Kapitel 1

Die 8086/80186/80286-Basis-Architektur

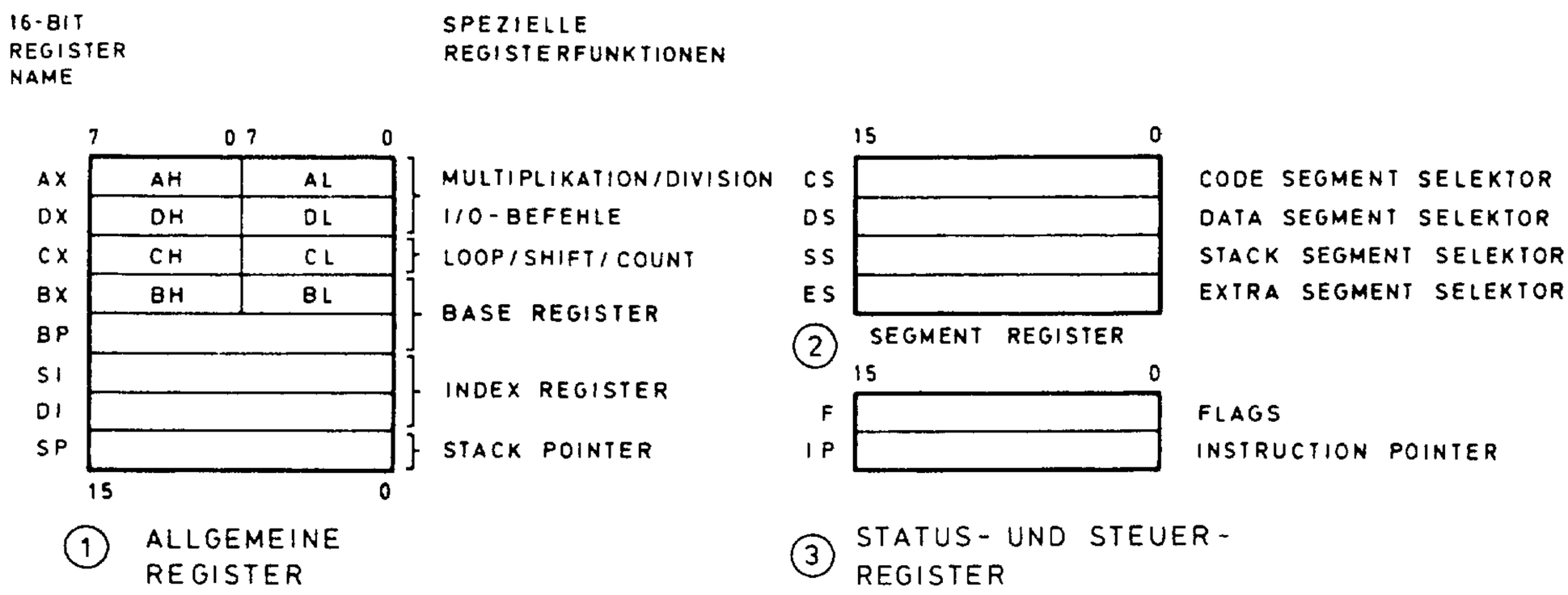
(für MASM-, ASM86- und ASM286-Programmierer)

1.1 Register-Struktur-Übersicht

Die Mikroprozessoren 8086/80186/80286 beinhalten den gleichen Basissatz von 14 Registern. Diese Register sind in folgende Funktionsgruppen eingeteilt:

- Allgemeine Register
- Segment-Register
- Status- und Steuer-Register

8086/80186/80286-Register-Struktur:



Die vier allgemeinen Register AX, DX, CX und BX sind für den Assembler-Programmierer 16-Bit-Einheiten. Dabei sind die oberen und unteren Hälften dieser Register als 8-Bit allgemeine Register separat adressierbar. Sie heißen AL, AH, BL, BH, CL, CH, DL und DH. Die Nachsilbe H oder L kennzeichnet das High Order- bzw. Low Order-Byte eines 16-Bit-Registers.

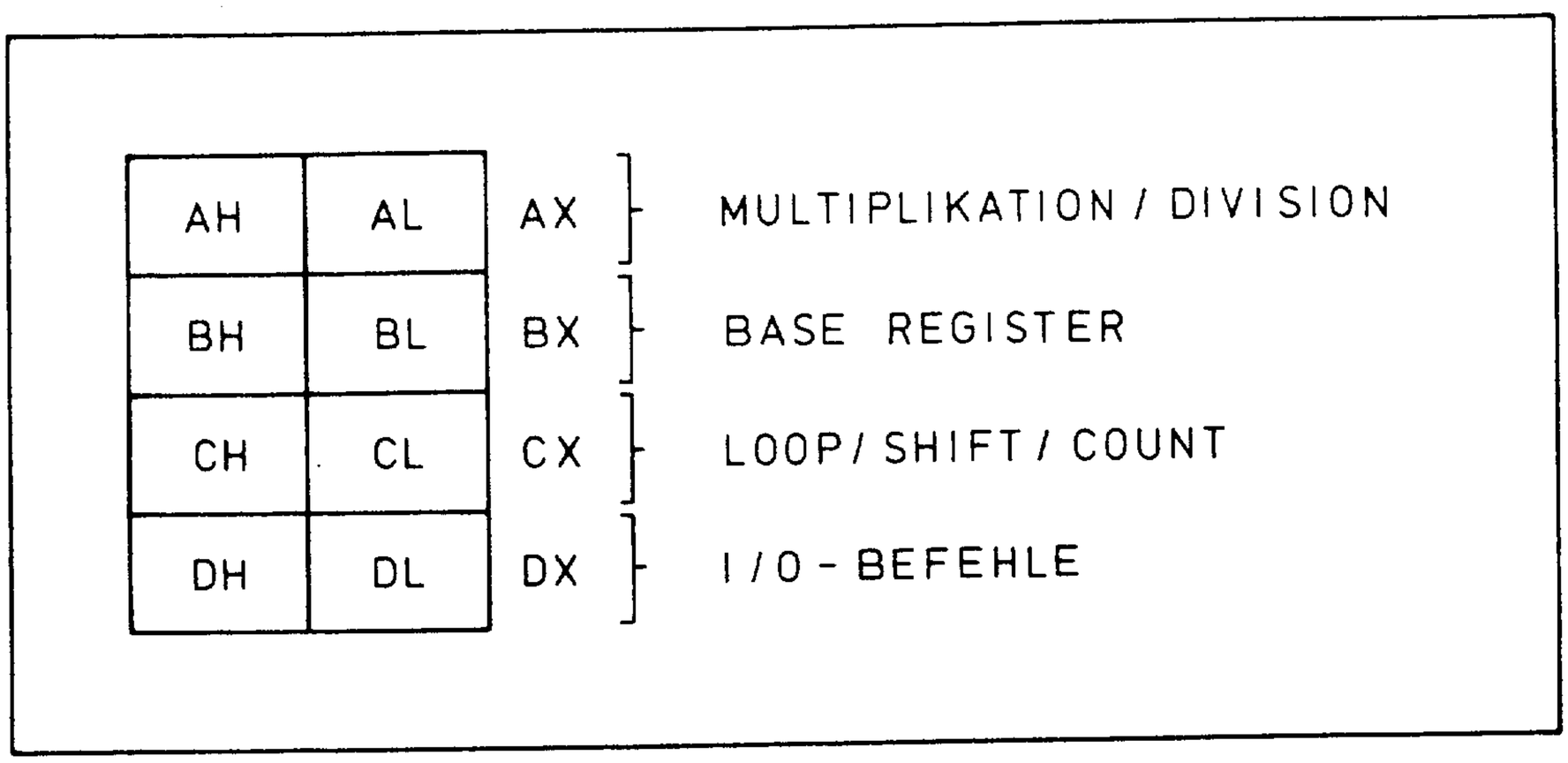
Die restlichen allgemeinen Register BP, SI, DI und SP, die Segment-Register CS, DS, SS und ES und die Status- und Steuer-Register F und IP werden in der Assemblersprache als 16-Bit-Einheiten behandelt.

1.2 Die allgemeinen Register

Die allgemeinen Register sind in zwei Gruppen unterteilt:

- Daten-Register
- Index- und Pointer-Register

1.2.1 Daten-Register

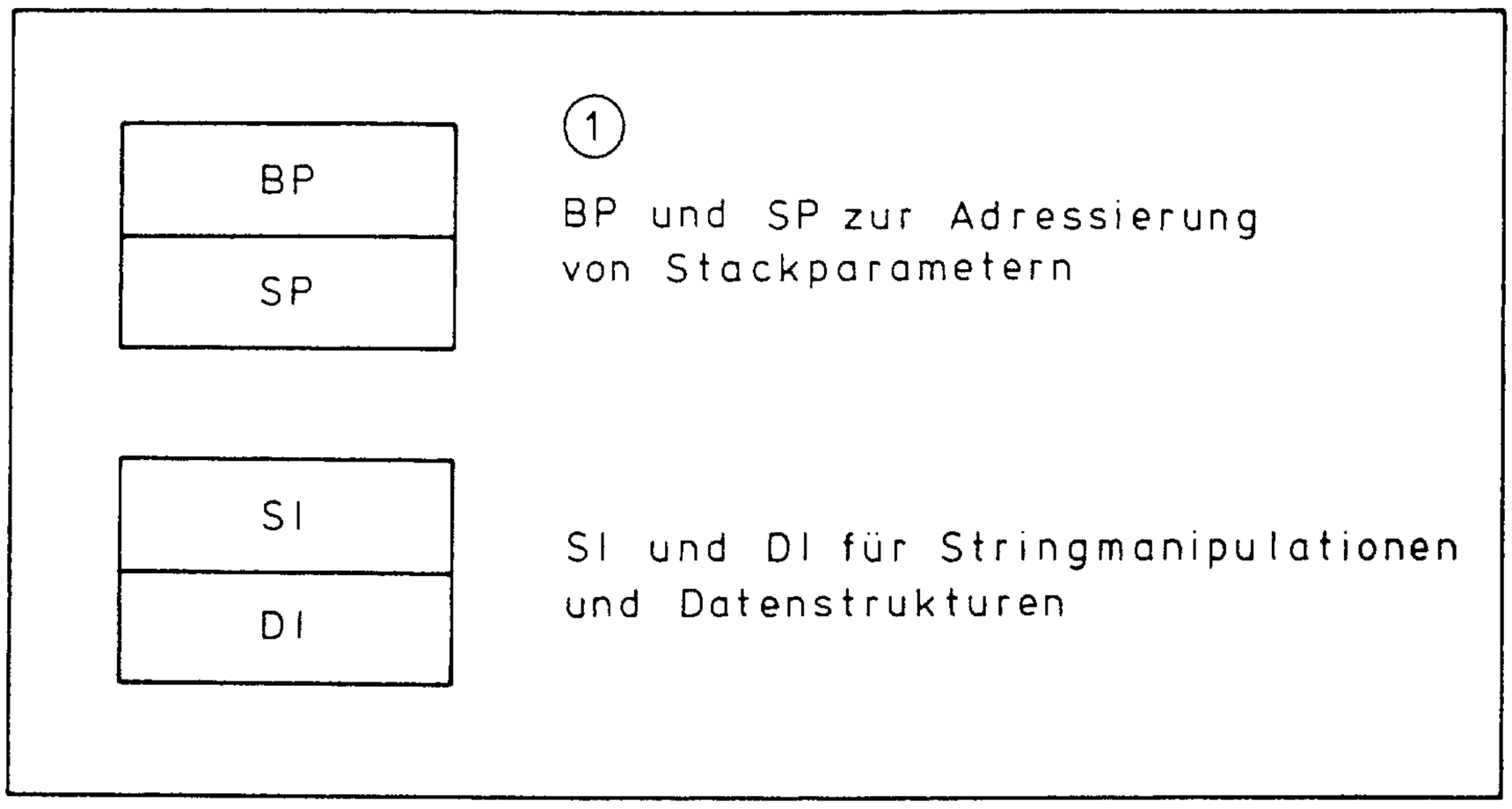


Im Allgemeinen werden diese Register für logische und arithmetische Operationen benützt. Einige Register haben zusätzliche **spezielle** Funktionen, die bei der Ausführung bestimmter Befehle geleistet werden.

- Beispielsweise wird der Inhalt des CX-Registers oft als Zählervariable für Wiederholungsbefehle verwendet.
- Das BX-Register kann als Basis-Register bei einigen Adressierungsarten benützt werden.
- Das DX-Register kann die 16-Bit-Portadressen für die indirekte Adressierung von I/O-Bausteinen enthalten.
- Word- oder Byte-Multiplikationen bzw. Word- oder Byte-Divisionen benützen **implizit** die Register AX bzw. AH und AL.

Es gilt, daß die implizite Anwendung der Register für spezielle Funktionen sehr leistungsfähige Maschinenbefehle erlaubt.

1.2.2 Pointer- und Index-Register



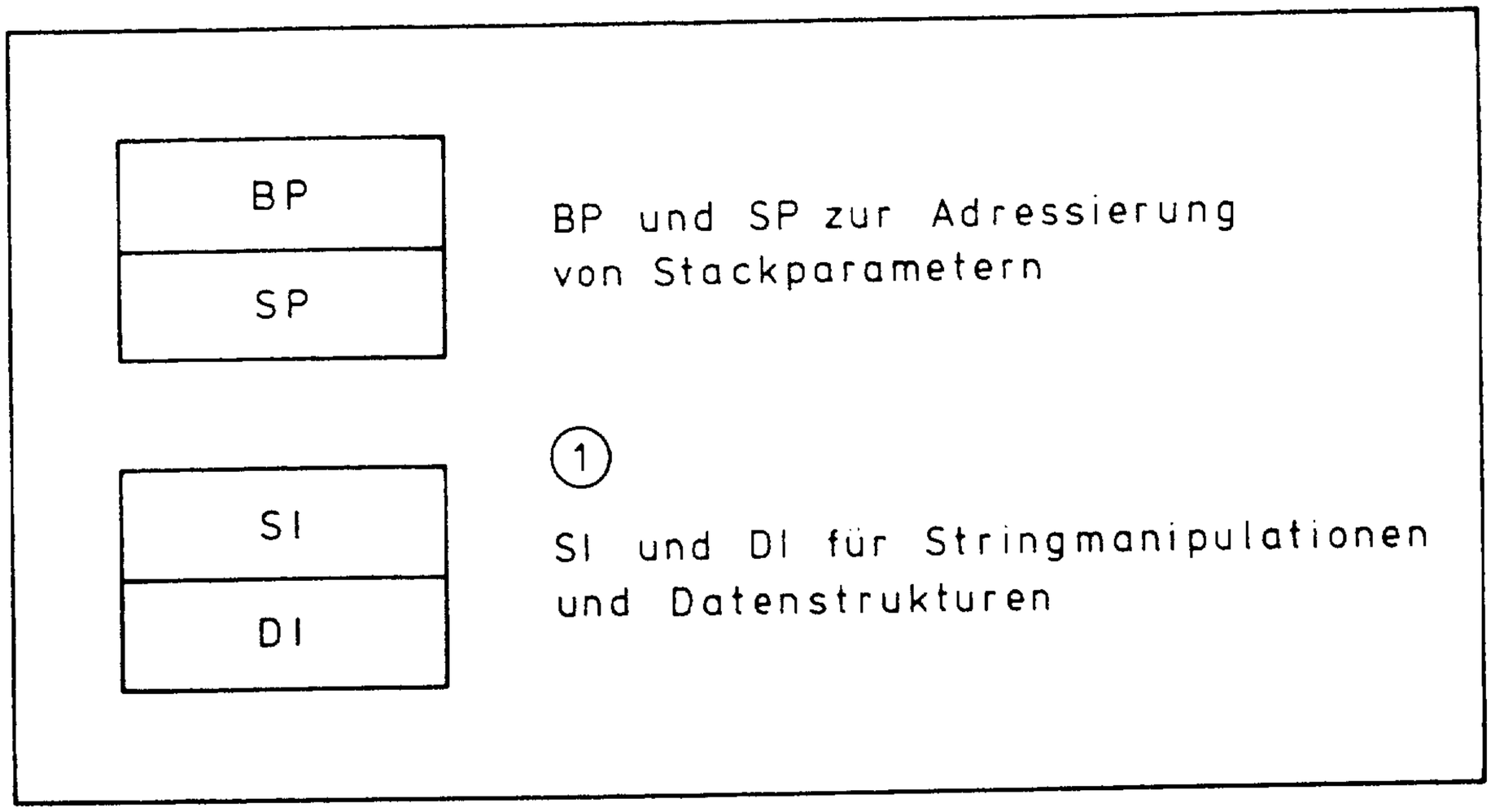
Die meisten Mikroprozessoren haben ein einzelnes Stackpointer-Register, das im Allgemeinen mit SP (Stack Pointer) bezeichnet wird. Das SP-Register zeigt auf ein lineares Feld im Speicher (STACK).

In diesem Feld werden Unterprogramm-Parameter und Unterprogramm-Rücksprungadressen hinterlegt oder vorübergehende Daten, die während eines Programmablaufs auftreten, zwischengespeichert.

Die Mikroprozessoren 8086/80186/80286 haben neben dem SP-Register einen zusätzlichen Zeiger zum STACK, der mit BP (Base Pointer) bezeichnet wird ①.

Das BP-Register kann einen alten Stackpointer-Wert enthalten oder unabhängig vom SP-Register einen Speicherplatz im Stack markieren. Dadurch ist es möglich, mit dem BP-Register bestimmte Unterprogramm-Parameter oder Zwischendaten im Stack gezielt zu adressieren, ohne das SP-Register verändern zu müssen.

Neben seiner speziellen Funktion, kann das BP-Register auch für arithmetische und logische Operationen verwendet werden.



Die zwei Index-Register sind das SI-Register (Source Index) und das DI-Register (Destination Index) ①.

Diese Register werden implizit bei Stringverarbeitungsbefehlen benützt. Zusätzlich ist es möglich, die Index-Register beim Aufbau von 8086/80186/80286-Datenstrukturen oder bei einigen Adressierungsarten zu verwenden.

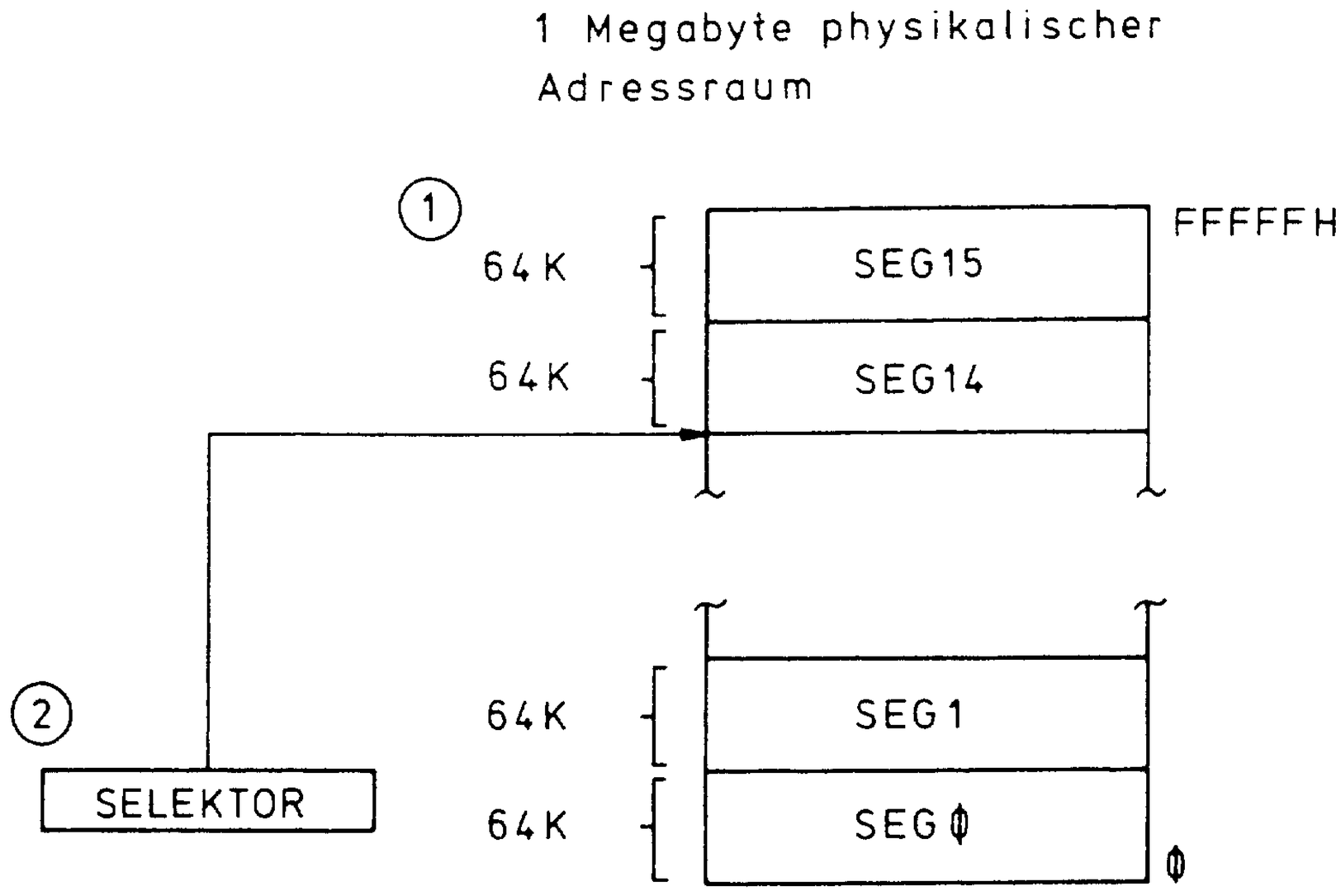
Sowohl das SI-Register, als auch das DI-Register haben die Fähigkeit, daß sie automatisch inkrementiert und dekrementiert werden können.

Neben ihren speziellen Funktionen ist es möglich, die Index-Register als allgemeine Register für arithmetische und logische Operationen zu verwenden.

1.3 Segment-Register

Die Mikroprozessoren 8086/80186 und der 80286 im Real-Mode haben die Möglichkeit maximal 1 Megabyte (1.048.576 Bytes) physikalisch zu adressieren.

Dieser 1 Megabyte große Speicherbereich ist so organisiert, daß er in unterschiedliche Regionen, sogenannte **Segmente** zu je 64 KByte (65.536 Bytes) aufgeteilt ist ①.



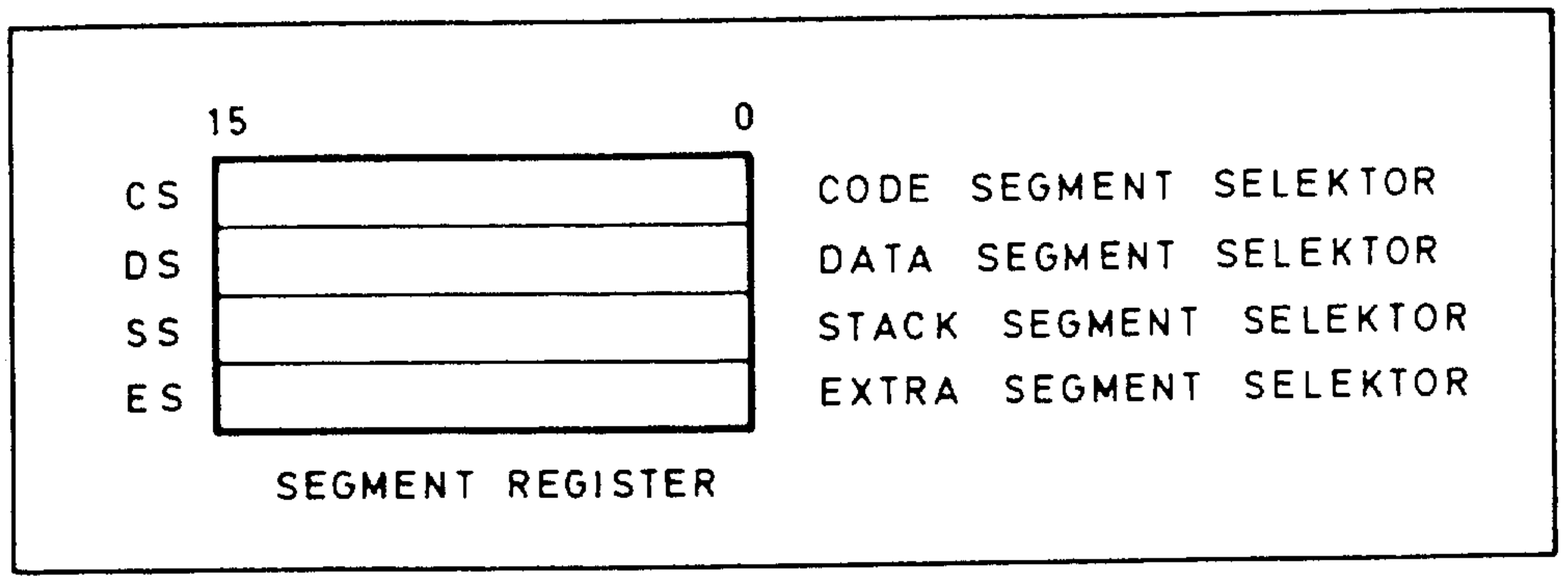
Da jedes Segment aus maximal 64 KByte bestehen kann, können im 1-MByte-Speicherraum mindestens 16 unterschiedliche Segmente (SEG0-SEG15) existieren ①.

Um den Datenbestand in den einzelnen Segmenten zu erreichen, ist zunächst das Segment auszuwählen, dessen Datenbestand bearbeitet werden soll.

Die Auswahl eines aktuellen Segments erfolgt bei den Mikroprozessoren 8086/80186/80286 durch einen sogenannten Selektor ②. Dabei ist ein Selektor identisch mit einem der vier Segment-Register CS, DS, ES oder SS.

Im Allgemeinen besteht ein Assembler-Programm aus vielen unterschiedlichen Segmenten. Dabei darf sich in einem Segment nur ein bestimmter **Typ** von Daten befinden. So ist es möglich, daß ein Programm aus mehreren Segmenten zusammengesetzt ist, wobei einige Segmente ausschließlich Code enthalten und andere nur variable Daten.

Während der Programmausführung wird aber nur eine kleine Teilmenge dieser Programm-Segmente benützt. Im Allgemeinen besteht ein Assembler-Programm mindestens aus einem Code-Segment, einem Segment mit variablen Daten und möglicherweise einem Stack-Segment.



①

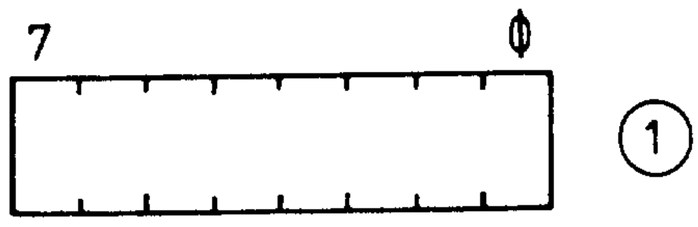
Die Mikroprozessoren 8086/80186/80286 haben vier verschiedene Selektoren, die unmittelbar während eines Programmablaufs auf vier unterschiedliche Segmenttypen zugreifen können ①.

- Das Code-Segment-Register (CS-Register) selektiert ein Code-Segment
- Das Data-Segment-Register (DS-Register) selektiert ein Daten-Segment
- Das Stack-Segment-Register (SS-Register) selektiert ein Stack-Segment
- Das Extra-Segment-Register (ES-Register) selektiert ein zweites Daten-Segment.

1.4 Fundamentale Datentypen

MASM, ASM86 und ASM286 sind typorientierte Assemblersprachen. Dies bedeutet, daß jeder Operand in diesen Sprachen mit einem sogenannten **Typ**-Attribut versehen ist, das dem Assembler-Sprachübersetzer (Assembler) Auskunft über den Typ des zu verarbeitenden Operanden gibt.

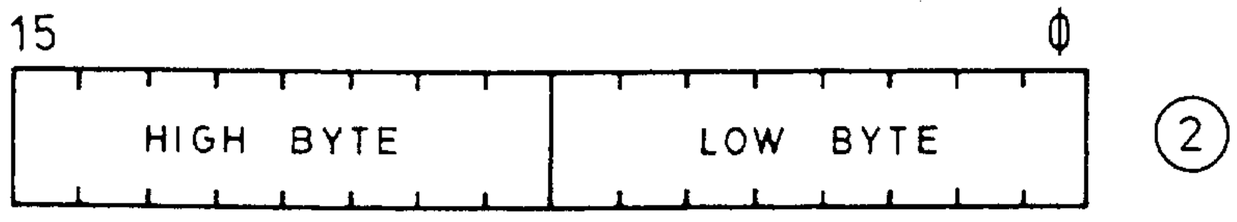
Die fundamentalen Datentypen in MASM, ASM86 und ASM286 sind **Bytes** und **Words**.



①

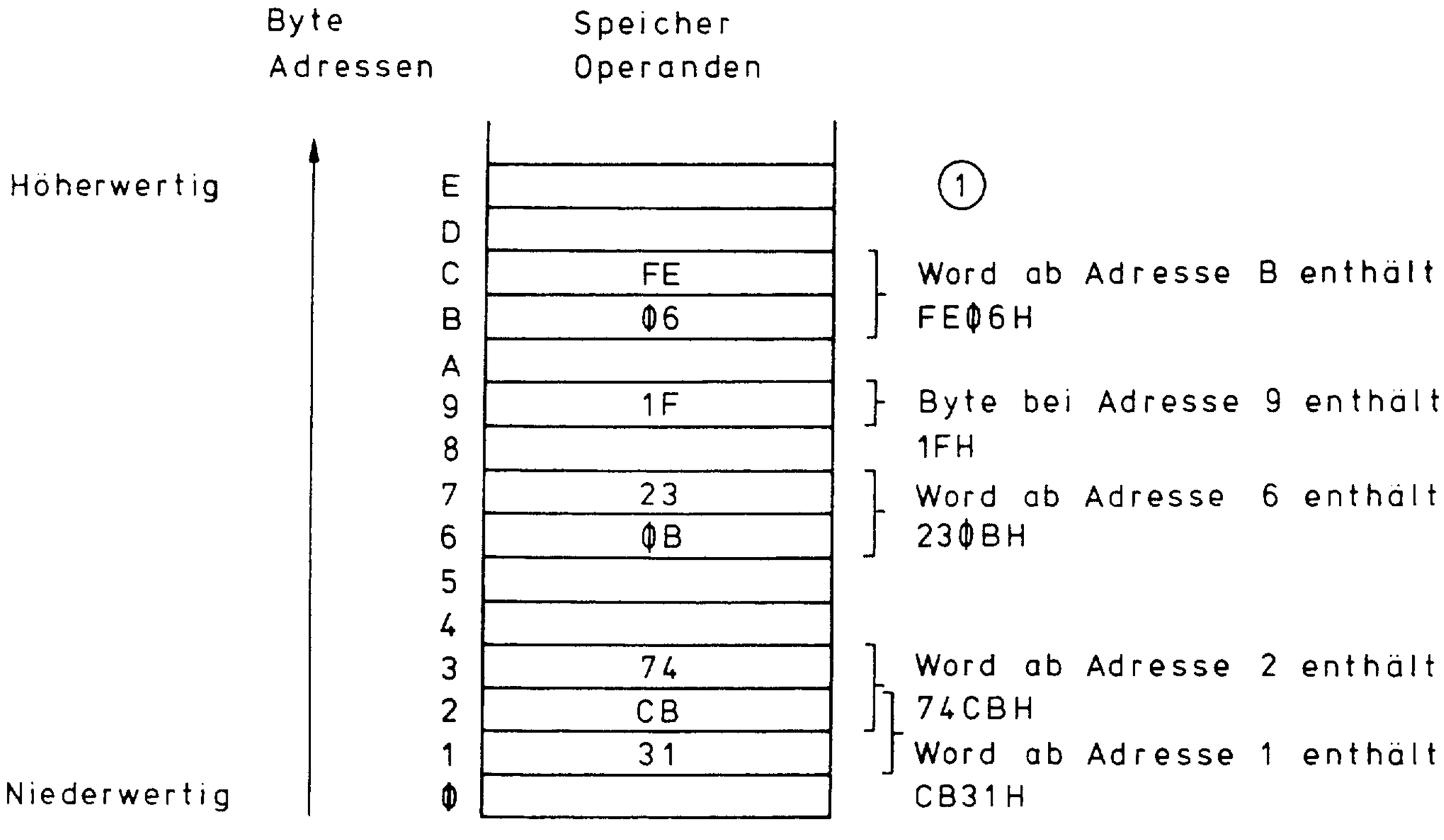
– Ein **Byte** besteht aus 8 Bit, das an einer beliebigen Adresse im Speicher hinterlegt ist. Die Bits sind von 0 bis 7 durchnumeriert, wobei Bit 0

als LSB (Least Significant Bit) und Bit 7 als MSB (Most Significant Bit) bezeichnet wird ①.



– Ein **Word** besteht aus zwei zusammenhängenden Bytes und ist ab einer beliebigen Adresse im Speicher hinterlegt. Somit enthält ein Word 16 Bits, die von 0 bis 15 durchnumeriert sind. Bit 15 ist das Most Significant Bit und Bit 0 ist das Least Significant Bit. Das Byte, das Bit 0 enthält wird als **Low Byte** und das Byte, das Bit 15 enthält, wird als **High Byte** bezeichnet ②.

1.4.1 Bytes und Words im Speicher



Jedes Byte in einem Word hat seine eigene Adresse. Die niederwertigere der beiden Adressen wird als Adresse des Words benützt. Das Byte an der niederwertigeren Adresse enthält die 8 niederwertigen Bits des Words, während die höherwertige Adresse die 8 höherwertigen Bits des Words enthält ①.

Words müssen nicht ab einer geraden Adresse (Word boundary) im Speicher ausgerichtet sein. Sollten sie ab einer ungeraden Adresse (Byte boundary) gespeichert sein, bedeutet dies, daß die 8086/80186/80286-CPUs für einen Word-Transfer 2 Speicherzyklen benötigen.

Für einen Word Transfer unter gerader Adresse ist nur ein Transferzyklus erforderlich. Um eine maximale Systemleistung zu erreichen, sollten daher in Datenstrukturen (z.B. Stacks) die Wordoperanden an geraden Adressen ausgerichtet sein.

1.5 Zusammenhang zwischen Datentypen und Register

Da in Assembler die 8086/80186/80286-Register entweder als 8-Bit- oder 16-Bit-Einheiten behandelt werden, bedeutet dies, daß in den 16-Bit-Registern Operanden vom Typ **Word** und in den 8-Bit-Registern Operanden vom Typ **Byte** gespeichert werden können.

Damit ergeben sich folgende mögliche Registertypen:

- Allgemeine **Byte**-Register (8 Bit)
- Allgemeine **Word**-Register (16 Bit)
- Segment-**Word**-Register (16 Bit)

Die Register und die mit ihnen verbundenen Typen zeigt folgende Tabelle:

8086/80186/80286-Register

Allgemeine Register		Segment Register
Typ Word	**Typ Byte**	**Typ WORD**
AX BX CX DX SI DI SP BP	AL,AH BL,BH CL,CH DL,DH	CS DS SS ES

1.6 Segmentierung

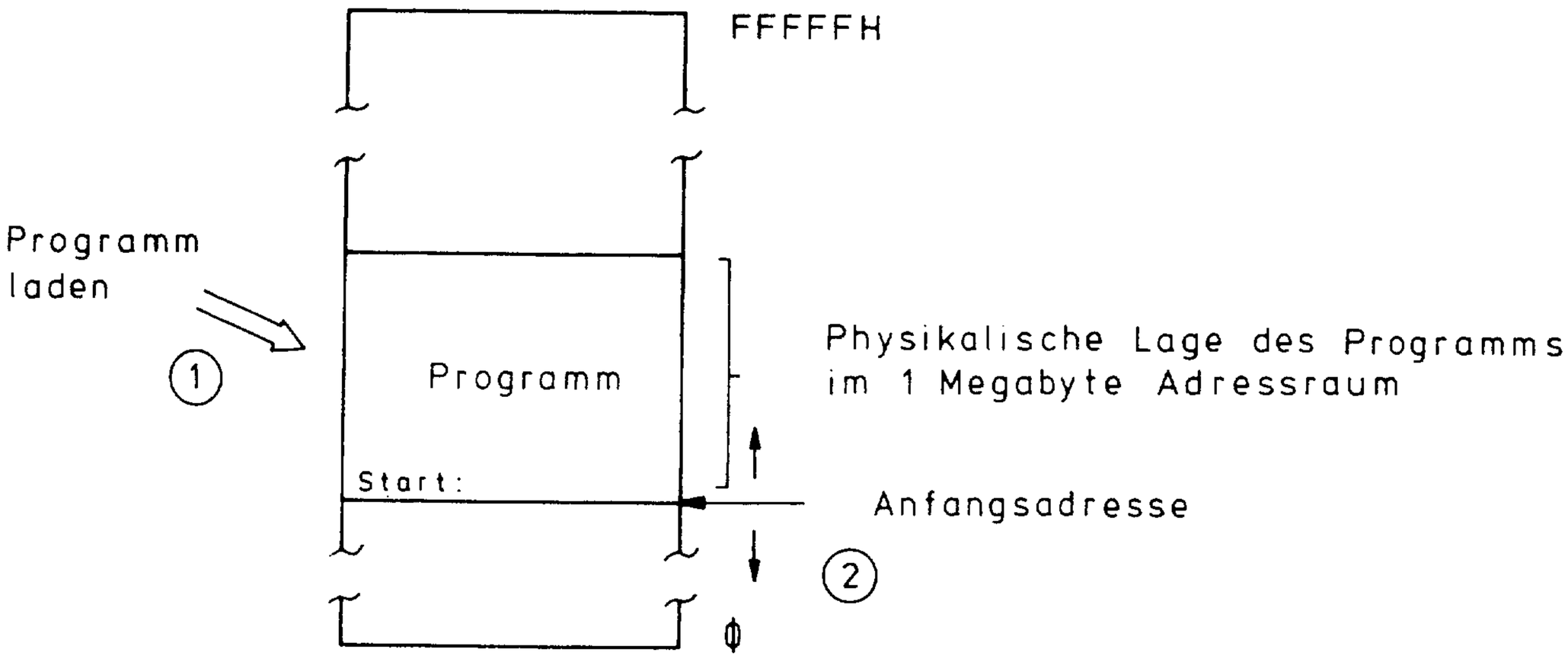
Wie bereits erwähnt, ist der physikalisch adressierbare Adreßraum der 8086/80186/80286-CPUs in Segmente eingeteilt. Man sagt: Die Mikroprozessoren 8086/80186/80286 haben eine **segmentierte** Architektur.

Nachfolgend soll die Rückwirkung der Segmentierung auf die Assembler-Programmiersprache erklärt werden.

1.6.1 Die Teile eines Programms: Code, Daten und Stack

Es sei vorausgesetzt, daß ein einfaches Assembler-Programm entwickelt wurde und in einer Quelldatei (Source File) existiert.

Das geschriebene Programm stellt eine funktionale Einheit dar und wird nach der Übersetzung in den Maschinencode zu Test- und Ablaufzwecken in einen bestimmten Bereich des physikalischen Adreßraums geladen ①.

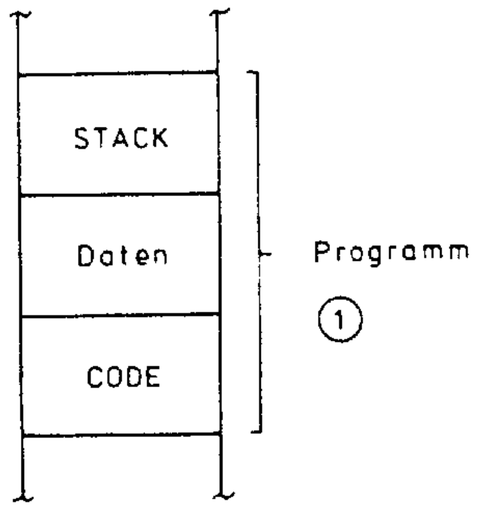


Dabei stellt die Anfangsadresse, ab der das Programm in den Speicher geladen wurde, die Startadresse des Programms dar ②.

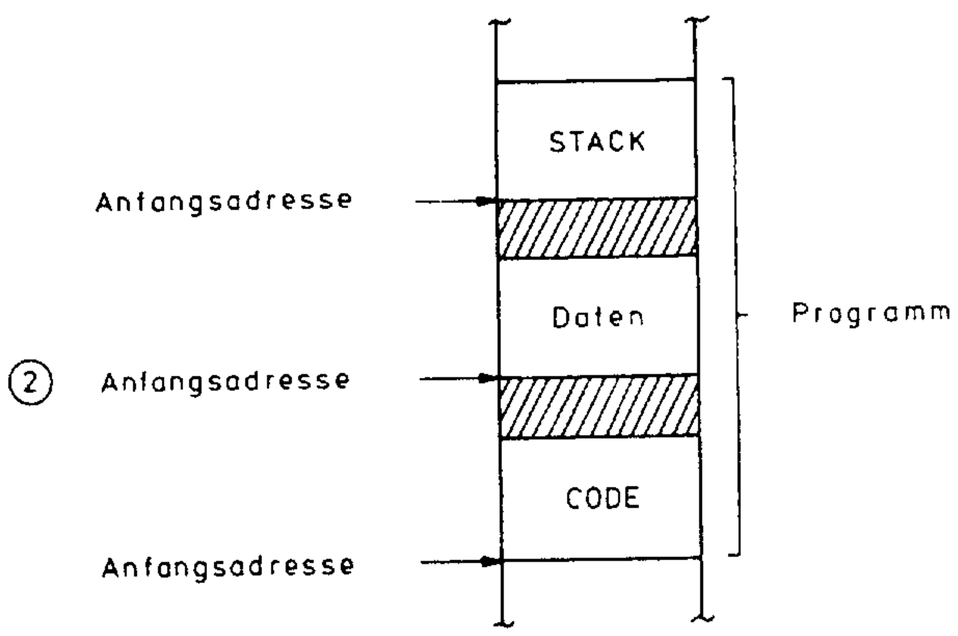
Die Anfangsadresse einer zu ladenden Programmeinheit ist **veränderbar** (relocatable), so daß ein Programm ab beliebigen Positionen gestartet werden kann.

Aus der Sicht des Assembler-Programmierers ist ein Programm in drei Regionen eingeteilt:

- Code-Region
- Daten-Region
- Stack-Region ①



Diese konzeptionell unterschiedlichen Teile eines Programms liegen streng voneinander getrennt in eigenen Speicherbereichen. Dadurch werden Mischungen dieser Regionen ausgeschlossen. So ist es nicht möglich, daß Daten als Code behandelt werden oder Stackdaten in die Daten-Region geschrieben werden.



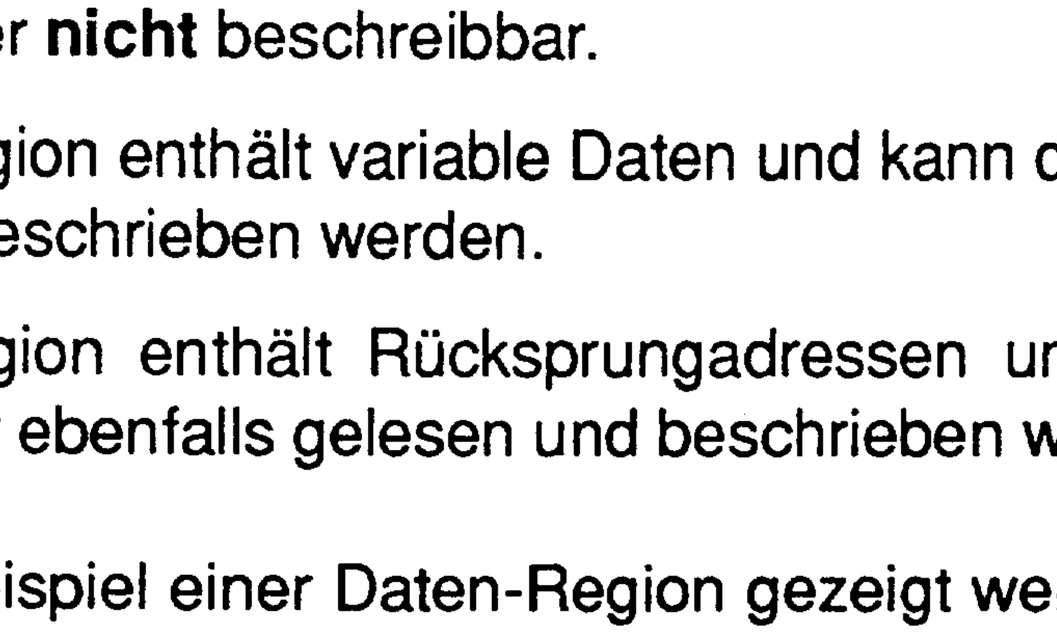
Die Programm-Regionen müssen nicht bündig aneinandergereiht sein, sondern können ab beliebigen Anfangsadressen im physikalischen Speicherraum beginnen ②.

1.6.2 Adressierung ab der Basis-Position

Jede Programm-Region enthält Daten unterschiedlichen Typs. Dies bedeutet, daß die Regionen auch unterschiedlich behandelt werden.

- Eine Code-Region kann neben Befehlen (Code) auch Konstanten enthalten.
- Daher ist diese Region sowohl lesbar (Konstanten), als auch ausführbar (Code), aber **nicht** beschreibbar.
- Eine Daten-Region enthält variable Daten und kann daher sowohl gelesen, als auch beschrieben werden.
- Eine Stack-Region enthält Rücksprungadressen und Zwischenwerte und kann daher ebenfalls gelesen und beschrieben werden.

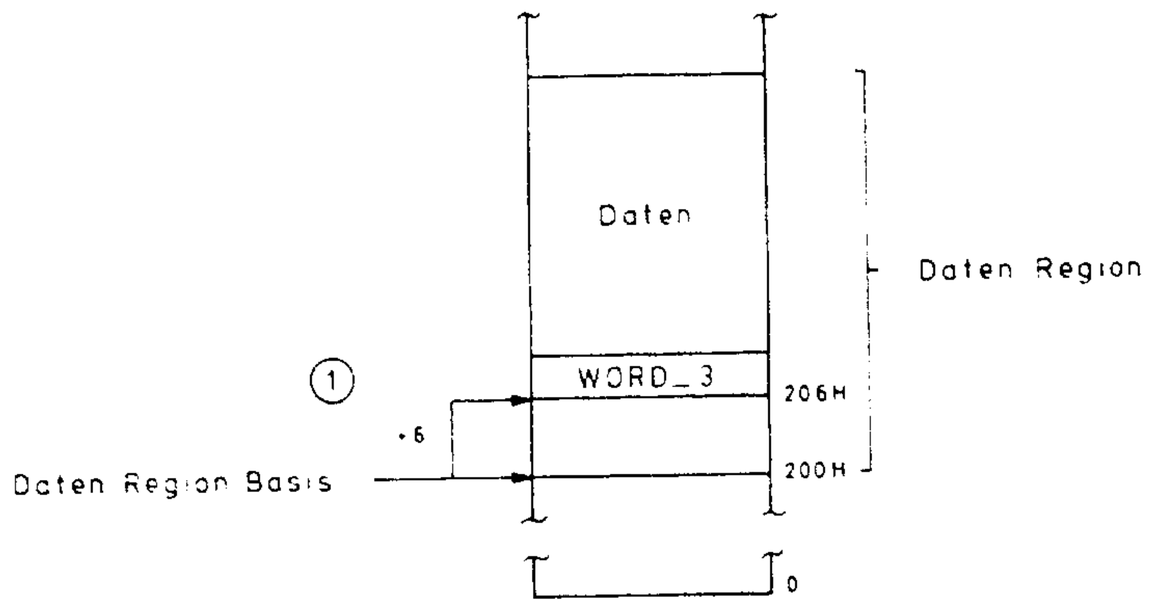
Es soll nun am Beispiel einer Daten-Region gezeigt werden, wie die Variable WORD_3 zu erreichen ist.



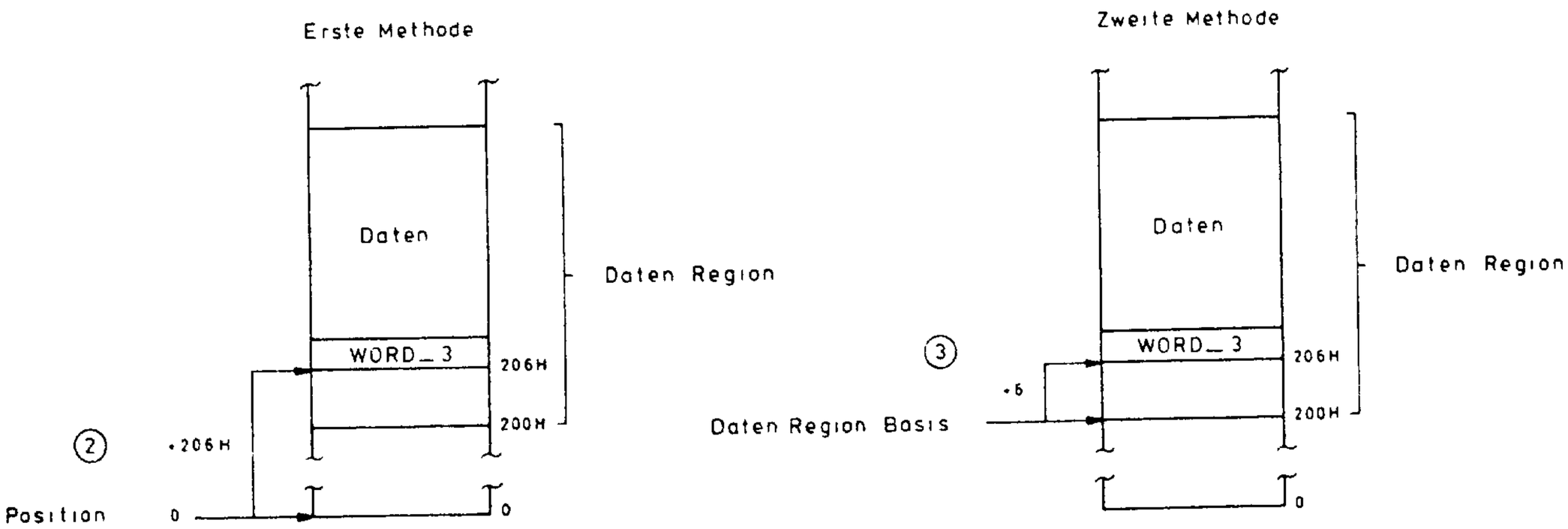
Eine Möglichkeit besteht darin, den Abstand (Offset) zwischen Position 0 und der augenblicklichen Lage der Variablen WORD_3 anzugeben. Physikalisch liegt die Variable WORD_3 bei 206H, daher beträgt der relative Abstand zur Position 0 +206H Bytes ①.

Eine alternative Methode, die Variable WORD_3 zu adressieren, ist den Abstand relativ zur bekannten Basisadresse der Datenregion anzugeben.

Im Beispiel beginnt die Daten-Region bei der Basisadresse 200H. Die Variable WORD_3 ist vom Beginn der Daten-Region im Abstand von 6 Bytes gespeichert ①. Man sagt: Die Variable WORD_3 liegt bei **Offset 6** ab der Daten-Region-Basis.



Die beiden Methoden der Variablen-Adressierung sind äquivalent.



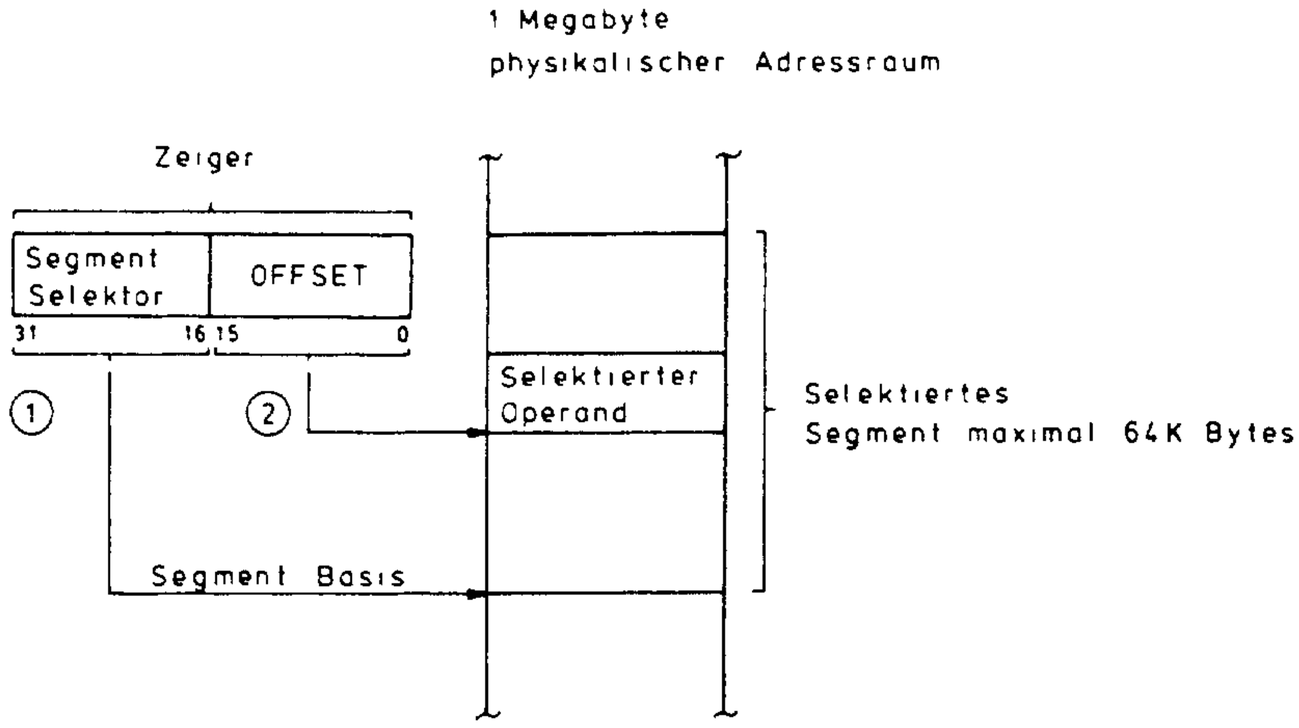
Bei der ersten Methode liegt WORD_3 bei Offset 206H ab der Basisadresse 0 ②.

Bei der zweiten Methode liegt WORD_3 bei Offset 6 ab der Daten-Region-Basis 200H ③.

In beiden Fällen hat die Variable WORD_3 die physikalische Adresse 206H.

1.6.3 Adressierung durch 8086/80186/80286

Bei den Mikroprozessoren 8086/80186/80286 gilt die Vorstellung, daß Adressierungen relativ zu bekannten Basispositionen durchgeführt werden.



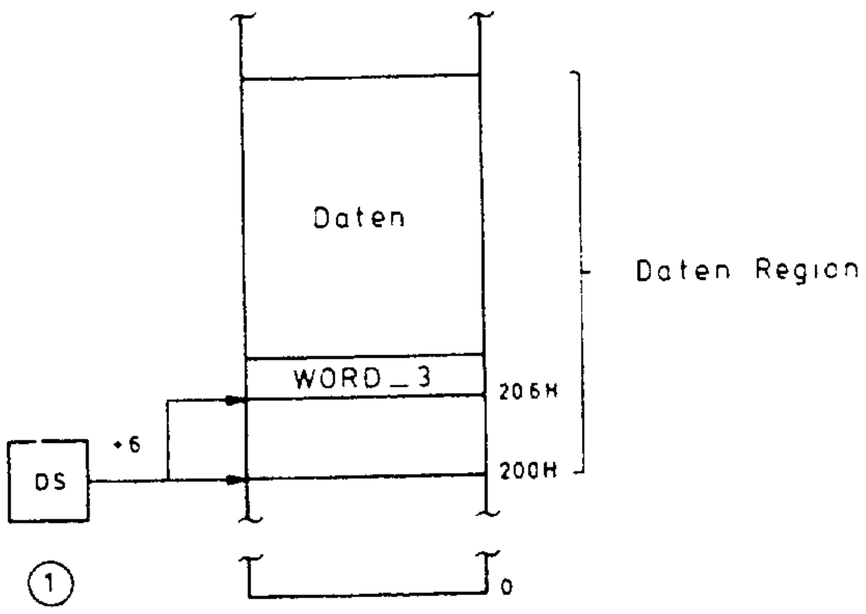
Dabei kann die physikalische Adresse eines Operanden im 8086/80186/80286-Adreßraum als ein 32-Bit-Zeiger aufgefaßt werden, der aus zwei Komponenten besteht:

- Einem Segment-Selektor (16 Bit) ① und
- dem Offset (16 Bit) ②

Die Selektorkomponente einer 32-Bit-Operandenadresse zeigt immer auf den **Anfang** eines maximal 64 KByte großen Segments im 1 Megabyte physikalischen Adreßraum ①.

Die Offsetkomponente zeigt dann innerhalb des selektierten Segments zum gewünschten Operanden ①.

Im Beispiel zeigt das DS-Register (Selektorkomponente) zum Anfang einer Daten-Region mit der Adresse 200H ①.



Die Variable WORD_3 in dieser Region kann als **"Offset 6 von DS"** ausgedrückt werden.

Um einen 32-Bit-Zeiger in Assembler anzugeben, gilt folgende Kurzschreibweise:

```
Base  :  Offset
```

Offsetkomponente (Typ Word)

Verkettungszeichen zwischen beiden Komponenten

Selektorkomponente (Typ Word)

Hinweis: Das Zeichen (:) gibt an, daß beide Komponenten zusammengehören, also ein Wertepaar sind und gemeinsam eine Operandenadresse bilden.

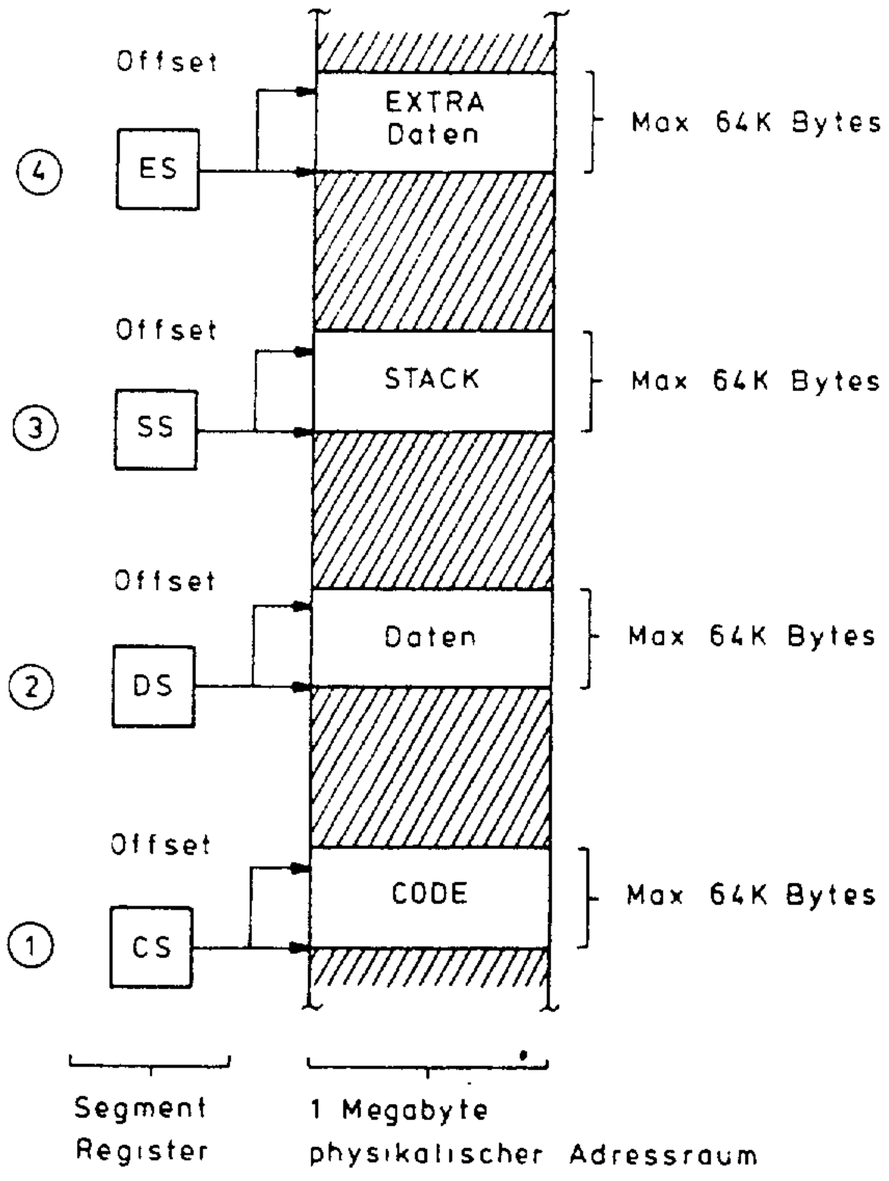
Wird die Notation Base:Offset auf das angegebene Beispiel angewendet, kann als Adresse für die Variable WORD_3 geschrieben werden:

```
DS  :  6
```

Der Offset beträgt 6 Bytes

Name des Datensegment-Selektors

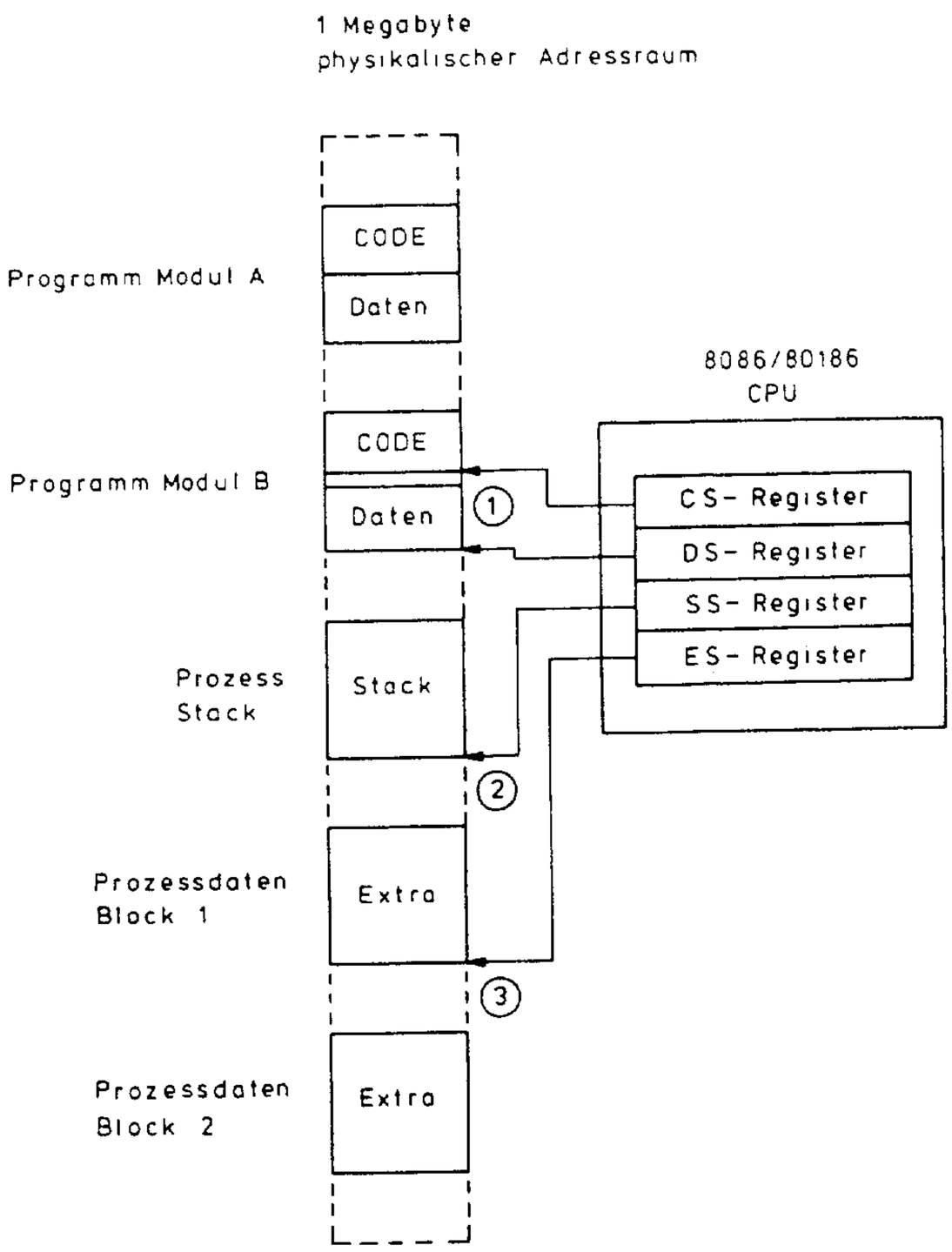
Die Mikroprozessoren 8086/80186/80286 kennen vier verschiedene Selektoren (16-Bit-Segment-Register), um die Basisadressen von vier unterschiedlichen Segmenttypen anzugeben.



- Das CS-Register wird benützt, um die Anfangsadresse einer Code-Region festzulegen ①.
- Das DS-Register zeigt auf den Anfang einer Daten-Region ②.
- Das SS-Register legt den Beginn einer Stack-Region fest ③.
- Das ES-Register kennzeichnet die Basis einer zweiten Daten-Region (Extra-Region) ④.

Hinweis: Das erste Zeichen eines Segment-Registernamens gibt den Typ des selektierten Segments an: **CS** für Code, **DS** für Daten, **SS** für Stack und **ES** für Extra. Das Zeichen "S" in jedem Namen bedeutet "Segment".

Beispiel: Anwendung der Speichersegmentierung



Es sollen zwei Programm-Module A und B existieren, die je ein Code-Segment und ein Daten-Segment enthalten.

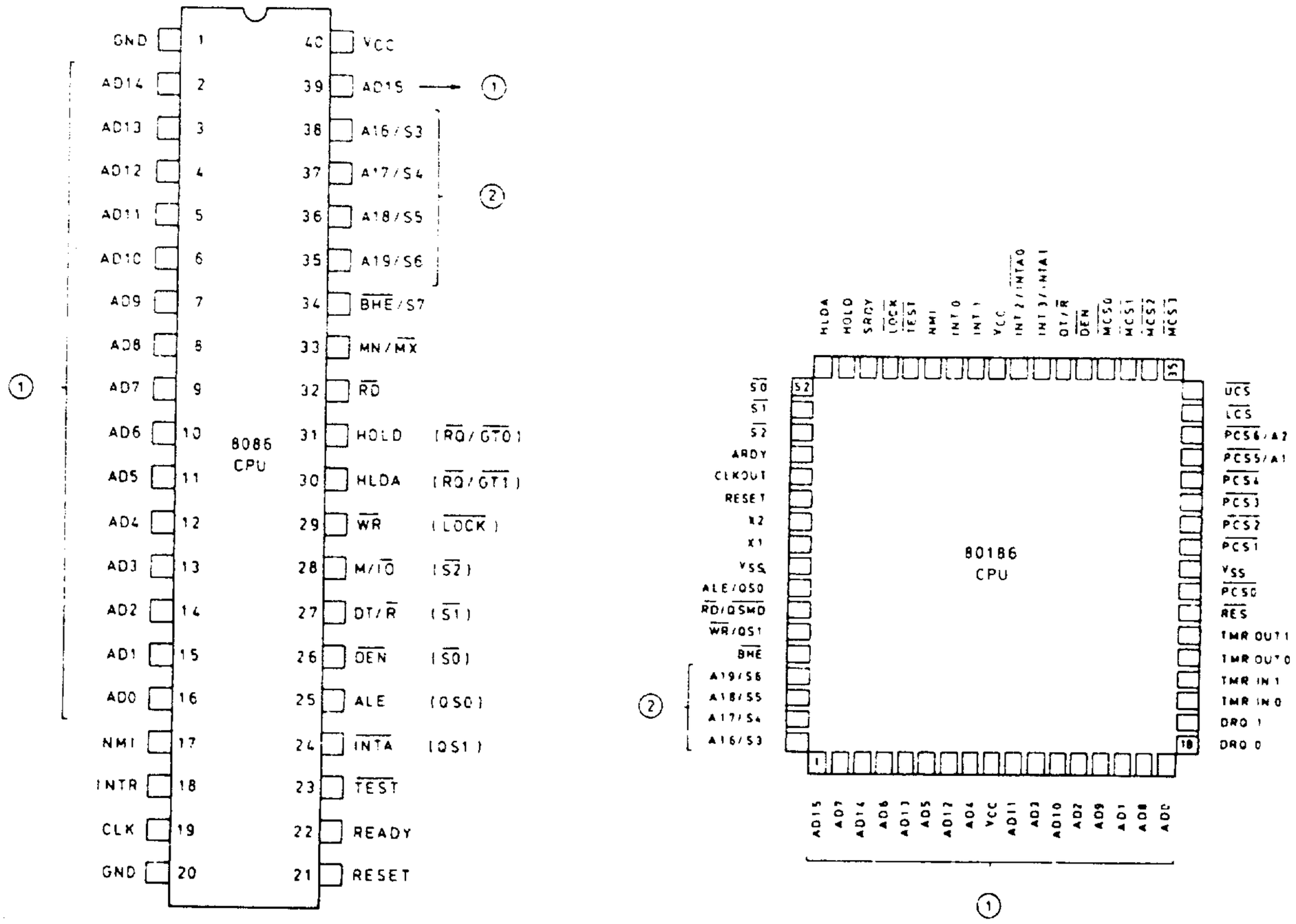
Das zu bearbeitende Modul wird durch einen entsprechenden Code-Segment-Selektor CS und den zugehörigen Daten-Segment-Selektor DS ausgewählt ①.

Beide Module beziehen sich auf einen gemeinsamen Stack-Bereich (Prozess-Stack), der durch den Stack-Segment-Selektor SS festgelegt ist ②.

Weiterhin werden externe Prozeßdaten von beiden Modulen verarbeitet. Der notwendige Prozeßdaten-Block wird durch den Extra-Segment-Selektor ES ausgewählt ③.

1.6.4 Basis-Positionen und physikalische Adressen

Um 1.048.576 verschiedene Adressen (physikalische Adressen) bilden zu können, sind 20 Adreßleitungen (A19 – A0) erforderlich.



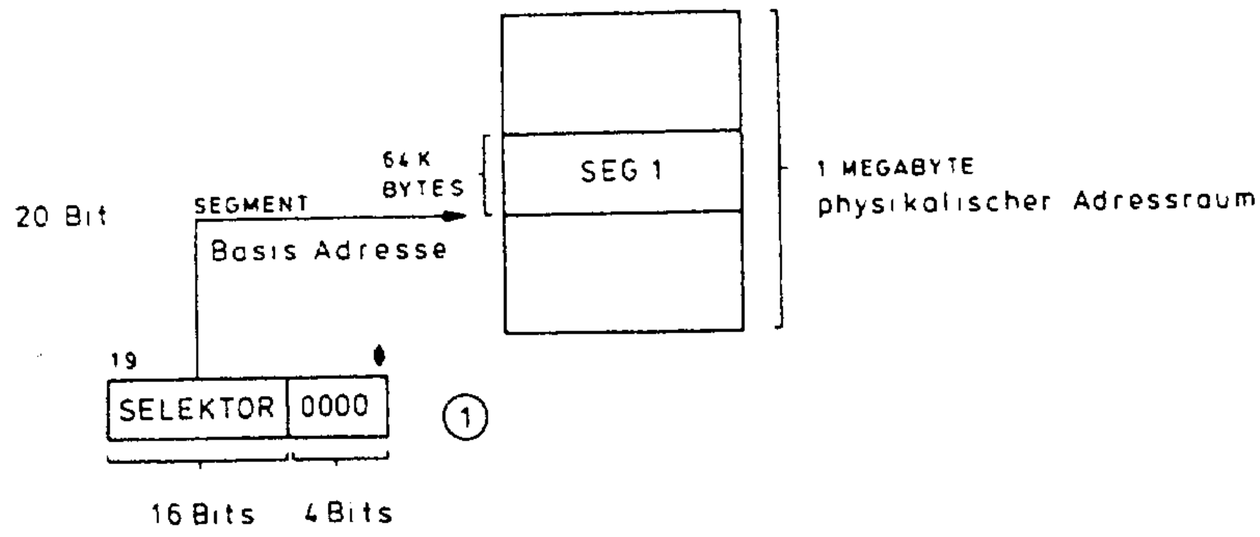
Bei den Mikroprozessoren 8086 und 80186 stehen 20 Adreßleitungen als Multiplexleitungen zur Verfügung. Es sind dies die

- Adreß-/Datenbus-Leitungen AD15 – AD0 ① und die
- Adreß-/Status-Leitungen A19/S6
  A18/S5
  A17/S4
  A16/S3 ②

Beide Prozessoren geben jeweils im Taktzyklus T1 eines Buszyklus eine 20-Bit-Adresse an die zwei Leitungssysteme aus. Dabei werden die Adressen A15-A0 an den Adreß-/Datenbus und die Adressen A19-A16 an die Adreß-/Status-Leitungen ausgegeben.

Hinweis: Beim 80286 sind die Adreßleitungen **nicht** gemultiplext.

Da die 16-Bit-Segment-Selektoren nur Daten vom Typ Word speichern können, repräsentieren sie die oberen 16 Bits einer physikalischen 20-Bit-Basisadresse.

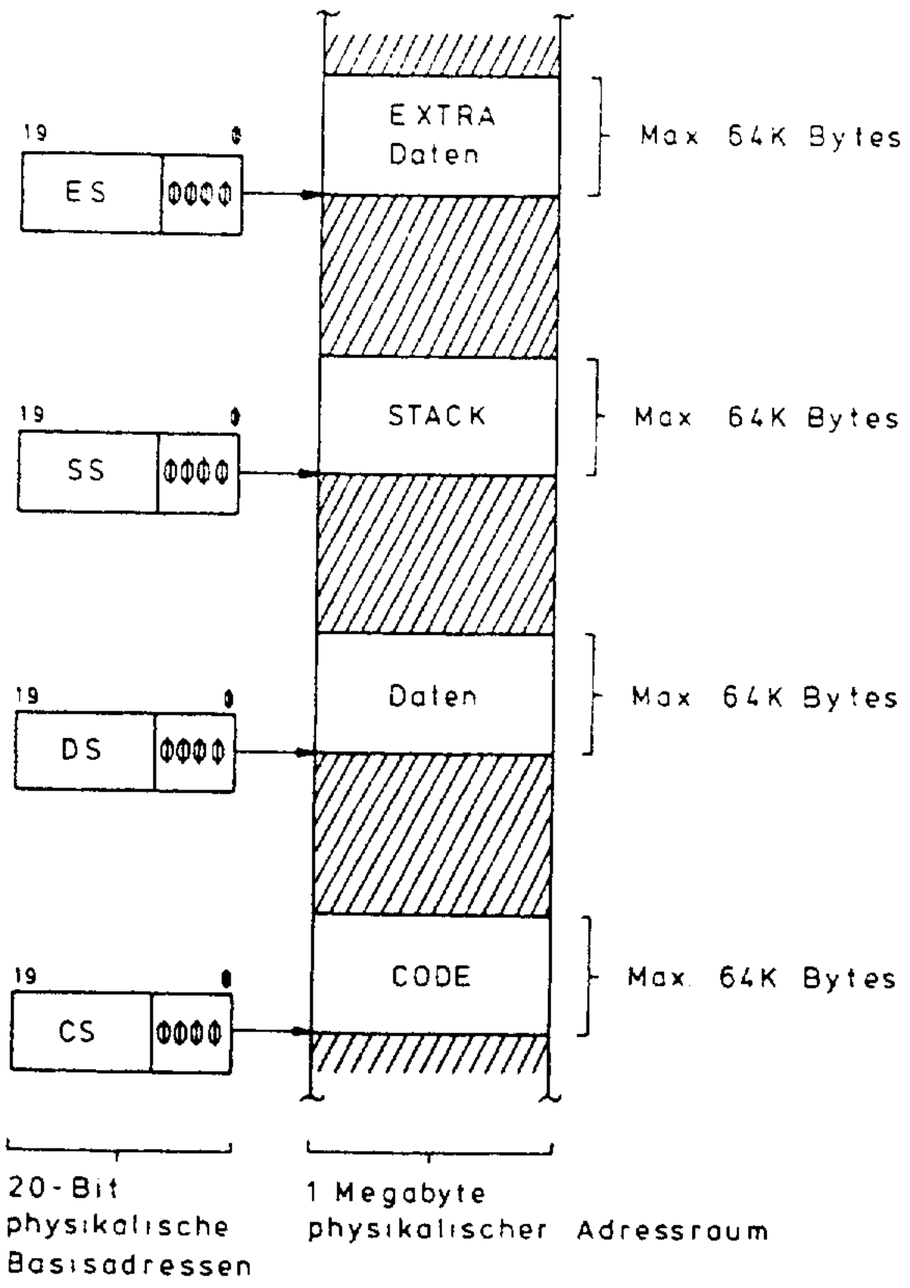


Jedes Segment-Register bildet eine 20-Bit physikalische Basisadresse, indem der augenblickliche Registerinhalt um **vier** Bitpositionen nach **links** verschoben wird. D.h. der Registerwert wird mit 16 multipliziert.

Dies bedeutet, daß die unteren 4 Bits einer physikalischen Basisadresse immer Null sind ①.

Beispiel: Enthält das DS-Register den Wert 1234H, dann zeigt DS auf ein physikalisches Segment mit der Basisadresse 12340H.

Jedes Segment im 8086/80186 Adreßraum bzw. im 80286-Real-Mode-Adreßraum ist dadurch gekennzeichnet, daß die niederwertigen vier Bits (Low Nibble) der 20-Bit-Basisadresse mit Nullen belegt sind.



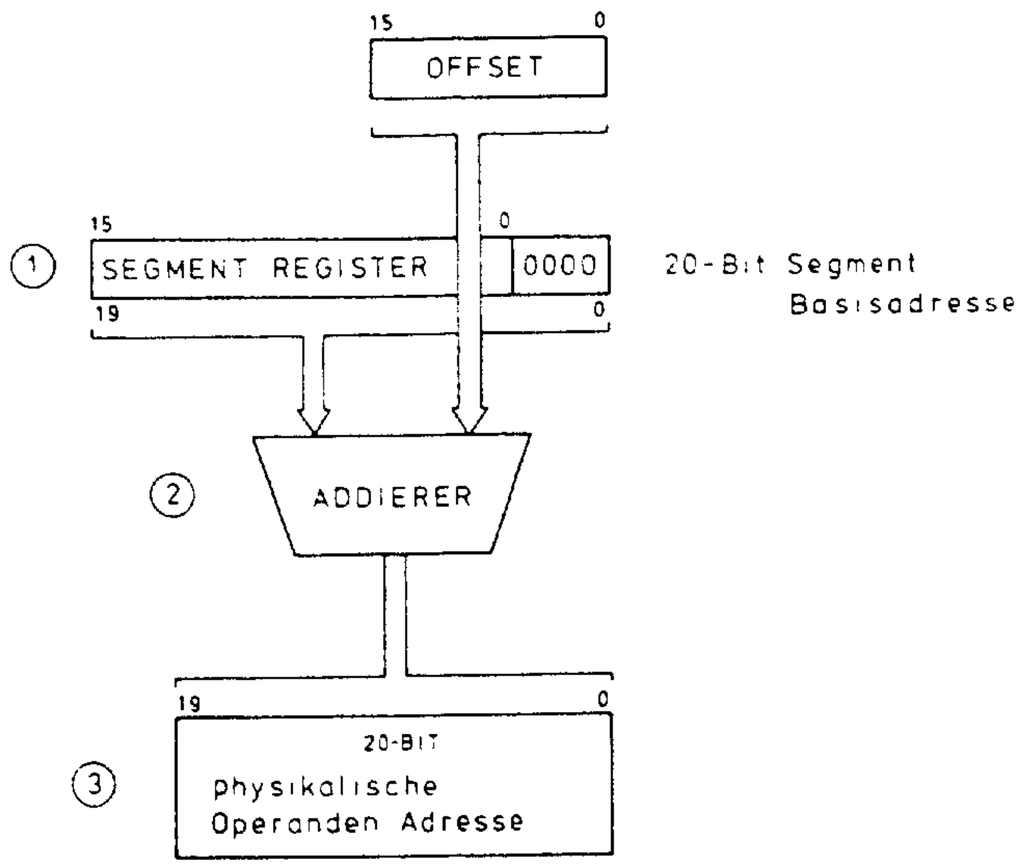
Terminologie:

Man sagt: Eine Adresse, die durch 16 teilbar ist (Low Nibble = Null), liegt an einer sogenannten Paragraphgrenze bzw. **Paragraph Boundary**.

Weil physikalische Segmente immer an einer solchen Adresse beginnen, sagt man, sie sind an Paragraphgrenzen ausgerichtet, bzw. sie sind **Paragraph-aligned**.

Da der Wert in einem Segment-Register eine bestimmte Paragraphgrenze festlegt, wird für den oberen 16-Bit-Anteil einer 20-Bit physikalischen Basisadresse der Term **Paragraph Number** benützt.

Es soll nun gezeigt werden, wie unter Anwendung des Base:Offset-Komponentenpaares die physikalische Adresse eines Speicheroperanden ermittelt wird.



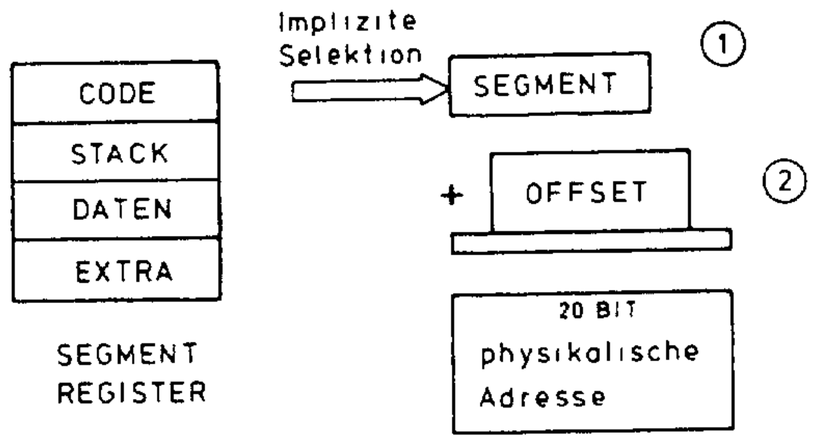
Zunächst berechnen die Mikroprozessoren 8086/80186 und 80286 im Real-Mode die 20-Bit-Segment-Basisadresse, indem sie die Paragraph Number von einem aktuellen Segment-Register mit 16 multiplizieren ①.

Anschließend wird der 16-Bit Offset zu diesem 20-Bit-Basiswert addiert ②.

Das 20-Bit-Ergebnis ③ wird nun an die Adreßleitungen A19-A0 ausgegeben und damit eine der 1.048.576 Bytepositionen spezifiziert.

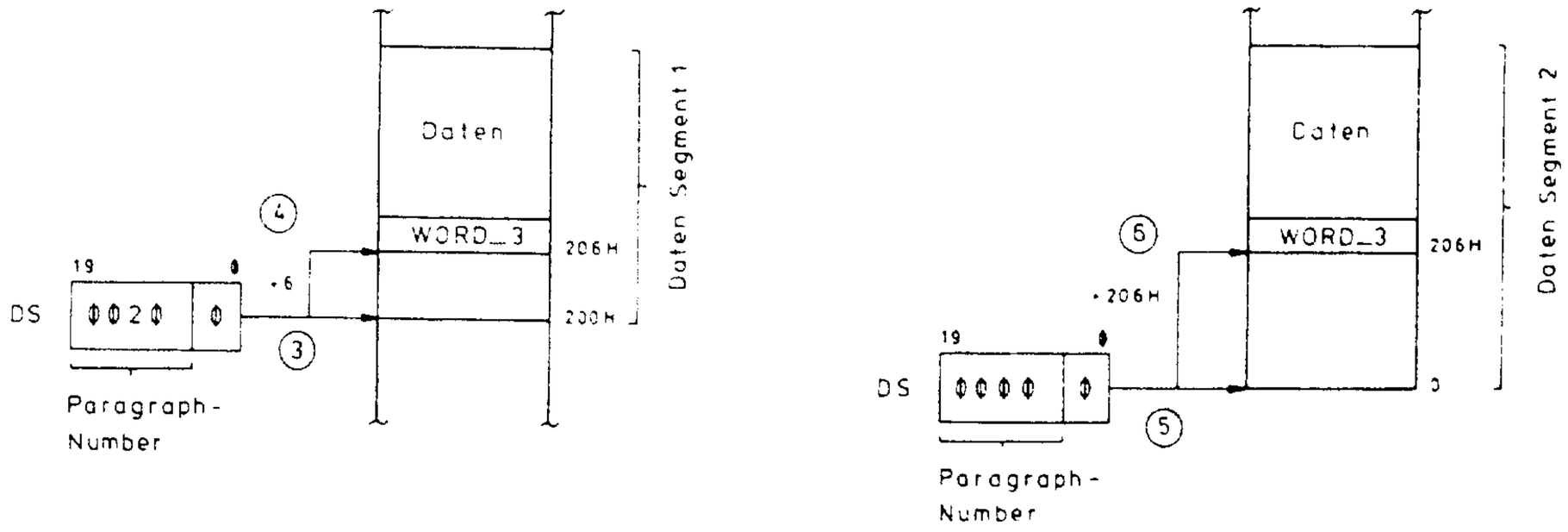
Die Berechnung einer physikalischen Adresse aus der Selektorkomponente (Paragraph Number) und der Offsetkomponente ist eine Funktion, die von den 8086/80186/80286-CPUs automatisch geleistet wird.

Das angegebene Berechnungsverfahren zur Ermittlung physikalischer Positionen innerhalb eines selektierten Segments gilt für alle Segmenttypen.



Ist von einem bestimmten Segment-Register das **zugehörige** Segment ausgewählt worden (Implizite Selektion) ①, sind für den Assembler-Programmierer nur noch die Offset-Werte von Interesse ②.

Beispiel:



In beiden Daten-Segmenten liegt die Variable WORD_3 bei der physikalischen Adresse 206H.

Das Daten-Segment 1 beginnt physikalisch bei 00200H (0020H * 16) ③. Dadurch liegt die Variable WORD_3 bei Offset 6 ④.

Das Daten-Segment 2 beginnt physikalisch bei 00000H (0000H * 16) ⑤. Dadurch liegt die Variable WORD_3 bei Offset 206H ⑥.