Die komplexen Befehle

Diese Befehle setzen sich zusammen aus:

- einem Operationstyp
- einem Nebenschlüssel N, der die Anzahl der durchzuführenden Operationen angibt
- den Adressen der entsprechenden Ein-/Ausgabemedien.

Zu diesen Befehlen gehören:

- die Befehle der Mehrfachüberträge, bei denen die Adressen durch die Verschiebungen D (die Basisregister sind einbegriffen) oder durch die Nummer R des numerischen Registers dargestellt werden.
- die Ein-/Ausgabebefehle, bei denen die Randeinheiten durch Spezialcodes C und die Ein-/Ausgabezonen im Zentralspeicher durch die Nummern R der numerischen Register identifiziert werden.

Die speziellen Formate dieser Befehle werden in Abschnitt 3.5 und 3.6 beschrieben.

Format der Parameter

. Der Operationstyp ist ein Binärschlüssel und belegt ein Byte.

. Der Nebenschlüssel N ist eine zweistellige Dezimalzahl in gepackter Form auf einem Byte.

. Der Nebenschlüssel K ist ein Binärwert oder eine Kombination des IS$\emptyset$-Codes auf einem Byte.

. Die Adressen belegen in der Form von zwei-, drei- oder vierstelligen gepackten Dezimalzahlen ein oder zwei Bytes.

Die Art der Adressierung ist dabei gleichgültig:

- absolute Adressierung: 4 Ziffern (2 Bytes) bezeichnen die Nummer eines Bytes.
- Adressierung der Register: 2 Ziffern (1 Byte) bezeichnen die Nummer eines numerischen Registers oder eines Basisregisters, das die Adresse einer Speicherzone enthält.
- Adressierung einer Sektion in einer Speicherzone:
  . mit ausführlicher Basis: 4 Ziffern (2 Bytes), von denen die 1. die Nummer des Basisregisters und die drei anderen die Verschiebung darstellen.

mit einbezogener Basis: - 2 Ziffern (1 Byte) stellen eine Verschiebung ⩽ 99 dar,

- 3 Ziffern mit vorangestelltem Buchstaben A (2 Bytes) stellen eine Verschiebung zwischen 100 und 999 dar.

. Der Anschlußcode C ist ein Binärcode und belegt 1 Byte.

Die Länge der Befehle ist variabel, aber immer gleich einer Zahl ganzer Bytes. Sie wird bestimmt durch die Analyse des Operationstyps und des Nebenschlüssels bei komplexen Befehlen.

3.1.2.2. Die Befehle der Grundsprache

Die Elementarbefehle

Die Elementarbefehle, aus denen ein Programm besteht, werden vom Programmierer im Format der Maschinenbefehle geschrieben, die ihnen entsprechen. Wie die Maschinenbefehle besitzt auch der Elementarbefehl einen Operationstyp, evtl. einen Nebenschlüssel und eine oder mehrere Adressen.

. Der Operationstyp wird symbolisch durch einen numerischen Schlüssel dargestellt, der sich aus Buchstaben und Ziffern für bestimmte Befehle zusammensetzt.

Beispiel:

	OT symb.	Binärcode
- Addition in einfacher Länge	ADD	1010 0001

Der Binärcode des OT in Maschinensprache erscheint bei einem Kernspeicherausdruck oder bei Auflisten eines Programms in Form von zwei hexadezimalen Zeichen nach den Normen, die in Abschnitt 2.1.2.1 beschrieben wurden. (z.B. 1010 0001 = A1)

. Der Nebenschlüssel N, dargestellt durch Dezimalziffern, kann eine Zahl von 00 bis 99 sein. Bei bestimmten Befehlen kann dieser Wert jedoch niedriger sein.

. Der Nebenschlüssel K und die Anschlußcodes C werden dargestellt durch zwei hexadezimale Zeichen (s. 2.1.2.1). Die Tafel des internen Codes in Kapitel 6 zeigt den Zusammenhang zwischen den Binär- und hexadezimalen Kombinationen.

Beispiele:

1) K = A4; K = 31; K = BC
2) Anschlußcode Drucker MB = CA

. Die Adressen werden durch Dezimalziffern nach den gleichen Bedingungen wie bei den Maschinenbefehlen dargestellt.

Beispiele:

- absolute Adresse: 4021 bezeichnet das Byte 4021 im Zentralspeicher
- Adresse der Register: 04 bezeichnet das 4. numerische Register.
- Basisadresse: 4021 bezeichnet als Basisregister: Register 4 und als Verschiebung 21. Enthält Reg. 4 z.B. 2320, so lautet die absolute Adresse: 2341 (2320 + 21).
- Adresse mit einbezogener Basis: 62 bezeichnet eine Verschiebung von 62 Bytes; A 105 eine von 105 Bytes. Die einbezogene Basis ist eins der Register 6-9. Enthält Reg. 8 z.B. 2114, so ist die absolute Adresse im 1. Fall: 2176 und im 2. Fall: 2219.

Der Anfang der Sektionen eines Programms wird durch ein symbolisches Bezugszeichen gekennzeichnet, das sich aus zwei hexadezimalen Zeichen (*) zusammensetzt. Dieses Symbol bezeichnet bei Verwendung in einem Pseudobefehl die Adresse des ersten Befehls einer Sektion.

(*) Anmerkung: Die Kombinationen F1 - FF sind verboten. Sollte der Programmierer eines dieser Zeichen unbedingt verwenden müssen, ist es erforderlich, innerhalb des Programms eine Serie zu programmieren, in der diese Zeichen durch logische Operationen zusammengesetzt werden. Dieser Vorgang ist nicht anwendbar bei den symbolischen Bezugszeichen. Es bleiben jedoch 240 Kombinationen zur freien Verfügung.

Die Pseudobefehle

Die Pseudobefehle werden in der gleichen Weise geschrieben wie die Elementarbefehle und enthalten einen symbolischen Operationstyp und evtl. symbolische Parameter.

3.1.3. Darstellung der Befehle

Die in den folgenden Abschnitten beschriebenen Befehle sind in folgende Gruppen eingeteilt:

- Befehle zur Programmverknüpfung (3.2)
- Registeroperationen (3.3)
- Operationen auf Zeichenbasis (3.4)
- Mehrfachüberträge (3.5)
- Ein/Ausgabeoperationen (3.6)

- Umschlüsselungsbefehle (3.7.)
- Hilfsbefehle (3.8.)
- Pseudobefehle (3.9.)

Für jede Befehlsgruppe enthält ein Einführungsparagraph die gemeinsamen Charakteristika.

Die besonderen Eigenschaften jedes Befehls werden in folgender Weise dargestellt:

- symbolischer Operationstyp und englische Bezeichnung,
- Befehlsformat in Maschinensprache: Operationscode und Ergänzungsparameter,
- Länge des Befehls in Bytes,
- besondere Normen der Anwendung und der Parametrierung im gegebenem Fall,
- Wirkung, Anwendung und Besonderheiten,
- Beispiele.

Falls mehrere Befehle identische Eigenschaften haben, werden sie in einer gemeinsamen Beschreibung zusammengefaßt. Am Anfang des Abschnittes sind in einer Tabelle die verschiedenen symbolischen Operationstypen und die entsprechenden OT im Maschinencode angegeben.

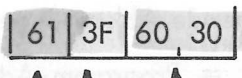
Innerhalb der Beispiele sind die Werte, die im Speicher enthalten sind, wie folgt dargestellt:

- der größte Teil in hexadezimaler Form (1 Zeichen je Byte),
- einige in Binärdarstellung (acht Ziffern (0/1) je Byte),
- einige in Klartext (2 Zeichen je Byte).

Eine Information oder ein Befehl besteht aus einer Folge von Zeichen, die immer in ein Schema gezeichnet sind, das die Speicherstellen (oder Bytes) darstellt, das diesen Wert enthält. Die interessanten Zonen (Register, Information) sind mit Markierungen versehen.

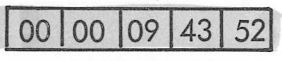
Beispiel:

Ein Befehl:



Byte 1 : OT
Byte 2 : K
Byte 3 + 4 : Adresse

Ein Register: (5 Bytes)



Am Ende des Handbuches geben Tabellen Auskunft über die Ablaufzeit eines Befehles und verweisen auf die entsprechenden Abschnitte , in denen der Befehl beschrieben wurde.

Definitionen

Die verschiedenen Elemente eines Befehls werden durch folgende Zeichen dargestellt:

OT = Operationstyp

N = Nebenschlüssel dezimal, oder zwei gepackte Dezimalziffern.

K = Nebenschlüssel hexadezimal (Binärwert) oder zwei gepackte hexadezimale Zeichen.

C = Anschlußcode (hexadezimal) einer Randeinheit.

R = Nummer eines Registers (2 Dezimalziffern)

A = 1 alphanumerisches ungepacktes Zeichen

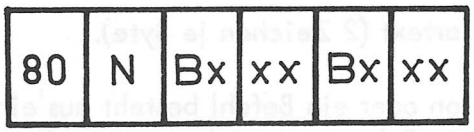
xxxx = absolute Adresse (dezimal)

Bxxx = Basisadresse (dezimal) oder:
B = Nummer des Basisregisters, das die Basisadresse enthält
xxx = Verschiebung oder Ergänzung der Adresse. Dieser Wert wird zur Basis addiert, um die absolute Adresse zu erhalten.

Dn = Verschiebung (dezimal) in bezug auf eine Basis n.

Die Befehle werden so eingerahmt, daß ihr Format und die Anzahl der Bytes sichtbar werden.

Beispiel:



Binärcodes des Operationstypes

Folgende Abkürzungen werden ebenfalls verwendet:

PAR = Programm-Adreß-Register

ANR = Adresse nach Rücksprung

3.2. Die Befehle zur Programmverknüpfung

3.2.1. Adressierung der Sprünge

Der Programmierer hat die Wahl zwischen zwei Formulierungen: Er kann die Sprünge mit reellen Adressen oder mit Symbolen programmieren.

a) reelle Adressen

Der Programmierer prüft die effektive Einspeicherung des Programmes im Zentralspeicher, um die Adresse jedes Sprunges zu bestimmen. Die anzugebende Adresse ist die Stelle, die der Operationstyp des Befehls belegt, mit dem nach dem Sprung fortgefahren werden soll.

b) Symbole

Der Programmierer springt aus dem Programm auf Sprungniveaus, die durch den Pseudobefehl LEVEL (Abschnitt 3.9.) dargestellt werden. Jedes Niveau erhält ein besonderes Etikett, das die Sprungadresse ersetzt.

Der Programmierer setzt dieses Symbol in den Sprungbefehl in folgender Weise ein:

| F1 | K |

F1 = festes Zeichen, das besagt, daß es sich um ein Bezugszeichen handelt.

K = ein Zeichen in hexadezimaler Form, das dieses Bezugszeichen darstellt. Die Kombinationen F1 bis FF sind verboten.

Um von der Maschine verarbeitet werden zu können, werden die Symbole durch reelle Adressen des Sprungniveaus ersetzt. Dieser Austausch findet beim Laden des Programmes in den Kernspeicher durch das Standardladeprogramm statt.

Wahl der Methode

Die erste Methode setzt voraus, daß das Programm völlig fertiggestellt ist, damit die reellen Sprungadressen festgelegt werden können. Weiterhin ist die Veränderung eines Programmteiles mit einer Revision auch der Sprungbefehle, die nicht direkt berührt werden, verbunden. Das ist eine ständige Quelle des Ärgers und der Schwierigkeiten.

Die zweite Methode enthält nicht die vorgenannten Unbequemlichkeiten.

Die symbolische Adressierung kann in dem Augenblick festgelegt werden, in dem das Programm entsteht. Der Programmierer setzt die Sprungniveaus in das Flußdiagramm ein und ordnet die Bezugszeichen zu. Nach Maßgabe

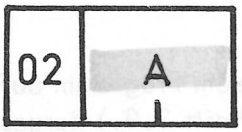
des Programmes schiebt er die Pseudobefehle LEVEL ein und vermerkt die Symbole in den Sprungbefehlen.

Spätere Veränderungen des Programmes bleiben ohne Einfluß auf die Sprungbefehle, wenn sie nicht gerade selbst benutzt werden. In diesen Fällen werden die neuen reellen Adressen automatisch während des Ladens des veränderten Programmes in den Kernspeicher festgelegt.

3.2.2. Sprungbefehle

3.2.2.1. Unbedingter Sprung

J R T	Jump; save adress for Return



A (Sprungadresse)	reelle Adresse	: | xx | xx |
	symbolische Bezugszeichen	: | F1 | K |

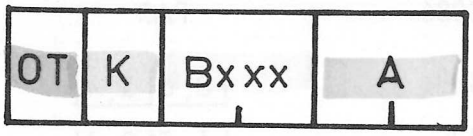
Der Inhalt des Programm-Adress-Registers wird im Register der Adresse bei Rücksprung (ANR) gespeichert (Adresse des dem JRT folgenden Befehls). Die im JRT festgelegte Sprungadresse wird in das PAR übertragen. Das Programm fährt mit dieser Adresse fort.

Beispiele:

Adresse	Programm		PAR	ANR
734	JRT | 0 2 | 1 2 | 8 7 |	vorher:	0734	0000
737	ADD | A 1 | 3 2 | 1 6 |	nachher:	1287	0737

3.2.2.2. Bedingte Sprünge

JIERT	Jump if Immediate octet Equal to store octet; save adresse for ReTurn
JIURT	Uniqual



OT = 04 JIERT
05 JIURT

K = Bezugszeichen

Bxxx = Adresse der zu untersuchenden Stellen

A = Sprungadresse { reelle Adresse |xx |xx|
Etikett des Niveaus |F1 |xx|

Der Programmsprung ist abhängig vom Vergleichsergebnis zwischen dem Zeichen K und dem Zeichen, das sich an der Adresse Bxxx befindet.

- bei JIERT wird der Sprung ausgeführt bei Gleichheit zwischen den beiden Zeichen.

- bei JIURT wird der Sprung durchgeführt, wenn Ungleichheit zwischen den Zeichen besteht.

a) Bei Sprungausführung

Der Inhalt des PAR wird im ANR gespeichert (Adresse des Befehles, der auf den Befehl JIERT oder JIURT folgt)

Die Sprungadresse des JIERT oder JIURT wird in das PAR übertragen und das Programm fährt an der Stelle fort, die dieser Adresse entspricht.

b) Bei Nichtausführung des Sprunges

Der Inhalt des PAR bleibt unverändert und das Programm fährt mit dem Befehl, der dem JIERT oder JIURT folgt, fort.

Beispiele:

a) JIERT (Sprung wird ausgeführt)

Adresse 654 |0,4|3,E|0,0|8,6|1,5|0,1| Basis 0 = 0000

	Stelle 0086 (VG)	PAR	ANR
vorher	: 3 E	|0,6|5,4|	|x,x|x,x|
nachher	: 3 E	|1,5|0,1|	|0,6|6,0|

b) JIURT (Sprung wird nicht ausgeführt)

Adresse 654 |0,5|0,0|0,0|9,1|1,2|1,8| Basis 0 = 0000

	Stelle 0091 (Kap)	PAR	ANR
vorher	: 0 0	|0,6|5,4|	|x,x|x,x|
nachher	: 0 0	|0,6|6,0|	|x,x|x,x|

3.3. Registeroperationen

3.3.1. Allgemeines

Die Operationen, die in den Registern durchgeführt werden, beziehen sich im wesentlichen auf die Verarbeitung von numerischen Daten, die in gepackter, algebraischer Form dargestellt werden. (s. 2.1.2.2.) Die Verarbeitung erfolgt in einfacher oder doppelter fester Länge (s. 2.1.3.3.).

Die Befehle dieser Gruppe enthalten:

- Rechen- und algebraische Vergleichsbefehle,
- Versetzungs- und Rundungsbefehle,
- Übertragungsbefehle.

Ihre Beschreibung und die angeführten Beispiele beziehen sich nur auf Dezimalzahlen.

Die Benutzung dieser Befehle setzt die folgenden allgemeinen Richtlinien voraus.

Löschen der Register

Es besteht kein besonderer Löschbefehl.
Ein Löschen der Register besteht darin, jedes Halbbyte auf Null zu setzen.

Man erreicht dies, indem man das Register mit dem Zeichen für "Nichts" (00) belegt, entweder durch den Befehl "Einsetzen eines Zeichens" -INC- (s. 3.4.3.2.) oder durch den Übertrag des Inhaltes eines anderen Registers, das nichts enthält.

Da die Zeichen "0" (30) und "Leerzeichen" (20) ein Halbbyte ungleich Null enthalten, können diese Zeichen nicht benutzt werden.

Versetzung

Die Rechenbefehle und der Vergleich erfordern Operanden, die in den numerischen Registern rechtsbündig stehen. Wird ein Operand nach links verschoben, so daß die rechten Stellen mit Null aufgefüllt werden, wird der Wert so behandelt, als ob er mit 10^n multipliziert worden wäre.
n entspricht der Anzahl der Nullen rechts.

Die Rechenergebnisse stehen immer rechtsbündig in den Registern.

Kapazitätsüberschreitung

Je nachdem es sich um ein Einfach- oder Doppelregister handelt, kann ein numerisches Register 9 oder 19 Ziffern und ein Vorzeichen (C oder 9) enthalten.

Es ist Aufgabe des Programmierers darauf zu achten, daß die Rechenoperanden die besonderen Normen jedes Befehls beachten, so daß sie niemals die erlaubte Maximalkapazität überschreiten. Die Kapazität, die man für einen Operanden in Betracht ziehen muß, ist gleich der Maximalzahl von Ziffern, die er enthält, zuzüglich eventueller Nullen rechts im Register.

Beispiel:	Register	zu beachtende Kapazität
Operand rechtsbündig:	0,0,0,0,0,x,x,x,x,x	5 Ziffern
Operand versetzt:	0,0,x,x,x,x,x,0,0,0	8 Ziffern

Am Ende der Rechenoperationen prüft die Maschine das erhaltene Ergebnis. Diese Kontrolle besteht darin, den Wert der Vorzeichenstelle des Resultats zu analysieren. Ein Wert ungleich 0 oder 9 zeigt an, daß es sich um eine Kapazitätsüberschreitung handelt und daß dadurch das Resultat unauswertbar ist (falsches Vorzeichen). In diesem Fall wird ein Testzeichen in das Register "Kapazitätsüberschreitung" gesetzt.

Bei der Division wird dieses Testzeichen dargestellt durch AA. Bei den anderen Operationen ist es ein unbestimmbarer Wert; nämlich der Inhalt des äußersten linken Bytes des Resultats. Dieser Inhalt ist immer ungleich 00 und variabel.

Um sich von der Art des Resultats zu überzeugen, geht der Programmierer nach folgender Methode vor:

- Vor der Rechenoperation setzt er das Register "Kap" auf Null (00).

- nach der Operation analysiert er diese Stelle durch einen Vergleich gegen 00, um festzustellen, ob sich der Inhalt geändert hat.

Diese Methode ist unabhängig vom Wert des Testzeichens gültig.

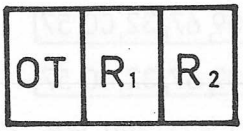
Nicht dezimale Werte

Das Vorhandensein von nicht dezimalen Werten (größer 9) in den Registern wird von den Befehlen nicht festgestellt. Sie laufen ohne einen eventuellen Fehler anzuzeigen ab. Bei den Rechen- und Rundungsbefehlen erhält man falsche Resultate. Bei Versetzungsbefehlen dagegen kann es geduldet werden.

3.3.2. Rechenbefehle und algebraischer Vergleich

3.3.2.1. Additionen und Subtraktionen

ADD	ADd Decimal single length	A 1
ADDD	ADd Decimal Double length	A 2
SBD	SuBtract Decimal single length	B 1
SBDD	SuBtract Decimal Double length	B 2



OT : siehe Maschinencode in obenstehender Tabelle

R1 und R2 : Einfach- oder Doppelregister je nach OT

Der Inhalt des Registers 1 wird auf den Inhalt des Registers 2 addiert oder von ihm subtrahiert. Das Resultat steht in Register 2. Wenn Register 1 ein anderes als Register 2 ist, wird sein Inhalt durch die Operation nicht verändert.

Der Befehl wird algebraisch durchgeführt.

Wenn R1 und R2 das gleiche Register bezeichnen, wird der Inhalt dieses Registers verdoppelt oder gelöscht, je nachdem, ob eine Addition oder Subtraktion vorlag.

Die Operation bezieht sich auf Register von

- 5 Bytes (9 Ziffern und Vorzeichen) bei ADD und SBD (einfache Länge)

- 10 Bytes (19 Ziffern und Vorzeichen) bei ADDD und SBDD (dopp. Länge)

Überschreitet das Ergebnis 9 oder 19 Ziffern, liegt Kapazitätsüberschreitung vor. Das Resultat bleibt erhalten, das höchste Byte wird aber in das Register Kap (0091) übertragen.

Beispiele:

A) Addition, einfache Länge: ADD | A1 | 27 | 35 |

	Reg. Kap.	Reg. 27	Reg. 35
a) vorher:	00	00,76,93,21,69	99,99,83,29,67
nachher:	00	00,76,93,21,69	00,76,76,51,36

(+ 76.932.169) + (- 167.033) = (+ 76.765.136)

Es fand keine Kapazitätsüberschreitung statt.

b) vorher:	00	09,67,32,00,57	03,99,50,63,39
nachher:	13	09,67,32,00,57	13,66,82,63,96

(+ 967.320.057) + (+ 399.506.339) = (+ 1.366.826.396)

Es fand eine positive Kapazitätsüberschreitung statt.

B) Substraktion, einfache Länge: SBD | B1 | 50 | 97 |

	Reg. Kap.	Reg. 50	Reg. 97
a) vorher:	00	00,76,93,21,69	00,00,16,70,33
nachher:	00	00,76,93,21,69	99,23,23,48,64

(+ 167.033) - (+ 76.932.169) = (- 76.765.136)

Es fand keine Kapazitätsüberschreitung statt.

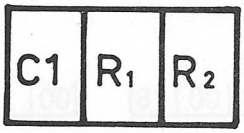
b) vorher:	00	07,36,29,92,02	95,21,07,63,22
nachher:	87	07,36,29,92,02	87,84,77,71,20

(- 478.923.678) - (+ 736.299.202) = (- 1.215.222.880)

Es fand eine negative Kapazitätsüberschreitung statt.

3.3.2.2. Multiplikation

MPD — Multi Ply Decimal



R1 = Einfachregister Multiplikator

R2 = Doppelregister Multiplikand

Der Inhalt des Doppelregisters R2 wird multipliziert mit dem Inhalt des Einfachregisters R1. Das Produkt befindet sich am Ende der Operation im Register 2. Ist das Register 1 ein anderes als Doppelregister 2, so wird sein Inhalt durch die Operation nicht verändert.

Die Operation erfolgt algebraisch.

Größe der Faktoren:
- Multiplikator : 1 - 9 Ziffern + VZ
- Multiplikand : 1 - 19 Ziffern + VZ
- Produkt : 1 - 19 Ziffern + VZ

Die Maximalkapazität der Operanden errechnet sich nach der Formel:

$M + m \leqslant 19$

M = Anzahl Stellen Multiplikand

m = Anzahl Stellen Multiplikator

Ansonsten gelten die Regeln des Abschnittes 3.3.1.

Überschreitet das Produkt 19 Ziffern, liegt Kapazitätsüberschreitung vor. Man kann es jedoch nur erkennen, wenn die 20. Ziffer ungleich 0 oder 9 ist. Das Resultat bleibt erhalten, das äußerste linke Byte des Registers 2 wird jedoch in das Register "Kap" (0091) übertragen. Überschreitet das Produkt 20 Stellen, gehen die überzähligen Stellen verloren.

Ist R1 = R2 wird die Zahl, die sich in R1 befindet, quadriert. Diese Zahl kann 9 Stellen nicht übersteigen. R1 kann auch gleich R2 - 1 sein.

Beispiele:

MPD | C1 | 59 | 32 |

Reg. 59 Reg. 31 und 32

a) +x+

vorher : |00|00|00|00|18| |00| ... |00|08|34|79|

nachher : |00|00|00|00|18| |00| ... |01|50|26|22|

0251

(+ 83.479) x (+ 18) = (+ 1.502.622)

b) +x-

vorher : |99|99|99|99|82| |00| .... |00|08|34|79|

nachher : |99|99|99|99|82| |99| .... |98|49|73|78|

(+ 83.479) x (- 18) = (- 1.502.622)

3.3.2.3. Division DVD

Für die Division existiert ein Unterprogramm, das vom Benutzer frei in den Kernspeicher eingesetzt werden kann. Es muß in den Programmkartensatz zwischen KA17 und KA31/32 eingefügt werden.
Die UP-Karten sind aufsteigend von 0001 bis 0014 numeriert, die Reihenfolge darf nicht geändert werden.
Das UP benötigt 249 Kernspeicherstellen. Die Adresse der UP-Divission wird vom Ladeprogramm auf 2 Bytes gespeichert; und zwar bei einer Kernspeichergröße von

2500 Bytes auf die Stellen 2300 und 2301,
5000 " " " " 4800 und 4801,
10000 " " " " 9800 und 9801.

Der Dividend muß im Doppelregister (13) 14, der Divisor im Register 15 gespeichert sein. Das Register 00 muß stets Null sein.

Der Anruf dieses Unterprogramms erfolt automatisch durch den Divisionsbefehl:

DVD DiVide Decimal

| D1 | R_1 | R_2 |

R1 = Einfachregister (Divisor) = R 15
R2 = Doppelregister (Dividend) = RR 14

Der Inhalt des Doppelregisters R2 (RR14) wird durch den Inhalt des Einfachregisters R1 (R15) dividiert.

Ergebnis:
- Der Quotient steht in den 5 linken Bytes (Einfachregister mit der Nummer R2 - 1 = R13)
- Der Rest steht in den 5 rechten Bytes (Einfachregister mit der Nummer R2 = R14)

Nach der Operation ist der Dividend (RR14) verändert; der Divisor (R15) bleibt erhalten.

Wenn das Register R1 ein anderes als das Doppelregister R2 ist, wird sein Inhalt durch die Operation nicht verändert.

Die Division erfolgt nicht algebraisch. Dividend und Divisor müssen positiv sein.

Größe der Faktoren:
- Dividend : 1 - 17 Ziffern + VZ
- Divisor : 1 - 8 Ziffern + VZ
- Quotient : 1 - 9 Ziffern + VZ

Die Maximalgröße der Operanden errechnet sich nach folgender Formel:

$$D - d + 1 \leqslant 9$$

- D = Anzahl Stellen Dividend
- d = Anzahl Stellen Divisor

Ansonsten gelten die Regeln des Abschnittes 3.3.1.

Überschreitet der Quotient 9 Stellen oder ist der Divisor Null, empfängt das Register "Kap" (0091) das Zeichen AA.

Beispiel:

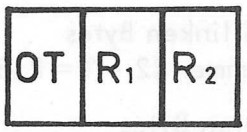
DVD | D1 | 15 | 14 |

	Reg. 15	Reg. 13	Reg. 14
vorher :	00 00 00 05 24	00 00 00 00 00	00 00 20 19 87
nachher:	00 00 00 05 24	00 00 00 03 85	00 00 00 02 47
		Quotient	Rest

(+ 201.987) : (+ 524) = (+ 385)

3.3.2.4. Algebraischer Vergleich

CMD	CoMpare Decimal single length	91
CMDD	CoMpare Decimal Double length	92



OT = 91 oder 92

R1 + R2 = Einfach- oder Doppelregister, nach OT.

Der Inhalt des Registers 1 wird mit dem Inhalt des Registers 2 verglichen. Der Inhalt beider Register wird durch die Operation nicht verändert.

Der Vergleich erfolgt algebraisch. Das Ergebnis wird in dem Vergleichsregister (0086) in Form eines Zeichens gespeichert:

3 C bei R1 < R2
3 D bei R1 = R2
3 E bei R1 > R2

Anmerkung:

Der Inhalt des Vergleichsregisters bleibt bis zum nächsten Vergleich, der den Inhalt verändert, erhalten.

Vor einer Veränderung muß der Inhalt ausgewertet worden sein.

Größe der Operanden:

- CMD : 9 Ziffern + VZ (Register mit 5 Bytes)
- CMDD : 19 Ziffern + VZ (Register mit 10 Bytes)

Beispiele:

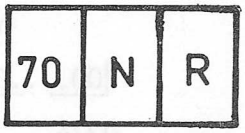
CMD |91|67|59|

	R1 = Reg. 67	R2 = Reg. 59	VG (0086)
a) vorher :	...|00|46|38|	...|00|46|38|	|xx|
nachher :	...|00|46|38|	...|00|46|38|	|3D|
			R1 = R2
b) vorher :	...|00|03|78|	...|99|86|54|	|3D|
nachher :	...|00|03|78|	...|99|86|54|	|3E|
	(+ 378)	(- 1.346)	R1 > R2
c) vorher :	...|99|76|34|	...|99|94|12|	|3E|
nachher :	...|99|76|34|	...|99|94|12|	|3C|
	(- 2.366)	(- 588)	R1 < R2

3.3.3. Versetzungs- und Rundungsbefehle

3.3.3.1. Versetzung nach links

SLD — Shift Left Decimal



N = kann variieren von 00 bis 19

R = Doppelregister

Der Inhalt des Doppelregisters R wird um N Dezimalstellen (Halbbytes) nach links verschoben. Die N ersten Stellen rechts im Register werden nach der Versetzung auf Null gesetzt. Das Vorzeichen des Operanden, der sich im Register R befindet, wird unter der Voraussetzung nicht verändert, daß der Operand beim Versetzen nicht die dafür vorgesehene Stelle erreicht.

Kapazitätsüberschreitung wird nicht angezeigt. Ist die Zahl N der Versetzungen größer als die Zahl N' der zur Verfügung stehenden Stellen links des Operanden, gehen N' - N Ziffern einschließlich des Vorzeichens verloren.

Bei N = 00 findet keine Versetzung statt, der Befehl wird nicht ausgeführt.

Bei N > 19 ist die Zahl der durchgeführten Versetzungen gleich dem Rest der Division von N durch 20.

Beispiel:
N = 77; Zahl der Versetzungen 17.

Das Register kann auch nicht numerische Zeichen enthalten.

Sie werden in der gleichen Weise, Halbbyte für Halbbyte, versetzt.

Anmerkung:

Dieser Befehl ermöglicht die Multiplikation einer Zahl mit 10^n.

Beispiele:

SLD | 70 | 03 | 37 |

vorher:

| 00 | ... | 00 | 00 | 23 | 75 |

0276 0285

nachher:

| 00 ... | 02 | 37 | 50 | 00 |

0276 0285

2.375 ⟶ 2.375.000

vorher:

| 99 | ... | 99 | 99 | 96 | 25 |

0276 0285

nachher:

| 99 ... | 99 | 62 | 50 | 00 |

0276 0285

- 375 ⟶ - 375.000

3.3.3.2. Versetzung nach rechts

SRD — Shift Right Decimal

72	N	R

N = 00 - 19

R = Doppelregister

Der Inhalt des Doppelregisters R wird um N Dezimalstellen (Halbbytes) nach rechts verschoben.

Die Ziffern, die sich an den N ersten Stellen rechts befinden, verschwinden und gehen verloren.

Die Ziffer, die sich an der ersten Stelle von links befindet, bleibt erhalten und wird auf den N folgenden Stellen eingesetzt. Dadurch bleibt die algebraische Form des Registerinhaltes erhalten.

Bei N = 00 findet keine Versetzung statt; der Befehl wird nicht ausgeführt.

Bei N > 19 ist die Zahl der durchgeführten Versetzungen gleich dem Rest der Division von N durch 20.

Beispiel: N = 35; Zahl der Versetzungen 15

Das Register kann auch nicht numerische Zeichen enthalten. Sie werden in der gleichen Weise, Halbbyte für Halbbyte, versetzt.

Anmerkung:

Dieser Befehl ermöglicht eine Division durch 10 unter Vernachlässigung des Restes. Ebenso kann eine Abrundung erfolgen.