

Multiplikation

Anleitung zum Programmieren

1. Der Multiplikand und der Multiplikator müssen vor der Multiplikation gepackt sein.
2. Die Länge des gepackten Multiplikanden und des gepackten Multiplikators in Bytes ergeben addiert die Länge des Produktfeldes in Bytes.
3. Vor der Multiplikation muß der Multiplikand in die rechten Stellen des Produktfeldes gebracht werden. Die nicht durch den Multiplikanden belegten Stellen des Produktfeldes müssen Nullen enthalten.
4. Das Produkt bildet sich im Produktfeld. Das linke Halbbyte enthält immer eine "0".
5. Das Vorzeichen des Produktes ist ein hexadezimaler C bei positivem und ein hexadezimaler D bei negativem Produkt.
6. Ist zu runden und hat das Produkt wechselnde Vorzeichen, so ist vor der Rundung das Vorzeichen des Produktes auf die Rundungskonstante zu übertragen.
7. Ist das Produkt unter Vernachlässigung von Dezimalstellen auszugeben, so ist eine Stellenversetzung nach rechts erforderlich. (Ist um eine ungerade Anzahl Stellen zu versetzen, ist MVO erforderlich.)
8. Vor oder nach dem Stellenversetzen ist evtl. das Vorzeichen des Produktes zu übertragen bzw. zu versetzen.

D i v i s i o n

Hinweise für die Programmierung

A Ermitteln der Länge des Rechenfeldes

1. Bestimmen der Länge des Dividenden.
Der Divisor und der Dividend werden mit der gleichen Zehnerpotenz multipliziert (aber nur im Geiste!), so daß der Divisor keine Dezimalstellen mehr enthält.

Bei der Division erhält der Quotient so viele Dezimalstellen, wie der Dividend hat. Soll der Quotient weitere Dezimalstellen erhalten, so müssen dem Dividenden entsprechend viele Nullen angehängt werden.

2. Bestimmen der Länge des Rechenfeldes.
 $\text{Länge gepackter Dividend} + \text{Länge gepackter Divisor} = \text{Länge Rechenfeld. Länge jeweils in Bytes.}$

Anm.: Bei der Länge des Dividenden sind die evtl. angehängten Nullen mitzuzählen.

B Durchführung der Programmierung

1. Packen Dividend und Divisor (falls nicht schon gepackt).
2. Einbringen des Dividenden in das Rechenfeld.
Sind dem Dividenden Nullen anzuhängen, so ist eine Stellenversetzung nach links erforderlich. - Es ist darauf zu achten, daß nur im rechten Byte des Rechenfeldes ein Vorzeichen steht. Die durch den Dividenden links im Rechenfeld nicht belegten Stellen müssen vor der Division Nullen enthalten.
3. Prüfen des Divisors auf null, wenn man nicht sicher ist, daß er immer $\neq 0$ (ungleich null) ist. (Ein Divisor "0" ist nicht erlaubt.)

3 4. Divisionsinstruktion

Der Quotient bildet sich in den linken, der Rest in den rechten Stellen des Rechenfeldes. - Der Rest belegt so viele Bytes wie der Divisor.

Das Vorzeichen des Quotienten bildet sich nach den algebraischen Regeln (hexadezimal C oder D). - Das Vorzeichen des Restes ist gleich dem Vorzeichen des Dividenden.

5. Runden des Quotienten, falls erforderlich.

Hat der Quotient wechselnde Vorzeichen, so ist das jeweilige Vorzeichen vor der Rundung auf die Rundungskonstante zu übertragen.

Anm.: Ist mit dem nicht gerundeten Quotienten weiterzurechnen, so ist dieser vor der Rundung wegzuspeichern.

6. Ausgeben des Quotienten (bzw. Bereitstellen für eine Weiterrechnung).

Sind Dezimalstellen des Quotienten zu vernachlässigen, so ist eine Stellenversetzung nach rechts erforderlich.

7. Übertragen bzw. Versetzen des Vorzeichens, falls erforderlich. Dies kann vor oder nach dem Stellenversetzen erforderlich sein.

SYMBOLTABELLE - USINGTABELLE

Symbolische Instruktion

Name				Operation		Operand und Be			
1	8	10	14			20		25	
				M.V.C.		S.N.R.		E.N.R.	

Maschineninstruktion

OP	L	B1	D1	B2	D2
----	---	----	----	----	----

SYMBOL-TABELLE

Symbol	Adresse
KCENDE	1200
...	...
SNR	1289
...	...
ENR	1308
EVERBR	1313

USING-TABELLE

Register	Adresse
10	802
11	1313

Die Umwandlung eines symbolischen Namens in die Nummer eines Basisregisters und die relative Adresse geschieht wie folgt:

1. Aufsuchen SNR in der Symboltabelle
2. Entnehmen der effektiven Adresse 1289
3. Vergleichen der in der USING-Tabelle angegebenen Adressen mit der Adresse 1289, bis sich gleich oder kleiner ergibt. - Dabei wird bei der höchsten Adresse in der USING-Tabelle begonnen.
In unserem Beispiel erhalten wir beim zweiten Vergleich (b) "kleiner".
4. Errechnen der relativen Adresse. - Hierbei wird die bei dem Vergleich ermittelte (gleich große oder kleinere) Adresse als Basisadresse genommen.
5. Einsetzen der Registernummer und der relativen Adresse in die Maschineninstruktion.

ZUM LEHRGANG /360 ASS-PU

I Allgemeines

Sie haben im Laufe dieses Lehrgangs die Grundzüge des Programmierens in der Assemblersprache für das System /360 Modell 25 (und höhere) kennengelernt.

Es ist jedoch zu beachten, daß es für die

- a) Kartensysteme der /360 Modelle 20 und 25 je eine Basic - Assemblersprache gibt.
- b) Band- und/oder Plattensysteme der /360 Modelle 20 und 25 je eine Assemblersprache gibt.

Welche dieser Programmiersprachen zur Codierung eines Programms zu verwenden ist, wird durch das vorhandene IBM System /360 bestimmt.

Zwischen den einzelnen Assemblersprachen bestehen Unterschiede. Diese sind in diesem Anhang (III) aufgeführt.

Außerdem ist zu beachten, daß es modellabhängige Unterschiede (Modell 20 $\longleftrightarrow$ Modell 25 und höhere) gibt. - Diese sind ebenfalls in diesem Anhang (II) zusammengefaßt.

In den Folgelehrgängen erhalten Sie die entsprechende Systemliteratur.

II Unterschiede Modell 20 $\longleftrightarrow$ Modell 25 (und höhere)

Das System /360 ist sowohl eine Stellen- als auch eine Wortmaschine.

Zu beachten ist jedoch, daß Dualdaten beim Modell 20 immer nur Halbwortgröße haben.

Deshalb entfallen bei Modell 20 alle auf "Wort" oder "Doppelwort" bezogenen Assembleranweisungen und Maschineninstruktionen. Eine Begrenzungszuordnung erfolgt nur auf Halbwortgrenzen.

Es entfallen:

Die Maschineninstruktionen A, L, LA, LR.

Es gelten folgende Einschränkungen:

DS: Nicht erlaubt sind die Feldschlüssel F und D (nur C und H sind erlaubt).

DC: Nicht erlaubt sind die Typenschlüssel F und A.
Bei Modell 20 wird eine Adreßkonstante durch den Typenschlüssel Y definiert. Die Speicherung der Adreßkonstanten erfolgt in einem Halbwort.

Register: Die Register haben nur Halbwortgröße. Das Modell 20 hat nur 8 Allgemeine Register. Diese haben die Adressen (Nummern) 8-15.

Instruk-
tionsfor-
mate:

Bei Instruktionen im RX-Format ist bei Modell 20 keine Indizierung möglich (in der echten Instruktion ist X2 immer 0).

Die Operation BALR heißt bei Modell 20 BASR (mit gleicher Funktion).

Wenn bei den Operationen AP, SP, ZAP und CP der 2. Operand länger ist als der 1. Operand, dann erfolgt bei dem Modell 20 ein Maschinenstop. Die Operationen werden nicht ausgeführt.

III Unterscheide Basic-Assembler $\longleftrightarrow$ Assembler
bei den Modellen 20 und 25 (und höhere)

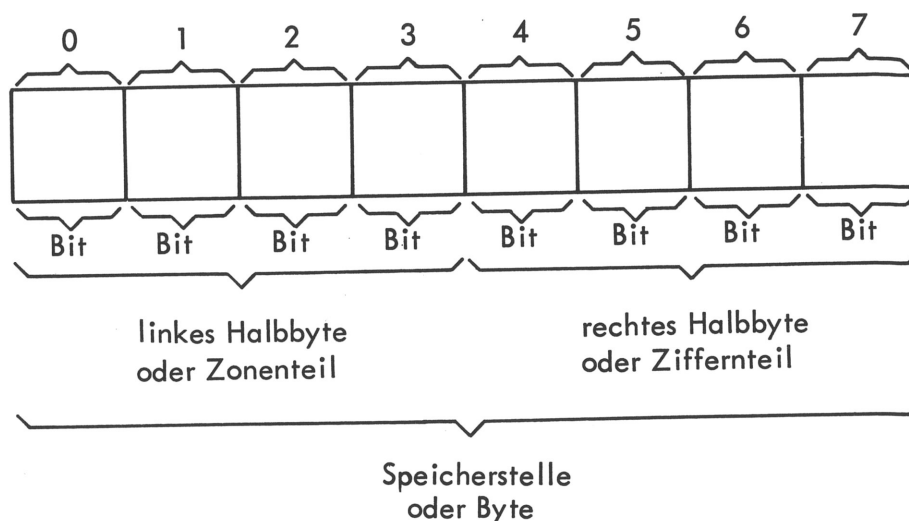
	Mod. 20 Basic-Assembler	Mod. 20 Assembler	Mod. 25 Basic-Assembler	Mod. 25 Assembler
Länge der Symbole (Namen)	max. 4 Stellen	max. 8 Stellen	max. 6 Stellen	max. 8 Stellen
Mögliche Zeichen in Symbolen	A - Z 0 - 9	A - Z 0 - 9 \$, #, @	A - Z 0 - 9	A - Z 0 - 9 \$, #, @
Größe der Zeichenkonstanten (C)	max. 16 Bytes	max. 32 Bytes	max. 16 Bytes	max. 256 Bytes
Größe der Gepackten Konstan- ten (P)	* entfällt	max. 16 Bytes	* entfällt	max. 16 Bytes
Größe der Hexadezimalkon- stanten (X)	max. 16 Bytes	max. 32 Bytes	max. 16 Bytes	max. 256 Bytes

* Gepackte Konstanten (P) gibt es in der Basic-Assemblersprache nicht.

Unterschiede bezüglich der Verzweigungsoperationen siehe Merkblatt 408.

VERSCHLÜSSELUNG DER ZEICHEN IM HAUPTSPESICHER
NACH DEM EBCD-CODE

Bit-Position:



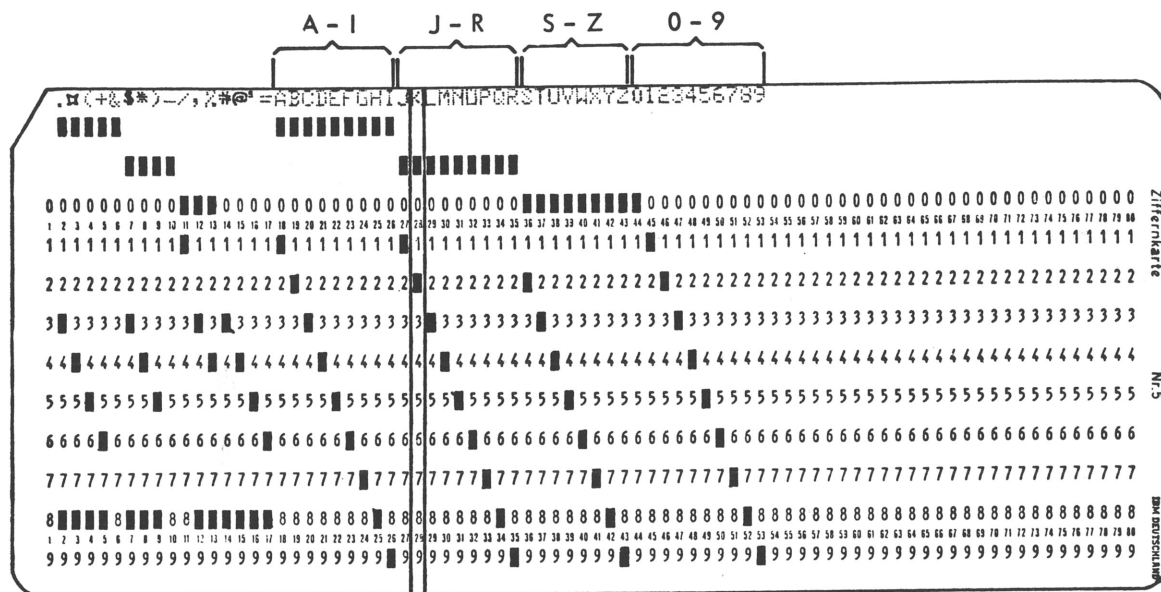
Ein Bit (Magnetkern) kann die Binärziffern 0 und 1 speichern.
Ein Byte kann ein Zeichen (Buchstabe, Ziffer, Sonderzeichen) speichern.

Hexadezimalziffern (Verschlüsselung in 4 Bits):

ein Bit	ein Bit	ein Bit	ein Bit	Hexadez. Ziffer
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F
8	4	2	1	Stellenwert

Anmerkung:

Zu jedem Byte gehören zusätzlich zu den 8 Bits noch 1 (Modelle 30, 40 usw.) bzw. 2 (Modell 20) sogenannte Prüfbits. Diese Prüfbits dienen nicht der Verschlüsselung der Zeichen in einem Byte, sondern allein der Kontrolle, ob die Daten in technischer Hinsicht fehlerlos im Speicher bewegt werden. Um diese Bits braucht sich der Programmierer nicht zu kümmern. Daher wurden sie bisher nicht erwähnt.



Buchstabe "K" in einer Lochspalte

1	1	0	1	0	0	1	0
D				2			

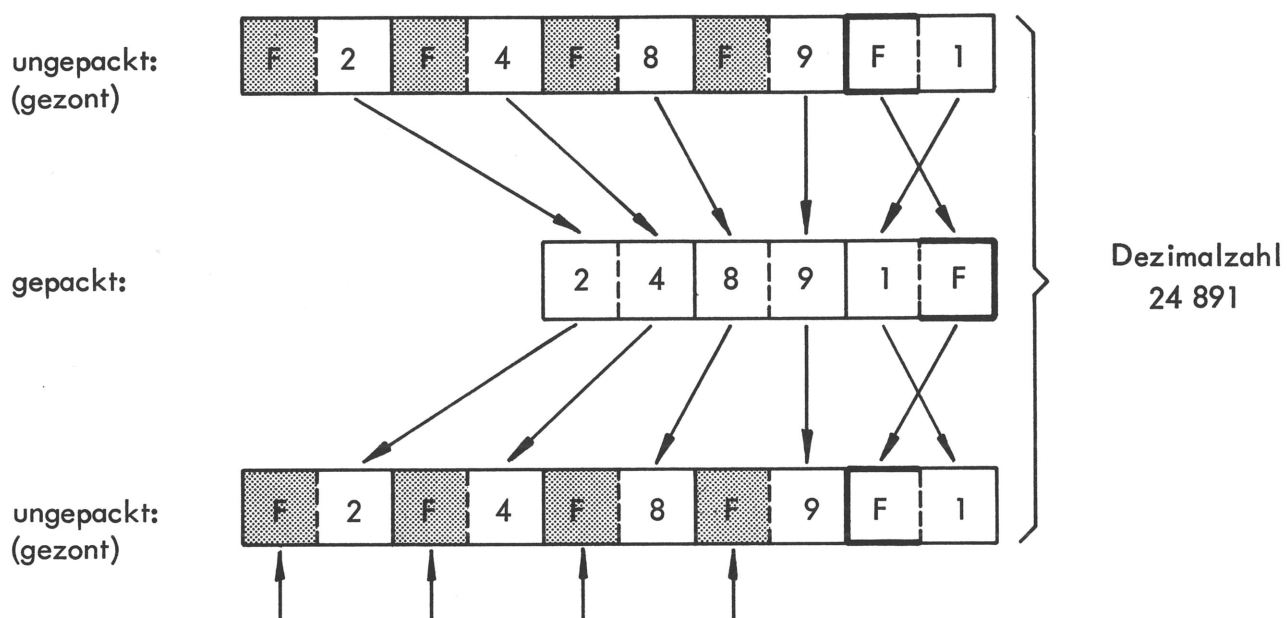
- Buchstabe "K" in einem Byte im Speicher

- hexadezimale Schreibweise

		rechtes Halbbyte (Ziffernteil)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
hexa-dezimal		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
linkes Halbbyte (Zonen-teil)	0	0000															
	1	0001															
	2	0010															
	3	0011															
	4	0100	blank										.	□	(	+	
	5	0101	&										§	*	)		
	6	0110	-	/									,	%			
	7	0111											#	©	'	=	
	8	1000															
	9	1001															
	A	1010															
	B	1011															
	C	1100		A	B	C	D	E	F	G	H	I					
	D	1101		J	K	L	M	N	O	P	Q	R					
	E	1110			S	T	U	V	W	X	Y	Z					
	F	1111		0	1	2	3	4	5	6	7	8	9				

UNGEPACKT - GEPACKT

2 Darstellungsformen für Dezimalzahlen



Die Vorzeichen-Verschlüsselung steht im rechten Byte des Zahlenfeldes.

Beispiel zur Verwendung der Darstellungsformen
gepackt - ungepackt:

00000000000000000000000000000000																									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20						
11111111111111111111111111111111																									
22222222222222222222222222222222																									
33333333333333333333333333333333																									
44444444444444444444444444444444																									
55555555555555555555555555555555																									
66666666666666666666666666666666																									
77777777777777777777777777777777																									
88888888888888888888888888888888																									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
99999999999999999999999999999999																									

WERT1



00000000000000000000000000000000																										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21						
11111111111111111111											■	1111111111111111														
22222222222222222222											■	2222222222222222														
33333333333333333333											■	3333333333333333														
44444444444444444444											■	4444444444444444														
55555555555555555555											■	5555555555555555														
66666666666666666666											■	6666666666666666														
77777777777777777777											■	7777777777777777														
88888888888888888888											■	8888888888888888														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
99999999999999999999999999999999																										

WERT2



+

00000000000000000000000000000000																									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21					
1111111111								1	1	11111111111111															
22222222222222222222222222222222																									
33333333333333333333333333333333																									
44444444444444444444444444444444																									
55555555555555555555555555555555																									
66666666666666666666666666666666																									
77777777777777777777777777777777																									
8888888888								8	1	888888888888888888888888															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
99999999999999999999999999999999																									

SUMME



=

12

VOLLSTÄNDIGE TABELLE DER VORZEICHENVERSCHLÜSSELUNGEN
BEI DEZIMALZAHLEN

Verschlüsselung		Vorzeichen	entsteht bei Rechenergebnis	entspricht Zonenlochung in Zeile
hexadezimal	binär			
A	1010	+		
B	1011	-		
X C	1100	+	positiv	12
X D	1101	-	negativ	11
E	1110	+		
X F	1111	(+)		(keine)

13

1. UMWANDELN DEZIMAL $\rightarrow$ DUAL

Stellenwerte im Dualsystem:

256	128	64	32	16	8	4	2	1
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$$137_{10} = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

↓
1

↓
0

↓
0

↓
0

↓
1

↓
0

↓
0

↓
1

137_{10} entspricht 10001001_2

2. UMWANDELN DUAL $\rightarrow$ DEZIMAL

100101_2 das ist:

$$\begin{array}{rcl}
 1 \cdot 2^0 & = & 1 \\
 0 \cdot 2^1 & = & 0 \\
 1 \cdot 2^2 & = & 4 \\
 0 \cdot 2^3 & = & 0 \\
 0 \cdot 2^4 & = & 0 \\
 1 \cdot 2^5 & = & 32 \\
 \hline
 & & 37_{10}
 \end{array}$$

14

3. ADDITION IM DUALSYSTEM

Arithmetische Regeln:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \quad (\text{und Übertrag})$$

Beispiel einer Addition:

	1001011_2	$\hat{=}$	75_{10}
	$+ 1011110_2$	$\hat{=}$	$+ 94_{10}$
(Überträge:)	$\begin{array}{r} 1001011 \\ + 1011110 \\ \hline 1\ 1111 \end{array}_2$	$\hat{=}$	$\begin{array}{r} 75 \\ + 94 \\ \hline 1 \end{array}_{10}$
Ergebnis:	10101001_2	$\hat{=}$	169_{10}

DIE DATENFORMATE FÜR DUALDATEN

Halbwort



2 Bytes

Wort



4 Bytes

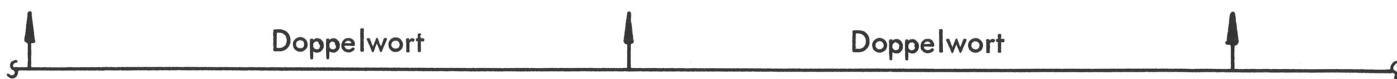
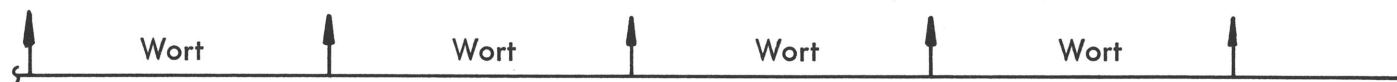
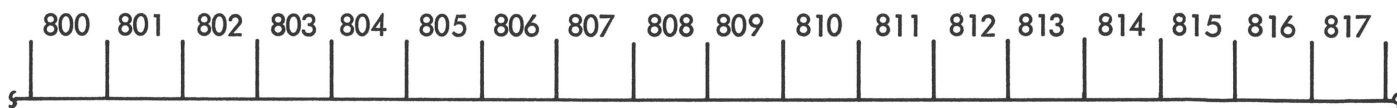
Doppelwort



8 Bytes

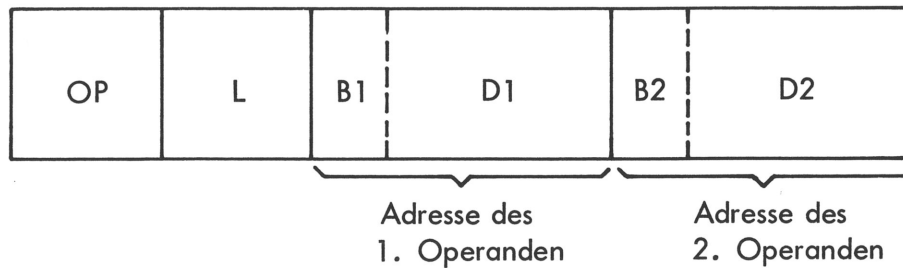
Halbwort	=	2 Bytes	=	1 Vorzeichen + 15 Bits
Wort	=	4 Bytes	=	1 Vorzeichen + 31 Bits
Doppelwort	=	8 Bytes	=	1 Vorzeichen + 63 Bits

ADRESSIERBARE EINHEITEN DER ZENTRALEINHEIT



1. Ein Feld variabler Länge kann ab jeder beliebigen Speicherstelle gespeichert werden.
2. Felder fester Länge unterliegen der Begrenzungszuordnung.
 - a) Halbwort-Adressen müssen durch 2 teilbar sein,
 - b) Wort-Adressen müssen durch 4 teilbar sein,
 - c) Doppelwort-Adressen müssen durch 8 teilbar sein.

ADRESSIERUNG DES HAUPTSPEICHERS IN DEN
MASCHINENINSTRUKTIONEN



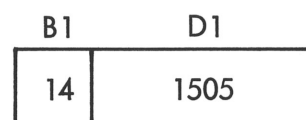
D = Relative Adresse (Displacement)
12 Dual-Stellen: Adressierung von 0 bis 4095

B = Adresse des Registers, das die Basisadresse enthält.
4 Dual-Stellen: Adressierung der Register (0 bis 15)

$$\text{Effektive Adresse} = \text{Relative Adresse} + \text{Inhalt des Basisregisters.}$$

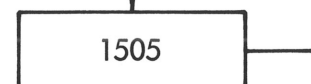
Beispiel zur Bildung einer effektiven Adresse

Adresse des 1. Operanden in der
Maschineninstruktion



Aufgliederung des Operanden in:

a) Relative Adresse (D1)

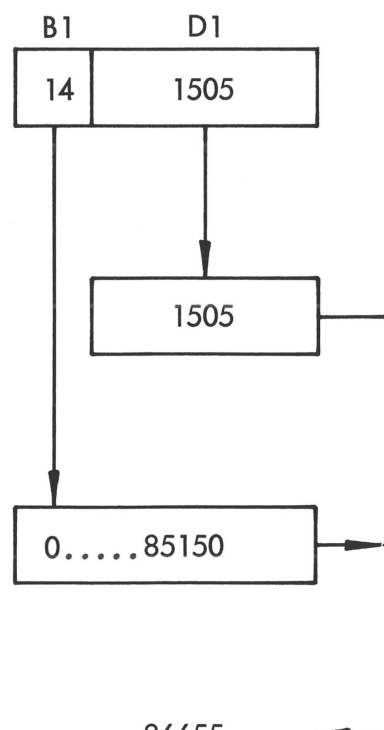


b) Basisadresse
(Inhalt Register 14/B1)



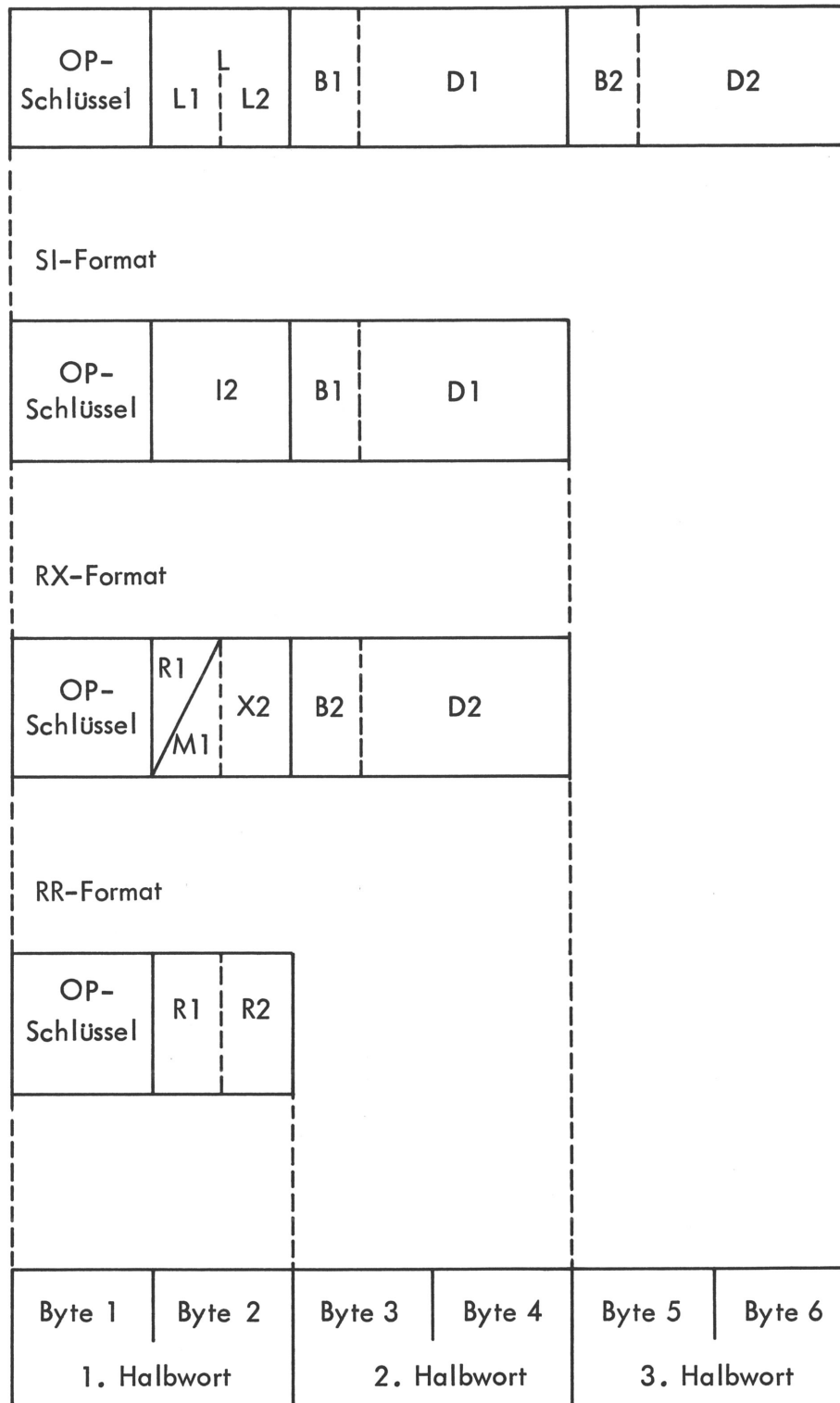
Effektive Adresse

86655



4 INSTRUKTIONSFORMATE DES SYSTEMS /360

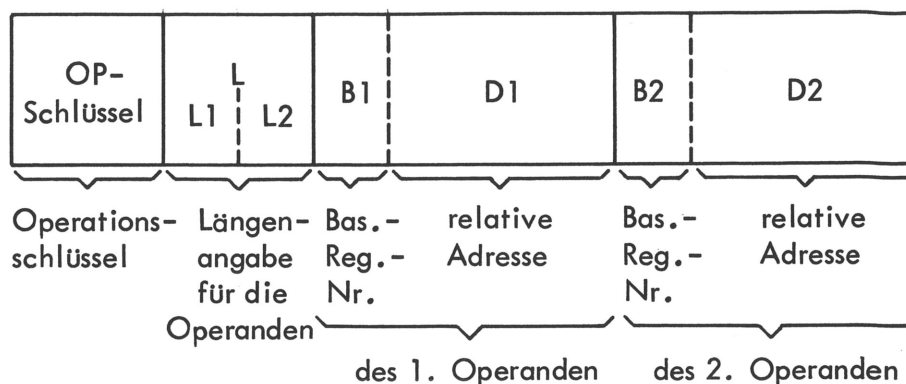
SS-Format
 $\swarrow$ logisch
 $\searrow$ arithmetisch



Die einzelnen Angaben in den echten Instruktionen sind jeweils dual verschlüsselt. Zur Wiedergabe dieses dualen Inhalts werden Hexadezimalziffern verwendet (z.B. "D2" für den Operationsschlüssel "Übertragen Zeichen").

SS-FORMAT

logisch:
arithmetisch:



Das SS-Format hat eine Länge von 6 Bytes.

Die beiden Operanden sind Felder im Hauptspeicher.

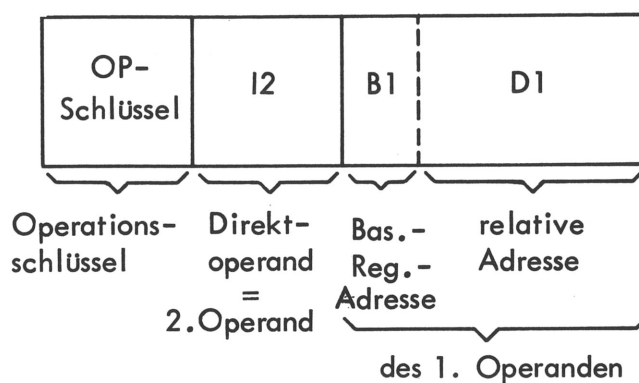
Die Längenangaben im 2. Byte (L bzw. L1 und L2) sind jeweils um 1 kleiner als die Byteanzahl der Felder.

Zur arithmetischen Form des SS-Formates gehören 2 Längenangaben (L1 und L2).

Für die Feldlänge der Operanden gelten folgende Grenzen:

	Längenangabe		Feldlänge in Bytes		Maximaler Feldinhalt
	untere Grenze	obere Grenze	untere Grenze	obere Grenze	
eine Längenangabe L	0	255	1	256	256 8-Bit-Zeichen
zwei Längenangaben L1 / L2	0	15	1	16	31 Dezimal- ziffern und ein Vorzeichen

SI-FORMAT



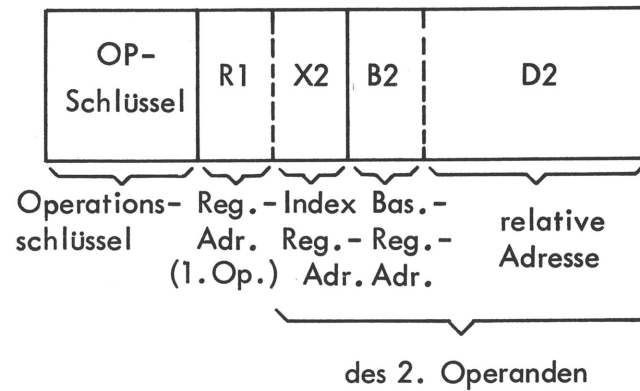
Das SI-Format hat eine Länge von 4 Bytes.

Der 1. Operand ist ein Feld im Hauptspeicher und der 2. Operand ein Direktoperand (im 2. Byte der Instruktion als 8-Bit-Direktzeichen).

Eine Längenangabe in der Instruktion ist überflüssig. Die Feldlänge beträgt immer 1 Byte.

21

RX-FORMAT



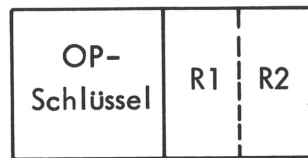
Das RX-Format hat eine Länge von 4 Bytes.

Der 1. Operand ist ein Register und der 2. Operand ein Feld im Hauptspeicher.

Ist $X2 = 0$, so erfolgt keine Indizierung.

An der Stelle von R1 kann auch eine Maske (M1) stehen (bei Verzweiginstruktionen).

RR-FORMAT



Operations- Reg.- Reg.-
schlüssel Adr. Adr.
 (1. Op.) (2. Op.)

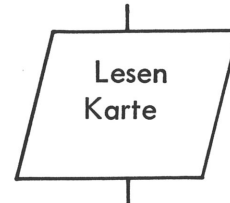
Das RR. Format hat eine Länge von 2 Bytes.

Beide Operanden sind Register.

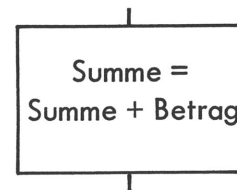
DIE SYMBOLE DES PROGRAMMABLAUFPLANS

1. Die Arbeitssymbole

a) Eingabe/Ausgabe

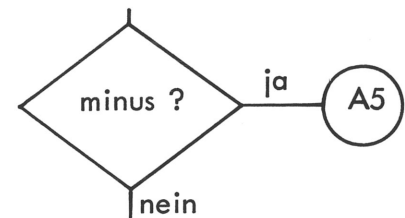


b) Verarbeitung



2. Die Folgesymbole

a) Entscheidung



b) Anschlußpunkt bzw. Schritt-kennzeichnung



c) Programm-Modifikation

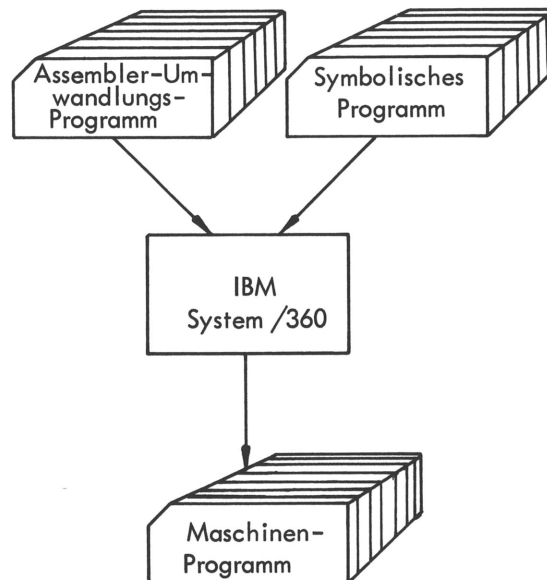


d) Begrenzung

Anmerkung :

Die Zeichenschablone, die den Arbeitsmitteln beigelegt ist, enthält neben den hier beschriebenen Symbolen des Programmablaufplans noch weitere Symbole. Diese Symbole, auf die in diesem Lehrgang nicht eingegangen werden kann, dienen zur Erstellung von Datenflußplänen. Sie sind in DIN 66 001 beschrieben.

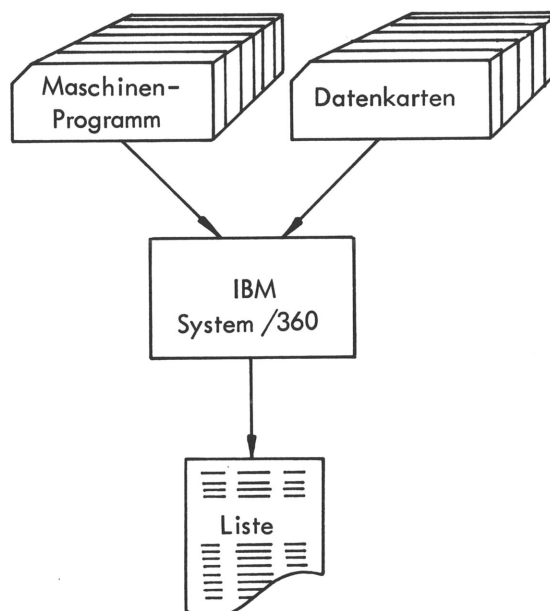
DAS ERSTELLEN DES MASCHINENPROGRAMMS



Anmerkung:

Statt "symbolisches Programm" sind auch die Bezeichnungen "Symbolprogramm", "Quellenprogramm" oder "Ursprungsprogramm" gebräuchlich.

DIE VERARBEITUNG DER DATEN



Anmerkung:

Statt "Maschinenprogramm" werden auch die Bezeichnungen "echtes Programm" oder "Objektprogramm" verwendet.

1. REGELN ZUM PROGRAMMSCHEMA

- a) Das Programmschema ist eine Ablochkarte, die das Ablocken des symbolischen Programms in Lochkarten erleichtert.
- b) Pro Zeile des Programmschemas ist nur eine Anweisung erlaubt.

2. REGELN FÜR DIE WAHL DER SYMBOLE

- a) Ein Symbol kann 1 bis 8 Zeichen lang sein.
- b) Folgende Zeichen dürfen verwendet werden:
 - die Buchstaben A-Z,
 - die Ziffern 0-9,
 - die 3 Zeichen \$, #, @
- c) Das erste Zeichen eines Symbols darf jedoch keine Ziffer sein.
- d) Leerstellen sind innerhalb eines Symbols unzulässig.

3. REGELN FÜR DIE BEMERKUNGEN

- a) Der Stern (*) in der Stelle 1 des Programmschemas kennzeichnet eine Bemerkungszeile.
- b) Bemerkungen dürfen nur bis Stelle 71 reichen und können in der nächsten Zeile fortgesetzt werden. Jede Bemerkungszeile ist hierbei durch einen Stern zu kennzeichnen.
- c) Bemerkungen rechts von den Operanden benötigen keinen Stern. Zwischen Operand(en) und Bemerkung muß sich jedoch mindestens 1 Leerstelle befinden.

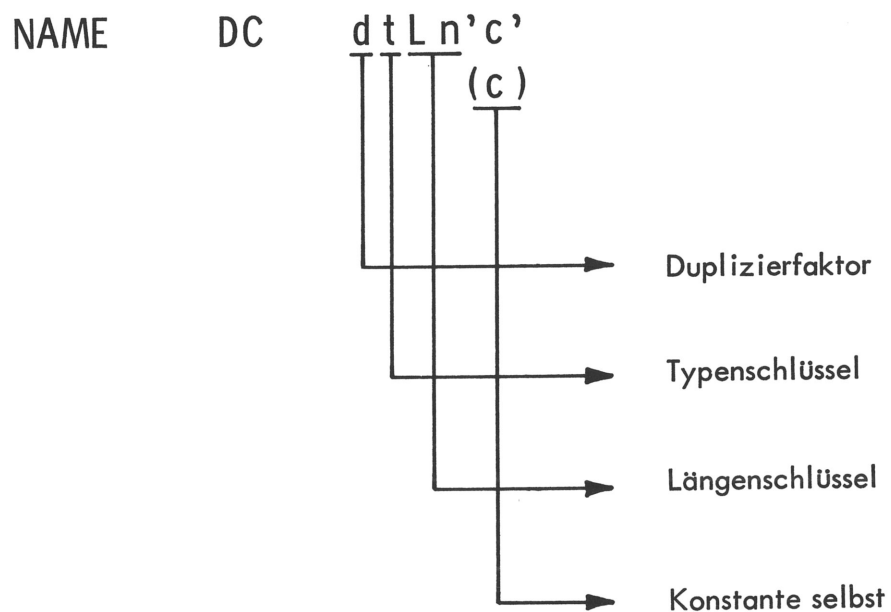
DIE KONSTANTENTYPEN C, P, X

Konstantentyp	Beispiel	Inhalt der Bytes	Verschlüsselung	Max. Länge	Bei Längenschlüssel Wegfall oder Auffüllen	
						w.aufgef.
C Zeichenkonstante	DC C'DM'	D M 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 0	Jedes Zeichen in 1 Byte (8-Bit- EBCD-Code)	256	rechts	Blank
P Gepackte Kon- stante	DC P'250'	2 5 0 C 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 0	Ziffern und Vorzeichen je- weils dual in einem Halbbyte	16	links	Null
X Hexadezimal- konstante	DC X'F1F5'	F 1 F 5 1 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1	Jede Hexadezi- malziffer dual in einem Halbbyte	256	links	Null

DIE KONSTANTENTYPEN F, H, A

Konstantentyp	Beispiel	Inhalt der Bytes	Verschüsselung
F Wortkonstante	DC F'815'	V 815	Dual rechtsbündig im Wort
H Halbwortkonstante	DC H'294'	V 294	Dual rechtsbündig im Halbwort
A Adresskonstante	DC A(EFELD)	EFELD sei Adresse 850	Dual rechtsbündig im Wort

DAS FORMAT DES OPERANDEN BEI DER DEFINITION DER KONSTANTEN



Der Duplizierfaktor, eine Dezimalzahl, kann wahlweise benutzt oder weggelassen werden.

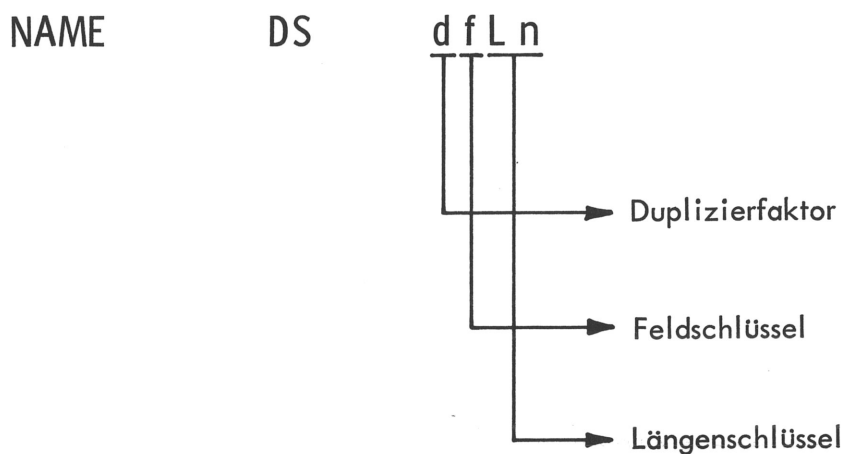
Die behandelten Typenschlüssel:

- C - Zeichenkonstante
- P - Gepackte Konstante
- X - Hexadezimalkonstante
- F - Wortkonstante
- H - Halbwortkonstante
- A - Adresskonstante

Ein Längenschlüssel kann in der Form Ln angegeben werden. n bezeichnet hierbei die Länge in Anzahl Bytes.

Die Konstante selbst wird mit Ausnahme der Adresskonstanten, die in Klammern gesetzt wird, in Hochkommas geschrieben.

DAS FORMAT DES OPERANDEN BEI DER DEFINITION
DER FELDER UND BEREICHE



Der Duplizierfaktor, eine Dezimalzahl, kann wahlweise benutzt oder weggelassen werden.

Folgende Feldschlüssel sind möglich:

- C - Zeichen (Bytes)
- H - Halbwort
- F - Wort
- D - Doppelwort

Ein Längenschlüssel wird nur bei dem Feldschlüssel C in der Form Ln angegeben. n bezeichnet hierbei die Anzahl Bytes, die für ein Feld reserviert werden sollen.

DEFINITIONSANWEISUNGEN

Name	Operation	Operand	Reserviert werden:	Feldlänge	Bemerkungen
FELD1	DS	CL80	ein 80 Bytes großes Feld	80	
FELD2	DS	4CL20	vier 20 Bytes große Felder, das heißt insgesamt 80 Bytes	20	
FELD3	DS	0CL80	0 Bytes	80	Der Zuordnungszähler wird nicht weitergestellt.
FELD4	DS	H	ein Halbwort = 2 Bytes	2	Beginnend auf Halbwortadresse
FELD5	DS	5F	5 Worte = 20 Bytes	4	Beginnend auf Wortadresse
FELD6	DS	0D	0 Bytes		Der Zuordnungszähler wird auf eine Doppelwortgrenze eingestellt.

DIE MÖGLICHEN OPERANDEN-FORMATE FÜR
INSTRUKTIONEN IM SS-FORMAT

Arithmetische Operationen

1. Operand:	2. Operand:
S1	S2
S1(L1)	S2(L2)
D1(L1, B1)	D2(L2, B2)

Logische Operationen

1. Operand:	2. Operand:
S1	S2
S1(L)	-
D1(L, B1)	D2(B2)

Beispiele:

S1: FELDA bzw. FELDA+5
S2(L2): FELDA(3) bzw. FELDA+5(3)
D1(L, B1): 100(5, 8)
D2(B2): 100(8)

Anmerkung:

- S bedeutet die symbolische Adresse (ein einfacher oder zusammengesetzter Ausdruck) eines Feldes im Hauptspeicher.
- L bezeichnet die explizite Länge eines Feldes im Hauptspeicher.
- D bedeutet die relative Adresse eines Feldes im Hauptspeicher, explizit angegeben.
- B bezeichnet ein Basisregister, das die Basisadresse enthält.

MVC

ÜBERTRAGEN ZEICHEN

(MoVe Characters)

Diese Operation überträgt Zeichen von einem Speicherfeld in ein anderes.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10
14	20	25
30		
	M,V,C,	A,F,E,L,D,,E,F,E,L,D,

Wirkungsweise:

Der Inhalt des zweiten Operanden (im Beispiel EFELD) wird in die Speicherstellen (Bytes) des ersten Operanden (im Beispiel AFELD) übertragen.

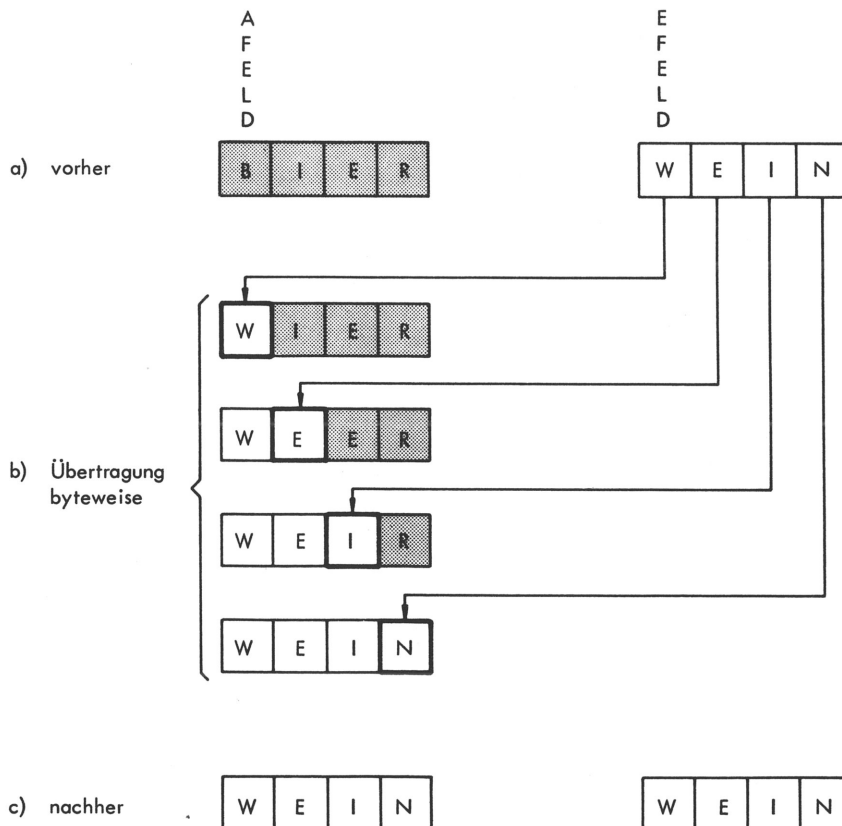
Der Inhalt des zweiten Operanden wird nicht verändert.

Der bisherige Inhalt des ersten Operanden wird zerstört.

Die Länge der Operation ist variabel. Es gibt aber eine obere Grenze: Eine MVC-Operation kann maximal den Inhalt von 256 Bytes übertragen.

Verarbeitet wird byteweise von links nach rechts in der Länge des ersten Operanden.

Durchführung der Operation:



MVZ

ÜBERTRAGEN ZONEN (MoVe Zones)

Diese Operation überträgt den Inhalt von linken Halbbytes. Im übrigen arbeitet sie genauso wie MVC.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10
14	20	25
30		
	M.V.Z.	W.E.R.T.A., W.E.R.T.B.

Wirkungsweise:

Der Inhalt der linken Halbbytes des zweiten Operanden (im Beispiel WERTB) wird in die linken Halbbytes des ersten Operanden (im Beispiel WERTA) übertragen.

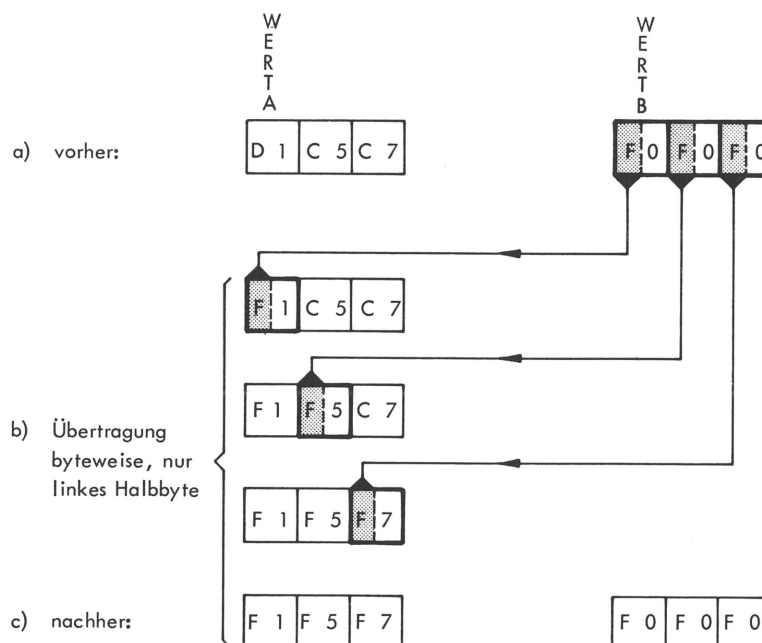
Der Inhalt des zweiten Operanden wird nicht verändert.

Der bisherige Inhalt der linken Halbbytes des ersten Operanden wird zerstört.

Die maximale Länge der Operanden ist 256 Bytes.

Verarbeitet wird byteweise von links nach rechts in der Länge des ersten Operanden.

Durchführung der Operation:



MVN

ÜBERTRAGEN ZIFFERN

(MoVe Numerics)

Diese Operation überträgt den Inhalt von rechten Halbbytes. Im übrigen arbeitet sie wie MVC.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10 -14
		20 25 30
	MVN	FELDA, FELDB

Wirkungsweise:

Der Inhalt der rechten Halbbytes des zweiten Operanden (im Beispiel FELDB) wird in die rechten Halbbytes des ersten Operanden (im Beispiel FELDA) übertragen.

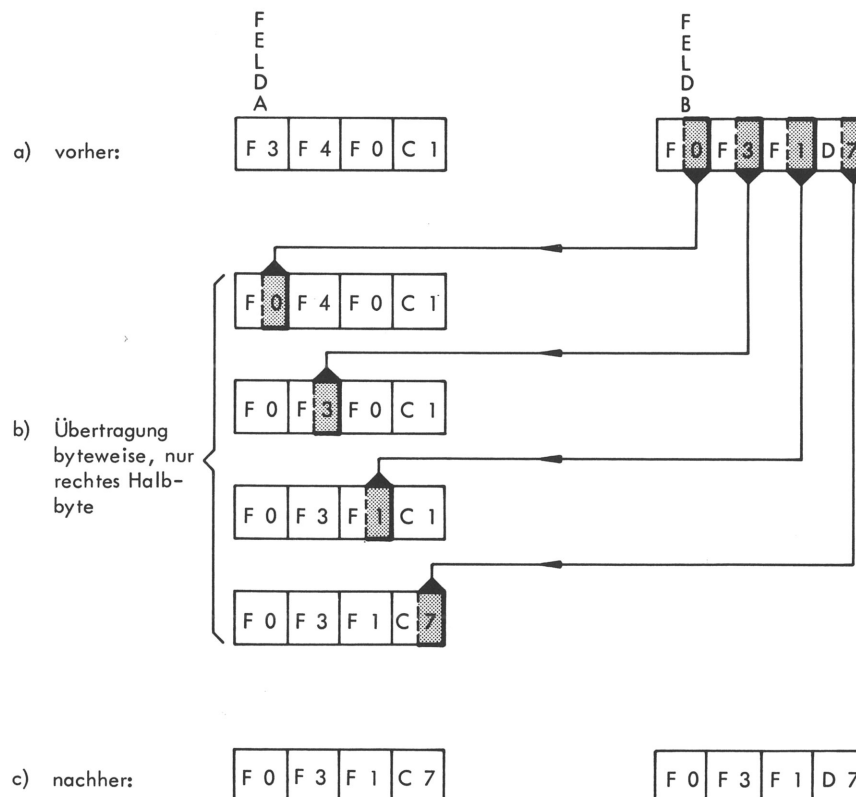
Der Inhalt des zweiten Operanden wird nicht verändert.

Der bisherige Inhalt der rechten Halbbytes des ersten Operanden wird zerstört.

Die maximale Länge der Operanden ist 256 Bytes.

Verarbeitet wird byteweise von links nach rechts in der Länge des ersten Operanden.

Durchführung der Operation:



35

MVI

ÜBERTRAGEN DIREKTZEICHEN

(MoVe Immediate)

Diese Operation überträgt ein einzelnes Zeichen, das direkt in der Instruktion angegeben wird, in ein Byte im Speicher.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1 8	10 14	20 25 30
	M.V.I.	A.G.R.U.P.P.E., C', A'

Wirkungsweise:

Der zweite Operand ist ein Direktoperand (im Beispiel C'A').

Das Direktzeichen - auch Direktdate - des zweiten Operanden (im Beispiel "A") wird in den ersten Operanden (im Beispiel AGRUPPE) übertragen.

Die Operation bezieht sich immer auf ein Byte.

Das Direktzeichen im zweiten Operanden bleibt unverändert.

Durchführung der Operation:

Name	Operation	Operand und Bemerkungen
1 8	10 14	20 25 30
	M.V.I.	A.G.R.U.P.P.E., C', A'

a) vorher:

A
G
R
U
P
P
E

K

b) nachher:

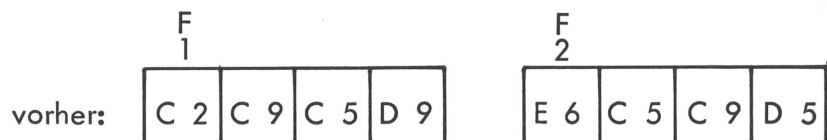
A

ÜBERTRAGUNGSOPERATIONEN

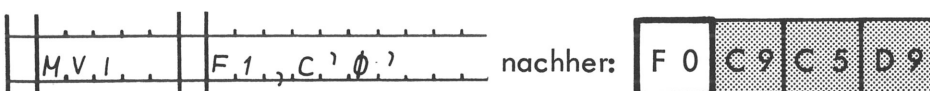
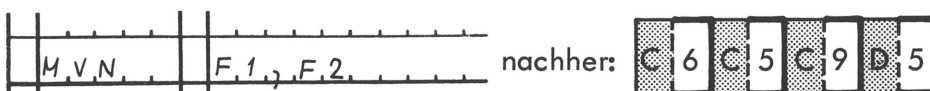
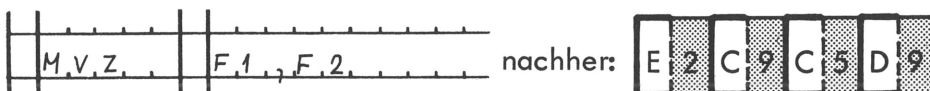
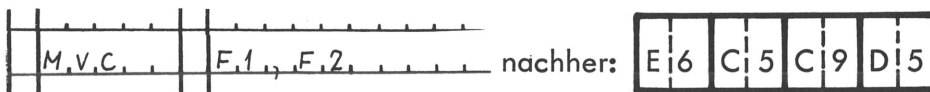
Schlüssel	Operation	Verarbeit.- Richtung	Verarbeitet werden
	Bezeichnung		
MVC	Übertragen Zeichen	→	Bytes
MVZ	Übertragen Zonen	→	linke Halbbbytes
MVN	Übertragen Ziffern	→	rechte Halbbbytes
MVI	Übertragen Direktzeichen		1 Byte

Für MVC, MVZ und MVN gilt:
Übertragungslänge = Länge 1. Operand
max. 256 Bytes

Instruktionsbeispiele:



Operation	Operand und l
10	14
20	25



CLC

VERGLEICHEN LOGISCH ZEICHEN

(Compare Logical Characters)

Diese Operation vergleicht den Inhalt zweier Speicherfelder und zeigt das Ergebnis im Bedingungsschlüssel an.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10 14
		20 25 30
	C.L.C.	A.N.R., E.N.R.

Wirkungsweise:

Der Inhalt des ersten Operanden (im Beispiel ANR) wird mit dem Inhalt des zweiten Operanden (im Beispiel ENR) verglichen.

Das Vergleichsergebnis wird im Bedingungsschlüssel angezeigt. Es bezieht sich immer auf den ersten Operanden.

Folgende Vergleichsergebnisse sind möglich:

- gleich (erster Operand gleich zweiter Operand)
- kleiner/niedrig (erster Operand kleiner als zweiter Operand)
- größer/hoch (erster Operand größer als zweiter Operand)

Verarbeitet wird Bit für Bit von links nach rechts in der Länge des ersten Operanden.

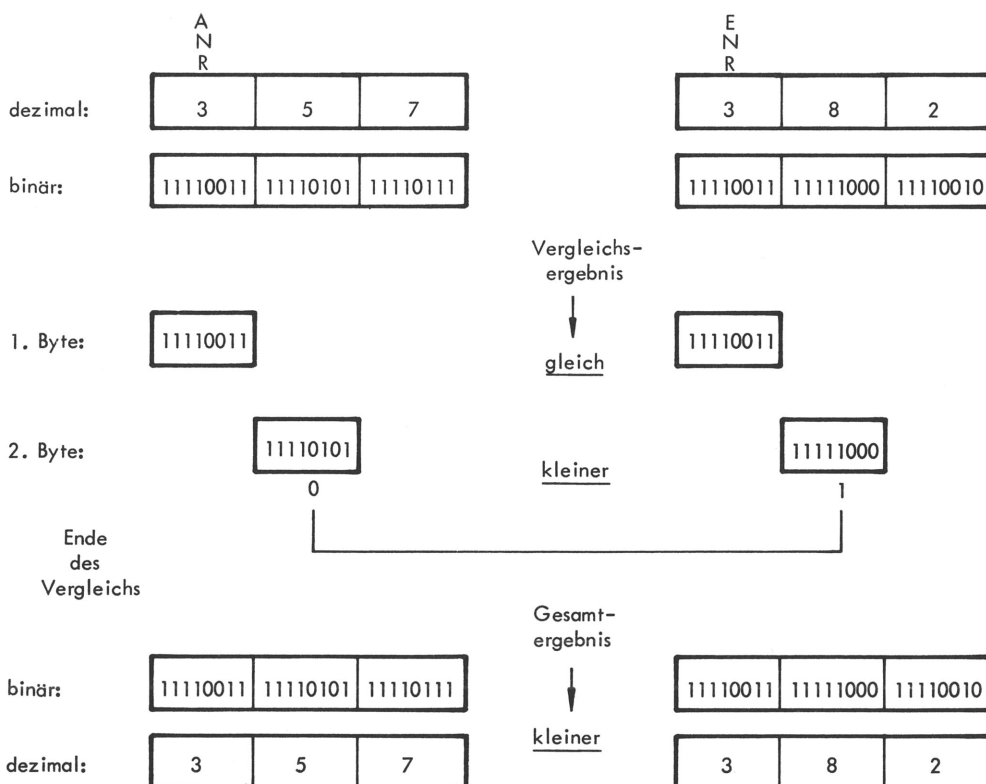
Der Vergleich endet bei der ersten Ungleichheit.

Jede der 256 möglichen Bit-Kombinationen in einem Byte ist erlaubt.

Der Inhalt beider Operanden bleibt unverändert.

Die Operanden können maximal 256 Bytes lang sein.

Durchführung der Operation:



CLI

VERGLEICHEN LOGISCH DIREKTZEICHEN

(Compare Logical Immediate)

Diese Operation vergleicht den Inhalt eines Bytes im Speicher mit einem Direktzeichen. Das Ergebnis wird im Bedingungsschlüssel angezeigt.

Beispiel einer Instruktion:

Name				Operation				Operand und Bemerkungen											
1		8		10		14		20				25				30			
				C, L, I				A, R, T, C, 'L, ' ,											

Wirkungsweise:

Der zweite Operand ist ein Direktoperand (im Beispiel C'L').

Verarbeitet wird 1 Byte und zwar Bit für Bit von links nach rechts.

Der Inhalt des ersten Operanden (ein Byte, im Beispiel ART) wird mit dem Direktzeichen (auch Direktdate, im Beispiel L) des zweiten Operanden verglichen.

Der Vergleich endet bei der ersten Ungleichheit.

Das Vergleichsergebnis wird im Bedingungsschlüssel angezeigt. Es bezieht sich immer auf den ersten Operanden.

Jede der 256 Bit-Kombinationen in einem Byte ist erlaubt.

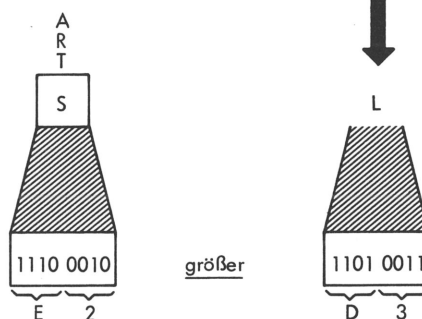
Folgende Vergleichsergebnisse sind möglich:

Der Inhalt beider Operanden bleibt unverändert.

- gleich (erster Operand gleich Direktzeichen)
- kleiner/niedrig (erster Operand kleiner als Direktzeichen)
- größer/hoch (erster Operand größer als Direktzeichen)

Durchführung der Operation:

Name				Operation				Operand und Bemerkungen											
1		8		10		14		20				25				30			
				C, L, I				A, R, T, C, 'L'											



bzw. hoch

VERZWEIGUNGSOPERATIONEN

OPERATIONS-		Schlüssel		Es wird verzweigt, wenn der Bedingungsschlüssel anzeigt:
Bezeichnung		f. Kartensystem		
Verzweigen Unbedingt (Branch)	B	BC	15	(Unabhängig von der Anzeige. Es wird immer verzweigt.)
Nach Vergleichs-Operationen:				
Verzweigen bei Größer (Branch on High)	BH	BC	2	größer (hoch)
Verzweigen bei Kleiner (Branch on Low)	BL	BC	4	kleiner (niedrig)
Verzweigen bei Gleich (Branch on Equal)	BE	BC	8	gleich
Verzweigen bei Nicht Gleich (Branch on Not Equal)	BNE	BC	7	größer oder kleiner
Verzweigen bei Nicht Kleiner (Branch on Not Low)	BNL	BC	11	größer oder gleich
Verzweigen bei Nicht Größer (Branch on Not High)	BNH	BC	13	kleiner oder gleich
Nach arithmetischen Operationen:				
Verzweigen bei Positiv (Branch on Plus)	BP	BC	2	positiv (größer als 0)
Verzweigen bei Negativ (Branch on Minus)	BM	BC	4	negativ (kleiner als 0)
Verzweigen bei Null (Branch on Zero)	BZ	BC	8	null (gleich 0)
Verzweigen bei Nicht Null (Branch on Not Zero)	BNZ	BC	7	positiv oder negativ
Verzweigen bei Nicht Negativ (Branch on Not Minus)	BNM	BC	11	positiv oder null
Verzweigen bei Nicht Positiv (Branch on Not Plus)	BNP	BC	13	negativ oder null
Verzweigen bei Überlauf (Branch on Overflow)	BO	BC	1	Überlauf
Keine Operation (No Operation)	NOP	BC	0	(Unabhängig von der Anzeige. Es wird nie verzweigt.)

Ergänzung:

Bei Benutzung der Kartenversion des Assembler darf nur die Schreibweise in der rechten Spalte benutzt werden.

Instruktionsbeispiel:

sonst:

Name	Operation	Operand und Be
1 8	10 14	20 25
	B.H.	E.5.

Für Kartenversion:

	B.C.	2, E.5.
--	------	---------

BC - "Verzweigen bei Bedingung"
(Branch on Condition)

Art der Bedingung:
"größer"

In diesem Lehrgang werden nur die Operationsschlüssel aus der linken Spalte der obigen Übersicht benutzt.

PACK

PACKEN

(PACK)

Diese Operation ändert die Speicherungsform von numerischen Daten von "ungepackt" in "gepackt".

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1 8	10 14	20 25 30
P A C K		P W E R T , E W E R T

Wirkungsweise:

Der ungepackte Inhalt des zweiten Operanden (im Beispiel EWERT) wird gepackt und in den ersten Operanden (im Beispiel PWERT) übertragen.

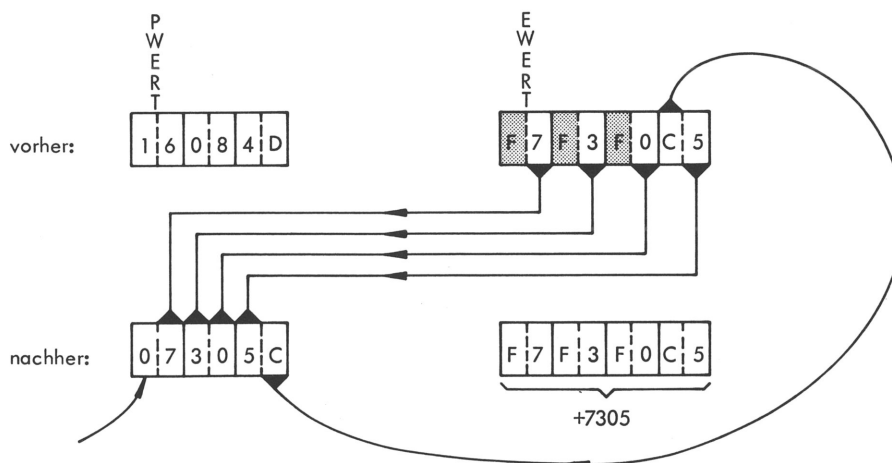
Verarbeitet wird byteweise von rechts nach links in der Länge des ersten Operanden.

Der Inhalt des zweiten Operanden wird nicht verändert.

Die Feldlängen müssen nicht übereinstimmen. Wenn der erste Operand zu lang ist, werden links führende Nullen eingeschoben. Wenn der erste Operand zu kurz ist, werden die höchsten (linken) Ziffern aus dem zweiten Operanden nicht übertragen.

Die Feldlänge jedes Operanden darf maximal 16 Bytes betragen.

Durchführung der Operation:



UNPK

ENTPACKEN

(UNPack)

Diese Operation ändert die Speicherungsform von numerischen Daten von "gepackt" in "ungepackt".

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10 14 20 25 30
.....	U,N,P,K,	A,S,U,M,M,E,,R,S,U,M,M,E, ..

Wirkungsweise:

Der gepackte Inhalt des zweiten Operanden (im Beispiel RSUMME) wird entpackt und in den ersten Operanden (im Beispiel ASUMME) übertragen.

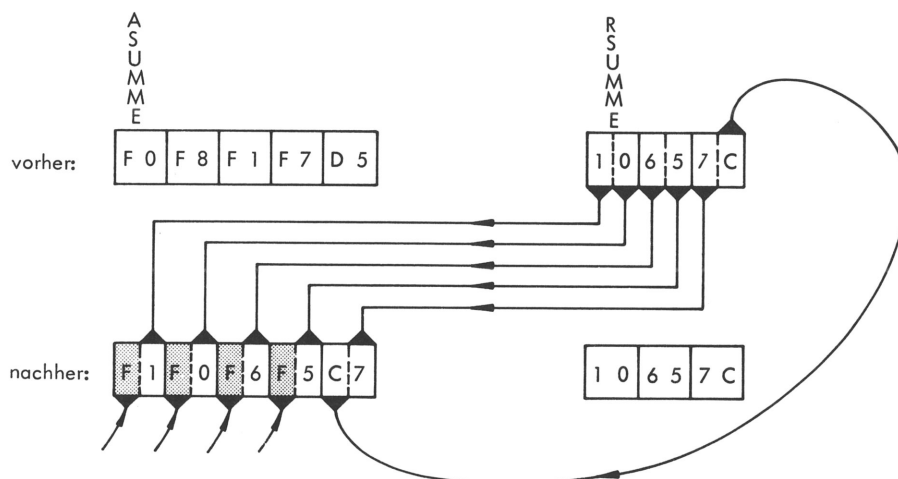
Verarbeitet wird byteweise von rechts nach links in der Länge des ersten Operanden.

Der Inhalt des zweiten Operanden wird nicht verändert.

Die Feldlängen müssen nicht übereinstimmen. Wenn der erste Operand zu lang ist, werden links führende Nullen eingeschoben. Wenn der erste Operand zu kurz ist, werden die höchsten (linken) Ziffern aus dem zweiten Operanden nicht übertragen.

Die Feldlänge jedes Operanden darf maximal 16 Bytes betragen.

Durchführung der Operation:



AP

ADDIEREN DEZIMAL (GEPACKT)

(Add decimal Packed)

Diese Operation addiert den (gepackten) Inhalt zweier Speicherfelder.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1 8	10 14	20 25 30
	A.P.	S.U.M.M.E., P.W.E.R.T.

Wirkungsweise:

Der Inhalt des zweiten Operanden (im Beispiel PWERT) wird zum Inhalt des ersten Operanden (im Beispiel SUMME) addiert und das Ergebnis im ersten Operanden gespeichert.

Vor Durchführung der Addition werden beide Operanden auf Gültigkeit (gepackte Daten) geprüft.

Verarbeitet wird byteweise von rechts nach links in der Länge des längeren Operanden.

Das kürzere Feld wird so verarbeitet, als ob es führende Nullen hätte.

Das Vorzeichen wird nach den Regeln der Algebra gebildet.

Verschlüsselung: C für +
D für -

Ein Ergebnis von Null ist stets positiv.

Der Bedingungsschlüssel wird entweder auf "Überlauf" oder auf "positiv", "negativ" oder "null" eingestellt. Die Anzeige des Überlaufs hat Vorrang.

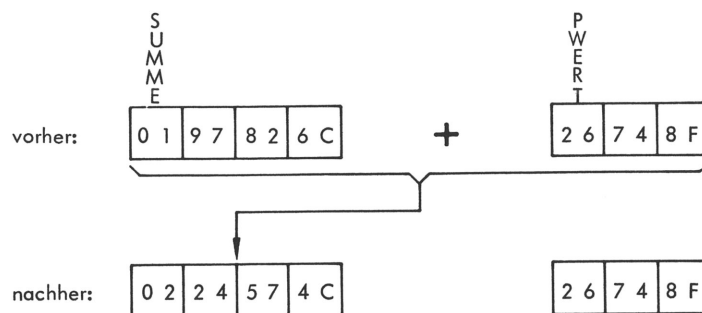
Ein Überlauf liegt vor, wenn der erste Operand zu kurz ist, um alle Ziffern der Summe - außer führenden Nullen - aufzunehmen.

Der Inhalt des zweiten Operanden wird nicht verändert.

Jeder Operand darf maximal 16 Bytes lang sein.

Bei einem Gültigkeitsfehler oder Überlauf tritt Programmunterbrechung ein.

Durchführung der Operation:



SP

SUBTRAHIEREN DEZIMAL (GEPACKT)

(Subtract decimal Packed)

Diese Operation berechnet die Differenz aus dem (gepackten) Inhalt zweier Speicherfelder.

Beispiel einer Instruktion:

Name		Operation	Operand und Bemerkungen			
1	8	10	14	20	25	30
		S,P		E,R,G	P,W,E,R,T	

Wirkungsweise:

Diese Operation entspricht in ihrer Arbeitsweise der AP-Operation.

Einzigster Unterschied: Der zweite Operand wird nicht zum ersten addiert, sondern vom ersten subtrahiert.

Im Einzelnen:

Der Inhalt des zweiten Operanden (im Beispiel PWERT) wird vom Inhalt des ersten Operanden (im Beispiel ERG) subtrahiert. Das Ergebnis wird im ersten Operanden gespeichert.

Vor Durchführung der Subtraktion werden beide Operanden auf Gültigkeit (gepackte Daten) geprüft.

Verarbeitet wird byteweise von rechts nach links in der Länge des längeren Operanden.

Das kürzere Feld wird so verarbeitet, als ob es führende Nullen hätte.

Das Vorzeichen wird nach den Regeln der Algebra gebildet.

Verschlüsselung: C für +
D für -

Ein Ergebnis von Null ist stets positiv.

Der Bedingungsschlüssel wird entweder auf "Überlauf" oder auf "positiv", "negativ" oder "null" eingestellt.

Die Anzeige des Überlaufs hat Vorrang.

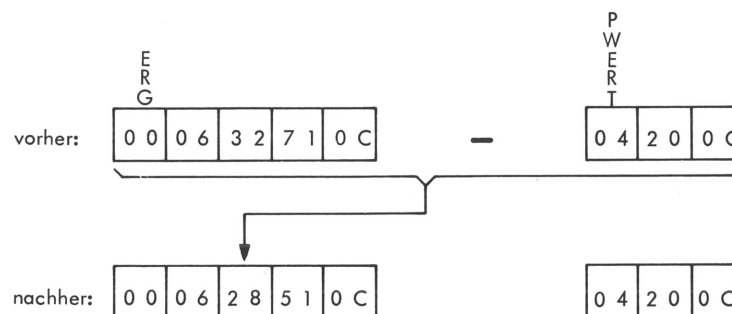
Ein Überlauf liegt vor, wenn der erste Operand zu kurz ist, um alle Ziffern des Ergebnisses - außer führenden Nullen - aufzunehmen.

Der Inhalt des zweiten Operanden wird nicht verändert.

Jeder Operand darf maximal 16 Bytes lang sein.

Bei einem Gültigkeitsfehler oder Überlauf tritt Programmunterbrechung ein.

Durchführung der Operation:



ZAP

LÖSCHEN UND ADDIEREN (GEPACKT)

(Zero and Add Packed)

Diese Operation ist gleichbedeutend mit der Addition eines Speicherfeldes in ein anderes Feld, das den Wert Null enthält.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1 8	10 14	20 25 30
	ZAP	RFELD, PWERT

Wirkungsweise:

Der Inhalt des zweiten Operanden (im Beispiel PWERT) wird so in den ersten Operanden (im Beispiel RFELD) gebracht, als ob er zum Inhalt Null im ersten Operanden addiert würde.

Nur der zweite Operand wird vor Durchführung der Operation auf Gültigkeit (gepackte Daten) geprüft.

Verarbeitet wird byteweise von rechts nach links in der Länge des längeren Operanden.

Das kürzere Feld wird so verarbeitet, als ob es führende Nullen hätte.

Die Verschlüsselung der Vorzeichen:

C für +
D für -

Ein Ergebnis von Null ist stets positiv.

Der Bedingungsschlüssel wird entweder auf "Überlauf" oder auf "positiv", "negativ" oder "null" eingestellt.

Die Anzeige des Überlaufs hat Vorrang.

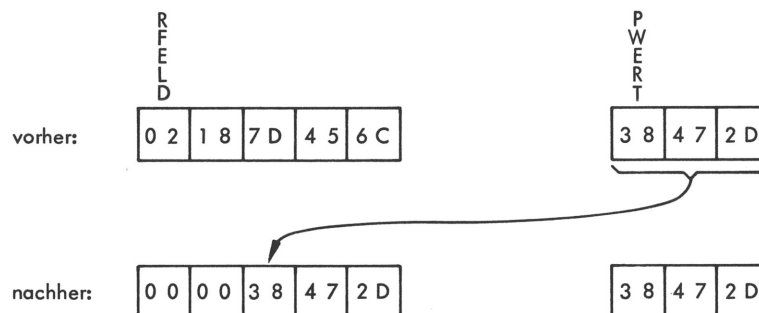
Ein Überlauf liegt vor, wenn der erste Operand zu kurz ist, um alle Ziffern des zweiten Operanden - außer führenden Nullen - aufzunehmen.

Der Inhalt des zweiten Operanden wird nicht verändert.

Jeder Operand darf maximal 16 Bytes lang sein.

Bei einem Gültigkeitsfehler oder Überlauf tritt Programmunterbrechung ein.

Durchführung der Operation:



Operation Schlüssel	Bezeichnung	Verarbeit.- Richtung	Verarbeitet werden	Kürzerer Operand mit führenden Nullen ver- arbeitet		Gültigkeitsprüfung (gepackte Daten)		Bed.-Schlüssel gesetzt auf Überlauf oder positiv bzw. negativ bzw. null
				1. Operand	2. Operand	1. Operand	2. Operand	
PACK	Packen	→	Bytes		X			
UNPK	Entpacken	→	Bytes		X			
AP	Addieren Dezimal (Gepackt)	→	Bytes	X	X	X	X	X
SP	Subtrahieren Dezi- mal (Gepackt)	→	Bytes	X	X	X	X	X
ZAP	Löschen und Addieren (Gepackt)	→	Bytes	X	X		X	X

Verarbeitet wird:

bei PACK und UNPK in der Länge des 1. Operanden (kürzerer 2. Operand – führende Nullen eingeschoben
längerer 2. Operand – linke Ziffern nicht verarbeitet)

bei AP, SP und ZAP in der Länge des längeren Operanden
(kürzerer Operand – führende Nullen eingeschoben, evtl. Fehleranzeige und Bed.-Schlüssel auf
"Überlauf" gesetzt)

Maximale Länge der Operanden: 16 Bytes

MP

MULTIPLIZIEREN DEZIMAL (GEPACKT)

(Multiply decimal Packed)

Diese Operation multipliziert den (gepackten) Inhalt zweier Speicherfelder miteinander.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10 14 20 25 30
	P A C K	P R O D, E M A N D
	Z A P	P R O D, P M A N D
	M P	P R O D, P M O R

entweder
oder
} vor MP

Wirkungsweise:

Der Multiplikand (MAND) muß vor der Ausführung der Multiplikation rechtsbündig mit führenden Nullen im Produktfeld (im Beispiel PROD) stehen.

Das Produkt aus dem ersten Operanden (im Beispiel PROD) und dem zweiten Operanden (im Beispiel PMOR) wird errechnet und im Feld des ersten Operanden gespeichert.

Vor Durchführung der Multiplikation werden beide Operanden auf Gültigkeit (gepackte Daten) geprüft.

Das Vorzeichen wird nach den Regeln der Algebra gebildet, auch wenn einer oder beide Operanden null sind.

Verschlüsselung: C für +
D für -

Der Bedingungsschlüssel wird nicht verändert.

Der Inhalt des zweiten Operanden bleibt erhalten.

Für die Länge des Produktfeldes gilt folgende Regel:

$$\text{Länge PROD} = \text{Länge MAND} + \text{Länge MOR}$$

Anmerkung:

Länge = Anzahl Bytes, die ein Wert in gepackter Form braucht,

PROD = Produkt, MAND = Multiplikand,

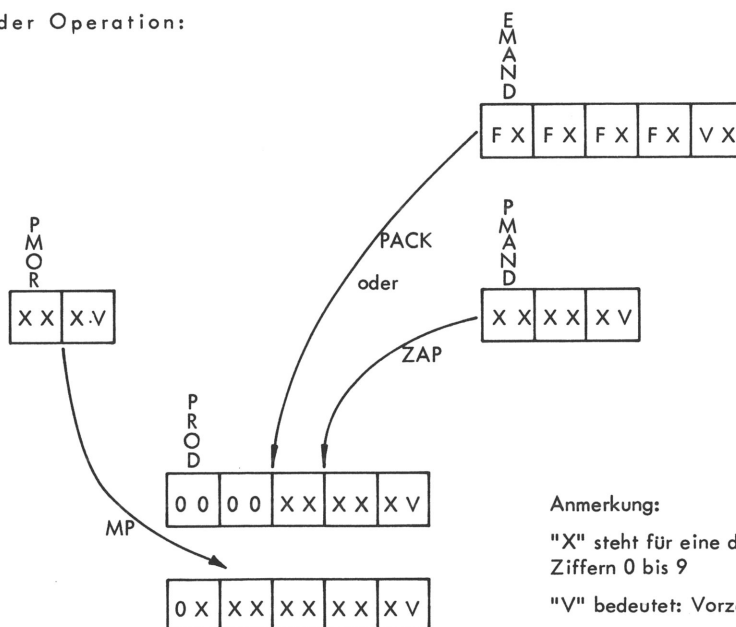
MOR = Multiplikator.

Längenbegrenzung:

Erster Operand (PROD) maximal 16 Bytes, zweiter Operand (PMOR) maximal 8 Bytes, dabei unbedingt zweiter Operand kürzer als erster Operand.

Bei Gültigkeitsfehler oder falscher Operandenlänge tritt Programmunterbrechung ein.

Durchführung der Operation:



Anmerkung:

"X" steht für eine der Ziffern 0 bis 9

"V" bedeutet: Vorzeichen

DP

DIVIDIEREN DEZIMAL (GEPACKT)

(Divide decimal Packed)

Diese Operation dividiert den (gepackten) Inhalt eines Feldes durch den (gepackten) Inhalt eines anderen Feldes.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10 14 20 25 30
	PACK	RFELD, EDEND
	ZAP	RFELD, PDEND
	DP	RFELD, PDOR

entweder } vor DP
oder }

Wirkungsweise:

Der Dividend (DEND) muß vor der Ausführung der Division rechtsbündig mit führenden Nullen im Rechenfeld (im Beispiel RFELD) stehen.

Der erste Operand (im Beispiel RFELD) wird durch den zweiten Operanden (im Beispiel PDOR) dividiert. Das Ergebnis wird im ersten Operanden gespeichert und zwar auf folgende Weise:
Der Rest steht rechtsbündig in der Länge des Divisors (DOR) im Rechenfeld.
Der Quotient (QUOT) schließt sich links an den Rest an. Seine Adresse ist die Adresse des Rechenfeldes.

Der Inhalt des zweiten Operanden (PDOR) bleibt erhalten.

Ist der Divisor null, so wird die Division nicht durchgeführt und ein Fehler angezeigt.

Außerdem werden vor Durchführung der Division beide Operanden auf Gültigkeit (gepackte Daten) geprüft.

Das Vorzeichen des Quotienten wird nach den Regeln der Algebra gebildet.

Verschlüsselung: C für +
D für -

Der Rest hat das Vorzeichen des Dividenten. Diese Regeln gelten auch dann, wenn Quotient und Rest null sind.

Der Bedingungsschlüssel wird nicht verändert.

Für die Länge des Rechenfeldes gilt folgende Regel:

Länge RFELD = Länge DEND + Länge DOR

Anmerkung:

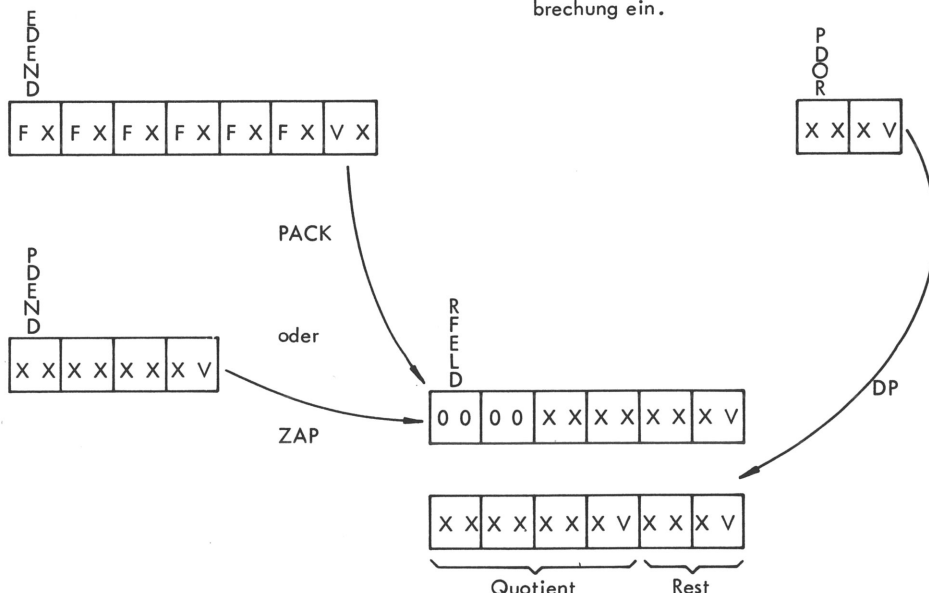
Länge = Anzahl Bytes, die ein Wert in gepackter Form braucht,
RFELD = Rechenfeld, DEND = Dividend,
DOR = Divisor.

Längenbegrenzung:

Erster Operand (RFELD) maximal 16 Bytes,
zweiter Operand (PDOR) maximal 8 Bytes,
dabei unbedingt zweiter Operand kürzer als erster Operand.

Bei Gültigkeitsfehler, Divisor gleich null oder falscher Operandenlänge tritt Programmunterbrechung ein.

Durchführung der Operation:



MVO

UMPACKEN

(MoVe with Offset)

Diese Operation überträgt den Inhalt eines Feldes um ein halbes Byte nach links versetzt in ein anderes Feld.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1 8	10 14	20 25 30
	M.V.O.	F.E.L.D.A., F.E.L.D.B.

Wirkungsweise:

Der Inhalt des zweiten Operanden (im Beispiel FELDB) wird in das Feld des ersten Operanden (im Beispiel FELDA) übertragen und zwar um ein halbes Byte nach links versetzt. Das rechte Halbbyte im rechten Byte des ersten Operanden wird nicht verändert.

Der Bedingungsschlüssel wird nicht verändert.

Der Inhalt des zweiten Operanden bleibt erhalten.

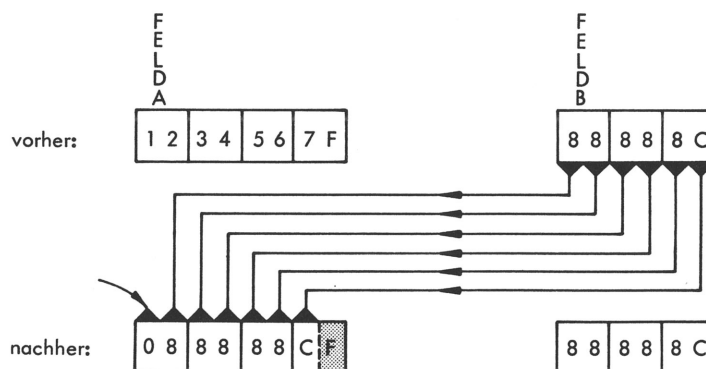
Keiner der Operanden wird auf Gültigkeit geprüft.

Die Feldlänge jedes Operanden beträgt maximal 16 Bytes.

Verarbeitet wird byteweise von rechts nach links.

Ist der erste Operand zu lang, werden führende Nullen eingeschoben. Ist der erste Operand zu kurz, so werden die linken Daten aus dem zweiten Operanden nicht übertragen.

Durchführung der Operation:



CP

VERGLEICHEN DEZIMAL (GEPACKT)

(Compare decimal Packed)

Diese Operation vergleicht vorzeichengerecht den (gepackten) Inhalt zweier Speicherfelder.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1 8	10 14	20 25 30
	C,P	F,E,L,D,A,,F,E,L,D,B,

Wirkungsweise:

Der Inhalt des ersten Operanden (im Beispiel FELDA) wird mit dem Inhalt des zweiten Operanden (im Beispiel FELDB) verglichen.

Das Vergleichsergebnis, das sich immer auf den ersten Operanden bezieht, wird im Bedingungs-
schlüssel angezeigt.

Es lautet kleiner (FELDA kleiner als FELDB)
oder größer (FELDA größer als FELDB)
oder gleich (FELDA gleich FELDB).
Ein Überlauf kann nicht auftreten.

Vor Durchführung der Operation werden beide Operanden auf Gültigkeit (gepackte Daten) geprüft.

Verarbeitet wird byteweise von rechts nach links in der Länge des längeren Operanden.

Das kürzere Feld wird so verarbeitet, als ob es führende Nullen hätte.

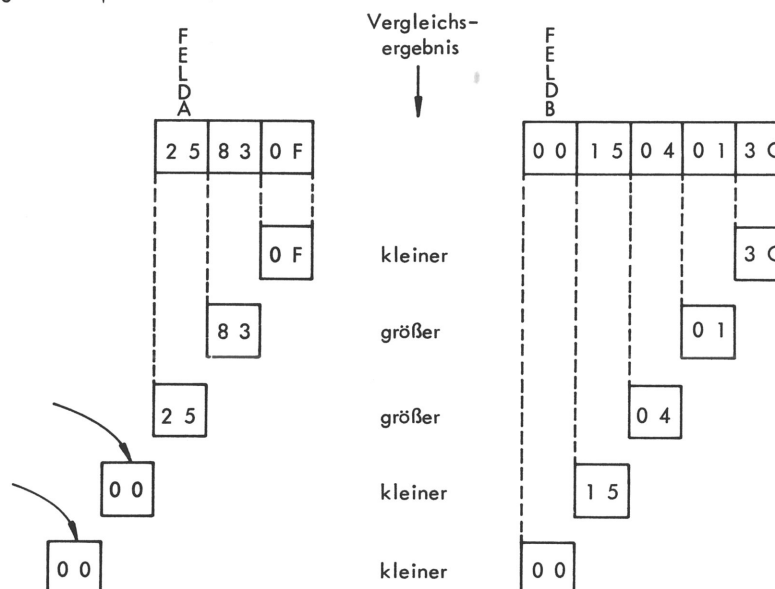
Der Vergleich erfolgt algebraisch (vorzeichen-
gerecht). Negative Zahlen sind immer kleiner als positive, unabhängig von der Verschlüsselung der Vorzeichen.
Sonderfall: "plus null" ist gleich "minus null".

Der Inhalt beider Operanden bleibt unverändert.

Jeder Operand darf maximal 16 Bytes lang sein.

Bei einem Gültigkeitsfehler tritt Programmunterbrechung ein.

Durchführung der Operation:



Endergebnis:

FELDA kleiner als FELDB

Operation		Verarbeit. Richtung	Verarbeitet werden	Kürzerer Operand mit führenden Nullen ver- arbeitet		Gültigkeitsprüfung (gepackte Daten)		Bed.-Schlüs- sel gesetzt auf gleich od. kleiner od. größer
Schlüssel	Bezeichnung			1. Operand	2. Operand	1. Operand	2. Operand	
MP	Multiplizieren Dezimal (Gepackt)		Bytes			X	X	
DP	Dividieren Dezimal (Gepackt)		Bytes			X	X	
MVO	Umpacken	→	Bytes		X			
CP	Vergleichen De- zimal (Gepackt)	→	Bytes	X	X	X	X	X

Ergänzung zur Länge des Produktfeldes (vgl. Merkblatt 415):

- Das Produktfeld kann länger sein als die Regel verlangt. Das Produkt erhält dann entsprechend viele führende Nullen.
- Das Produktfeld kann auch kürzer gewählt werden. Es muß jedoch lang genug sein für Ziffern und Vorzeichen des Produktes und zusätzlich mindestens eine führende Null.

Ergänzung zur Länge des Divisionsfeldes (vgl. Merkblatt 416):

- Das Rechenfeld kann länger sein als die Regel verlangt. Der Quotient erhält dann entsprechend viele führende Nullen.
- Das Rechenfeld kann auch kürzer gewählt werden. Es muß jedoch folgende Bedingungen erfüllen:
 - Der Dividend muß vor der Division mit mindestens einer führenden Null im Rechenfeld stehen.
 - Das Rechenfeld muß lang genug sein, um den Rest (in der Länge des Divisors) und den Quotienten einschließlich des Vorzeichens aufnehmen zu können.

ED

AUFBEREITEN ZUM DRUCKEN

(EDit)

Diese Operation bewirkt, daß eine gepackte Zahl mit Hilfe einer Schablone ein durch Punkt, Komma, Minuskennzeichnung usw. aufbereitetes Druckbild erhält.

Beispiel einer Instruktion:

Name	Operation	Operand und Bemerkungen
1	8	10 16 20 25 30
SCHAB	DC	C, ' ▽ ▽ . ▽ I ▽ , ▽ ▽ - '
	MVC	ABETRAG, SCHAB
	ED	ABETRAG, PBETRAG

Wirkungsweise:

Diese Operation setzt voraus, daß vor ihrer Durchführung eine entsprechende Schablone (im Beispiel SCHAB) im Ergebnisfeld (im Beispiel ABETRAG) steht.

Die Schablone wird als Konstante definiert. Zur Übertragung dient die MVC-Instruktion.

Der Inhalt des zweiten Operanden (im Beispiel PBETRAG) wird durch ED vom gepackten in das ungepackte Format überführt und unter Kontrolle der Schablone im ersten Operanden aufbereitet.

Die Schablone im ersten Operanden wird nicht auf Gültigkeit geprüft. Sie kann beliebige Zeichen enthalten. Der zweite Operand muß gepacktes Format haben. Genauer gesagt: Jedes linke Halbbyte muß eine der Ziffern 0 bis 9 enthalten. In jedem rechten Halbbyte darf eine der Ziffern 0 bis 9 oder ein Vorzeichen stehen.

Verarbeitet wird byteweise von links nach rechts in der Länge des ersten Operanden.

Der Bedingungsschlüssel wird neu gesetzt – abhängig vom Inhalt des zweiten Operanden – auf positiv, negativ oder null.

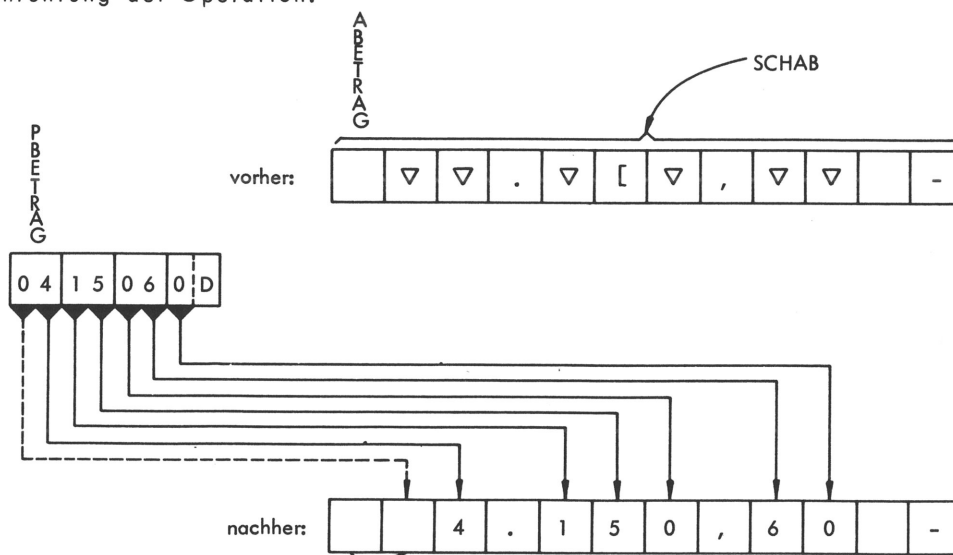
Ein Überlauf wird nicht angezeigt. Ist der erste Operand zu kurz, so wird der rechte Teil des zweiten Operanden nicht aufbereitet.

Der Inhalt des zweiten Operanden bleibt erhalten.

Der erste Operand darf maximal 256 Bytes lang sein.

Bei Gültigkeitsfehler tritt Programmunterbrechung ein.

Durchführung der Operation:



DIE SCHABLONE

Name	Symbol	Ver- schlüsselung	Lochkarten- Code
Ziffern- Auswahl- Zeichen	▽	0010 0000	11,0,9,8,1
Start-Zei- chen für geltende Ziffern	[	0010 0001	0,9,1
Feld- Trennungs- Zeichen	]	0010 0010	0,9,2

- Jedem der Zeichen "▽" und "[" in der Schablone wird von links nach rechts eine Ziffer aus dem Datenfeld zugeordnet. Die Anzahl der Zeichen "▽" und "[" muß der Anzahl der Ziffern genau entsprechen.
- Das linke Byte der Schablone enthält immer das Füllzeichen. Als Füllzeichen dient ein beliebiges Zeichen und zwar im allgemeinen das Blank.
- Zu Beginn der Druckaufbereitung ist die Unterdrückung führender Nullen, kurz "Nullenunterdrückung", wirksam.
 - Die Nullenunterdrückung wird aufgehoben für die nächste (rechts folgende) Stelle
 - durch eine der Ziffern 1 bis 9 aus dem Datenfeld,
 - durch das "Start-Zeichen für geltende Ziffern" ("[" in der Schablone.
 - Die Nullenunterdrückung wird wieder wirksam
 - durch ein positives Vorzeichen in einem rechten Halbbyte des Datenfeldes,
 - durch das "Feld-Trennungs-Zeichen" ("]").
- Während die Nullenunterdrückung wirksam ist, wird das Füllzeichen eingesteuert
 - statt führender Nullen,
 - statt "anderer" Zeichen (z.B. Punkt, Komma, DM, CR usw.).
- Ist die Nullenunterdrückung nicht wirksam, so
 - werden führende Nullen nicht unterdrückt,
 - bleiben "andere" Zeichen (z.B. Punkt, DM, CR usw.) für das Druckbild erhalten.
- Ziffern 1 bis 9 aus dem Datenfeld werden nie durch das Füllzeichen ersetzt. Das Zeichen "]" in der Schablone wird immer durch das Füllzeichen ersetzt.
- Ein positives Vorzeichen im Datenfeld hat keine andere Wirkung als die unter 3. c) 1) angegebene. Ein negatives Vorzeichen bewirkt gar nichts.

BEISPIELE

Schablone:

*	▽	┌	▽	,	▽	▽	C	R
---	---	---	---	---	---	---	---	---

DATEN	D.S.	C.L.9
SCHAB	D.C.	C',*▽┌,▽,▽,▽C,R'
AFELD	D.S.	C.L.9
MVC	AFELD	SCHAB
ED	AFELD	DATEN

Daten:

1	2	3	4	5	C
0	2	3	4	5	C
0	0	3	4	5	C
0	0	0	4	5	C
0	0	0	0	5	C
0	0	0	0	0	C

Druckbild:

*	1	2	3	,	4	5	*	*
*	*	2	3	,	4	5	*	*
*	*	*	3	,	4	5	*	*
*	*	*	0	,	4	5	*	*
*	*	*	0	,	0	5	*	*
*	*	*	0	,	0	0	*	*

1	2	3	4	5	D
0	2	3	4	5	D
0	0	3	4	5	D
0	0	0	4	5	D
0	0	0	0	5	D
0	0	0	0	0	D

*	1	2	3	,	4	5	C	R
*	*	2	3	,	4	5	C	R
*	*	*	3	,	4	5	C	R
*	*	*	0	,	4	5	C	R
*	*	*	0	,	0	5	C	R
*	*	*	0	,	0	0	C	R

II Schablone:

▽	┌	▽	,	▽	▽	-	D	M
---	---	---	---	---	---	---	---	---

SCHAB	D.C.	C',▽┌,▽,▽,▽-D,M'
PWERT	D.S.	C.L.9
DRUCK	D.S.	C.L.1.0
MVC	DRUCK	SCHAB
ED	DRUCK(B)	PWERT

Daten:

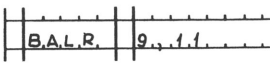
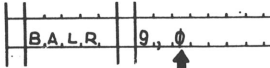
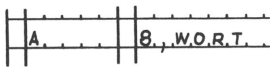
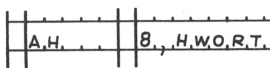
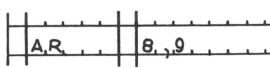
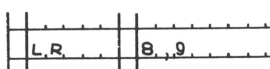
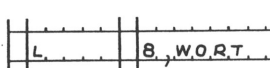
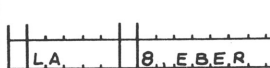
1	2	3	4	5	F
0	2	3	4	5	F
0	0	3	4	5	F
0	0	0	4	5	F
0	0	0	0	5	F
0	0	0	0	0	F

Druckbild:

1	2	3	,	4	5		D	M
	2	3	,	4	5		D	M
		3	,	4	5		D	M
		0	,	4	5		D	M
		0	,	0	5		D	M
		0	,	0	0		D	M

1	2	3	4	5	D
0	2	3	4	5	D
0	0	3	4	5	D
0	0	0	4	5	D
0	0	0	0	5	D
0	0	0	0	0	D

1	2	3	,	4	5	-	D	M
	2	3	,	4	5	-	D	M
		3	,	4	5	-	D	M
		0	,	4	5	-	D	M
		0	,	0	5	-	D	M
		0	,	0	0	-	D	M

Operation		Beispiel		Bed.-Schl. gesetzt auf "Überlauf" oder "positiv", "negativ" od. "null"
Schlüssel	Bezeichnung	Instruktion	Wirkungsweise	
BALR	Verzweigen und Laden Folgeadresse - Verzweigadresse im Reg. -		1. Zwischenspeichern Verzweigadresse aus dem 2. Operanden (Register 11) 2. Speichern Folgeadresse in den 1. Operanden (Register 9) 3. Verzweigen zu der zwischengespeicherten Adresse	
			Kein Verzweigen Nur: Speichern Folgeadresse nach Register 9	
A	Addieren		Addition 2-er Dualzahlen von Wortlänge 1. Operand: ein allgemeines Register 2. Operand: ein Feld im Hauptspeicher (WORD) Ergebnis im 1. Operanden (Register 8)	X
AH	Addieren Halbwort		Addition 2-er Dualzahlen 1. Operand: ein allgemeines Register 2. Operand: ein Halbwort im Hauptsp. (HWORT) Ergebnis im 1. Operanden (Register 8)	X
AR	Addieren Register		Addition 2-er Dualzahlen von Wortlänge Beide Operanden: allgemeine Register Ergebnis im 1. Operanden (Register 8)	X
LR	Umladen		Umladen des Inhalts eines allgemeinen Registers (2. Operand - Register 9 -) in ein anderes allgemeines Register (1. Operand - Register 8 -)	
L	Laden		Laden einer Dualzahl von Wortlänge aus einem Feld im Hauptspeicher (2. Operand - WORD -) in ein allgemeines Register (1. Operand - Register 8 -)	
LA	Laden Adresse		Laden der (effektiven) Hauptspeicheradresse des 2. Operanden (Adresse von EBER) in ein allgemeines Register (Register 8)	

DIE INSTRUKTIONEN UND IHRE FORMATE

SS-Format Logisch:

OP	L	B1	D1	B2	D2
					1 Byte

SS-Format Arithmetisch:

OP	L1	L2	B1	D1	B2	D2
----	----	----	----	----	----	----

SI-Format

OP	I	B1	D1
----	---	----	----

RX-Format

OP	M1	X2	B2	D2
----	----	----	----	----

RX-Format

OP	R1	X2	B2	D2
----	----	----	----	----

RR-Format

OP	R1	R2
----	----	----

Op.- Schlüssel	Verarbeit.- Richtung	Verarbeitet werden	Kürzerer Operand mit führenden Nullen verarbeitet 1. Operand 2. Operand	
-------------------	-------------------------	-----------------------	--	--

MVC	→	Bytes		
MVZ	→	linke		
MVN	→	Halbbbytes		
		rechte		
CLC	→	Halbbbytes		
ED	→	Bits		
		Bytes		

PACK	←	Bytes		X
UNPK	←	Bytes		X
AP	←	Bytes	X	X
SP	←	Bytes	X	X
ZAP	←	Bytes	X	X
MP		Bytes		
DP		Bytes		
MVO	←	Bytes		X
CP	←	Bytes	X	X

MVI	→	1 Byte		
CLI	→	8 Bits		

B				
und alle bedingten und unbedingten Verzweigungen- vgl. Merkblatt 408				

A		1 Wort		
AH		1 Halbwort		X
L		1 Wort		
LA		1 Wort		

BALR				
AR		1 Wort		
LR		1 Wort		

