** die Einheit, die auf den physikalischen Speicher abgebildet wird.

Die Abbildung erfolgt in diesem Fall über eine ***Segment-Tabelle,*** in der sogenannte ***Segment-Deskriptoren*** für jedes Segment gespeichert sind.

Ein Segment-Deskriptor enthält die physikalische Startadresse und die Länge eines Segments. Die Segment-Selektorkomponente einer logischen Adresse wird jetzt als ***Index*** benützt, um einen Segment-Deskriptor in der Segment-Tabelle auszuwählen.

Zur Startadresse des Segments, die im Segment-Deskriptor gespeichert ist, wird anschließend das Displacement addiert. Dadurch wird die physikalische Adresse im ausgewählten Segment gebildet.

Das folgende Bild soll diesen Mechanismus verdeutlichen. Die Hardware kann nun bequem das Displacement überprüfen, um sicher zu gehen, daß die Länge des Segments nicht überschritten wird.



1.9 Segmenttypen und Zugriffsrechte

Neben der physikalischen Startadresse und der Länge des Segments enthalten Segment-Deskriptoren auch Attribute, die das Zugriffsrecht für ein Segment festlegen. Das heißt, jedes individuelle Segment ist mit einem bestimmten Zugriffsrecht verbunden. Hat beispielsweise ein Segment das Attribut ***RO*** (Read Only), ist das Segment ***von allen Programm-Modulen nur lesbar.***

Die starre Festlegung von Zugriffsrechten für Segmente ist allerdings dann von Nachteil, wenn verschiedenen Modulen der Zugriff auf das ***gleiche*** Segment mit wechselnden Zugriffsrechten erlaubt werden soll.

1.10 Überwachung des Zugriffs

Ein weiterer Nachteil der beschriebenen Segment-Mappingtechnik ist, daß der Zugriff eines Programms auf Segmente eines anderen Programms schwer zu verhindern ist.

Da die Segment-Tabelle alle Segment-Deskriptoren enthält, kann jedes Programm jedes Segment erreichen. Es ist lediglich notwendig, den richtigen Indexwert anzugeben, um den entsprechenden Segment-Deskriptor in der Segment-Tabelle auszuwählen.

Dieses Problem kann allerdings durch mehrere Segment-Tabellen gelöst werden. So ist es möglich, jedem Programm seine eigene Segment-Tabelle zu geben. Die Deskriptoren, die in solch einer Tabelle enthalten sind, beziehen sich dann nur auf die Segmente, die das Programm benötigt.