AEG TELEFUNKEN DATENVERARBEITUNG

TR 440

Befehls-Lexikon

Inhalt

1. Einleitung

In diesem Befehlslexikon werden die mikroprogrammtechnisch organisierten Befehle zum Programmieren von Operatoren für den Digital-Rechner TR 440 beschrieben.

Die Befehle sind in alphabetischer Reihenfolge der Befehlscodes, die sich aus den mnemotechnichen Abkürzungen der Befehlsbezeichnungen ergeben, geordnet.

Eine nach Sachgebieten gegliederte Aufstellung wird in der Druckschrift "TR 440 - Kleine Befehlsliste" angeboten. Als handliche Arbeitsunterlage steht dem Programmierer die "TR 440 - Große Befehlsliste" zur Verfügung. In knapper Tabellenform werden die nötigen Informationen vermittelt. Das Zusammenwirken der Befehle innerhalb der Programme wird im TAS-Handbuch erläutert.

Tabellen zu den Befehlen befinden sich hinter den Befehlsformularen.

2. Einteilung des Formulars

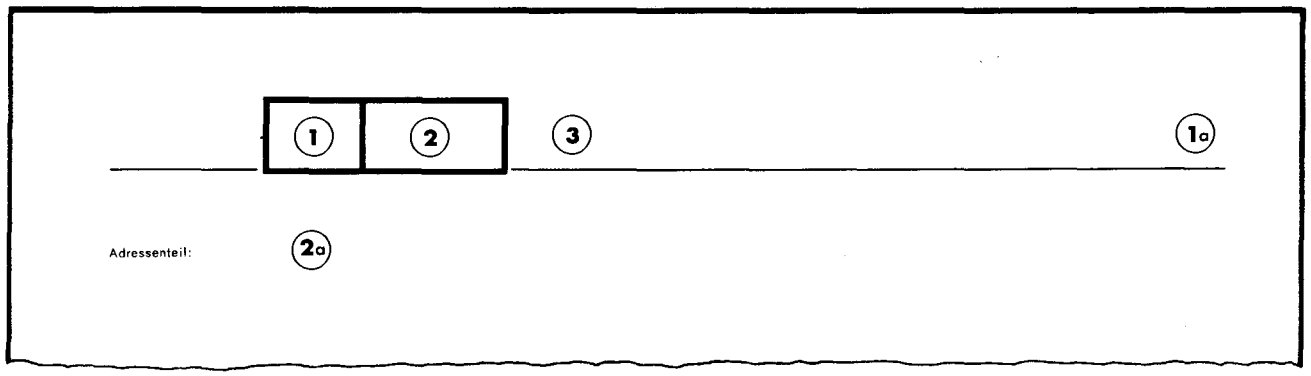
Die Einteilung des Formulars soll dazu verhelfen, die Wirkung der Befehle übersichtlich geordnet zu verfolgen. Die Beschreibung der jeweiligen Operation erfolgt immer in der gleichen Weise und an der gleichen Stelle des Formulars. Der Normalfall der Wirkung wird stets auf der Vorderseite des Formulars beschrieben. Die aufgeführten Gleichungen für den Normalfall werden textlich erläutert. Selten auftretende Fälle und Sonderfälle (bei evtl. nicht eingehaltenen Voraussetzungen) sind auf der Rückseite aufgeführt. Für diese Fälle werden nur Gleichungen ohne textliche Erläuterungen angegeben.

Abkürzungen, Buchstaben und Zeichen werden unter 3 "Erklärung der Zeichen" beschrieben.

16 Bits können bei der Angabe der Speicheradresse eines Ganzwortes n Werte von 0 bis 65 535 erreicht werden; (n hat nur geradzahlige Adressen, ungeradzahlige werden um 1 vermindert).

Bei der Speicheradresse eines Halbwortes m können ebenfalls Werte von 0 bis 65 535 angegeben werden.

Steht z (Zahl) im Adressenteil des Befehls, so können hier ebenso Werte von 0 bis 65 535 angegeben werden (Ausnahme: Befehle AA, SBA).



(1) Code:

Der Befehlscode bildet sich aus der mnemotechnischen Abkürzung der Befehlsbezeichnung, die unter (3) angegeben ist. In der rechten oberen Ecke steht dieser Code in großen Buchstaben (1a) um ein leichtes Auffinden beim Durchblättern der Beschreibungen zu ermöglichen.

(2) Adressenteil:

An dieser Stelle des Kästchens wird angegeben, was der Adressenteil eines Befehls enthalten muß. Auch wenn die hier verwendeten Buchstaben ohne Bedeutung für die Operation sind, so müssen für diese Buchstaben bei der Niederschrift eines Programms stets Werte eingesetzt werden (evtl. Null).

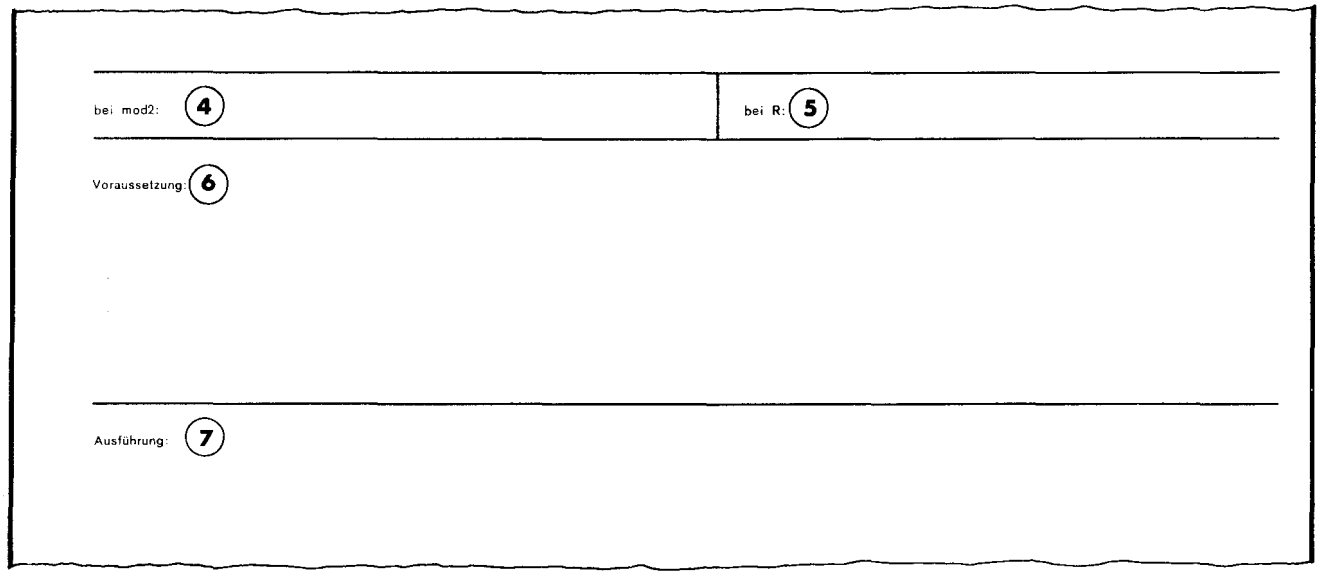
(2a) Hier werden die im Kästchen (2) angegebenen Buchstaben näher erklärt. Der Adressenteil eines Befehls ist 16 Bits groß. In diesen Der 16 Bits große Adressenteil des Befehls wird in der Abrufphase (links durch 0-Bits) auf 24 Bits erweitert. Durch Modifizierung können die Werte im Adressenteil eines Befehls erhöht werden.

Wenn der Buchstabe i (Indexadresse) im Adressenteil steht, können Werte von 0 bis 255 angegeben werden.

Im Adressenteil eines Befehls können auch mehrere Angaben stehen. Spezifikationen wie s können durch Indizes s_1, s_2 ... unterteilt werden. Bei der Angabe von i, p und s kann durch die Indizes L für "links" oder R für "rechts" unterschieden werden.

(3) Bezeichnung:

Die Bezeichnung des Befehls gibt in wenigen Worten die Wirkung an. Buchstaben, die zum Befehlscode (Mnemocode) (1) führen, sind unterstrichen.



④ mod2:

Ein Modifikator 2. Art ist eine 24 Bit-Größe, die vom Vorbefehl vorhanden sein kann.

An dieser Stelle ist angegeben, ob und wie dieser mod2 auf den Befehl einwirkt.

Es gibt 3 Möglichkeiten für mod2

1. w i r d m o d i f i z i e r t:

mod2 wird während der Abrufphase zum Adressenteil des Befehls addiert. Anschließend wird mod2 gelöscht.

```
adr  := +mod2
mod2 := 0
```

2. wird s p e z i e l l modifiziert:

Der Befehl wird während der Ausführungsphase modifiziert. Die Art dieser Modifizierung wird hier jeweils unter "Ausführung" 7 beschrieben.

3. wird n i c h t modifiziert:

Der Befehl wird nicht modifiziert, mod2 wird in der Abrufphase gelöscht. mod2 := 0

⑤ bei R:

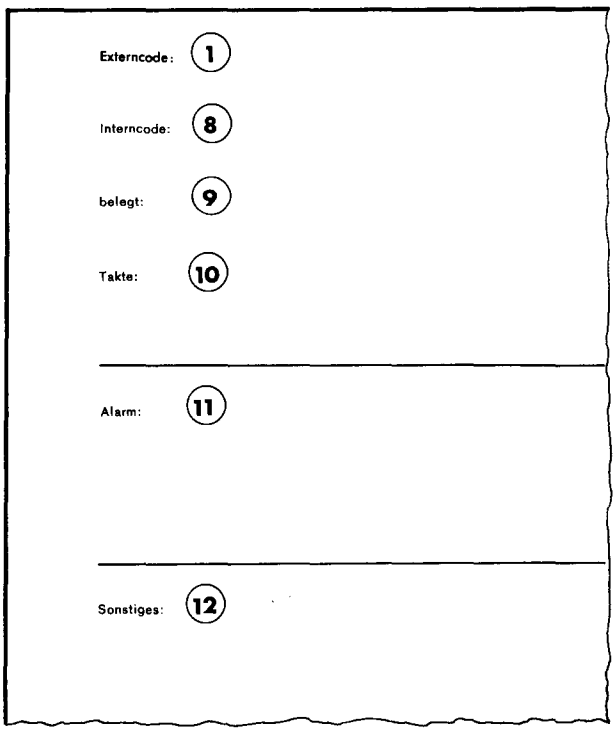
Es wird angegeben, ob der Befehl als Zweitcode beim Befehl R (Registeradressierung) zugelassen ist. Eine Aufstellung aller bei R zugelassenen Befehlen befindet sich auf der Rückseite des Befehls R und in der "Alphabetischen Liste der Befehle".

⑥ Voraussetzung:

Hier wird angezeigt unter welchen Voraussetzungen die unter ⑦ aufgeführte Ausführung auftritt. Die Wirkungen bei nicht eingehaltenen Voraussetzungen sind auf der Rückseite unter "Sonstiges" ⑫ beschrieben.

⑦ Ausführung:

Die Wirkung des Befehls wird für den Normalfall (bei erfüllten Voraussetzungen) an dieser Stelle formal beschrieben. Gleichungen werden in der Reihenfolge der Ausführung angegeben. Es wird davon ausgegangen, daß die vorhergehenden Gleichungen jeweils bereits ausgeführt sind. Hier ist nicht angestrebt den internen Ablauf des Befehls, sondern die Wirkung anzuzeigen, wie sie für den Benutzer in Erscheinung tritt. Nicht aufgeführte Register werden durch den Befehl nicht verändert. Die aufgeführten Gleichungen werden anschließend textlich erläutert um ein besseres Verständnis der Zeichen, Buchstaben und Indizes für den Ungeübten zu ermöglichen. Aus gleichem Grund wird auf einigen Formularen die Beschreibung durch Zeichnungen unterstützt.



(8) Interncode:

Unter dem auf der Rückseite der Formulare nochmals aufgeführten Befehlscode (Externcode) (1) steht hier jeweils die interne Darstellung des Befehlscodes in zwei Sedezimalen (2 x 4 Bits). Eine Übersicht über die Interncodes und die entsprechenden Befehlscodes ist in der Tabelle "Interncode - Externcode" zu finden.

(9) belegt:

Hier wird angegeben, ob der Befehl bei seiner Ausführung das Befehlswerk, das Rechenwerk oder beide Werke belegt. Zwischen Rechenwerk und Befehlswerk ist eine parallele Arbeit möglich.

(10) Takte:

Die Ausführungszeit des jeweiligen Befehls ist hier in Takten angegeben. Es handelt sich zum Teil um Mittelwerte. Sind unter "belegt" (9) Befehlswerk und Rechenwerk aufgeführt, so gilt die Zeit für beide Werke. Für das Rechenwerk kann in vielen Fällen eine kürzere Zeit benötigt werden, die aber dann ohne Vorteil ist. Ist unter Werk nur das Rechenwerk aufgeführt, können während der angegebenen Zeit parallel dazu die Ausführungsphase und alle Befehle ablaufen, die nur das Befehlswerk ansprechen.

Zu den auf den Formularen angegebenen Zeiten der Ausführungsphase müssen noch die Zeiten für die Abrufphase, die das Befehlswerk belegen, hinzugerechnet werden.

Für die einfache Zeitberechnung gelten für die Abrufphase folgende Zeiten als Mittelwerte:

	Takte
Befehlsabruf	8
belegt Rechenwerk	1
Modifizierung 1. Art	8
Modifizierung 2. Art (wenn bei mod2: +)	8
bei Operand aus dem Speicher	8
bei erfüllter Sprungbed.	5

Spezieller aufgeführt sind die benötigten Zeiten im Abschnitt 4 "Zeitberechnung"

(11) Alarm:

An dieser Stelle des Formulars wird angegeben, unter welchen Bedingungen ein Alarm auftritt. Es ist zu unterscheiden zwischen Typenkennungsalarm (TK-Alarm) und Bereichsüberschreitungsalarm (BÜ-Alarm).

Ein Typenkennungsalarm tritt auf, wenn die unter "Voraussetzung" (6) angegebene Typenkennung nicht eingehalten wurde.

Ein Bereichsüberschreitungsalarm zeigt an, ob der zulässige Zahlenbereich überschritten wurde.

Die bei einem Alarm auftretende Wirkung wird unter "Sonstiges" (12) beschrieben.

(12) Sonstiges:

Alle Wirkungen eines Befehls die nicht dem Normalfall oder den Voraussetzungen entsprechen werden hier aufgeführt. Dazu gehört auch die Ausführung bei auftretenden Alarmen. Gleichungen werden hier nicht mehr textlich erläutert. Falls es erforderlich ist wird hier die Internspezifikation des jeweiligen Befehls angegeben. Eine Zusammenstellung von Internspezifikationen ist auch in der "Großen Befehlsliste" zu finden.

3. Erklärung der Zeichen

In diesem Befehlslexikon werden folgende Bezeichnungen verwendet.

Bezeichnung der Register im Rechenwerk:

A	Akkumulator	48 Bits Informationen und 2 Bits Typenkennung
Q	Quotientenregister	
D	Multiplikandenregister	
H	Hilfsregister	
Y	Schiftzähler	8 Bits
M	Markenregister	1 Bit
A,Q	doppeltlanges Register A und Q	
H,Q	doppeltlanges Register H und Q	

Bezeichnung der Register im Befehlswerk:

B	Bereitadressenregister	24 Bits
F	Befehlsfolgeregister	
X	Indexbasisregister	22 Bits
K	Merklichterregister	8 Bits
U	Unterprogrammregister	

Variable:

mod1	Modifikator erster Art	24-Bit-Größe im Register B oder in einem nicht adressierbaren Register
mod2	Modifikator zweiter Art	
op	Operationscode eines Befehls	8 Bits
adr	Inhalt des Adressenteils eines Befehls	16 Bits, links (durch 0-Bits) auf 24 Bits erweitert
n	Speicheradresse eines Ganzwortes nur geradzahlige Adressen, ungeradzahlige werden um 1 vermindert	Adressenteil 16 Bits, wird in der Abrufphase, links (durch 0-Bits) auf 24 Bits erweitert
m	Speicheradresse eines Halbwortes	Der Zahlenbereich kann durch Modifizierung erhöht werden.
z	Zahl oder Operand	
i	Indexadresse	Indizes: L für links
p	Parameter	
s	Spezifikation (s evtl. unterteilt in $s_1, s_2, \ldots$)	R für rechts
c	Zweitcode (Code für den Zweitbefehl)	

Zeichen und ihre Bedeutung:

〈 〉 Inhalt eines Registers, einer Speicher- oder Indexzelle,
Beispiel: 〈A〉
Inhalt des Registers A

〈 〉,〈 〉 Inhalt von zwei getrennten Registern,
Beispiel: 〈A〉,〈Q〉
Inhalt der Register A und Q

〈 , 〉 Inhalt zweier Register, die zu einem doppeltlangen Register zusammengefaßt sind,
Beispiel: 〈A,Q〉
Inhalt der zu einem doppeltlangen Register vereinigten Register A und Q

| | Betrag einer Größe,
Beispiel: |〈n〉|
Betrag vom Inhalt der Speicherzelle n

:= Die links stehende Zielgröße ergibt sich aus der rechts stehenden Quellengröße,
Beispiel: 〈A〉 := 〈n〉
Der Inhalt des Registers A ergibt sich aus dem Inhalt der Speicherzelle n

:=: links- und rechtsstehende Größen werden miteinander vertauscht,
Beispiel: 〈A〉 :=: 〈H〉
Die Inhalte der Register A und H werden vertauscht

t_x; Typenkennung im Register x oder in der Speicherzelle x (Die Typenkennung steht vor dem Semikolon),
Beispiel: 〈A〉 := t_n;〈n〉
Der Inhalt des Registers A ergibt sich aus dem Inhalt der Speicherzelle n; die Typenkennung des Registers A ergibt sich ebenso aus der Typenkennung der Speicherzelle n.

1,1; Typenkennung in beiden Registern eines doppeltlangen Registers,
Beispiel: 〈A,Q〉 := 1,1;〈A,Q〉 + 〈n〉
Der Inhalt des doppeltlangen Registers A,Q ergibt sich aus dem Inhalt von A,Q plus dem Inhalt der Speicherzelle n; die Typenkennung in beiden Teilen des doppeltlangen Registers A,Q ergibt sich zu 1.

0, Der linke Teil des betreffenden Registers ist mit Null aufgefüllt,
Beispiel: 〈A〉 := t_m;0,〈m〉
Der Inhalt des Registers A ergibt sich im rechten Teil aus dem Inhalt des Halbwortes der Speicherzelle m, der linke Teil des Registers A wird mit Null aufgefüllt; die Typenkennung von A ergibt sich aus der Typenkennung des Halbwortes m.

v, · Der linke Teil des betreffenden Registers ist vorzeichengleich aufgefüllt,
Beispiel: 〈A〉 := 1;v,〈m〉
Der Inhalt des Registers A ergibt sich im rechten Teil aus dem Inhalt des Halbwortes m, das 1. Bit (linke Bit) des Halbwortes m ist das Vorzeichen ($\langle m \rangle_v$), der linke Teil des Registers A wird mit vorzeichengleichen Bits aufgefüllt.

Indizes für Teile eines Wortes:

$\langle\ \rangle_1$ Bit 1 vom Inhalt,
Beispiel: $\langle A\rangle_1$
Bit 1 vom Inhalt des Registers A

$\langle\ \rangle_{41-48}$ Bit 1 - 48 vom Inhalt,
Beispiele: $\langle A\rangle_{1-48}$
Bits 1 bis 48 vom Inhalt des Registers A

$\langle n\rangle_{9-24}$
Bits 9 bis 24 vom Inhalt der Speicherzelle n (Drittelwort)

$\langle\ \rangle_{1,2}$ Bit 1 und 2 vom Inhalt,
Beispiel: $\langle A\rangle_{1,2}$
Bit 1 und 2 vom Inhalt des Registers A
$(\langle A\rangle_{1,2} = \langle A\rangle_v)$
zur Zählung der Bits siehe Seite IV Wortstruktur

$\langle\ \rangle_t$ Typenkennung vom Inhalt,
Beispiel: $\langle D\rangle_t = 0$
Die Typenkennung des Registers D ist Null.

$\langle\ \rangle_m$ Marke vom Inhalt,
Beispiel: $\langle M\rangle := \langle n\rangle_m$
Der Inhalt des Registers M ergibt sich aus der Marke vom Inhalt der Speicherzelle n. (Das 1. Bit vom Inhalt der Speicherzelle n ist das Markenbit, $\langle n\rangle_1 = \langle n\rangle_m$.)

$\langle\ \rangle_v$ Vorzeichen vom Inhalt,
Beispiel: $\langle n\rangle_2 = \langle n\rangle_v$
Das 2. Bit vom Inhalt der Speicherzelle n ist das Vorzeichenbit von n.

Logische Verknüpfungen:

$\wedge$ UND-Verknüpfung (Konjunktion)

a	:=	b	∧	c
0		0		0
0		0		L
0		L		0
L		L		L

$\vee$ ODER-Verknüpfung (Disjunktion)

a	:=	b	∨	c
0		0		0
L		0		L
L		L		0
L		L		L

$\nmid$ Antivalenz-Verknüpfung (exklusives ODER)

a	:=	b	∤	c
0		0		0
L		0		L
L		L		0
0		L		L

$\neg$ Negation

a	:=	¬	b
0			L
L			0

4. Zeitberechnung

Die Zeit, die von den Befehlen während ihres Ablaufs benötigt wird, setzt sich aus den Zeiten für die Abrufphase und der Ausführungsphase zusammen. Die Zeiten für die Abruf- und Ausführungsphase können parallel ablaufen. Auf dem Bild 4.2 ist ein Ablaufdiagramm der belegten Zeiten in der Abruf- und Ausführungsphase zu sehen.

Die Zeiten der Ausführungsphase sind jeweils auf der Rückseite der Befehlsformulare in Takten angegeben.

4.1. Die Abrufphase

Die Bereitstellung des nächsten Befehls erfolgt in der Abrufphase und belegt nur das Befehlswerk.

Die Abrufphase gliedert sich in nachstehend beschriebene Teile.

4.1.1. Der Befehlsabruf

Bei einem Befehlsabruf durch das Befehlswerk wird immer ein ganzes Befehlswort (2 Befehle) geholt. Der linke Befehl (mit gerader Adresse) kommt in das Befehlsregister während der rechte Befehl (mit ungerader Adresse) in das Reservebefehlsregister gebracht wird. Die Zeit, um ein ganzes Befehlswort zu holen, beträgt 10 Takte. Wird der zweite Befehl (mit ungerader Adresse) bereitgestellt, werden dazu nur noch 2 Takte benötigt.

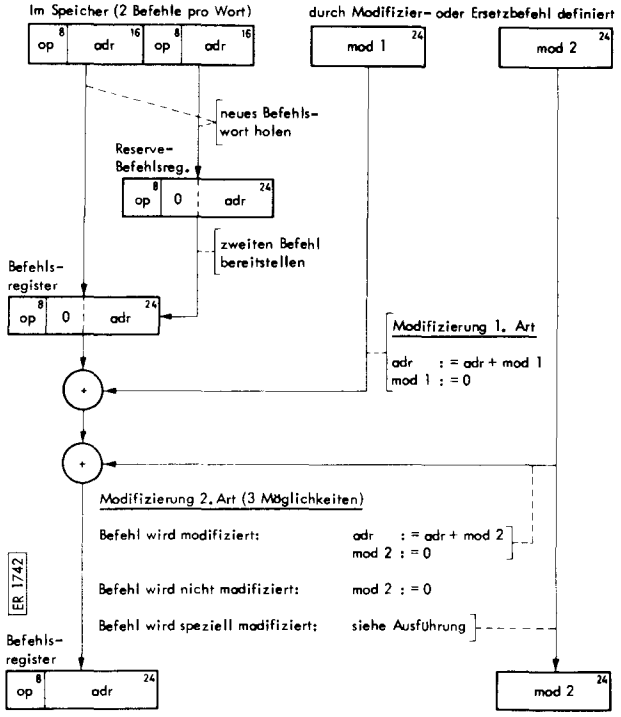


Bild 4.1 Holen eines Befehlswortes und Modifizieren

In diesem Ablaufdiagramm wird der vorstehend beschriebene Befehlsabruf gezeigt. Die hier aufgezeigten Möglichkeiten der Modifizierung werden im Abschnitt 4.1.3 beschrieben.

Die unter Abschnitt 2 "Einteilung der Formulare" und in der "Großen Befehlsliste" angegebenen Mittelwerte für eine einfache Zeitberechnung wurden wie folgt ermittelt:

Abruf eines neuen Befehlswortes (2 Befehle) = 10 Takte
Bereitstellen des zweiten Befehls (unger. Adr.) = 2 Takte

Mittelwert = 6 Takte

wie im Abschnitt 4.1.4. beschrieben werden 2 Takte hier zugerechnet

zus. = 8 Takte

4.1.2. Befehlsabruf nach Sprungbefehlen

Ist vor dem Befehlsabruf ein Befehl mit erfüllter Sprungbedingung abgehandelt worden, muß in jedem Fall ein neues Befehlswort geholt werden. Die Zeit, ein Befehlswort (2 Befehle) zu holen, beträgt 10 Takte. Auch hier wird der zweite Befehl in das Reservebefehlsregister gebracht. Wird dieser zweite Befehl (mit ungerader Adresse) bereitgestellt, werden zusätzlich 2 Takte benötigt.

Die Mittelwerte für eine einfache Zeitberechnung werden wie folgt ermittelt:

Steht der angesprungene Befehl in einem linken Halbwort (gerade Adresse), werden für den Befehlsabruf 10 Takte benötigt; steht der Befehl im rechten Halbwort (ungerade Adresse) werden, um den Befehl aus dem Reservebefehlsregister zu holen (zusätzlich 2 Takte) zusammen 12 Takte belegt.

Mittelwert = 11 Takte

Gegenüber des sequentiellen Abrufs hat der Mittelwert 5 Takte mehr.

Dieser Teil des Befehlsabrufs wird nach den Ersetzbefehlen (einschließlich des Befehls T) für die erzeugten Zweitbefehle nicht durchlaufen.

4.1.3. Modifizierung der Befehle

Liegt vom Vorbefehl ein Modifikator 1. oder 2. Art vor, so werden für die Ausführungsphase des gerade ablaufenden Befehls je nach Art der Modifizierung unterschiedliche Zeiten belegt. Eine Übersicht über die verschiedenen Arten der Modifizierung gibt das Bild 4.1

Modifizierung 1. Art

Liegt für den ablaufenden Befehl ein Modifikator 1. Art (mod1) vom Vorbefehl vor, so wird die Abrufphase um 8 Takte erhöht.

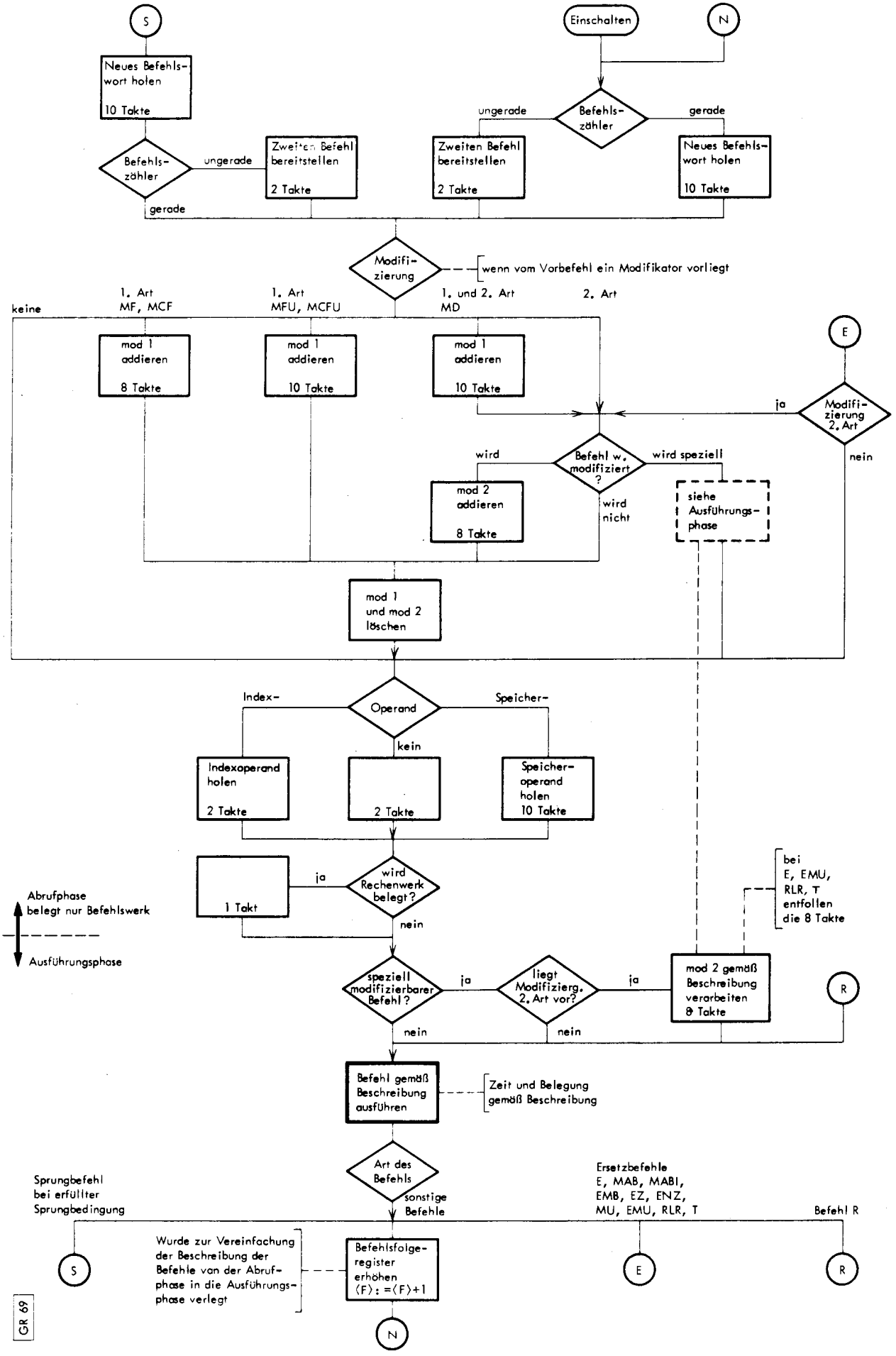


Bild 4.2 Abruf- und Ausführungsphase mit Rechenzeiten

Für die Modifizierung 1. Art wird das rechenfähige Register B benötigt. Da der Inhalt des Registers nicht verändert werden darf (bei den Befehlen MFU und MCFU), werden zur Wiederherstellung des ursprünglichen Zustandes (zusätzlich 2 Takte), zusammen 10 Takte benötigt.

Wird der ablaufende Befehl durch den Befehl MD doppelt modifiziert, dann steht im Register B der Modifikator 2. Art (mod2); es darf durch die Modifizierung 1. Art das Register B nicht verändert werden. Es gilt hier das im vorigen Abschnitt gesagte. Es werden 10 Takte belegt.

Modifizierung 2. Art

Steht für den gerade ablaufenden Befehl ein Modifikator 2. Art vom Vorbefehl bereit, hängt die benötigte Zeit davon ab, ob und wie dieser ablaufende Befehl modifiziert werden kann.
Es gibt für einen Modifikator 2. Art drei Möglichkeiten.

"w i r d m o d i f i z i e r t"

mod2 wird addiert, die Abrufphase des jeweiligen Befehls wird um 8 Takte erhöht.

"wird s p e z i e l l m o d i f i z i e r t"

Nicht zur Abrufphase, sondern zu der auf den Formularen angegebenen Zeit der Ausführungsphase des jeweiligen Befehls, der speziell modifiziert wird, werden 8 Takte addiert.

Die Befehle E, EMU, RLR und T verarbeiten mod2 nicht selbst, sie geben den mod2 an die (von ihnen erzeugten) Zweitbefehle weiter und benötigen daher keine zusätzliche Zeit. Die hier nicht benötigten 8 Takte werden nach den in diesem Abschnitt aufgestellten Regeln ggf. bei den Zweitbefehlen belegt.

Bei dem Mittelwert für die einfache Zeitberechnung werden für die Modifizierung stets 8 Takte angenommen.

"w i r d n i c h t m o d i f i z i e r t"

Der Modifikator 2. Art bleibt unberücksichtigt, es wird keine zusätzliche Zeit belegt.

4.1.4. Operanden werden geholt

Unabhängig davon, ob der laufende Befehl einen Indexoperanden oder keinen Operanden benötigt, werden jeweils 2 Takte belegt.

Dies gilt nur für einen Operanden (den ersten). Für Befehle die weitere Operanden benötigen (z.B. ZXX, TLI) ist die Zeit für das Holen dieser Operanden in der Ausführungszeit enthalten.

Für die einfache Zeitberechnung sind diese 2 Takte zu den Mittelwerten für den Befehlsabruf Abschnitt 4.1.1. zugerechnet.

Wird der Operand aus dem Speicher geholt, werden zusätzlich noch 8 Takte in der Abrufphase belegt.

Befehle, die in der Abrufphase einen Operanden aus dem Speicher holen, sind im Abschnitt 6 auf Blatt 1 gekennzeichnet.

Belegt der Befehl das Rechenwerk, wird zusätzlich 1 Takt benötigt.

Für die einfache Zeitberechnung sind auch die 8 Takte für den aus dem Speicher geholten Operanden als auch der eine Takt für die Belegung des Rechenwerks angegeben.

4.2. Die Ausführungsphase

Auf den Rückseiten der Befehlsformulare sind die Zeiten der Ausführungsphase angegeben. Die Ausführungsphase kann entweder das Befehlswerk, das Rechenwerk oder beide Werke belegen. Sind auf den Formularen unter "belegt" beide Werke aufgeführt und ist nur eine Zeit genannt, so gilt die Zeit für beide Werke. (Für das Rechenwerk kann in vielen Fällen eine kürzere Zeit benötigt werden, die aber dann ohne Vorteil ist.) Ist unter "belegt" nur das Rechenwerk aufgeführt, können während der angegebenen Zeit parallel dazu die Abrufphase und alle Befehle ablaufen, die nur das Befehlswerk belegen.

Zu den auf den Formularen angegebenen Zeiten der Ausführungsphase werden, wenn vom vorhergehenden Befehl ein Modifikator 2. Art (mod2) ansteht, für speziell modifizierbare Befehle noch zusätzlich 8 Takte benötigt (siehe 4.1.3 "wird speziell modifiziert"). Diese 8 Takte werden bei den Befehlen **E, EMU, RLR** und T nicht belegt.

4.3. Übergang Ausführungsphase - Abrufphase

Nachdem die Ausführungsphase beendet ist, folgt die nächste Abrufphase. Abhängig vom jeweiligen Befehl wird in die verschiedenen Stellen der Abrufphase gesprungen **(Bild 4.2).**

Nach Sprungbefehlen mit erfüllter Sprungbedingung, wird ein neues Befehlswort geholt (Eingang S).

Bei **Ersetz**befehlen (einschließlich des Befehls T) kommt als nächster der Zweitbefehl zur Ausführung; dazu braucht der Befehlsabruf nicht mehr durchlaufen zu werden und mod1 kommt nicht zur Wirkung (Eingang E).

Der durch R erzeugte Zweitbefehl benötigt die Abrufphase nicht. Die Modifizierung und das Holen eines Operanden entfällt für den Zweitbefehl. Der Operand wird aus dem Register genommen (Eingang R).

Bei allen in diesem Abschnitt nicht aufgeführten Befehlen und bedingten Sprungbefehlen, deren Sprungbedingungen nicht erfüllt sind, erfolgt die Erhöhung des Befehlsfolgezählers am Ende der Ausführungsphase (Eingang N).

Nur zur Vereinfachung der Beschreibung wurde die Erhöhung des Befehlsfolgezählers an das Ende der Ausführungsphase verlegt. Tatsächlich wird die Erhöhung des Befehlsfolgezählers in der Abrufphase unmittelbar hinter dem Befehlsabruf vorgenommen. Das trifft auch bei den Sprungbefehlen zu, nur ist es bei erfüllter Sprungbedingung ohne Bedeutung, da in der Ausführungsphase der Befehlsfolgezähler neu gesetzt wird.

A	n

Addiere

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle A \rangle := t_{max} ; \langle A \rangle + \langle D \rangle$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und zum Inhalt des Registers A addiert. Die Addition erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis im Register A erhält die höhere der beiden Typenkennungen.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: A

Interncode: '42'

belegt: Rechenwerk

Takte: 8

Alarm:

BÜ-Alarm: wenn beim Ergebnis:

$\langle A \rangle_t = 0$ oder 1 und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:

$\langle A \rangle_t$ oder $\langle n \rangle_t \neq 1$

Addition erfolgt wie bei TK = 1

Ergebnis: siehe Ausführung

bei über- oder untergelaufenem Ergebnis:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

Ergebnis: siehe Ausführung

BÜ-Alarm

bei über- oder untergelaufenen Operanden:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

Wird mit über- oder untergelaufenen Operanden gerechnet, so kann das Ergebnis falsch sein, ohne daß ein BÜ-Alarm erfolgt.

A2	m

Addiere Halbwort

A2

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$|\langle A\rangle| < 2^{22}$
$|\langle m\rangle| < 2^{22}$

Ausführung:

$\langle A\rangle_{25-48} := \langle A\rangle_{25-48} + \langle m\rangle$
$\langle A\rangle_{1-24} := \langle A\rangle_{25}$
$\langle A\rangle_{t} := \langle A\rangle_{t}$
$\langle D\rangle := t_A ;+0$

Der Inhalt der Speicherzelle m wird zu den rechten 24 Bits des Registers A addiert. Die Addition erfolgt unabhängig von der Typenkennung wie bei Festkommazahlen.

Das Ergebnis steht im Register A; die linken 24 Bits werden dem 25. Bit angeglichen.

Das Register D wird auf Null gelöscht und erhält die Typenkennung des Registers A.

Externcode: A2

Interncode: '7C'

belegt: Rechenwerk

Takte: 14

Alarm:

BÜ-Alarm: wenn beim Ergebnis $\langle A \rangle_{25} \neq \langle A \rangle_{26}$

Sonstiges:

Bei über- oder untergelaufenem Ergebnis:

$\langle A \rangle_{25} \neq \langle A \rangle_{26}$

Ergebnis: siehe unter Ausführung

BÜ-Alarm

Voraussetzungen nicht erfüllt:

$|\langle A \rangle| \geq 2^{22}$

$|\langle m \rangle| \geq 2^{22}$

Ergebnis: siehe unter Ausführung

Das Ergebnis kann um $\pm 2^{24} - 1$ falsch sein, ohne daß ein BÜ-Alarm erfolgt.

Der Einerrücklauf erfolgt von der 25. zur 48. Binärstelle.

AA	z

Addiere Adressenteil

AA

Adressenteil: z = Zahl bei TK = 0: 0...±127 *
TK = 1: 0... 65 535
TK = 2: 0... 65 535
TK = 3: 0... 65 535

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A\rangle_t = 0: |z| < 2^7$
$\langle A\rangle_t = 1: |z| < 2^{23}$
$\langle A\rangle_t = 2: z < 2^{16}$
$\langle A\rangle_t = 3: |z| < 2^{23}$

} nach der Modifizierung

Ausführung:

Die Ausführung hängt von $\langle A\rangle_t$ ab

bei $\langle A\rangle_t = 0$: $\langle D\rangle := 1;0,z$
$\langle A\rangle := 0;\langle A\rangle \cdot 16^{+z}$

bei $\langle A\rangle_t = 1$: $\langle D\rangle := 1;v,z$
$\langle A\rangle := 1;\langle A\rangle + v,z$

bei $\langle A\rangle_t = 2$: $\langle D\rangle := 1;0,z$
$\langle A\rangle := 2;\langle A\rangle_{1-32},(\langle A\rangle_{33-48} + z)$

bei $\langle A\rangle_t = 3$: $\langle D\rangle := 1;v,z$
$\langle A\rangle := 3;\langle A\rangle + v,z$

Die Zahl z wird in das Register D gebracht und gleichzeitig zum Inhalt des Registers A addiert.

Bei Gleitkommazahlen (Typenkennung = 0) wird die Zahl z zum Exponenten addiert, dies bedeutet Multiplikation mit dem Faktor 16^{+z}.

*Bei der Assemblierung, wird bei negativen Werten, eine Fehlermeldung "negativer Adressenteil" gegeben. Dies kann ignoriert werden, wenn das Register A die Typenkennung 0 hat.

Externcode: AA

Interncode: '98'

belegt: Rechenwerk

Takte: 13

Alarm:

BÜ-ALarm: wenn bei $\langle A \rangle_t = 0$ beim Ergebnis der Exponent übergelaufen ist
wenn bei $\langle A \rangle_t = 1$ das Ergebnis über- oder untergelaufen ist

---

Sonstiges:

Bei über- oder untergelaufenem Ergebnis:

Ergebnis: siehe Ausführung
BÜ-Alarm
bei $\langle A \rangle_t = 0$: zusätzlich $\langle A \rangle := \langle A \rangle \cdot 16^{-127}$

Einerrücklauf: bei $\langle A \rangle_t = 0$ von $\langle A \rangle_{41}$ nach $\langle A \rangle_{48}$
bei $\langle A \rangle_t = 1$ von $\langle A \rangle_{1}$ nach $\langle A \rangle_{48}$
bei $\langle A \rangle_t = 2$ von $\langle A \rangle_{32}$ nach $\langle A \rangle_{48}$
bei $\langle A \rangle_t = 3$ von $\langle A \rangle_{1}$ nach $\langle A \rangle_{48}$

Voraussetzung nicht erfüllt: das Ergebnis wird falsch

Das Ergebnis wird deshalb falsch, weil nach der Modifizierung von z nur folgende Bits verwendet werden:

bei $\langle A \rangle_t = 0$: rechte 8 Bits; linke 40 Bits = 0
bei $\langle A \rangle_t = 1$: rechte 24 Bits; linke 24 Bits = vorzeichengleich *
bei $\langle A \rangle_t = 2$: rechte 16 Bits; linke 32 Bits = 0
bei $\langle A \rangle_t = 3$: rechte 24 Bits; linke 24 Bits = vorzeichengleich *

* Vorzeichen ist beim auf 24 Bits verlängerten Adressenteil das linke Bit.

AB	n

Addiere Betrag

AB

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_n \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle A \rangle := t_{max} ; \langle A \rangle + |\langle D \rangle|$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Der Betrag vom Inhalt des Registers D wird zum Inhalt des Registers A addiert. Die Addition erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis erhält die höhere der beiden Typenkennungen.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: AB

Interncode: '40'

belegt: Rechenwerk

Takte: 8

Alarm:

BÜ-Alarm: wenn beim Ergebnis:
$\langle A \rangle_t = 0$ oder 1 und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:
$\langle A \rangle_t$ oder $\langle n \rangle_t \neq 1$ bei $\langle n \rangle_t = 2$ oder 3

Addition erfolgt wie bei TK = 1 $|\langle D \rangle| = \langle D \rangle$

Ergebnis: siehe Ausführung

bei über- oder untergelaufenem Ergebnis:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

Ergebnis: siehe Ausführung
BÜ-Alarm

bei über- oder untergelaufenen Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

Wird mit über- oder untergelaufenen Operanden gerechnet, so kann das Ergebnis falsch sein, ohne daß BÜ-Alarm erfolgt.

AC	n

Addiere im Speicher

AC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_{\blacksquare} \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle D \rangle := t_{\blacksquare ax} ; \langle A \rangle + \langle D \rangle$

$\langle n \rangle := t_b ; \langle D \rangle$

$\langle n \rangle_{\blacksquare} := \langle n \rangle_{\blacksquare}$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und dort zum Inhalt des Registers A addiert. Die Addition erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis steht mit der höheren der beiden Typenkennungen im Register D und wird ebenso in die Speicherzelle n abgespeichert. Die ursprüngliche Marke in der Speicherzelle bleibt erhalten.

Der Inhalt des Registers A bleibt unverändert.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: AC

Interncode: '43'

belegt: Befehlswerk und Rechenwerk

Takte: 17

Alarm:

BÜ-Alarm: wenn beim Ergebnis:

$\langle D \rangle_t = 0$ oder 1 und $\langle D \rangle_1 \neq \langle D \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:

$\langle A \rangle_t$ oder $\langle n \rangle_t \neq 1$

Addition erfolgt wie bei TK = 1

Ergebnis: siehe Ausführung

bei über- oder untergelaufenem Ergebnis:

$\langle D \rangle_1 \neq \langle D \rangle_2$ und $\langle D \rangle_t = 0$ oder 1

$\langle D \rangle := t_{max}; \langle n \rangle + \langle A \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_n \end{array}\right\}$ nur bei $t_n = 0$ oder 1

BÜ-Alarm

Das im Register D stehende Ergebnis wird nicht in die Speicherzelle n abgespeichert.

Die Marke in n bleibt jedoch erhalten.

bei über- oder untergelaufenen Operanden:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

Wird mit über- oder untergelaufenen Operanden gerechnet, so kann das Ergebnis falsch sein, ohne daß BÜ-Alarm erfolgt.

AQ	n

Addiere in A,Q

AQ

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle Q\rangle_t = \langle n\rangle_t = 1$

$\left.\begin{array}{l}\langle A\rangle_1 = \langle A\rangle_2 \\ \langle Q\rangle_1 = \langle Q\rangle_2\end{array}\right\}\ \langle A\rangle_v = \text{oder} \neq \langle Q\rangle_v$

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_s$

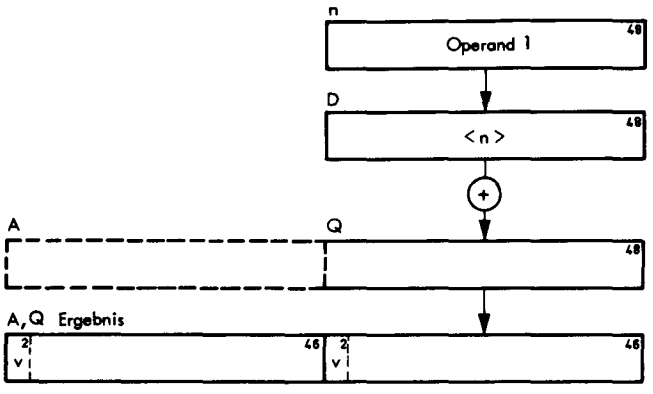
$\langle A,Q\rangle := 1,1 ; \langle A,Q\rangle + \langle n\rangle$

$\langle A\rangle_v$ kann $\neq \langle Q\rangle_v$

Die Typenkennung der Register A und Q sowie der Speicherzelle n muß = 1 sein; die Register dürfen nicht über- oder untergelaufen sein, sie können aber verschiedenen Vorzeichen haben.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und zum Inhalt des doppeltlangen Registers A,Q addiert, dabei wird zum Inhalt des Registers Q addiert und ein Überlauf erfolgt in das Register A. Das Ergebnis kann in jedem der beiden Register verschiedene Vorzeichen haben, da die Register getrennt verarbeitet werden.

Die Marke wird berücksichtigt.



Externcode: AQ

Interncode: '7E'

belegt: Rechenwerk

Takte: 17

Alarm:

TK-Alarm: wenn $\langle A\rangle_t$ oder $\langle Q\rangle_t$ oder $\langle n\rangle_t \neq 1$

BÜ-Alarm: wenn beim Ergebnis $\langle A\rangle_1 \neq \langle A\rangle_2$

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t$ oder $\langle Q\rangle_t$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_n\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

6 Takte

bei über- oder untergelaufenen Operanden:

$\langle A\rangle_1 \neq \langle A\rangle_2$ oder $\langle Q\rangle_1 \neq \langle Q\rangle_2$

Die Befehlsausführung wird falsch.

bei über- oder untergelaufenem Ergebnis:

$\langle A\rangle_1 \neq \langle A\rangle_2$

Ergebnis: siehe Ausführung

BÜ-Alarm

17 Takte

AT	n

Addiere Teilwort

AT

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle Q \rangle$ = Maske

$\langle A \rangle + nr < 2^{48}$

Ausführung:

$\langle D \rangle := t_Q ; \langle Q \rangle$

Hilfsgröße qr := $\langle Q \rangle$ um p Stellen nach rechts im Kreis geschiftet

Hilfsgröße nr := $\langle n \rangle$ um p Stellen nach rechts geschiftet

p: Anzahl der L-Bits, die rechtsbündig im Register Q stehen
Falls $\langle Q \rangle$ = LL...L, dann p = 0

$nr_x := 0$ falls $qr_x = L$

$\langle A \rangle_x := L$ falls $qr_x = L$

$\langle A \rangle := t_A ; \langle A \rangle_x + nr_x$

$\langle A \rangle_x = 0$ falls $qr_x = L$

x: 1,2,...48
$\langle A \rangle$ und nr werden als zusammenhängende, positive und vorzeichenlose Festkommazahlen aufgefaßt.

Im Register Q steht eine Maske. Es wird aus dem Inhalt des Registers Q eine Hilfsgröße gebildet. Diese Hilfsgröße wird qr genannt. Sie ergibt sich aus dem Inhalt des Registers Q um p Stellen nach rechts im Kreis geschiftet, wobei p die Anzahl der im Register Q rechtsbündig stehenden L-Bits angibt. Falls der Inhalt des Registers Q nur aus L-Bits besteht, wird p zu Null. Auch aus dem Inhalt der Speicherzelle n wird eine Hilfsgröße nr gebildet, die sich aus dem um p Stellen nach rechts geschifteten Inhalt ergibt.

Durch das Nullfeld der Maske in der Hilfsgröße qr werden aus dem Inhalt des Registers A und aus der Hilfsgröße nr Teilwörter ausgeschnitten.

Diese Teile werden addiert. Gleichgültig welche Bits aus den Ganzwörtern ausgeschnitten werden, erfolgt die Addition der Teilwörter immer so, als wären sie zusammenhängende, positive und vorzeichenlose Festkommazahlen. Das Ergebnis der Addition steht mit der ursprünglichen Typenkennung im Register A. Die Binärstellen, die dem L-Feld der Hilfsgröße qr entsprechen, werden zu Null.

Externcode: AT

Interncode: 'F4'

belegt: Rechenwerk

Takte: 21 + 2p

Alarm:

BÜ-Alarm: wenn Ergebnis übergelaufen $\geq 2^{48}$

Sonstiges:

Bei übergelaufenem Ergebnis:

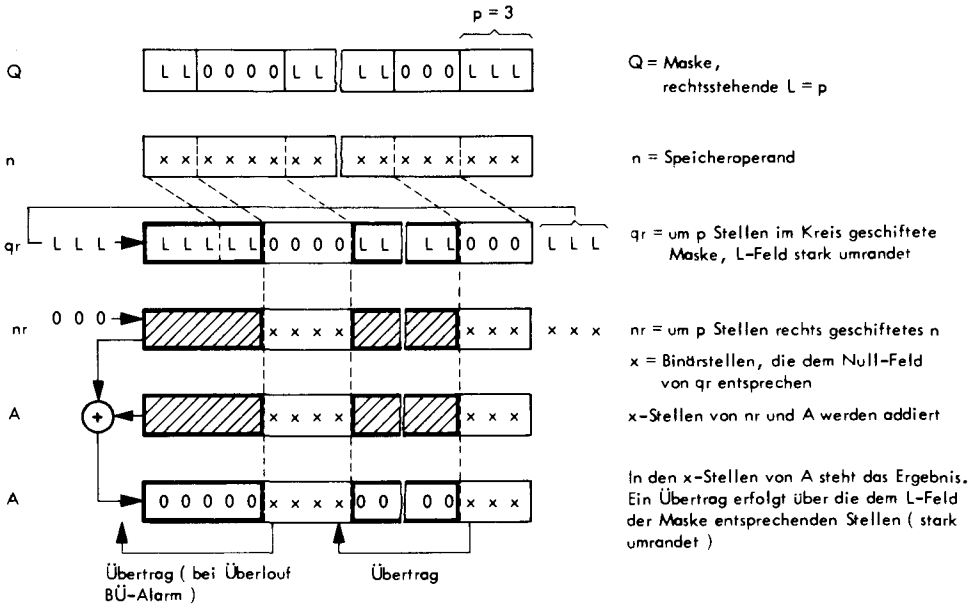
$\langle A \rangle + nr \geq 2^{48}$

Ergebnis: siehe Ausführung

$\langle A \rangle := t_A ; \langle A \rangle + nr - 2^{48}$

BÜ-Alarm

Ist beim Ergebnis ein L in eine links neben der 1. Bitstelle gedachte Stelle übergelaufen, so geht dieses L verloren, da kein Einerrücklauf erfolgt. Es erfolgt BÜ-Alarm. Das Ergebnis ist dann um -2^{48} falsch.



ATA	z

ATA-Adressenteil ("Antivalenz"-Verknüpfung)

ATA

Adressenteil: z: Zahl = 0...65 535

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨D⟩ := 1;0,z
⟨A⟩ := t_H ;⟨H⟩ ≢ ⟨D⟩

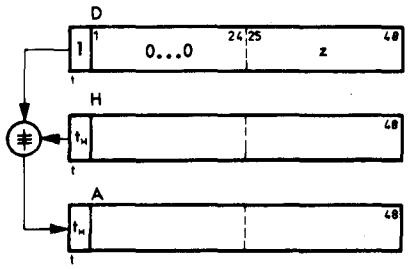
In die rechte Hälfte des Registers D (Binärstellen 25 - 48) wird der Wert z gebracht; die linke Hälfte des Registers (Binärstellen 1 - 24) ergibt sich zu 0, die Typenkennung zu 1.

Der Inhalt des Registers D wird mit dem Inhalt des Registers H durch die Boolesche Operation "Antivalenz" (exklusives ODER) verknüpft.

Das Ergebnis steht im Register A mit der Typenkennung aus dem Register H.

"Antivalenz"-Verknüpfung

a := b ≢ c		
0	0	0
L	0	L
L	L	0
0	L	L



Externcode: ATA

Interncode: '89'

belegt: Rechenwerk

Takte: 8

Alarm:

Sonstiges:

AU	n

Addiere unnormalisiert

AU

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A \rangle := 0 ; \langle A \rangle + \langle D \rangle$ nicht normalisiert und gerundet

$\langle Q \rangle := 0 ; +0$

$\langle Y \rangle := +0$

Beide Operanden müssen die Typenkennung = 0 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und zum Inhalt des Registers A addiert. Das Ergebnis steht mit Typenkennung = 0 im Register A und ist weder normalisiert noch gerundet.

Das Register Q wird mit Typenkennung = 0 auf Null gelöscht. Der Inhalt des Registers Y ergibt sich zu Null.

Die Marke wird berücksichtigt.

Externcode: AU

Interncode: '49'

belegt: Rechenwerk

Takte: 26

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)
wenn Exponent des Ergebnisses > +127

Sonstiges: Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm

2 Takte

bei übergelaufenem Operanden:

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0; undefiniert
$\langle Q\rangle$:= 0;+0
$\langle Y\rangle$:= undefiniert
evtl. BÜ-Alarm
26 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe Ausführung
$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$
BÜ-Alarm
26 Takte

AUT	n

AUT ("Antivalenz"-Verknüpfung - exklusives ODER) **AUT**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_s\end{array}\right\}$ bei t_n = 0 oder 1

$\langle A\rangle := t_{s\,a\,x} ; \langle A\rangle \neq \langle D\rangle$

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht.

Der Inhalt des Registers A wird mit dem Inhalt des Registers D durch die Boolesche Operation "Antivalenz" verknüpft.

Das Ergebnis steht im Register A und erhält die größere der beiden Typenkennungen der Register A oder D.

Bei Zahlwörtern wird die Marke berücksichtigt.

"Antivalenz"-Verknüpfung

a :=	b ≠	c
0	0	0
L	0	L
L	L	0
0	L	L

Externcode: AUT

Interncode: '69'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

B	n

Bringe (nach A)

B

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle A \rangle_1 := \langle A \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_{\blacksquare} \end{array}\right\}$ nur bei $t_n = 0$ oder 1

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register A gebracht.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: B

Interncode: '70'

belegt: Rechenwerk

Takte: 2

Alarm:

Sonstiges:

B2	m

Bringe Halbwort

B2

Adressenteil: m = Adresse eines Halbwortes

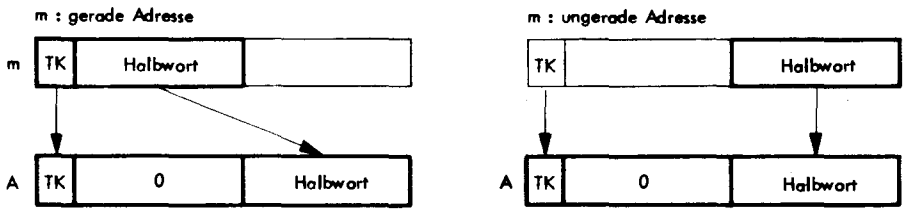
bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := t_m ; 0, \langle m \rangle$

Der Inhalt der Speicherzelle m wird rechtsbündig in das Register A gebracht, der linke Teil des Registers wird auf Null gelöscht. Der Transport erfolgt einschließlich Typenkennung



Externcode: B2

Interncode: '6E'

belegt: Rechenwerk

Takte: 9

Alarm:

Sonstiges:

B2V	m

Bringe Halbwort mit Vorzeichen

B2V

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

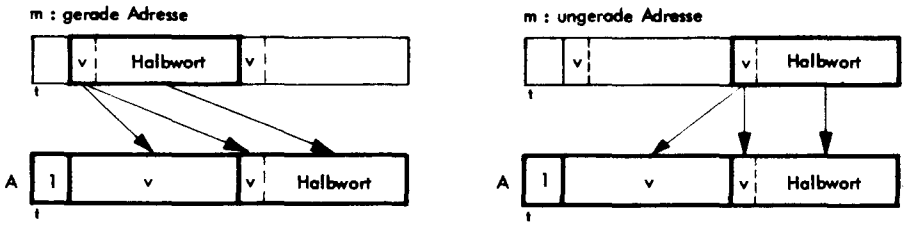
Voraussetzung:

$\langle m \rangle_1 = \langle m \rangle_v$

Ausführung:

$\langle A \rangle := 1;v,\langle m \rangle$

Der Inhalt der Speicherzelle m wird rechtsbündig in das Register A gebracht. Das 1. Bit der Speicherzelle m wird als Vorzeichen aufgefaßt. Die linke Hälfte des Registers A wird diesem Vorzeichen angeglichen. Das Register A erhält die Typenkennung = 1.



Externcode: B2V

Interncode: '6F'

belegt: Rechenwerk

Takte: 10

Alarm:

Sonstiges:

B2VN	m

Bringe Halbwort mit Vorzeichen negativ

B2VN

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

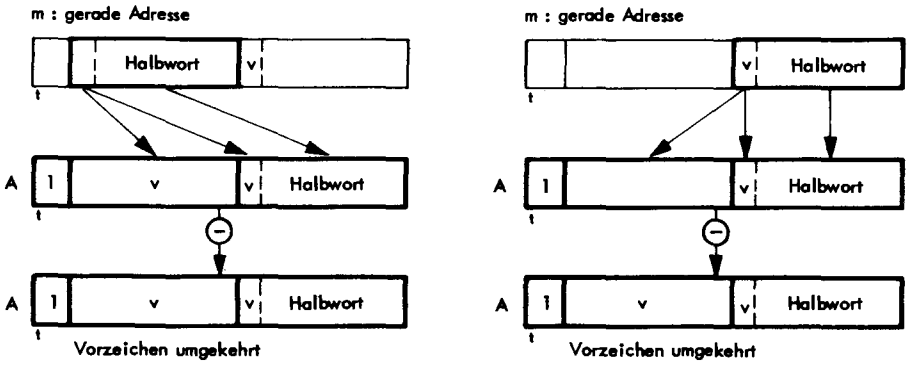
Voraussetzung:

$\langle m \rangle_1 = \langle m \rangle_v$

Ausführung:

$\langle A \rangle := 1;-(v,\langle m \rangle)$

Der Inhalt der Speicherzelle m wird mit umgekehrtem Vorzeichen in die rechte Hälfte des Registers A gebracht. Das 1. Bit der Speicherzelle m wird als Vorzeichen aufgefaßt. Die linke Hälfte des Registers A erhält die Typenkennung = 1.



Externcode: B2VN

Interncode: '67'

belegt: Rechenwerk

Takte: 12

Alarm:

Sonstiges:

B3	m	Bringe Drittelwort	**B3**

Adressenteil: m = Adresse eines Halbwortes

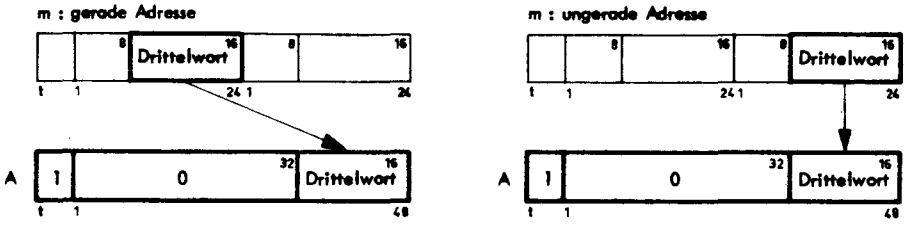
bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := 1;0,\langle m \rangle_{9-24}$

Der Inhalt des Drittelwortes der Speicherzelle m wird rechtsbündig in das Register A gebracht, der linke Teil des Registers wird auf Null gelöscht. Das Register A erhält die Typenkennung = 1.



Externcode: B3

Interncode: '6C'

belegt: Rechenwerk

Takte: 9

Alarm:

Sonstiges:

B3V	m

Bringe Drittelwort mit Vorzeichen

B3V

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

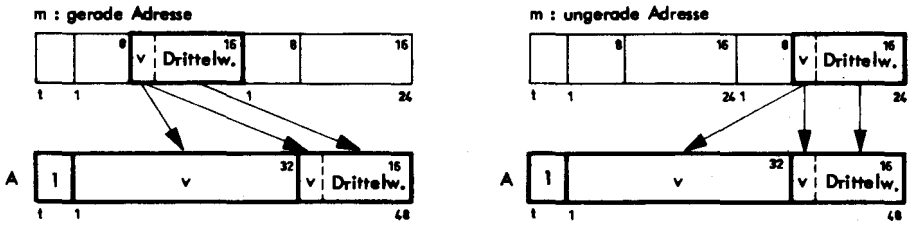
Voraussetzung:

$\langle m \rangle_9 = \langle m \rangle_v$

Ausführung:

$\langle A \rangle := 1;v,\langle m \rangle_{9-24}$

Der Inhalt des Drittelwortes der Speicherzelle m wird rechtsbündig in das Register A gebracht. Das 9. Bit der Speicherzelle m wird als Vorzeichen aufgefaßt. Der linke Teil des Registers A wird diesem Vorzeichen angeglichen. Das Register erhält die Typenkennung = 1.



Externcode: B3V

Interncode: '6D'

belegt: Rechenwerk

Takte: 10

Alarm:

Sonstiges:

BA	z

Bringe Adressenteil

BA

Adressenteil: z = Zahl 0...65 535 (vor der Modifizierung)

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

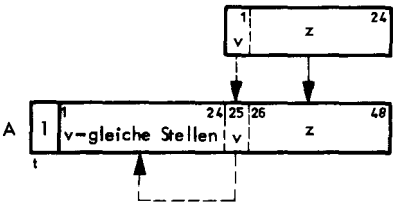
Voraussetzung:

$|z| < 2^{23}$ (nach Modifizierung)

Ausführung:

⟨A⟩ := 1;v,z

Die Zahl z wird in die rechte Hälfte (Binärstellen 25 - 48) des Registers A gebracht. Das linke Bit des auf 24 Bits verlängerten Adressenteils gilt als Vorzeichen. Die linke Hälfte des Registers A wird diesem Vorzeichen angeglichen. Das Register A erhält die Typenkennung = 1.



Externcode: BA

Internode: '8E'

belegt: Rechenwerk

Takte: 2

Alarm:

Sonstiges:

BAN	z

Bringe Adressenteil negativ

BAN

Adressenteil: z = Zahl 0...65 535 (vor der Modifizierung)

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

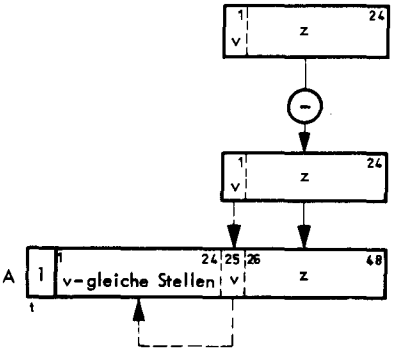
Voraussetzung:

$|z| < 2^{23}$ (nach Modifizierung)

Ausführung:

⟨A⟩ := 1;-(v,z)

Die Zahl z wird mit umgekehrten Vorzeichen in die rechte Hälfte (Binärstellen 25 - 48) des Registers A gebracht. Das linke Bit des auf 24 Bits verlängerten Adressenteils gilt als Vorzeichen. Die linke Hälfte des Registers A wird diesem Vorzeichen angeglichen. Das Register A erhält die Typenkennung = 1.



Externcode: BAN

Interncode: 'DF'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

BANR	z

Bringe Adressenteil negativ und reserviere **BANR**

Adressenteil: z = Zahl 0...65 535 (vor der Modifizierung)

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

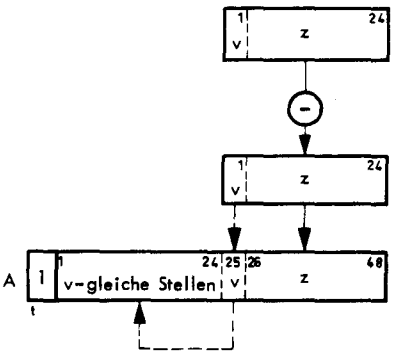
$|z| < 2^{23}$ (nach Modifizierung)

Ausführung:

$\langle H \rangle := t_A ; \langle A \rangle$
$\langle A \rangle := 1 ; -(v,z)$

Der Inhalt des Registers A wird einschließlich Typenkennung in das Register H gebracht und dort reserviert.

Die Zahl z wird mit umgekehrten Vorzeichen in die rechte Hälfte (Binärstellen 25 - 48) des Registers A gebracht. Das linke Bit des auf 24 Bits verlängerten Adressenteils gilt als Vorzeichen. Die linke Hälfte des Registers A wird diesem Vorzeichen angeglichen. Das Register A erhält die Typenkennung = 1.



Externcode: BANR

Interncode: 'DD'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

BAR	z

Bringe Adressenteil und reserviere

BAR

Adressenteil: z = Zahl 0...65 535 (vor der Modifizierung)

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$|z| < 2^{23}$ (nach Modifizierung)

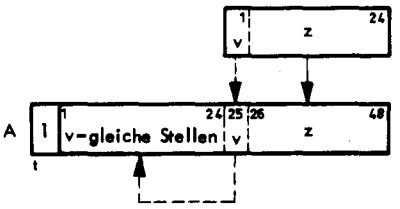
Ausführung:

$\langle H \rangle := t_A ; \langle A \rangle$

$\langle A \rangle := 1; v, z$

Der Inhalt des Registers A wird einschließlich Typenkennung in das Register H gebracht und dort reserviert.

Die Zahl z wird in die rechte Hälfte (Binärstellen 25 - 48) des Registers A gebracht. Das linke Bit des auf 24 Bits verlängerten Adressenteils gilt als Vorzeichen. Die linke Hälfte des Registers A wird diesem Vorzeichen angeglichen. Das Register A erhält die Typenkennung = 1.



Externcode: BAR

Interncode: 'DC'

belegt: Rechenwerk

Takte: 3

Alarm:

Sonstiges:

BB	n

Bringe Betrag

BB

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

bei t_n = 0 oder 1:

$\langle A \rangle := t_n ; |\langle n \rangle|$

$\langle A \rangle_1 = \langle A \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle m$

bei t_n = 2 oder 3:

$\langle A \rangle := t_n ; \langle n \rangle$

Bei Zahlwörtern wird der Betrag vom Inhalt der Speicherzelle n einschließlich Typenkennung in das Register A gebracht.

Bei Nichtzahlwörtern ist die Betragsbildung ohne Bedeutung; die Wirkung ist die gleiche wie bei dem Befehl B.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BB

Interncode: '74'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

BC	n

Bringe und speichere

BC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle A \rangle_t$ = 0 oder 1: $\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$t_A ; \langle A \rangle :=: t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle A \rangle_1 := \langle A \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei t_n = 0 oder 1

$\langle n \rangle_m := \langle n \rangle_m$ wenn t_A und t_n = 0 oder 1

Der Inhalt der Speicherzelle n wird mit dem Inhalt des Registers A vertauscht. Der Tausch erfolgt einschließlich der jeweiligen Typenkennung.

War der Inhalt des Registers A ein Zahlwort, so darf es nicht über- oder untergelaufen sein. Im anderen Fall erfolgt BÜ-Alarm und der Inhalt des Registers A geht verloren.

Bei Zahlwörtern wird die Marke berücksichtigt.

War der Inhalt des Registers A und der Inhalt der Speicherzelle n ein Zahlwort, so bleibt die ursprüngliche Markenstelle in der Speicherzelle n nach dem Tausch erhalten.

Externcode: BC

Interncode: 'A3'

belegt: Befehlswerk und Rechenwerk

Takte: 13

Alarm:

BÜ-Alarm: wenn $\langle A \rangle_t = 0$ oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei über- oder untergelaufenen Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

$\langle A \rangle := t_n ; \langle n \rangle$

$\langle A \rangle_1 := \langle A \rangle_2$
$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_n$

BÜ-Alarm

Das im Register A enthaltene über- oder untergelaufene Zahlwort geht verloren.

BCI	n

Bringe und speichere Indexbasis

BCI

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle n \rangle_{3-24}$ = Anfangsadresse des neuen Indexbereiches
$\langle n \rangle_{41-48}$ = gewünschter Wert für Unterprogrammregister

Ausführung:

$\langle X \rangle$:=: $\langle n \rangle_{3-24}$
$\langle U \rangle$:=: $\langle n \rangle_{41-48}$
$\langle n \rangle_{t}$:= 3
$\langle n \rangle_{1-2}$:= 0
$\langle n \rangle_{25-40}$:= 0

Mit dem Befehl kann das Indexbasisregister und das Unterprogrammregister neu besetzt werden und die alten Inhalte der Register abgespeichert werden.

Dazu wird der Inhalt der Indexbasisregister mit der linken Hälfte des Inhalts der Speicherzelle n ausgetauscht und der Inhalt des Unterprogrammregisters mit den rechten 8 Bits des Inhaltes der Speicherzelle n. Die restlichen Stellen werden auf Null gesetzt.

Mit diesem Befehl kann von einem Indexspeicherbereich auf einen anderen umgeschaltet werden unter Sicherstellung der Anfangsadresse des alten Indexbereiches und des Unterprogrammregisters.

Externcode: BCI

Interncode: 'B6'

belegt: Befehlswerk

Takte: 17x + 66 x = Anzahl der Register, welche zurückgespeichert werden

---

Alarm:

Sonstiges:

BCL	m

Bringe und speichere Merklichter

BCL

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle m \rangle_{17-24} :=: \langle K \rangle$

$\langle m \rangle_{1-16} := \langle m \rangle_{1-16}$

Der Merklichterstand im Register K wird ausgetauscht mit den rechten 8 Bits des Halbwortes in der Speicherzelle m; der linke Teil der Speicherzelle bleibt erhalten.

Externcode: BCL

Interncode: '06'

belegt: Befehlswerk

Takte: 18

Alarm:

Sonstiges:

BD	n	Bringe nach D	**BD**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$$\left.\begin{array}{l}\langle D\rangle := t_n ; \langle n\rangle \\ \langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\} \text{ nur bei } t_n = 0 \text{ oder } 1$$

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BD

Interncode: '71'

belegt: Rechenwerk

Takte: 3

Alarm:

Sonstiges:

BH	n

Bringe nach H

BH

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$$\left.\begin{array}{l} \langle H\rangle := t_n\ ;\langle n\rangle \\ \langle H\rangle_1 := \langle H\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m \end{array}\right\} \text{ nur bei } t_n = 0 \text{ oder } 1$$

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register H gebracht.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BH

Interncode: '73'

belegt: Rechenwerk

Takte: 3

Alarm:

Sonstiges:

BL	n	Bringe und lösche	**BL**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle_{1-48} := t_n ; \langle n \rangle_{1-48}$

$\langle n \rangle_{1-48} := 0 ; 0$

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung unverändert in das Register A gebracht. Das Markenregister bleibt unverändert.

Die Speicherzelle n wird mit Typenkennung 0 auf Null gelöscht.

Externcode: BL

Interncode: 'BO'

belegt: Befehlswerk und Rechenwerk

Takte: 17

Alarm:

Sonstiges:

BLEI	p	Bringe aus Leitblock	**BLEI**

Adressenteil: p : Adresse relativ zum Leitblockanfang
+0 ... 255

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

BL = Leitadressenregister
$\langle BL \rangle \cdot 2^8$ = Anfangsadresse

Ausführung:

$\langle A \rangle$:= t; $\langle \langle BL \rangle \cdot 2^8 + p \rangle$

Der Inhalt des p-ten Wortes des aktuellen Leitblocks wird unverändert einschließlich Typenkennung ins Register A gebracht.

Das Register BL enthält die linken 16 Bits der Anfangsadresse des aktuellen Leitblocks.

Aufbau des Leitblocks siehe Rückseite.

Okt. 69

Externcode: BLEI

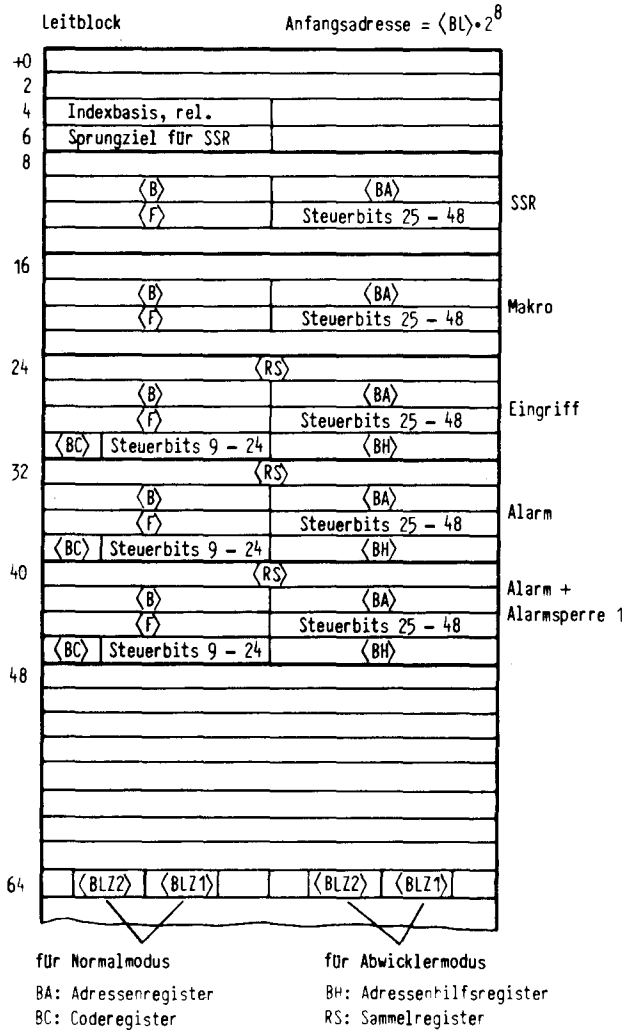
Interncode: 'BE'

belegt: Befehls- und Rechenwerk

Takte: 13

Alarm:

Sonstiges:



Nr.	Name	Bedeutung der Steuerbits
9	BEVA	In der Ausführungsphase wird ein Hauptspeicheroperand benötigt.
10	BESP	Die Dreierprobe darf nicht ersetzt werden.
11	BE30	Kennzeichnung verschiedener Ansprungstellen aus dem Mikroprogramm der Ausführungsphase
12	BE20	
13	BE10	
14	BEAC	Der Befehl wurde im Abspeicher-Nanoprogramm unterbrochen.
15	BEAA	Der Befehl wurde am Anfang der Abrufphase unterbrochen.
16	BEAB	Der Befehl wurde in der Abrufphase bei Auslesen eines Operanden unterbrochen.
17	BEIC	Der Befehl wurde bei Bildung der absoluten Indexbasis unterbrochen.
18	BEMQ	Der Befehl ist nicht zu Ende
19	BEEH	Hauptalarm (Stromausfall bzw. -Abschaltung)
20	BEER4	Rechneralarm vom 4. Rechnerkern
21	BEER3	Rechneralarm vom 3. Rechnerkern
22	BEER2	Rechneralarm vom 2. Rechnerkern
23	BEER1	Rechneralarm vom 1. Rechnerkern
24	BEFI	Technischer Fehler
25	BEMA	Der Befehl MF, MCF oder MD geht vorher
26	BEMO	Der Befehl MFU oder MCFU geht vorher
27	BEMN	Der vorhergehende Befehl definiert mod2
28	BEMM	Der Befehl MM geht vorher (im Modus 16)
29	BEMU	Der Befehl MU geht vorher
30	BEMB	Der Befehl MABI geht vorher
31	BEML	Der Befehl LEI geht vorher
32	BEMP	Der anstehende Befehl ist ein Sprungbefehl
33	BEBE	Steuerbit: Stop vor Abrufphase
34	BEBA	Steuerbit: Modus 16
35	BEBT	Steuerbit: Wartungsmodus
36	BEBF	Steuerbit: Stop nach Abrufphase
37	REAL	Typenkennungsal-Alarm
38	REBUE	Arithmetischer Alarm
39	BEEC	Speicherschutz-Alarm
40	BEEU	Alarm: Überlauf des Registers U
41	BEEK	Befehls-Alarm
42	BEEF	Stop-Alarm
43	BEFE	Eingriffssperre
44	BEEW	Wecker-Alarm
45	BEED	Dreierproben-Alarm
46	BEBO	Abwicklermodus
47	BEBN	Normalmodus
48	BEBY	Systemmodus

BN	n	Bringe negativ	**BN**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := t_n \; ; -\langle n \rangle$

$\left.\begin{array}{l} \langle A \rangle_1 := \langle A \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_* \end{array}\right\}$ nur bei $t_n = 0$ oder 1

Bei Zahlwörtern (TK = 0 oder 1) wird der Inhalt der Speicherzelle n einschließlich Typenkennung mit umgekehrtem Vorzeichen in das Register A gebracht (d.h. bei TK = 0 werden die Binärstellen der Mantisse invertiert, bei TK = 1 werden alle Binärstellen invertiert).

Bei Nichtzahlwörtern (TK = 2 oder 3) werden alle Binärstellen invertiert und einschließlich Typenkennung in das Register A gebracht.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BN

Interncode: '75'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

BNR	n

Bringe negativ und reserviere

BNR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle H \rangle := t_A ; \langle A \rangle$

$\langle A \rangle := t_n ; -\langle n \rangle$

$\left.\begin{array}{l} \langle A \rangle_1 := \langle A \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 0$ oder 1

Der Inhalt des Registers A wird einschließlich Typenkennung in das Register H gebracht und dort reserviert.

Bei Zahlwörtern (TK = 0 oder 1) wird der Inhalt der Speicherzelle n einschließlich Typenkennung mit umgekehrtem Vorzeichen in das Register A gebracht (d.h. bei TK = 0 werden die Binärstellen der Mantisse invertiert, bei TK = 1 werden alle Binärstellen invertiert).

Bei Nichtzahlwörtern (TK = 2 oder 3) werden alle Binärstellen invertiert und einschließlich Typenkennung in das Register A gebracht.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BNR

Interncode: '77'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

BNZ	i_L i_R	Bringe nächstes Zeichen	**BNZ**

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$a = \langle i_R \rangle$ + mod2 = laufende Adresse eines Wortes in der Liste
$b = \langle i_L \rangle_{17-24}$ = laufende Nummer eines Zeichens im Wort (0,1,...)
$d = (48/f) - 1$ = max. Nummer eines Zeichens im Wort, beginnend beim linken Zeichen
$f = \langle i_L \rangle_{9-12}$ = Anzahl der Bits pro Zeichen (4,6,8 oder 12)
die anderen Bits von $\langle i_L \rangle$ sind bedeutungslos

Ausführung:

$\langle A \rangle$:= t_a ;0, Zeichen gemäß a und b rechtsbündig

$\langle B \rangle$:= $\langle i_L \rangle$

$\langle D \rangle$:= undefiniert (einschl. TK)

$\langle Y \rangle$:= undefiniert

b := b + 1 wenn b < d

b := +0
$\langle i_R \rangle$:= $\langle i_R \rangle$ + 2 } wenn b = d

} siehe umseitig

Eine Liste enthält eine beliebige Anzahl Wörter. Jedes dieser Wörter enthält d Zeichen (linksbündig) mit einer Länge von f Bits. Die Numerierung beginnt beim linken Zeichen mit Null.

Durch diesen Befehl kann ein beliebiges Zeichen aus diesem Zeichenvorrat geholt werden und rechtsbündig in das Register A gebracht werden; das Register A wird dabei links mit Nullen aufgefüllt.

Durch wiederholte Anwendung des Befehls BNZ kann ein Zeichen nach dem anderen aus einer Liste von Zeichen in das Register A geholt werden.

Externcode: BNZ

Interncode: 'E6'

belegt: Befehlswerk und Rechenwerk

Takte: 41

Alarm:

Sonstiges: beim Vergleich (b < d) sind nur die Bits $\langle i_L \rangle_{21-24}$ von Bedeutung
beim Löschen (b = d) werden die Bits $\langle i_L \rangle_{17-24}$ auf +0 gelöscht

BQ	n

Bringe nach Q

BQ

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$$\left.\begin{array}{l} \langle Q\rangle := t_n\,;\langle n\rangle \\ \langle Q\rangle_1 := \langle Q\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m \end{array}\right\} \text{ nur bei } t_n = 0 \text{ oder } 1$$

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register Q gebracht.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BQ

Interncode: '72'

belegt: Rechenwerk

Takte: 3

Alarm:

Sonstiges:

BQB	n

Bringe nach Q und bringe (nach A)

BQB

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := t_n ; \langle n \rangle$
$\langle Q \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle A \rangle_1 := \langle A \rangle_2 \\ \langle Q \rangle_1 := \langle Q \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_{\blacksquare} \end{array}\right\}$ nur bei t_n = 0 oder 1

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register A und das Register Q gebracht.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BQB

Interncode: 'DA'

belegt: Rechenwerk

Takte: 2

Alarm:

Sonstiges:

BR	n

Bringe und reserviere

BR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$$\left.\begin{array}{l} \langle H\rangle := t_A\,;\langle A\rangle \\ \langle A\rangle := t_n\,;\langle n\rangle \\ \langle A\rangle_1 := \langle A\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m \end{array}\right\} \text{ nur bei } t_n = 0 \text{ oder } 1$$

Der Inhalt des Registers A wird im Register H reserviert. Danach wird der Inhalt der Speicherzelle n in das Register A gebracht. Der Transport erfolgt jeweils einschließlich Typenkennung.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: BR

Interncode: '76'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

BSS	s_L s_R	Bringe (und setze) Steuerbits (und Sperren)	**BSS**

Adressenteil:

s_L : Bits a_9 bis a_{18}

s_R : ohne Bedeutung, muß aber angegeben werden.

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Normalmodus eingestellt

Ausführung:

a_9 = L: $\langle A \rangle_t$:= 2

$\langle A \rangle_1$:= 0

$\langle A \rangle_2$:= Steuerbit 2

$\langle A \rangle_3$:= L wenn man sich im Rechnerkern 1 befindet

$\langle A \rangle_4$:= L wenn man sich im Rechnerkern 2 befindet

$\langle A \rangle_5$:= L wenn man sich im Rechnerkern 3 befindet

$\langle A \rangle_6$:= L wenn man sich im Rechnerkern 4 befindet

} sonst 0

$\langle A \rangle_{7-48}$:= Steuerbits 7 - 48

a_{10} - a_{24} : ohne Bedeutung

Nr.	Name	Bedeutung der Steuerbits
1	-	-
2	BEBT1	Prüfein- und Ausgabe
3	(R1	Rechnerkern 1)
4	(R2	Rechnerkern 2)
5	(R3	Rechnerkern 3)
6	(R4	Rechnerkern 4)
7	BEFA	Alarmsperre 1
8	BEFB	Alarmsperre 2
9	BEVA	In der Ausführungsphase wird ein Hauptspeicheroperand benötigt.
10	BESP	Die Dreierprobe darf nicht ersetzt werden.
11	BE30	Kennzeichnung verschiedener Ansprungstellen aus dem Mikroprogramm der Ausführungsphase
12	BE20	
13	BE10	
14	BEAC	Der Befehl wurde im Abspeicher-Nanoprogramm unterbrochen.
15	BEAA	Der Befehl wurde am Anfang der Abrufphase unterbrochen.
16	BEAB	Der Befehl wurde in der Abrufphase bei Auslesen eines Operanden unterbrochen.
17	BEIC	Der Befehl wurde bei Bildung der absoluten Indexbasis unterbrochen.
18	BEMQ	Der Befehl ist nicht zu Ende
19	BEEH	Hauptalarm (Stromausfall bzw. -Abschaltung)
20	BEER4	Rechneralarm vom 4. Rechnerkern
21	BEER3	Rechneralarm vom 3. Rechnerkern
22	BEER2	Rechneralarm vom 2. Rechnerkern
23	BEER1	Rechneralarm vom 1. Rechnerkern

Nr.	Name	Bedeutung der Steuerbits
24	BEFT	Technischer Fehler
25	BEMA	Der Befehl MF, MCF oder MD geht vorher
26	BEMO	Der Befehl MFU oder MCFU geht vorher
27	BEMN	Der vorhergehende Befehl definiert mod2
28	BEMM	Der Befehl MM geht vorher (im Modus 16)
29	BEMU	Der Befehl MU geht vorher
30	BEMB	Der Befehl MABI geht vorher
31	BEML	Der Befehl LEI geht vorher
32	BEMP	Der anstehende Befehl ist ein Sprungbefehl
33	BEBE	Steuerbit: Stop vor Abrufphase
34	BEBA	Steuerbit: Modus 16
35	BEBT	Steuerbit: Wartungsmodus
36	BEBF	Steuerbit: Stop nach Abrufphase
37	REAL	Typenkennungsal-Alarm
38	REBUE	Arithmetischer Alarm
39	BEEC	Speicherschutz-Alarm
40	BEEU	Alarm: Überlauf des Registers U
41	BEEK	Befehls-Alarm
42	BEEF	Stop-Alarm
43	BEFE	Eingriffssperre
44	BEEW	Wecker-Alarm
45	BEED	Dreierproben-Alarm
46	BEBO	Abwicklermodus
47	BEBN	Normalmodus
48	BEBY	Systemmodus

Externcode: BSS

Interncode: 'FB'

belegt: Befehls- und Rechenwerk

Takte: 3

Alarm:

Sonstiges: bei nicht erfüllter Voraussetzung

im Abwickler-, Spezial- und Systemmodus
gibt es darüber hinausgehende Wirkungen
wenn weitere Bits gesetzt sind

BT	n

Bringe Teilwort

BT

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

⟨Q⟩ = Maske

Ausführung:

$\langle A\rangle_x := t_n ; \langle n\rangle_x$ falls $\langle Q\rangle_x$ = 0

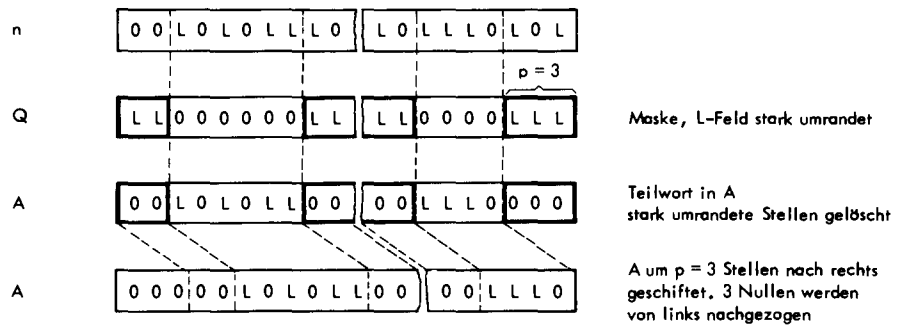
$\langle A\rangle_x := 0$ falls $\langle Q\rangle_x$ = L

} x: 1,2,...,48

⟨A⟩ := ⟨A⟩ um p Stellen nach rechts geschiftet, wobei von links Nullen nachgezogen werden

} p: Anzahl der L-Bits, die rechtsbündig in Q stehen

Im Register Q steht eine Maske. Der Inhalt der Speicherzelle n wird an den Stellen in das Register A gebracht, an denen in der Maske Null steht. Die übrigen Stellen werden im Register A auf Null gelöscht. Anschließend wird der Inhalt des Registers A um soviel Stellen nach rechts geschiftet, wie L-Bits rechtsbündig in der Maske stehen. Von links werden Nullen nachgezogen. Der Transport erfolgt einschließlich Typenkennung.



Externcode: BT

Interncode: 'F6'

belegt: Rechenwerk

Takte: 7 + 2p

Alarm:

Sonstiges:

BU	n	Bringe unverändert (nach A)	**BU**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := t_n ; \langle n \rangle$

Der Inhalt der Speicherzelle n wird unverändert einschließlich Typenkennung als Bitmuster in das Register A gebracht.

Im Gegensatz zum Befehl B wird die Marke nicht berücksichtigt.

Externcode: BU

Interncode: 'D3'

belegt: Rechenwerk

Takte: 2

Alarm:

Sonstiges:

BZ	n	Bringe zwei Wörter	**BZ**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := t_n ; \langle n \rangle$

$\langle Q \rangle := t_{n+2} ; \langle n + 2 \rangle$

$\left.\begin{array}{l} \langle A \rangle_1 := \langle A \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei t_n = 0 oder 1

$\left.\begin{array}{l} \langle Q \rangle_1 := \langle Q \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei t_{n+2} = 0 oder 1

Der Inhalt der Speicherzelle n wird in das Register A und der Inhalt der Speicherzelle n + 2 in das Register Q gebracht. Der Transport erfolgt jeweils einschließlich Typenkennung.

Bei Zahlwörtern wird jeweils die Marke beider Speicherzellen berücksichtigt.

Externcode: BZ

Interncode: 'D9'

belegt: Befehlswerk und Rechenwerk

Takte: 15

Alarm:

Sonstiges:

BZ2	m

Bringe zwei Halbwörter

BZ2

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

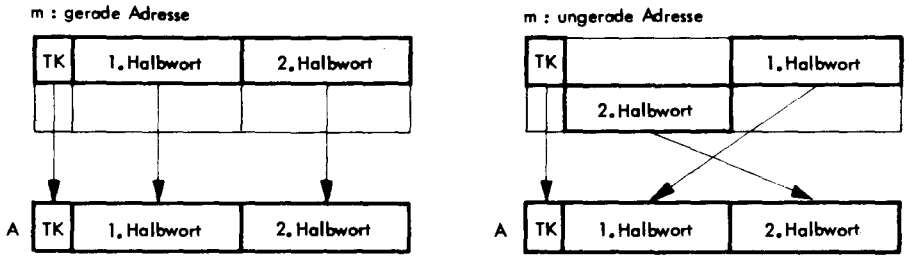
Voraussetzung:

Ausführung:

$\langle A \rangle := t_m ; \langle m, m+1 \rangle$

Der Inhalt der Speicherzelle m und der Speicherzelle m + 1 wird in das Register A gebracht.

Das Register A erhält die Typenkennung der Speicherzelle m.



Externcode: BZ2

Interncode: 'D8'

belegt: Befehlswerk und Rechenwerk

Takte: 10

Alarm:

Sonstiges:

BZN	n

Bringe zwei Wörter negativ

BZN

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle A \rangle := t_n ; -\langle n \rangle$

$\langle Q \rangle := t_{n+2} ; -\langle n + 2 \rangle$

$\left.\begin{array}{l} \langle A \rangle_1 := \langle A \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\left.\begin{array}{l} \langle Q \rangle_1 := \langle Q \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_{n+2} ; = 0$ oder 1

Bei Zahlwörtern (TK = 0 oder 1) wird der Inhalt der Speicherzelle n einschließlich Typenkennung und mit umgekehrtem Vorzeichen in das Register A gebracht. Ebenso wird der Inhalt der Speicherzelle n + 2 in das Register Q transportiert.

Bei Nichtzahlwörtern (TK = 2 oder 3) werden alle Binärstellen invertiert und einschließlich Typenkennung in die entsprechenden Register A und Q gebracht.

Bei Zahlwörtern wird jeweils die Marke beider Speicherzellen berücksichtigt.

Externcode: BZN

Interncode: 'D1'

belegt: Befehlswerk und Rechenwerk

Takte: 15

Alarm:

Sonstiges:

C	n

Speichere (aus Register A)

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle A \rangle_t = 0$ oder 1: $\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle n \rangle := t_A ; \langle A \rangle$

$\langle n \rangle_1 := 0$ nur bei $t_A = 0$ oder 1

Ist im Register A ein Zahlwort (TK = 0 oder 1) enthalten, so darf es nicht über- oder untergelaufen sein.

Der Inhalt des Registers A wird einschließlich Typenkennung und unmarkiert in die Speicherzelle n gebracht.

Externcode: 0

Internode: '80'

belegt: Befehlswerk und Rechenwerk

Takte: 8

Alarm:

BÜ-Alarm: wenn $\langle A \rangle_t = 0$ oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei über- oder untergelaufenen Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

BÜ-Alarm

C2	m

Speichere Halbwort

C2

Adressenteil: m = Adresse eines Halbwortes

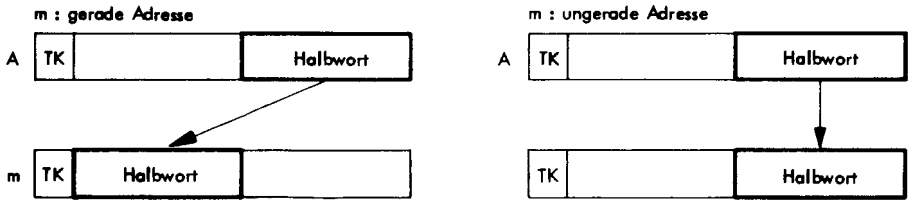
bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle m \rangle := \langle A \rangle_{25-48}$

Vom Inhalt des Registers A wird das rechte Halbwort unverändert in die Speicherzelle m gebracht. Auch die Typenkennung bleibt unverändert.



Externcode: C2

Interncode: 'AO'

belegt: Befehlswerk und Rechenwerk

Takte: 15

Alarm:

Sonstiges:

C3	m

Speichere Drittelwort

C3

Adressenteil: m = Adresse eines Halbwortes

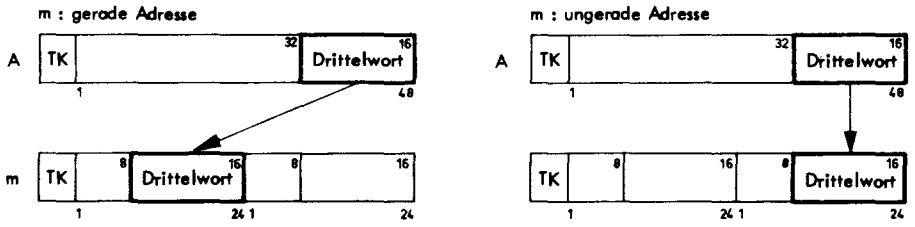
bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle m \rangle_{9-24} := \langle A \rangle_{33-48}$

Vom Inhalt des Registers A werden die rechten 16 Bits (Drittelwort) unverändert in die rechten 16 Stellen der Speicherzelle m gebracht. Auch die Typenkennung bleit unverändert.



Externcode: C3

Interncode: 'A1'

belegt: Befehlswerk und Rechenwerk

Takte: 15

Alarm:

Sonstiges:

CB	n	Speichere Betrag	**CB**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle A \rangle_t$ = 0 oder 1: $\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

bei t_A = 0 oder 1:

$\langle n \rangle := t_A ; |\langle A \rangle|$

$\langle n \rangle_s := 0$

bei t_A = 2 oder 3:

$\langle n \rangle := t_A ; \langle A \rangle$

Ist im Register A ein Zahlwort (TK = 0 oder 1) enthalten, so darf es nicht über- oder untergelaufen sein.

Bei Zahlwörtern wird der Betrag vom Inhalt des Registers A einschließlich Typenkennung und unmarkiert in die Speicherzelle n gebracht.

Bei Nichtzahlwörtern ist die Betragsbildung ohne Bedeutung; die Wirkung ist die gleiche wie bei dem Befehl C.

Externcode: CB

Internode: '85'

belegt: Befehlswerk und Rechenwerk

Takte: 9

Alarm:

BÜ-Alarm: wenn $\langle A \rangle_t = 0$ oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei über- oder untergelaufenem Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t$ und $\langle n \rangle_t = 0$ oder 1

BÜ-Alarm

CD	n

Speichere aus D

CD

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle D \rangle_t = 0$ oder 1: $\langle D \rangle_1 = \langle D \rangle_2$

Ausführung:

$\langle n \rangle := t_D ; \langle D \rangle$

$\langle n \rangle_t := 0$ nur bei $t_D = 0$ oder 1

Ist im Register D ein Zahlwort (TK = 0 oder 1) enthalten, so darf es nicht über- oder untergelaufen sein.

Der Inhalt des Registers D wird einschließlich Typenkennung und unmarkiert in die Speicherzelle n gebracht.

Externcode: CD

Interncode: '86'

belegt: Befehlswerk und Rechenwerk

Takte: 9

Alarm:

BÜ-Alarm: wenn $\langle D \rangle_t = 0$ oder 1
und $\langle D \rangle_1 \neq \langle D \rangle_2$

Sonstiges:

bei über- oder untergelaufenen Operanden:
$\langle D \rangle_1 \neq \langle D \rangle_2$ und $\langle D \rangle_t = 0$ oder 1

BÜ-Alarm

CH	n

Speichere aus H

CH

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle H \rangle_t = 0$ oder 1: $\langle H \rangle_1 = \langle H \rangle_2$

Ausführung:

$\langle n \rangle := t_H ; \langle H \rangle$

$\langle n \rangle_a := 0$ nur bei $t_H = 0$ oder 1

Ist im Register H ein Zahlwort (TK = 0 oder 1) enthalten, so darf es nicht über- oder untergelaufen sein.

Der Inhalt des Registers H wird einschließlich Typenkennung und unmarkiert in die Speicherzelle n gebracht.

Externcode: CH

Interncode: '8F'

belegt: Befehlswerk und Rechenwerk

Takte: 9

Alarm:

BÜ-Alarm: wenn $\langle H \rangle_t = 0$ oder 1
und $\langle H \rangle_1 \neq \langle H \rangle_2$

Sonstiges:

bei über- oder untergelaufenen Operanden:
$\langle H \rangle_1 \neq \langle H \rangle_2$ und $\langle H \rangle_t = 0$ oder 1

BÜ-Alarm

CMC	n

Speichere mit Marke aus Speicher

CMC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t$ und $\langle n \rangle_t$ = 0 oder 1

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle n \rangle := t_A ; \langle A \rangle$

$\langle n \rangle_m := \langle n \rangle_m$

In der Speicherzelle n muß ebenso wie im Register A ein Zahlwort (TK = 0 oder 1) enthalten sein. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt des Registers A wird einschließlich Typenkennung in die Speicherzelle n gebracht. Die ursprüngliche Markenstelle der Speicherzelle n bleibt erhalten.

Externcode: CMC

Interncode: 'A2'

belegt: Befehlswerk und Rechenwerk

Takte: 13

Alarm:

TK-Alarm: wenn $\langle A \rangle_t$ oder $\langle n \rangle_t$ = 2 oder 3

BÜ-Alarm: wenn $\langle A \rangle_t$ und $\langle n \rangle_t$ = 0 oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei falscher Typenkennung:
$\langle A \rangle_t$ oder $\langle n \rangle_t$ = 2 oder 3

TK-Alarm

bei über- oder untergelaufenen Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t$ und $\langle n \rangle_t$ = 0 oder 1

BÜ-Alarm

CMR	n

Speichere mit Marke aus Register

CMR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t$ = 0 oder 1

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle n \rangle := t_A ; \langle A \rangle$

$\langle n \rangle_m := \langle M \rangle$

Im Register A muß ein Zahlwort (TK = 0 oder 1) enthalten sein, welches nicht über- oder untergelaufen ist.

Der Inhalt des Registers A wird einschließlich Typenkennung in die Speicherzelle n gebracht. Die Markenstelle in der Speicherzelle n (1. Bit) ergibt sich aus dem Inhalt des Registers M (Marke).

Externcode: CMR

Interncode: '83'

belegt: Befehlswerk und Rechenwerk

Takte: 8

Alarm:

TK-Alarm: wenn $\langle A \rangle_t = 2$ oder 3

BÜ-Alarm: wenn $\langle A \rangle_t = 0$ oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei falscher Typenkennung:
$\langle A \rangle_t = 2$ oder 3

TK-Alarm

bei über- oder untergelaufenen Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

BÜ-Alarm

CMT	n

Speichere markiert

CMT

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t$ = 0 oder 1

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle n \rangle := t_A ; \langle A \rangle$

$\langle n \rangle_m := L$

Im Register A muß ein Zahlwort (TK = 0 oder 1) enthalten sein, welches nicht über- oder untergelaufen ist.

Der Inhalt des Registers A wird einschließlich Typenkennung in die Speicherzelle n gebracht. Die Markenstelle in der Speicherzelle n (1. Bit) wird auf L gesetzt; das Zahlwort ist markiert.

Externcode: CMT

Interncode: '82'

belegt: Befehlswerk und Rechenwerk

Takte: 8

Alarm:

TK-Alarm: wenn $\langle A \rangle_t = 2$ oder 3

BÜ-Alarm: wenn $\langle A \rangle_t = 0$ oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei falscher Typenkennung:
$\langle A \rangle_t = 2$ oder 3

TK-Alarm

bei über- oder untergelaufenen Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

BÜ-Alarm

CN	n

Speichere negativ

CN

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle A \rangle_t$ = 0 oder 1

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle n \rangle := t_A ; -\langle A \rangle$

$\langle n \rangle_{\blacksquare} := 0$ nur bei t_A = 0 oder 1

Ist im Register A ein Zahlwort (TK = 0 oder 1) enthalten, so darf es nicht über- oder untergelaufen sein.

Bei Zahlwörtern wird der Inhalt des Registers A einschließlich Typenkennung und mit umgekehrtem Vorzeichen in die Speicherzelle n gebracht. Es wird unmarkiert abgespeichert.

Bei Nichtzahlwörtern (TK = 2 oder 3) werden vom Inhalt des Registers A alle Binärstellen invertiert und einschließlich Typenkennung in die Speicherzelle n gebracht.

Externcode: CN

Interncode: '84'

belegt: Befehlswerk und Rechenwerk

Takte: 9

Alarm:

BÜ-Alarm: wenn $\langle A \rangle_t = 0$ oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei über- oder untergelaufenem Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

BÜ-Alarm

CNZ	i_L i_R	Speichere nächstes Zeichen	**CNZ**

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$a = \langle i_R \rangle$ + mod2 = laufende Adresse eines Wortes in der Liste

$b = \langle i_L \rangle_{17-24}$ = laufende Nummer eines Zeichens im Wort (0,1,...)

$d = (48/f) - 1$ = maximale Nummer eines Zeichens im Wort

$f = \langle i_L \rangle_{9-12}$ = Anzahl der Bits pro Zeichen (4,6,8 oder 12)

die anderen Bits von $\langle i_L \rangle$ sind bedeutungslos

Ausführung:

$\langle a \rangle$:= rechte f Bits von $\langle A \rangle$ eingesetzt gemäß a und b; der Rest des Wortes (einschließlich Typenkennung) bleibt unverändert

$\langle B \rangle := \langle i_L \rangle$

$\langle D \rangle$:= undefiniert (einschl. TK)

$\langle Q \rangle := t_A$; abzuspeicherndes Zeichen aus $\langle A \rangle$ gemäß b geschiftet, andere Bits in Q werden gelöscht.

$\langle Y \rangle$:= undefiniert

b := b + 1 wenn b < d

b := +0
$\langle i_R \rangle := \langle i_R \rangle + 2$ } wenn b = d

} siehe umseitig

Eine Liste enthält eine beliebige Anzahl Wörter. Jedes dieser Wörter enthält d Zeichen (linksbündig) mit einer Länge von f Bits. Die Numerierung beginnt beim linken Zeichen mit Null.
Durch diesen Befehl kann ein rechtsbündig im Register A stehendes Zeichen in der Liste abgespeichert werden. Im Register A steht nach der Ausführung des Befehls der neue Inhalt des Wortes, in dem das Zeichen abgespeichert wurde.
Durch wiederholte Anwendung des Befehls CNZ können Zeichen aufeinanderfolgend in der Liste abgespeichert werden.

Externcode: CNZ

Interncode: 'E7'

belegt: Befehlswerk und Rechenwerk

Takte: 55

Alarm:

Sonstiges: beim Vergleich (b <d) sind nur die Bits $\langle i_L \rangle_{21-24}$ von Bedeutung;
beim Löschen (b = d) werden die Bits $\langle i_L \rangle_{17-24}$ auf +0 gelöscht.

CQ	n

Speichere aus Q

CQ

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle Q \rangle_t = 0$ oder 1: $\langle Q \rangle_1 = \langle Q \rangle_2$

Ausführung:

$\langle n \rangle := t_Q ; \langle Q \rangle$

$\langle n \rangle_m := 0$ nur bei $t_Q = 0$ oder 1

Ist im Register Q ein Zahlwort (TK = 0 oder 1) enthalten, so darf es nicht über- oder untergelaufen sein.

Der Inhalt des Registers Q wird einschließlich Typenkennung und unmarkiert in die Speicherzelle n gebracht.

Externcode: CQ

Interncode: '87'

belegt: Befehlswerk und Rechenwerk

Takte: 9

Alarm:

BÜ-Alarm: wenn $\langle Q \rangle_t = 0$ oder 1
und $\langle Q \rangle_1 \neq \langle Q \rangle_2$

Sonstiges:

bei über- oder untergelaufenen Operanden:
$\langle Q \rangle_1 \neq \langle Q \rangle_2$ und $\langle Q \rangle_t = 0$ oder 1

Ergebnis: siehe Ausführung
BÜ-Alarm

CR	n

Speichere und bringe Reserve

CR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

bei $\langle A \rangle_t$ = 0 oder 1: $\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle n \rangle := t_A ; \langle A \rangle$

$\langle A \rangle := t_H ; \langle H \rangle$

$\langle n \rangle_m := 0$ nur bei t_A = 0 oder 1

Ist im Register A ein Zahlwort (TK = 0 oder 1) enthalten, so darf es nicht über- oder untergelaufen sein.

Der Inhalt des Registers A wird einschließlich Typenkennung und unmarkiert in die Speicherzelle n gebracht. Anschließend wird der Inhalt des Registers H einschließlich Typenkennung in das Register A gebracht.

Externcode: CR

Interncode: '81'

belegt: Befehlswerk und Rechenwerk

Takte: 9

Alarm:

BÜ-Alarm: wenn $\langle A \rangle_t = 0$ oder 1
und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

bei über- oder untergelaufenen Operanden:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

Ergebnis: siehe Ausführung
BÜ-Alarm

CT	n	Speichere Teilwort	**CT**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨Q⟩ = Maske

Ausführung:

⟨A⟩ := ⟨A⟩ um p Stellen nach links im Kreis geschiftet } p: Anzahl der L-Bits, die rechtsbündig im Register Q stehen

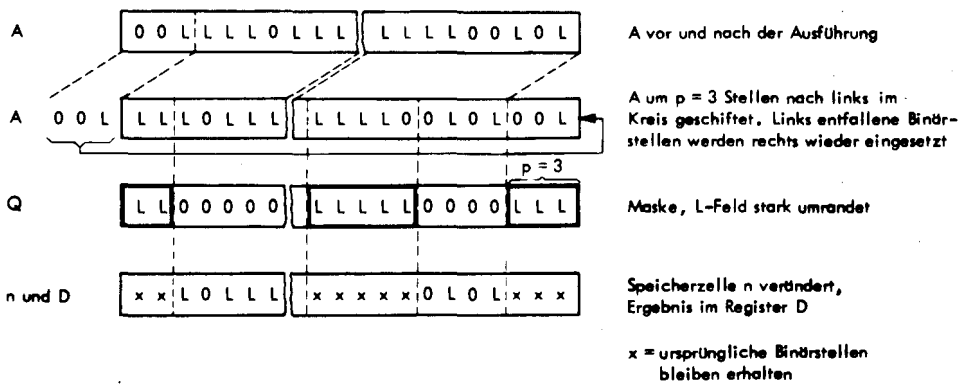
$\langle n\rangle_x := \langle A\rangle_x$ falls $\langle Q\rangle_x = 0$
$\langle n\rangle_x := \langle n\rangle_x$ falls $\langle Q\rangle_x = L$ } x: 1,2,...,48

⟨A⟩ := ⟨A⟩ um p Stellen im Kreis nach rechts in die ursprüngliche Lage zurückgeschiftet

$\langle D\rangle := t_n ; \langle n\rangle$

Im Register Q steht eine Maske. Der Inhalt des Registers A wird um p Stellen nach links im Kreis geschiftet, wobei p die Anzahl der L-Bits angibt, die rechtsbündig im Register Q stehen. Der Inhalt des geschifteten Registers A wird an den Stellen in die Speicherzelle n gebracht, an denen in der Maske Null steht; alle übrigen Stellen sowie die Typenkennung bleiben erhalten. Der so veränderte Inhalt von n wird in das Register D gebracht.

Das Register A wird um p Stellen im Kreis nach rechts wieder in die ursprüngliche Lage zurückgeschiftet.



Externcode: CT

Interncode: 'E7'

belegt: Befehlswerk und Rechenwerk

Takte: 22 + 2p

Alarm:

Sonstiges:

CU	n	Speichere unverändert	**CU**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨n⟩ := t_A ;⟨A⟩

Der Inhalt des Registers A wird einschließlich Typenkennung unverändert als Bitmuster in die Speicherzelle n gebracht.

Im Gegensatz zum Befehl C wird die Marke und ein Über- oder Unterlauf nicht berücksichtigt.

Externcode: CU

Interncode: 'DO'

belegt: Befehlswerk und Rechenwerk

Takte: 8

Alarm:

Sonstiges:

CZ	n

Speichere zwei Wörter aus Reg. A und Q

CZ

Adressenteil: n = Speicheradresse

bei mod2:	wird modifiziert	bei R:	nein

Voraussetzung:

bei $\langle A\rangle_t = 0$ oder 1: $\langle A\rangle_1 = \langle A\rangle_2$

$\langle Q\rangle_1 = \langle Q\rangle_2$

Ausführung:

$\langle n\rangle := t_A; \langle A\rangle$

$\langle n+2\rangle := t_Q; \langle Q\rangle$

$\langle n\rangle_m := 0$ nur bei TK = 0 oder 1

$\langle n+2\rangle_m := 0$ nur bei TK = 0 oder 1

Ist im Register A oder Q ein Zahlwort, so darf es nicht über- oder untergelaufen sein.

Der Inhalt des Registers A wird, einschließlich Typenkennung, in die Speicherzelle n und der Inhalt des Registers Q, einschließlich Typenkennung, in die Speicherzelle n+2 gebracht.

Externcode: CZ

Interncode: 'DB'

belegt: Befehlswerk und Rechenwerk

Takte: 16

Alarm: BÜ-Alarm: wenn $\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1
wenn $\langle Q \rangle_1 \neq \langle Q \rangle_2$ und $\langle Q \rangle_t = 0$ oder 1

Sonstiges:

bei $\langle A \rangle$ über- oder untergelaufen:

Ergebnis siehe Ausführung
BÜ-Alarm

bei $\langle Q \rangle$ über- oder untergelaufen:

Ergebnis siehe Ausführung
BÜ-Alarm

DA	n

Gleitkomma doppelter Genauigkeit: Addition **DA**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$
$\langle Q\rangle_t = \langle n+2\rangle_t = 1$ } richtige Typenkennung

$\langle A\rangle_1 = \langle A\rangle_2 = \langle Q\rangle_1 = \langle Q\rangle_2$ nicht über- oder untergelaufen
$\langle n\rangle_2 = \langle n+2\rangle_2$ } gleiche Vorzeichen

$\langle A,Q\rangle$ und $\langle n, n+2\rangle$ normalisiert

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$
$\langle H\rangle := t_{n+2} ; \langle n+2\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle H\rangle_1 := \langle H\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m \vee \langle n+2\rangle_m$

$\langle A,Q\rangle := 0{,}1; (\langle A,Q\rangle + \langle D,H\rangle)$ normalisiert und gerundet

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle Y\rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde
$\langle Y\rangle := +0$ wenn Ergebnis ± 0 oder bei Exponentenunterlauf

Beide Operanden müssen die für Gleitkommazahlen doppelter Genauigkeit vorgeschriebene Typenkennung haben. Die Vorzeichen in den beiden Wörtern jeder der beiden Operanden müssen gleich sein. Bei den beiden Wörtern der Operanden im Register müssen Bit 1 und 2 gleich sein (nicht über- oder untergelaufen). Es wird vorausgesetzt, daß beide Operanden normalisiert sind.

Die beiden Operanden werden addiert, anschließend normalisiert und gerundet. Die Anzahl der Binärstellen, um die normalisiert wurde, steht im Register Y. Ist das Ergebnis ± 0 oder trat ein Exponentenunterlauf auf, so wird das Register Y auf 0 gesetzt.

Die Register D und H werden mit Typenkennung 1 auf 0 gesetzt.

Die Marke beider Wörter wird berücksichtigt.

Okt. 69

Externcode: DA

Interncode: 'FO'

belegt: Befehls- und Rechenwerk

Takte: 106

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 0$ oder $\langle Q \rangle_t \neq 1$

wenn $\langle n \rangle_t \neq 0$ oder $\langle n+2 \rangle_t \neq 1$

BÜ-Alarm: wenn $\langle A \rangle_1 \neq \langle A \rangle_2$

wenn $\langle Q \rangle_1 \neq \langle Q \rangle_2$

wenn $\langle A \rangle_1 \neq \langle Q \rangle_1$

wenn $\langle n \rangle_2 \neq \langle n+2 \rangle_2$

wenn Exponent des Ergebnisses > +127

Sonstiges:

Bei falscher Typenkennung: $\langle A \rangle_t \neq 0$ oder $\langle Q \rangle \neq 1$ oder
$\langle n \rangle_t \neq 0$ oder $\langle n+2 \rangle_t \neq 1$

$\langle D \rangle := 1; +0$
$\langle H \rangle := 1; +0$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_a$ nur bei $t_n \leq 1$
$\langle M \rangle := \langle M \rangle \vee \langle n+2 \rangle_a$ nur bei $t_{n+2} \leq 1$

$\langle Y \rangle := +0$

TK-Alarm

bei über- oder untergelaufenem Operanden (nicht normalisiert):

$\langle A \rangle_1 \neq \langle A \rangle_2$ oder $\langle Q \rangle_1 \neq \langle Q \rangle_2$ und richtiger TK

$\langle D \rangle := 1; +0$
$\langle H \rangle := 1; +0$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_a \vee \langle n+2 \rangle_a$

$\langle Y \rangle := +0$

BÜ-Alarm

bei ungleichen Vorzeichen: $\langle A \rangle_1 \neq \langle Q \rangle_1$ oder $\langle n \rangle_2 \neq \langle n+2 \rangle_2$ und richtiger TK

$\langle D \rangle := 1; +0$
$\langle H \rangle := 1; +0$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_a \vee \langle n+2 \rangle_a$
$\langle Y \rangle := +0$

BÜ-Alarm

bei übergelaufenem Ergebnis: Exponent > +127

Ergebnis: siehe unter Ausführung

$\langle A \rangle := \langle A \rangle \cdot 16^{-255}$

BÜ-Alarm

bei nicht normalisierten Operanden:

Ergebnis ist u.U. nicht normalisiert und gerundet

DML	n	Gleitkomma doppelte Genauigkeit: Multipliziere	DML

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$
$\langle Q\rangle_t = \langle n+2\rangle_t = 1$ } richtige Typenkennung

$\langle A\rangle_1 = \langle A\rangle_2 = \langle Q\rangle_1 = \langle Q\rangle_2$ nicht über- oder untergelaufen
$\langle n\rangle_2 = \langle n+2\rangle_2$ } gleiche Vorzeichen

$\langle A,Q\rangle$ und $\langle n, n+2\rangle$ normalisiert

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$
$\langle H\rangle := t_{n+2} ; \langle n+2\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle H\rangle_1 := \langle H\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m \vee \langle n+2\rangle_m$

$\langle A,Q\rangle := 0,1; (\langle A,Q\rangle \bullet \langle D,H\rangle)$ normalisiert und gerundet

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle Y\rangle := +0$ oder 4 Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde
$\langle Y\rangle := +0$ wenn das Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die für Gleitkommazahlen doppelter Genauigkeit vorgeschriebene Typenkennung haben. Die Vorzeichen in den beiden Wörtern beider Operanden müssen gleich sein. Die Bits 1 und 2 müssen bei beiden Wörtern der Operanden im Register gleich sein (d.h. nicht über- oder untergelaufen). Es wird vorausgesetzt, daß beide Operanden normalisiert sind.

Die Wörter aus den Speicherzellen n und n+2 werden einschließlich Typenkennung als doppeltlanges Wort in die Register D und H gebracht und mit dem Inhalt des doppeltlangen Registers A,Q multipliziert. Das Ergebnis steht normalisiert und gerundet im Register A,Q. Die Typenkennung ist in beiden Teilen des Registers verschieden. Das Register A hat die Typenkennung =0, das Register Q die TK =1.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y. Ist das Ergebnis ±0 oder trat ein Exponentenunterlauf auf, so wird das Register Y auf 0 gesetzt.

Die Register D und H werden mit Typenkennung =1 auf +0 gelöscht.

Die Marke beider Wörter wird berücksichtigt.

Externcode: DML

Interncode: 'F2'

belegt: Befehls- und Rechenwerk

Takte: 243

Alarm: TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle Q\rangle_t \neq 1$
wenn $\langle n\rangle_t \neq 0$ oder $\langle n+2\rangle_t \neq 1$

BÜ-Alarm: wenn $\langle A\rangle_1 \neq \langle A\rangle_2$
wenn $\langle Q\rangle_1 \neq \langle Q\rangle_2$
wenn $\langle A\rangle_1 \neq \langle Q\rangle_1$
wenn $\langle n\rangle_2 \neq \langle n+2\rangle_2$

wenn Exponent des Ergebnisses > +127

Sonstiges: Bei falscher Typenkennung: $\langle A\rangle_t \neq 0$ oder $\langle Q\rangle \neq 1$ oder
$\langle n\rangle_t \neq 0$ oder $\langle n+2\rangle_t \neq 1$

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_a$ nur bei $t_n \leq 1$
$\langle M\rangle := \langle M\rangle \vee \langle n+2\rangle_a$ nur bei $t_{n+2} \leq 1$

$\langle Y\rangle := +0$

TK-Alarm

bei über- oder untergelaufenem Operanden (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ oder $\langle Q\rangle_1 \neq \langle Q\rangle_2$ und richtiger TK

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_a \vee \langle n+2\rangle_a$

$\langle Y\rangle := +0$

BÜ-Alarm

bei ungleichen Vorzeichen: $\langle A\rangle_1 \neq \langle Q\rangle_1$ oder $\langle n\rangle_2 \neq \langle n+2\rangle_2$ und richtiger TK

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_a \vee \langle n+2\rangle_a$
$\langle Y\rangle := +0$

BÜ-Alarm

bei übergelaufenem Ergebnis: Exponent > +127

Ergebnis: siehe unter Ausführung

$\langle A\rangle := \langle A\rangle \cdot 16^{-256}$

BÜ-Alarm

bei nicht normalisierten Operanden:

Ergebnis: siehe Ausführung
Das Ergebnis wird nur um maximal 4 Binärstellen normalisiert. Das bedeutet bei nicht normalisierten Operanden, daß das Ergebnis zwar gerundet, aber nicht in allen Fällen normalisiert ist.

DSB	n	Gleitkomma doppelte Genauigkeit: Subtrahiere	**DSB**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$
$\langle Q\rangle_t = \langle n+2\rangle_t = 1$ } richtige Typenkennung

$\langle A\rangle_1 = \langle A\rangle_2 = \langle Q\rangle_1 = \langle Q\rangle_2$ nicht über- oder untergelaufen } gleiche Vorzeichen
$\langle n\rangle_2 = \langle n+2\rangle_2$

$\langle A,Q\rangle$ und $\langle n,n+2\rangle$ normalisiert

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$
$\langle H\rangle := t_{n+2} ; \langle n+2\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle H\rangle_1 := \langle H\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_a \vee \langle n+2\rangle_a$

$\langle A,Q\rangle := 0,1;(\langle A,Q\rangle - \langle D,H\rangle)$ normalisiert und gerundet

$\langle D\rangle := 1;+0$
$\langle H\rangle := 1;+0$

$\langle Y\rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde
$\langle Y\rangle$:= +0 wenn das Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die für Gleitkommazahlen doppelter Genauigkeit vorgeschriebene Typenkennung haben. Die Vorzeichen in den beiden Wörtern beider Operanden müssen gleich sein. Die Bits 1 und 2 müssen bei beiden Wörtern der Operanden im Register gleich sein (d.h. nicht über- oder untergelaufen). Es wird vorausgesetzt, daß beide Operanden normalisiert sind.

Die Wörter aus den Speicherzellen n und n+2 werden einschließlich Typenkennung als doppeltlanges Wort in die Register D und H gebracht und vom Inhalt des doppeltlangen Registers A,Q subtrahiert. Das Ergebnis steht normalisiert und gerundet im Register A,Q. Die Typenkennung ist in beiden Teilen des Registers verschieden. Das Register A hat die Typenkennung =0, das Register Q die TK =1.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y. Ist das Ergebnis ±0 oder trat ein Exponentenunterlauf auf, so wird das Register Y auf 0 gesetzt.

Die Register D und H werden mit Typenkennung =1 auf +0 gelöscht.

Die Marke beider Wörter wird berücksichtigt.

Externcode: DSB

Interncode: 'F1'

belegt: Befehls- und Rechenwerk

Takte: 106

Alarm: TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle Q\rangle_t \neq 1$
wenn $\langle n\rangle_t \neq 0$ oder $\langle n+2\rangle_t \neq 1$

BÜ-Alarm: wenn $\langle A\rangle_1 \neq \langle A\rangle_2$
wenn $\langle Q\rangle_1 \neq \langle Q\rangle_2$
wenn $\langle A\rangle_1 \neq \langle Q\rangle_1$
wenn $\langle n\rangle_2 \neq \langle n+2\rangle_2$

wenn Exponent des Ergebnisses > +127

Sonstiges: Bei falscher Typenkennung: $\langle A\rangle_t \neq 0$ oder $\langle Q\rangle \neq 1$ oder
$\langle n\rangle_t \neq 0$ oder $\langle n+2\rangle_t \neq 1$

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$ nur bei $t_n \leq 1$
$\langle M\rangle := \langle M\rangle \vee \langle n+2\rangle_m$ nur bei $t_{n+2} \leq 1$

$\langle Y\rangle := +0$

TK-Alarm

bei über- oder untergelaufenem Operanden (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ oder $\langle Q\rangle_1 \neq \langle Q\rangle_2$ und richtiger TK

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m \vee \langle n+2\rangle_m$

$\langle Y\rangle := +0$

BÜ-Alarm

bei ungleichen Vorzeichen: $\langle A\rangle_1 \neq \langle Q\rangle_1$ oder $\langle n\rangle_2 \neq \langle n+2\rangle_2$ und richtiger TK

$\langle D\rangle := 1; +0$
$\langle H\rangle := 1; +0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m \vee \langle n+2\rangle_m$
$\langle Y\rangle := +0$

BÜ-Alarm

bei übergelaufenem Ergebnis: Exponent > +127

Ergebnis: siehe unter Ausführung

$\langle A\rangle := \langle A\rangle \cdot 16^{-256}$

BÜ-Alarm

bei nicht normalisierten Operanden:

Ergebnis ist u.U. nicht normalisiert und gerundet

DV	n	Dividiere	**DV**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 1$
$\langle A\rangle_1 = \langle A\rangle_2$
$\langle n\rangle \neq \pm 0$ Divisor nicht gleich Null
$|\langle A\rangle| < |\langle n\rangle|$ Dividend kleiner Divisor

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_{\blacksquare}$

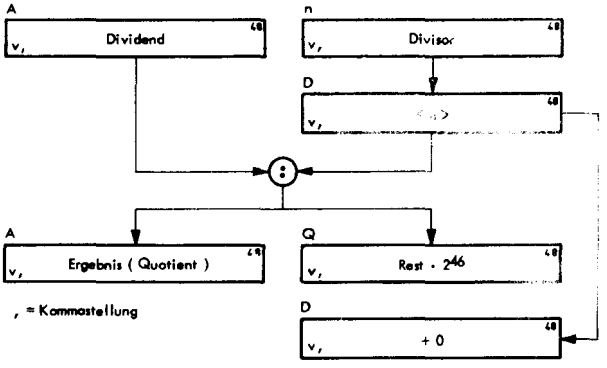
$\langle A\rangle := 1;\langle A\rangle : \langle D\rangle$
$\langle Q\rangle := 1;\text{Rest} \cdot 2^{46}$
} Komma wird hinter den Vorzeichenstellen angenommen
$\langle D\rangle := 1;+0$
$\langle Y\rangle := +0$

Dividend und Divisor müssen die Typenkennung = 1 haben, der Dividend darf weder über- noch untergelaufen sein. Der Divisor muß größer sein als der Dividend und darf nicht Null sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung als Divisor in das Register D gebracht. Der Inhalt des Registers A (Dividend) wird durch den Inhalt des Registers D (Divisor) geteilt. Das Ergebnis (Quotient) steht im Register A, der Rest im Register Q. Die Typenkennung in den Registern A und Q ist = 1. Anschließend wird das Register D auf Null gelöscht.

Das Komma wird hinter den Vorzeichenstellen angenommen. Ist das Komma beim Dividenden um x Stellen nach rechts und beim Divisor um y Stellen nach rechts verschoben, so gilt für den Quotienten eine Verschiebung um x - y Stellen nach rechts. Beim Rest ist das Komma um x Stellen nach rechts verschoben.

Die Marke wird berücksichtigt.



Externcode: DV

Interncode: '60'

belegt: Rechenwerk

Takte: 220

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

BÜ-Alarm: wenn $|\langle A\rangle| \geq |\langle n\rangle|$
wenn $\langle n\rangle = \pm 0$
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:
$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$ } nur bei $t_n = 0$

$\langle Q\rangle := 1;+0$

TK-Alarm
2 Takte

bei Dividend gleich oder größer als Divisor:
$|\langle A\rangle| \geq |\langle n\rangle|$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle A\rangle_1 = \langle A\rangle_2$ und $\langle n\rangle \neq \pm 0$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;\langle A\rangle$: $\langle D\rangle$ Gleitkommazahl, normalisiert, nicht gerundet
$\langle Q\rangle := 1;+0$
$\langle D\rangle := 1;+0$
$\langle Y\rangle := +0$

BÜ-Alarm
258 Takte

bei Divisor gleich Null:
$\langle n\rangle = \pm 0$ und $\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle D\rangle := t_A ;\langle A\rangle$
$\langle Q\rangle := 1;+0$
$\langle A\rangle := 1;+0$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$
$\langle Y\rangle := +0$

BÜ-Alarm
3 Takte

bei über- oder untergelaufenem Dividend:
$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle n\rangle \neq \pm 0$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0$ oder $1;\langle A\rangle$ undefiniert
$\langle Q\rangle := 1;\langle Q\rangle$ undefiniert
$\langle D\rangle := 1;+0$
$\langle Y\rangle := +0$

evtl. BÜ-Alarm
Ausführungszeit wie Normalfall 220 Takte oder wie Dividend ≥ Divisor 258 Takte

DVD	n	Dividiere doppelt lang	**DVD**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 1$
$\langle A\rangle_1 = \langle A\rangle_2$
$\langle Q\rangle_1 = \langle Q\rangle_2$
$\langle A\rangle_1$ kann $\neq \langle Q\rangle_1$
$\langle n\rangle \neq \pm 0$
$|\langle A\rangle| < |\langle n\rangle|$ (nach Vorzeichenangleich)

Ausführung:

$\langle A,Q\rangle := \langle A\rangle + \langle Q\rangle \cdot 2^{-46}$
$\langle Q\rangle_v := \langle A\rangle_v$
} Vorzeichenangleich , wenn $\langle A\rangle_1 \neq \langle Q\rangle_1$ (siehe Befehl VAQ)

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 1 ; \langle A,Q\rangle : \langle D\rangle$
$\langle Q\rangle := 1 ; \text{Rest} \cdot 2^{46}$
} Komma wird hinter den Vorzeichenstellen angenommen

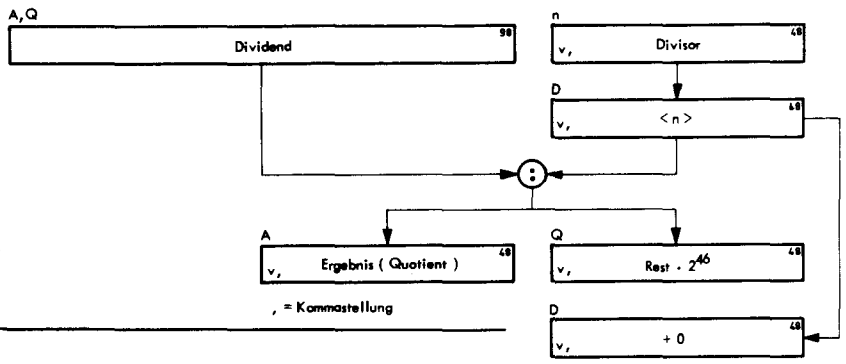
$\langle D\rangle := 1 ; +0$
$\langle Y\rangle := +0$

Dividend und Divisor müssen die Typenkennung = 1 haben, der Dividend darf nicht über- oder untergelaufen sein. Der Divisor muß größer sein als der Dividend und darf nicht Null sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung als Divisor in das Register D gebracht. Der Inhalt des doppeltlangen Registers A,Q (Dividend) wird durch den Inhalt des Registers D (Divisor) geteilt. Das Ergebnis (Quotient) steht im Register A, der Rest im Register Q. Die Typenkennung in den Registern A und Q ist = 1. Anschließend wird das Register D auf Null gelöscht.

Das Komma wird hinter den Vorzeichenstellen angenommen. Ist das Komma beim Dividenden um x Stellen nach rechts und beim Divisor um y Stellen nach rechts verschoben, so gilt für den Quotienten eine Verschiebung um x – y Stellen nach rechts. Beim Rest ist das Komma um x Stellen nach rechts verschoben.

Die Marke wird berücksichtigt.



Externcode: DVD

Interncode: '61'

belegt: Rechenwerk

Takte: 228

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$
wenn $\langle A\rangle_t = 1$ und $\langle n\rangle_t \neq 1$ und $\langle Q\rangle_1 = \langle Q\rangle_2$
wenn $\langle A\rangle_t = 1$ und $\langle n\rangle_t \neq 1$ und $\langle Q\rangle_1 \neq \langle Q\rangle_2$

BÜ-Alarm: wenn $|\langle A\rangle| \geq |\langle n\rangle|$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle A\rangle_1 = \langle A\rangle_2$ und $\langle Q\rangle_1 = \langle Q\rangle_2$ und $\langle n\rangle \neq \pm 0$
wenn $\langle n\rangle = \pm 0$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle Q\rangle_1 = \langle Q\rangle_2$
wenn $\langle n\rangle = \pm 0$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle Q\rangle_1 \neq \langle Q\rangle_2$
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ oder $\langle Q\rangle_1 \neq \langle Q\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle n\rangle \neq \pm 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung des Dividenden: $\langle A\rangle_t \neq 1$

$\langle D\rangle := t_n;\langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm 2 Takte

bei falscher Typenkennung des Divisors: $\langle n\rangle_t \neq 1$ und $\langle A\rangle_t = 1$ und $\langle Q\rangle_1 = \langle Q\rangle_2$

$\left.\begin{array}{l}\langle A,Q\rangle := 1,1;\langle A\rangle + \langle Q\rangle \circ 2^{-48} \\ \langle Q\rangle_v := \langle A\rangle_v\end{array}\right\}$ Vorzeichenangleich wenn $\langle A\rangle_1 \neq \langle Q\rangle$

$\langle D\rangle := t_n;\langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm 10 Takte

bei falscher Typenkennung des Divisors und über- oder untergelaufenem Dividend im Register Q (2. Teil): $\langle n\rangle_t \neq 1$ und $\langle A\rangle_t = 1$ und $\langle Q\rangle_1 \neq \langle Q\rangle_2$

$\langle D\rangle := t_n;\langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 0$

$\langle A\rangle,\langle Q\rangle := 1;$undefiniert

TK-Alarm 10 Takte

bei Dividend gleich oder größer als Divisor: $|\langle A\rangle| \geq |\langle n\rangle|$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle A\rangle_1 = \langle A\rangle_2$ und $\langle Q\rangle_1 = \langle Q\rangle_2$ und $\langle n\rangle \neq \pm 0$

Ergebnis: siehe Ausführung, jedoch
$\langle A\rangle := 0;\langle A\rangle : \langle D\rangle$ Gleitkommazahl, normalisiert, nicht gerundet
$\langle Q\rangle := 1;+0$

BÜ-Alarm 266 Takte

bei Divisor gleich Null: $\langle n\rangle = \pm 0$ und $\langle A\rangle_t = \langle n\rangle_t$ und $\langle Q\rangle_1 = \langle Q\rangle_2$

$\langle Q\rangle := 1;$ (Dividend 2. Teil nach Vorzeichenangleich zwischen A und Q)
$\langle D\rangle := 1;$ (Dividend 1. Teil nach Vorzeichenangleich)
$\langle A\rangle := 1;+0$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

BÜ-Alarm 11 Takte

bei Divisor gleich Null und
über- oder untergelaufenem Dividenden im Register Q (2. Teil):
$\langle n\rangle = +0$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle Q\rangle_1 \neq \langle Q\rangle_2$

$\langle A\rangle := 1;+0$
$\langle Q\rangle,\langle D\rangle := 1;$undefiniert
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$
$\langle Y\rangle := +0$

BÜ-Alarm 11 Takte

bei über- oder untergelaufenem Dividend:
$\langle A\rangle_1 \neq \langle A\rangle_2$ oder $\langle Q\rangle_1 \neq \langle Q\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle n\rangle \neq \pm 0$

$\langle A\rangle := 0$ oder 1; undefiniert
$\langle Q\rangle := 1;$undefiniert
$\langle D\rangle := 1;+0$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$
$\langle Y\rangle := +0$

evtl. BÜ-Alarm Ausführungszeit wie Normalfall 228 Takte oder
wie Dividend ≥ Divisor 266 Takte

DVI	n	Dividiere invers	**DVI**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 1$
$\langle A\rangle_1 = \langle A\rangle_2$
$\langle A\rangle \neq \pm 0$ Divisor nicht gleich Null
$|\langle n\rangle| < |\langle A\rangle|$ Dividend kleiner Divisor

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 1;\langle D\rangle : \langle A\rangle$
$\langle Q\rangle := 1;\text{Rest} \cdot 2^{46}$
} Komma wird hinter den Vorzeichenstellen angenommen

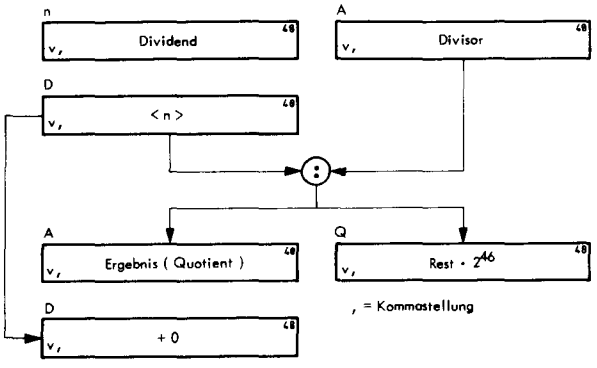
$\langle D\rangle := 1;+0$
$\langle Y\rangle := +0$

Dividend und Divisor müssen die Typenkennung = 1 haben, der Dividend darf weder über- noch untergelaufen sein. Der Divisor muß größer sein als der Dividend und darf nicht Null sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung als Dividend in das Register D gebracht und durch den Inhalt des Registers A (Divisor) geteilt. Das Ergebnis (Quotient) steht im Register A, der Rest im Register Q. Die Typenkennung in den Registern A und Q ist = 1. Anschließend wird das Register D auf Null gelöscht.

Das Komma wird hinter den Vorzeichenstellen angenommen. Ist das Komma beim Dividenden um x Stellen nach rechts und beim Divisor um y Stellen nach rechts verschoben, so gilt für den Quotienten eine Verschiebung um x - y Stellen nach rechts. Beim Rest ist das Komma um x Stellen nach rechts verschoben.

Die Marke wird berücksichtigt.



Externcode: DVI

Interncode: '62'

belegt: Rechenwerk

Takte: 222

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

BÜ-Alarm: wenn $|\langle n\rangle| \geq |\langle A\rangle|$
wenn $\langle A\rangle = \pm 0$
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:
$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_n\end{array}\right\}$ nur bei $t_n = 0$

$\langle Q\rangle := 1;+0$

TK-Alarm
2 Takte

bei Dividend gleich oder größer als Divisor:
$|\langle n\rangle| \geq |\langle A\rangle|$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle A\rangle_1 = \langle A\rangle_2$ und $\langle n\rangle \neq \pm 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_n$

$\langle A\rangle := 0;\langle D\rangle$: $\langle A\rangle$ Gleitkommazahl, normalisiert, nicht gerundet
$\langle Q\rangle := 1;+0$
$\langle D\rangle := 1;+0$
$\langle Y\rangle := +0$

BÜ-Alarm
260 Takte

bei Divisor gleich Null:
$\langle A\rangle = \pm 0$ und $\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_n$

$\langle A\rangle := 1;+0$
$\langle Q\rangle := 1;+0$
$\langle Y\rangle := +0$

BÜ-Alarm
3 Takte

bei über- oder untergelaufenem Divisor:
$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$ und $\langle A\rangle \neq \pm 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_n$

$\langle A\rangle := 0$ oder $1;\langle A\rangle$ undefiniert
$\langle Q\rangle := 1;\langle Q\rangle$ undefiniert
$\langle D\rangle := 1;+0$
$\langle Y\rangle := +0$

evtl. BÜ-Alarm
Ausführungszeit wie Normalfall 222 Takte oder wie Dividend ≥ Divisor 260 Takte

---

E	c i	Ersetze	E

Adressenteil: c = Befehlscode des Zweitbefehls
i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

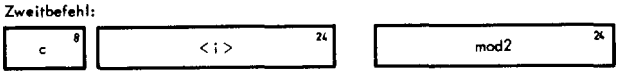
Ausführung:

op := c
adr := ⟨i⟩
mod2 := mod2

Es wird ein Zweitbefehl erzeugt mit c als Operationsteil und mit dem Inhalt der Indexzelle i als Adressenteil. Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wirkt nicht auf diesen Befehl, sondern auf den Zweitbefehl.

Es folgt die Ausführung des Zweitbefehls.

Zweitbefehl:



Externcode: E

Interncode: '29'

belegt: Befehlswerk

Takte: 4 (Mittelwert)

Alarm:

Sonstiges:

EMB	c i

Ersetze und modifiziere mit B

EMB

Adressenteil: c = Befehlscode des Zweitbefehls
i = Adresse einer Indexzelle

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

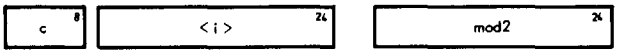
op := c
adr := ⟨i⟩
mod2 := ⟨B⟩

Es wird ein Zweitbefehl erzeugt mit c als Operationsteil und mit dem Inhalt der Indexzelle i als Adressenteil.

Der Inhalt des Registers B ergibt den Modifikator 2. Art für den Zweitbefehl.

Es folgt die Ausführung des Zweitbefehls.

Zweitbefehl:



Externcode: EMB

Interncode: '28'

belegt: Befehlswerk

Takte: 4 (Mittelwert)

Alarm:

Sonstiges:

EMU	c p

Ersetze nach Modifizierung über U

EMU

Adressenteil: c = Befehlscode des Zweitbefehls
p = Zahl ±0...±127

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

```
op   :=  c
adr  := ⟨⟨⟨U⟩⟩ + p⟩
mod2 := mod2
```

Dieser Befehl wird im Unterprogramm verwendet, um Operanden zu holen, deren Adressen im übergeordneten Programm in der Nähe des Befehls SU (Unterprogrammsprung) stehen. ⟨⟨U⟩⟩ ist die Rücksprungadresse im übergeordneten Programm.

Es wird ein Zweitbefehl erzeugt mit c als Befehlscode. Die Rücksprungadresse wird um p erhöht (bei negativem p vermindert) und der Inhalt dieser Adresse ist der Adressenteil des Zweitbefehls. Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wirkt nicht auf diesen Befehl, sondern auf den Zweitbefehl.

Es folgt die Ausführung des Zweitbefehls.

Zweitbefehl:

Externcode: EMU

Interncode: '04'

belegt: Befehlswerk

Takte: 29,5 (Mittelwert)

Alarm:

Sonstiges:

ENZ	c i

Ersetze negativ zählend

ENZ

Adressenteil: c = Befehlscode des Zweitbefehls
i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

op := c
adr := ⟨i⟩ + mod2
mod2 := 0
⟨B⟩ := ⟨i⟩ - 2
⟨i⟩ := ⟨i⟩ - 2

Es wird ein Zweitbefehl erzeugt mit c als Operationsteil und mit dem Inhalt der Indexzelle i als Adressenteil; ein ggf. vom Vorbefehl vorhandener Modifikator 2. Art wird zum Adressenteil des Zweitbefehls addiert und anschließend gelöscht.

Der Inhalt der Indexzelle i wird um 2 vermindert und der verminderte Wert gleichzeitig ins Register B gebracht.

Es folgt die Ausführung des Zweitbefehls.

Es ist zu beachten, daß im Gegensatz zum Befehl EZ der Adressenteil für den Zweitbefehl mit dem ursprünglichen Inhalt der Indexzelle gebildet wird.

Zweitbefehl:

Externcode: ENZ

Interncode: '2A'

belegt: Befehlswerk

Takte: 8

Alarm:

Sonstiges:

ET	n

ET ("UND"-Verknüpfung - Konjunktion)

ET

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_{\blacksquare} \end{array}\right\}$ bei t_n = 0 oder 1

$\langle A \rangle := t_{max} ; \langle A \rangle \wedge \langle D \rangle$

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht.

Der Inhalt des Registers A wird mit dem Inhalt des Registers D durch die Boolesche Operation "UND" verknüpft.

Das Ergebnis steht im Register A und erhält die größere der beiden Typenkennungen der Register A oder D.

Bei Zahlwörtern wird die Marke berücksichtigt.

Verknüpfung durch "UND"

a := b ∧ c		
0	0	0
0	0	L
0	L	0
L	L	L

Externcode: ET

Interncode: '6A'

belegt: Befehlswerk

Takte: 5

Alarm:

Sonstiges:

Externcode: ETA

Interncode: '8A'

belegt: Rechenwerk

Takte: 8

Alarm:

Sonstiges:

EZ	c i

Ersetze zählend

EZ

Adressenteil: c = Befehlscode des Zweitbefehls
i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

```
op   :=  c
adr  := ⟨i⟩ + 2 + mod2
mod2 :=  0
⟨B⟩  := ⟨i⟩ + 2
⟨i⟩  := ⟨i⟩ + 2
```

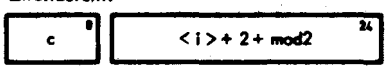
Es wird ein Zweitbefehl erzeugt mit c als Operationsteil und mit dem um 2 erhöhten Inhalt der Indexzelle i als Adressenteil; ein ggf. vom Vorbefehl vorhandener Modifikator 2. Art wird zum Adressenteil des Zweitbefehls addiert und anschließend gelöscht.

Der Inhalt der Indexzelle i wird um 2 erhöht und der erhöhte Wert gleichzeitig ins Register B gebracht.

Es folgt die Ausführung des Zweitbefehls.

Es ist zu beachten, daß im Gegensatz zum Befehl ENZ der Adressenteil mit dem neuen Inhalt der Indexzelle gebildet wird.

Zweitbefehl:



Mär 4

Externcode: EZ

Interncode: '2B'

belegt: Befehlswerk

Takte: 5 (Mittelwert)

Alarm:

Sonstiges:

GA	n

Gleitkomma: Addiere

GA

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$

$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_*$

$\langle A \rangle$:= 0;($\langle A \rangle$ + $\langle D \rangle$) normalisiert und gerundet

$\langle Q \rangle$:= 0;+0

$\langle Y \rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y \rangle$:= +0 wenn das Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und zum Inhalt des Registers A addiert. Das Ergebnis wird normalisiert und gerundet und steht mit Typenkennung = 0 im Register A.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GA

Interncode: '4B'

belegt: Rechenwerk

Takte: 28

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges: Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$
} nur bei $t_n = 1$

TK-Alarm

2 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe unter Ausführung

$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$

BÜ-Alarm

28 Takte

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0; undefiniert
$\langle Q\rangle$:= 0;+0
$\langle Y\rangle$:= undefiniert
evtl. BÜ-Alarm
28 Takte

bei nicht normalisiertem Operand:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

GAB	n	Gleitkomma: Addiere Betrag	**GAB**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle A\rangle$ und $\langle n\rangle$ normalisiert

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;(\langle A\rangle + |\langle D\rangle|)$ normalisiert und gerundet

$\langle Q\rangle := 0;+0$

$\langle Y\rangle :=$ Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y\rangle :=$ +0 wenn Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Der Betrag vom Inhalt dieses Registers wird zum Inhalt des Registers A addiert. Das Ergebnis wird normalisiert und gerundet und steht mit Typenkennung = 0 im Register A.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit der Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GAB

Interncode: '52'

belegt: Rechenwerk

Takte: 28

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm
2 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe unter Ausführung
$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$
BÜ-Alarm
28 Takte

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0; undefiniert
$\langle Q\rangle$:= 0;+0
$\langle Y\rangle$:= undefiniert
evtl. BÜ-Alarm
28 Takte

bei nicht normalisiertem Operand:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

GAC	n

Gleitkomma: Addiere im Speicher

GAC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$

$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_M$

$\langle D \rangle := 0 ; (\langle D \rangle + \langle A \rangle)$ normalisiert und gerundet

$\langle n \rangle := t_0 ; \langle D \rangle$

$\langle n \rangle_M := \langle n \rangle_M$

$\langle Q \rangle := 0 ; +0$

$\langle Y \rangle :=$ Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y \rangle := +0$ wenn Ergebnis = ± 0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht, hierzu wird der Inhalt des Registers A addiert. Das Ergebnis wird normalisiert und gerundet und steht mit Typenkennung = 0 im Register D und wird ebenso in die Speicherzelle n abgespeichert. Die ursprüngliche Marke in der Speicherzelle n bleibt erhalten. Der Inhalt des Registers A bleibt unverändert.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit der Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GAC

Interncode: '4A'

belegt: Befehls- und Rechenwerk

Takte: 37

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges: Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_n\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm

2 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe Ausführung

$\langle D\rangle := \langle D\rangle + \langle A\rangle \cdot 16^{-255}$

BÜ-Alarm

Ergebnis wird nicht abgespeichert

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_n$

$\langle D\rangle :=$ 0;undefiniert

$\langle n\rangle :=$ $\langle n\rangle$ undefiniert

$\langle n\rangle_n := \langle n\rangle_n$

$\langle Q\rangle :=$ 0;+0

$\langle Y\rangle :=$ undefiniert

evtl. BÜ-Alarm

37 Takte

bei nicht normalisiertem Operand:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

GDV	n

Gleitkomma: Dividiere

GDV

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$
$\langle n \rangle \neq \pm 0$
$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$
$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_\blacksquare$

$\langle A \rangle := 0;(\langle A \rangle : \langle D \rangle)$ normalisiert und gerundet
$\langle D \rangle := 0;+0$
$\langle Q \rangle := 0;+0$
$\langle Y \rangle :=$ Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde
$\langle Y \rangle :=$ +0 wenn Ergebnis = ± 0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Der Inhalt des Registers A (Dividend) wird durch den Inhalt des Registers D (Divisor) geteilt. Das Ergebnis (Quotient) wird normalisiert, gerundet und steht mit Typenkennung = 0 im Register A. Die Register D und Q werden mit Typenkennung = 0 auf Null gelöscht.

Im Register Y steht die Anzahl der Binärstellen, um die das Ergebnis im Register A normalisiert wurde.

Die Marke wird berücksichtigt.

Externcode: GDV

Interncode: '64'

belegt: Rechenwerk

Takte: 213

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn $\langle n\rangle = \pm 0$
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)
wenn $\langle A\rangle \neq \pm 0$ und Exponent des Ergebnisses > +127

Sonstiges:

Bei falscher Typenkennung:
$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$ } nur bei $t_n = 1$

$\langle Q\rangle := t_A ; \langle Q\rangle$
TK-Alarm
2 Takte

bei Divisor gleich Null:
$\langle n\rangle = \pm 0$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;+0$
$\langle D\rangle := 0;\langle A\rangle$
$\langle Q\rangle := 0;+0$
$\langle Y\rangle := +0$
BÜ-Alarm
3 Takte

bei über- oder untergelaufenem Dividend (nicht normalisiert):
$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ und $\langle n\rangle \neq \pm 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0$; undefiniert
$\langle D\rangle := 0;+0$
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ undefiniert
evtl. BÜ-Alarm
213 Takte

bei Dividend $\neq \pm 0$ und übergelaufenem Ergebnis:
$\langle A\rangle \neq \pm 0$ und Exponent > +127

Ergebnis: siehe Ausführung
BÜ-Alarm
213 Takte

GDVI	n

Gleitkomma: Dividiere invers

GDVI

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$
$\langle A \rangle \neq \pm 0$
$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$
$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_*$

$\langle A \rangle$:= 0;($\langle n \rangle$: $\langle A \rangle$) normalisiert und gerundet
$\langle D \rangle$:= 0;+0
$\langle Q \rangle$:= 0;+0
$\langle Y \rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde
$\langle Y \rangle$:= +0 wenn Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Der Inhalt des Registers D (Dividend) wird durch den Inhalt des Registers A (Divisor) geteilt. Das Ergebnis (Quotient) wird normalisiert, gerundet und steht mit Typenkennung = 0 im Register A. Die Register D und Q werden mit Typenkennung = 0 auf Null gelöscht.

Im Register Y steht die Anzahl der Binärstellen, um die das Ergebnis im Register A normalisiert wurde.

Die Marke wird berücksichtigt.

Externcode: GDVI

Interncode: '66'

belegt: Rechenwerk

Takte: 216

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn $\langle A\rangle = \pm 0$
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)
wenn $\langle n\rangle \neq \pm 0$ und Exponent des Ergebnisses > +127

Sonstiges:

Bei falscher Typenkennung:
$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle Q\rangle := t_A ; \langle Q\rangle$
TK-Alarm
2 Takte

bei Divisor gleich Null:
$\langle A\rangle = \pm 0$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;+0$
$\langle D\rangle := 0;\langle A\rangle$
$\langle Q\rangle := 0;+0$
$\langle Y\rangle := +0$
BÜ-Alarm
3 Takte

bei über- oder untergelaufenem Divisor (nicht normalisiert):
$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ und $\langle n\rangle \neq \pm 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0$; undefiniert
$\langle D\rangle := 0;+0$
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ undefiniert
evtl. BÜ-Alarm
216 Takte

bei Dividend $\neq \pm 0$ und übergelaufenem Ergebnis:
$\langle n\rangle \neq \pm 0$ und Exponent > +127

Ergebnis: siehe Ausführung
BÜ-Alarm
216 Takte

GMAN	n

Gleitkomma:
Multipliziere akkumulierend negativ

GMAN

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = \langle H\rangle_t = 0$
$\langle A\rangle$ und $\langle n\rangle$ und $\langle H\rangle$ normalisiert
Produkt nicht übergelaufen (Exponent $\leq +127$)

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_*$

$\langle A\rangle := 0;(-\langle A\rangle \cdot \langle D\rangle)$ normalisiert und gerundet Produkt

$\langle A\rangle := 0;(\langle A\rangle + \langle H\rangle)$ normalisiert und gerundet Ergebnis
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ Anzahl der Binärstellen, um die das Ergebnis der Addition normalisiert wurde
$\langle Y\rangle := +0$ wenn das Ergebnis = ±0 oder bei Exponentenüberlauf

Beide Operanden und der Addend (Inhalt vom Register H) müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Produkt erhält umgekehrte Vorzeichen, wird normalisiert, gerundet und steht mit Typenkennung = 0 im Register A. Ist dieses Produkt nicht übergelaufen, wird dazu der Inhalt des Register H addiert.

Das Ergebnis wird gerundet, normalisiert und steht mit Typenkennung = 0 im Register A.

Das Register Q ist mit Typenkennung = 0 auf Null gelöscht. Im Register Y steht die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde.

Die Marke wird berücksichtigt.

Externcode: GMAN

Interncode: '5D'

belegt: Rechenwerk

Takte: 97

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$
wenn $\langle H\rangle_t \neq 0$

BÜ-Alarm: wenn Produkt übergelaufen (Exponent > +127)
wenn Ergebnis übergelaufen (Exponent > +127)
wenn $\langle A\rangle$ unnormalisiert und $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)
wenn $\langle H\rangle$ unnormalisiert und $\langle H\rangle_1 \neq \langle H\rangle_2$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei Falscher Typenkennung der Faktoren: $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ;\langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm
2 Takte

bei übergelaufenem Produkt: Exponent > +127

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;-\langle A\rangle \cdot \langle D\rangle \cdot 16^{-255}$ normalisiert und gerundet
$\langle Q\rangle := 0;+0$
$\langle Y\rangle := 0$ oder 4
BÜ-Alarm
54 Takte

bei falscher Typenkennung des Addenden: $\langle H\rangle_t \neq 0$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;-\langle A\rangle \cdot \langle D\rangle$ gerundet und normalisiert
$\langle Q\rangle := 0;+0$
$\langle Y\rangle := 0$ oder 4
TK-Alarm
54 Takte

Operanden nicht normalisiert: $\langle A\rangle$ oder $\langle n\rangle$ oder $\langle H\rangle$ nicht normalisiert und $\langle A\rangle_1 = \langle A\rangle_2$, $\langle H\rangle_1 = \langle H\rangle_2$

Ergebnis: siehe Ausführung
$\langle Y\rangle := 0$ oder 4
Das Ergebnis wird nur um maximal 4 Binärstellen normalisiert. Das bedeutet bei nicht normalisierten Operanden, daß das Ergebnis zwar gerundet, aber nicht in allen Fällen normalisiert ist.

Operanden nicht normalisiert und $\langle A\rangle$ über- oder untergelaufen:
$\langle A\rangle$ oder $\langle n\rangle$ oder $\langle H\rangle$ nicht normalisiert und $\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0$;undefiniert
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ undefiniert
evtl. BÜ-Alarm und TK-Alarm möglich
97 Takte

Operanden nicht normalisiert und $\langle H\rangle$ über- oder untergelaufen:
$\langle A\rangle$ oder $\langle n\rangle$ oder $\langle H\rangle$ nicht normalisiert und $\langle H\rangle_1 \neq \langle H\rangle_2$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle :=$ undefiniert
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ undefiniert
evtl. BÜ-Alarm
97 Takte

GML	n

Gleitkomma: Multipliziere

GML

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle A\rangle$ und $\langle n\rangle$ normalisiert

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0;($\langle A\rangle \cdot \langle D\rangle$) normalisiert und gerundet

$\langle Q\rangle$:= 0;+0

$\langle Y\rangle$:= +0 oder 4 Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y\rangle$:= +0 wenn Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Ergebnis wird normalisiert, gerundet und steht mit Typenkennung = 0 im Register A.

Das Register Q wird mit Typenkennung = 0 auf Null gelöscht. Im Register Y steht die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde.

Die Marke wird berücksichtigt.

Externcode: GML

Interncode: '5E'

belegt: Rechenwerk

Takte: 54

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 0$ oder $\langle n \rangle_t \neq 0$

BÜ-Alarm: wenn Ergebnis übergelaufen (Exponent > +127)
wenn $\langle A \rangle$ unnormalisiert und $\langle A \rangle_1 \neq \langle A \rangle_2$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:
$\langle A \rangle_t \neq 0$ oder $\langle n \rangle_t \neq 0$

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm
2 Takte

bei übergelaufenem Ergebnis:
Exponent > +127 und beide Operanden $\neq \pm 0$

Ergebnis: siehe Ausführung
$\langle A \rangle := \langle A \rangle \cdot 16^{-255}$
BÜ-Alarm
54 Takte

Operand nicht normalisiert und $\langle A \rangle$ über- oder untergelaufen:
$\langle A \rangle$ oder $\langle n \rangle$ oder $\langle M \rangle$ nicht normalisiert und
$\langle A \rangle_1 \neq \langle A \rangle_2$

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$
$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A \rangle := 0$; undefiniert
$\langle Q \rangle := 0;+0$
$\langle Y \rangle :=$ undefiniert
evtl. BÜ-Alarm
54 Takte

Operanden nicht normalisiert:
$\langle A \rangle$ und $\langle n \rangle$ nicht normalisiert

Ergebnis: siehe Ausführung
$\langle Y \rangle := 0$ oder 4
Das Ergebnis wird nur um maximal 4 Binärstellen normalisiert. Das bedeutet bei nicht normalisierten Operanden, daß das Ergebnis zwar gerundet, aber nicht in allen Fällen normalisiert ist.

GMLA	n

Gleitkomma: Multipliziere akkumulierend

GMLA

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = \langle H\rangle_t = 0$

$\langle A\rangle$ und $\langle n\rangle$ und $\langle H\rangle$ normalisiert

Produkt nicht übergelaufen (Exponent > +127)

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_*$

$\langle A\rangle := 0;(\langle A\rangle \cdot \langle D\rangle)$ normalisiert und gerundet Produkt

$\langle A\rangle := 0;(\langle A\rangle + \langle H\rangle)$ normalisiert und gerundet Ergebnis

$\langle Q\rangle := 0;+0$

$\langle Y\rangle$:= Anzahl der Binärstellen, um die das Ergebnis der Addition normalisiert wurde

$\langle Y\rangle := +0$ wenn das Ergebnis = ±0 oder bei Exponentenüberlauf

Beide Operanden und der Addend (Inhalt vom Register H) müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Produkt wird normalisiert, gerundet und steht mit Typenkennung = 0 im Register A. Ist dieses Produkt nicht übergelaufen, wird dazu der Inhalt des Register H addiert.

Das Ergebnis wird gerundet, normalisiert und steht mit Typenkennung = 0 im Register A.

Das Register Q ist mit Typenkennung = 0 auf Null gelöscht. Im Register Y steht die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde.

Die Marke wird berücksichtigt.

Externcode: GMLA

Interncode: '5F'

belegt: Rechenwerk

Takte: 97

Alarm: TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$
wenn $\langle H\rangle_t \neq 0$

BÜ-Alarm: wenn Produkt übergelaufen (Exponent > +127)
wenn Ergebnis übergelaufen (Exponent > +127)
wenn $\langle A\rangle$ unnormalisiert und $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)
wenn $\langle H\rangle$ unnormalisiert und $\langle H\rangle_1 \neq \langle H\rangle_2$ (nicht unbedingt BÜ-Alarm)

Sonstiges: Bei falscher Typenkennung der Faktoren: $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ;\langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm
2 Takte

bei übergelaufenem Produkt: Exponent > +127

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;\langle A\rangle \cdot \langle D\rangle \cdot 16^{-255}$ normalisiert und gerundet
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ 0 oder 4
BÜ-Alarm
54 Takte

bei falscher Typenkennung des Addenden: $\langle H\rangle_t \neq 0$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;\langle A\rangle \cdot \langle D\rangle$ gerundet und normalisiert
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ 0 oder 4
TK-Alarm
54 Takte

Operanden nicht normalisiert: $\langle A\rangle$ oder $\langle n\rangle$ oder $\langle H\rangle$ nicht normalisiert und
$\langle A\rangle_1 = \langle A\rangle_2$, $\langle H\rangle_1 = \langle H\rangle_2$

Ergebnis: siehe Ausführung
$\langle Y\rangle :=$ 0 oder 4
Das Ergebnis wird nur um maximal 4 Binärstellen normalisiert. Das bedeutet bei nicht normalisierten Operanden daß das Ergebnis zwar gerundet, aber nicht in allen Fällen normalisiert ist.

Operanden nicht normalisiert und $\langle A\rangle$ über- oder untergelaufen:
$\langle A\rangle$ oder $\langle n\rangle$ oder $\langle H\rangle$ nicht normalisiert und $\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle :=$ 0;undefiniert
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ undefiniert
evtl. BÜ-Alarm und TK-Alarm möglich
97 Takte

Operanden nicht normalisiert und $\langle H\rangle$ über- oder untergelaufen:
$\langle A\rangle$ oder $\langle n\rangle$ oder $\langle H\rangle$ nicht normalisiert und $\langle H\rangle_1 \neq \langle H\rangle_2$

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle :=$ undefiniert
$\langle Q\rangle := 0;+0$
$\langle Y\rangle :=$ undefiniert
evtl. BÜ-Alarm
97 Takte

GMLN	n

Gleitkomma: Multipliziere negativ

GMLN

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle A\rangle$ und $\langle n\rangle$ normalisiert

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0;(-$\langle A\rangle \cdot \langle D\rangle$) normalisiert und gerundet

$\langle Q\rangle$:= 0;+0

$\langle Y\rangle$:= 0 oder 4 Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y\rangle$:= +0 wenn Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Ergebnis erhält umgekehrte Vorzeichen, wird normalisiert, gerundet und steht mit Typenkennung = 0 im Register A.

Das Register Q wird mit Typenkennung = 0 auf Null gelöscht. Im Register Y steht die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde.

Die Marke wird berücksichtigt.

Externcode: GMLN

Interncode: '5C'

belegt: Rechenwerk

Takte: 54

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 0$ oder $\langle n \rangle_t \neq 0$

BÜ-Alarm: wenn Ergebnis übergelaufen (Exponent > +127)
wenn $\langle A \rangle$ unnormalisiert und $\langle A \rangle_1 \neq \langle A \rangle_2$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:
$\langle A \rangle_t \neq 0$ oder $\langle n \rangle_t \neq 0$

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm
2 Takte

bei übergelaufenem Ergebnis:
Exponent > +127 und beide Operanden $\neq \pm 0$

Ergebnis: siehe Ausführung
$\langle A \rangle := \langle A \rangle \cdot 16^{-255}$
BÜ-Alarm
54 Takte

Operand nicht normalisiert und $\langle A \rangle$ über- oder untergelaufen:
$\langle A \rangle$ oder $\langle n \rangle$ oder $\langle M \rangle$ nicht normalisiert und
$\langle A \rangle_1 \neq \langle A \rangle_2$

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$
$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A \rangle$:= 0; undefiniert
$\langle Q \rangle$:= 0;+0
$\langle Y \rangle$:= undefiniert
evtl. BÜ-Alarm
54 Takte

Operanden nicht normalisiert:
$\langle A \rangle$ und $\langle n \rangle$ nicht normalisiert

Ergebnis: siehe Ausführung
$\langle Y \rangle$:= 0 oder 4
Das Ergebnis wird nur um maximal 4 Binärstellen normalisiert. Das bedeutet bei nicht normalisierten Operanden, daß das Ergebnis zwar gerundet, aber nicht in allen Fällen normalisiert ist.

GSB	n	Gleitkomma: Subtrahiere	**GSB**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$

$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A \rangle$:= 0;($\langle A \rangle$ - $\langle D \rangle$) normalisiert und gerundet

$\langle Q \rangle$:= 0;+0

$\langle Y \rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y \rangle$:= +0 wenn das Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und vom Inhalt des Registers A subtrahiert. Das Ergebnis wird normalisiert und gerundet und steht mit Typenkennung = 0 im Register A.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit der Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GSB

Interncode: '4F'

belegt: Rechenwerk

Takte: 28

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$ } nur bei $t_n = 1$

TK-Alarm
2 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe unter Ausführung
$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$
BÜ-Alarm
28 Takte

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0; undefiniert
$\langle Q\rangle$:= 0;+0
$\langle Y\rangle$:= undefiniert
evtl. BÜ-Alarm
28 Takte

bei nicht normalisiertem Operand:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

GSBB	n

Gleitkomma: Subtrahiere Betrag

GSBB

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$

$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A \rangle := 0;(\langle A \rangle - |\langle D \rangle|)$ normalisiert und gerundet

$\langle Q \rangle := 0;+0$

$\langle Y \rangle :=$ Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y \rangle :=$ +0 wenn Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Der Betrag vom Inhalt dieses Registers wird vom Inhalt des Registers A subtrahiert. Das Ergebnis wird normalisiert und gerundet und steht mit Typenkennung = 0 im Register D.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit der Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GSBB

Interncode: '53'

belegt: Rechenwerk

Takte: 28

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_{\blacksquare}\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm
2 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe unter Ausführung
$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$
BÜ-Alarm
28 Takte

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_{\blacksquare}$

$\langle A\rangle$:= 0; undefiniert
$\langle Q\rangle$:= 0;+0
$\langle Y\rangle$:= undefiniert
evtl. BÜ-Alarm
28 Takte

bei nicht normalisiertem Operand:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

GSBC	n

Gleitkomma: Subtrahiere im Speicher

GSBC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle A\rangle$ und $\langle n\rangle$ normalisiert

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle D\rangle$:= ($\langle D\rangle$ - $\langle A\rangle$) normalisiert und gerundet

$\langle n\rangle := t_b ; \langle D\rangle$

$\langle n\rangle_m := \langle n\rangle_m$

$\langle Q\rangle := 0;+0$

$\langle Y\rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y\rangle$:= +0 wenn Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht, vom Inhalt dieses Registers wird der Inhalt des Registers A subtrahiert. Das Ergebnis wird normalisiert und steht mit Typenkennung = 0 im Register D und wird ebenso in die Speicherzelle n abgespeichert.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit der Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GSBC

Interncode: '4E'

belegt: Befehlswerk und Rechenwerk

Takte: 37

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm

2 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe Ausführung

$\langle D\rangle := \langle D\rangle + \langle A\rangle \cdot 16^{-255}$

BÜ-Alarm

Ergebnis wird nicht abgespeichert

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle D\rangle := 0$; undefiniert

$\langle n\rangle :=$ undefiniert

$\langle n\rangle_m := \langle n\rangle_m$

$\langle Q\rangle := 0;+0$

$\langle Y\rangle :=$ undefiniert

evtl. BÜ-Alarm

37 Takte

bei nicht normalisiertem Operand:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

GSBD	n

Gleitkomma: Subtrahiere von D

GSBD

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle D\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle$ und $\langle n\rangle$ normalisiert

Ausführung:

$\langle A\rangle$:= 0;($\langle D\rangle$ - $\langle n\rangle$) normalisiert und gerundet

$\langle Q\rangle$:= 0;+0

$\langle M\rangle$:= $\langle M\rangle \vee \langle n\rangle_m$

$\langle Y\rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wird

$\langle Y\rangle$:= +0 wenn das Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird vom Inhalt des Registers D subtrahiert. Das Ergebnis wird normalisiert und gerundet und steht mit Typenkennung = 0 im Register A.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit der Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GSBD

Interncode: '4C'

belegt: Rechenwerk

Takte: 30

Alarm:

TK-Alarm: wenn $\langle D\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle D\rangle_1 \neq \langle D\rangle_2$ und $\langle D\rangle_t = \langle n\rangle_t = 0$

Sonstiges:

Bei falscher Typenkennung:

$\langle D\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle A\rangle := t_D ; \langle D\rangle$
$\langle D\rangle := t_n ; \langle n\rangle$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$ nur bei $t_n = 0$ oder 1
TK-Alarm
3 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe unter Ausführung
$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$
BÜ-Alarm
30 Takte

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle D\rangle_1 \neq \langle D\rangle_2$ und $\langle D\rangle_t = \langle n\rangle_t = 0$

$\langle A\rangle := 0$; undefiniert
$\langle Q\rangle := 0; +0$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle Y\rangle :=$ undefiniert
evtl. BÜ-Alarm
30 Takte

bei nicht normalisierten Operanden:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

GSBI	n

Gleitkomma: Subtrahiere invers

GSBI

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$

$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_{\blacksquare}$

$\langle A \rangle$:= 0;($\langle D \rangle$ - $\langle A \rangle$) normalisiert und gerundet

$\langle Q \rangle$:= 0;+0

$\langle Y \rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y \rangle$:= +0 wenn das Ergebnis = ±0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und normalisiert sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht, vom Inhalt dieses Registers wird der Inhalt des Registers A subtrahiert. Das Ergebnis wird normalisiert und gerundet und steht mit Typenkennung = 0 im Register A.

Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y.

Das Register Q wird mit der Typenkennung = 0 auf Null gelöscht.

Die Marke wird berücksichtigt.

Externcode: GSBI

Interncode: '48'

belegt: Rechenwerk

Takte: 28

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn Exponent des Ergebnisses > +127
wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$ (nicht unbedingt BÜ-Alarm)

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 1$

$\langle A\rangle := t_A ; -\langle A\rangle$

TK-Alarm

2 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe unter Ausführung

$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$

BÜ-Alarm

28 Takte

bei über- oder untergelaufenem Operand (nicht normalisiert):

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0; undefiniert

$\langle Q\rangle$:= 0;+0

$\langle Y\rangle$:= undefiniert

evtl. BÜ-Alarm

28 Takte

bei nicht normalisiertem Operand:

Ergebnis kann u. U. nicht normalisiert und nicht gerundet sein.

HALT	z	Halt

HALT

Adressenteil: z: ohne Bedeutung, muß aber angegeben werden

bei mod2:	wird nicht modifiziert	bei #:	nicht zugelassen

Voraussetzung:

BEBY = L — System- oder Spezialmodus

oder BEBT1 = L
oder BEBT = BEWA = L
oder BEBT = BEWH = L
} Wartungsvariante

Ausführung:

BEKH := L

Am Ende der Abrufphase (des nächsten Befehls) wird das Steuerbit BEKH abgefragt. Ist es L, so wird in der Abrufphase eine Warteschleife durchgeführt. Die Taste "HALT" am Bedienpult leuchtet auf.

Durch Drücken der Taste "HALT" am Bedienpult wird das Steuerbit BEKH gelöscht. Damit wird die Warteschleife verlassen und der nächste Befehl ausgeführt.

Externcode: HALT

Interncode: '9D'

belegt: Befehls- und Rechenwerk

Takte: 2

Alarm:

Sonstiges:

Voraussetzung nicht erfüllt:

Sprung in das Nanoprogramm "Makro"

Beschreibung siehe unter "Makro"

HBA	z

Erhöhe B um Adressenteil

HBA

Adressenteil: z: Zahl = 0...65 535

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle B \rangle + z$

Der Inhalt des Registers B wird um die Zahl z erhöht.

Externcode: HBA

Interncode: '11'

belegt: Befehlswerk

Takte: 7,5

Alarm:

Sonstiges:

HBC	m

Erhöhe B um Speicher

HBC

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle B \rangle + \langle m \rangle$

Der Inhalt des Registers B wird um den Inhalt der Speicheradresse m erhöht.

Externcode: HBC

Interncode: '3C'

belegt: Befehlswerk

Takte: 6,5

Alarm:

Sonstiges:

HBPX	p i

Erhöhe B um Parameter mal Indexzelle

HBPX

Adressenteil: p = Zahl (±1...±15)
i = Adresse einer Indexzelle

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle B \rangle + p \cdot \langle i \rangle$

Der Inhalt der Indexzelle i wird mit der Zahl p multipliziert und zum Inhalt des Registers B addiert. Ist die Zahl p negativ, so wird das Produkt vom Inhalt des Registers B subtrahiert.

Externcode: HBPX

Interncode: 'OF'

belegt: Befehlswerk

Takte: 5,5|p| + 3

Alarm:

Befehlsalarm: wenn |p| > 15

Sonstiges:

wenn p = ±0:

Ergebnis wie Leerbefehl (NULL-Befehl)
Ausführungszeit: 2 Takte

wenn |p| > 15:

Ergebnis wie Leerbefehl (NULL-Befehl)
Befehlsalarm
Ausführungszeit: 2 Takte

HXP	p i

Erhöhe Indexzelle um Parameter

HXP

Adressenteil: p = Zahl 0...±127
i = Adresse einer Indexzelle

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := ⟨i⟩ + p
⟨i⟩ := ⟨i⟩ + p

Der Inhalt der Indexzelle wird um die Zahl p erhöht (negatives p bedeutet Subtraktion).

Das Ergebnis wird gleichzeitig ins Register B gebracht.

Externcode: HXP

Interncode: '2C'

belegt: Befehlswerk

Takte: 10,5

Alarm:

Sonstiges:

HXX	i_L i_R	Erhöhe Indexzelle um Indexzelle	**HXX**

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle i_R \rangle + \langle i_L \rangle$

$\langle i_R \rangle := \langle i_R \rangle + \langle i_L \rangle$

Der Inhalt der Indexzelle i_R wird um den Inhalt der Indexzelle i_L erhöht.

Das Ergebnis wird gleichzeitig ins Register B gebracht.

Externcode: HXX

Interncode: '2E'

belegt: Befehlswerk

Takte: 14,5

Alarm:

Sonstiges:

IR	s

Invertiere Register (oder bilde Betrag im Register) **IR**

Adressenteil: s_1 : Angabe der Register A,Q,D und H

s_2 : leer = invertiere
B = Betrag bilden

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle s_1 \rangle := \neg\langle s_1 \rangle$ nur bei s_2 = leer

$\langle s_1 \rangle := |\langle s_1 \rangle|$ nur bei s_2 = B

Ist s_2 = leer, so wird der Inhalt der mit s_1 angegebenen Register invertiert, d. h. jede Binärstelle erhält den umgekehrten Wert. Bei Typenkennung = 0 werden nur die linken 40 Binärstellen (Mantisse) invertiert.

Ist s_2 = B, so wird der Betrag vom Inhalt der mit s_1 angegebenen Register gebildet. Dies ist nur sinnvoll bei Typenkennung = 0 oder 1. (Bei TK = 2 oder 3 keine Wirkung.)

Externcode: IR

Interncode: 'E1'

belegt: Rechenwerk

Takte: 4p + 1

p = Anzahl der adressierten Register

Alarm:

Sonstiges:

Adressenteil intern:

s_1: A = '--80'
Q = '--40'
D = '--20'
H = '--10'

s_2: leer = '--00'
B = '--08'

KB	z

Kein Befehl

KB

Adressenteil: z : muß angegeben werden, ist aber ohne Bedeutung

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprung in das Nanoprogramm "Makro".

März 70

Externcode: KB

Internecode: '9E'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

KDFR	p

Konvertiere Dezimalzahl in Festkommazahl rechtsbündig

KDFR

Adressenteil: p = Anzahl der Dezimalstellen die konvertiert werden sollen 1...13
Code siehe umseitig

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A,Q \rangle$ = positive Dezimalzahl, Komma rechts
Code siehe umseitig
nur die rechten p Tetraden sind von Bedeutung

Pseudodezimale: siehe umseitig

Ausführung:

$\langle A \rangle_{Festk.} := \langle A,Q \rangle_{Dez.}$
$\langle A \rangle_t := 1$
$\langle Q \rangle, \langle D \rangle, \langle H \rangle = +0$
$\langle Q \rangle_t, \langle D \rangle_t, \langle H \rangle_t = 1$
$\langle Y \rangle = +0$

Die im doppeltlangen Register A,Q stehende Dezimalzahl ist in Tetraden verschlüsselt und wird in eine Festkommazahl umgewandelt. Beide Zahlen sind ganze Zahlen, d. h. das Komma steht rechts.

Bei anderer Kommastellung muß eine Kommarechnung durchgeführt werden.

Es wird nur die durch p angegebene Anzahl von Tetraden rechts beginnend konvertiert. Alle anderen Stellen sind bedeutungslos.

Es werden auch Tetraden konvertiert, die größer als 9 sind (Pseudotetraden). Ausführung siehe umseitig.

Externcode: KDFR

Interncode: '94'

belegt: Rechenwerk

Takte: 7p + 22

Alarm:

Sonstiges:

p größer als 13:

Ergebnis: siehe unter Ausführung

⟨A⟩ = undefiniert

Dezimalcode:

0000 = 0	Dezimale	LOLO = 10	Pseudodezimale
000L = 1		LOLL = 11	
00LO = 2		LLOO = 12	
00LL = 3		LLOL = 13	
0LOO = 4		LLLO = 14	
0LOL = 5		LLLL = 15	
0LLO = 6			
0LLL = 7			
LOOO = 8			
LOOL = 9			

Pseudodezimale:

Es gilt für die Festkommazahl f:

$f = a_1 \cdot 10^0 + a_2 \cdot 10^1 + a_3 \cdot 10^2 + \ldots + a_p \cdot 10^{p-1}$

a = Tetraden 1 bis 13 mit a_1 = rechte Tetrade
(Dezimale und Pseudodezimale)

Adressenteil intern:

'--0x'

x = p = 1...13

KFLD	p

Konvertiere Festkommazahl linksbündig in Dezimalzahl

KFLD

Adressenteil: p = Anzahl der gewünschten Dezimalstellen 1...13

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle$ = positive Festkommazahl, Komma links

$\langle A \rangle_t$ = 0 oder 1

$\langle A \rangle_1 = \langle A \rangle_2 = 0$

Ausführung:

$\langle A,Q \rangle_{Dez.} := \langle A \rangle_{Festk.}$
p Tetraden, rechtsbündig
Komma links vor den p Tetraden

$\langle A \rangle_t := 1$
$\langle Q \rangle_t := 1$
$\langle D \rangle_{Festk.} :=$ Rest $\cdot 10^p$ (Komma links)
$\langle D \rangle_t := 1$
$\langle Y \rangle := +0$

<A> Festkommazahl:

```
ZTR  1A,
AA   1,
ML   C'384B94D092ED'1/1)
SH   ZL  3,
KFLD 13,
BT   AQ,
```

⇒ <A> konvertiert Dezimal-Zahl

Im Register A muß ein positiver Festkommabruch (Komma links) stehen mit der Typenkennung = 0 oder = 1. Er darf nicht über- oder untergelaufen sein.

Der Festkommabruch wird in einen Dezimalbruch konvertiert, der die mit p angegebene Anzahl von Dezimalstellen hat. Die p Tetraden des Dezimalbruches werden rechtsbündig in das doppeltlange Register A,Q gebracht. Das Komma steht links von den p Tetraden.

Der Konvertierungsrest steht als positiver Festkommabruch (Komma links) im Register D. Er ist mit dem Faktor 10^p multipliziert.

Es gilt die Gleichung
Dezimalbruch = Festkommabruch - Rest $\cdot 10^{-p}$.

Externcode: KFLD

Interncode: '95'

belegt: Rechenwerk

Takte: 9p + 5 bei $\langle A \rangle > 0$
2 bei $\langle A \rangle = 0$

Alarm:

TK-Alarm: wenn $\langle A \rangle_t$ = 2 oder 3

BÜ-Alarm: wenn $\langle A \rangle \leq -0$
wenn $\langle A \rangle_1 \neq \langle A \rangle_2$
wenn p $\neq$ 1...13

Sonstiges:

bei falscher Typenkennung:
$\langle A \rangle_t$ = 2 oder 3

TK-Alarm
2 Takte

bei $\langle A \rangle \leq -0$ oder $\langle A \rangle_1 \neq \langle A \rangle_2$:
BÜ-Alarm
2 Takte

bei p nicht gleich 1 bis 13:

$\langle A \rangle = \langle Q \rangle = \langle D \rangle$:= undefiniert
$\langle A \rangle_t = \langle Q \rangle_t = \langle D \rangle_t$:= 1
$\langle Y \rangle$:= +0
BÜ-Alarm
2 Takte

Dezimalcode:
0000 = 0
000L = 1
00L0 = 2
00LL = 3
0L00 = 4
0L0L = 5
0LL0 = 6
0LLL = 7
L000 = 8
L00L = 9

Adressenteil intern:
'--0x'

x = p = 1...13

LA	s	Lösche in A	**LA**

Adressenteil: s = Spezifikation
F, 2, E, 3, V und M: dürfen miteinander kombiniert werden
H oder T: dürfen nur einzeln oder mit M angegeben werden

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

s = F: $\langle A \rangle_{1-40}$:= 0 ≙ Mantissenteil
s = 2: $\langle A \rangle_{1-24}$:= 0 ≙ linkes Halbwort
s = E: $\langle A \rangle_{41-48}$:= 0 ≙ Exponententeil
s = 3: $\langle A \rangle_{33-48}$:= 0 ≙ rechtes Drittel
s = V: $\langle A \rangle_{1-2}$:= 0 ≙ Vorzeichenstellen
s = M: $\langle M \rangle$:= 0 ≙ Markenregister

Vorstehende Spezifikationen können kombiniert verwendet werden.

s = F: $\langle A \rangle_{1-40}$:= 0 ≙ ganzes Wort ohne rechte Oktade
s = H: $\langle A \rangle_{1-42}$:= 0 ≙ ganzes Wort ohne rechte Hexade
s = T: $\langle A \rangle_{1-44}$:= 0 ≙ ganzes Wort ohne rechte Tetrade

Vorstehende Spezifikationen dürfen nur einzeln verwendet werden oder mit der Spezifikation M kombiniert werden.

Es werden im Register A die durch die Spezifikation s bezeichneten Binärstellen gelöscht. Wenn im Adressenteil mehrere Spezifikationen F, 2, E, 3, V und M stehen, werden die bezeichneten Stellen im Register A, bzw. das Register M, gleichzeitig gelöscht.

Die Spezifikationen H und T dürfen nicht gleichzeitig benutzt werden, sie können aber mit M kombiniert werden. Eine Kombination mit den übrigen Spezifikationen ist nicht zugelassen.

Bei nicht zulässigen Spezifikationen erfolgt eine fehlerhafte Befehlsausführung.

Externcode: LA

Interncode: '8B'

belegt: Rechenwerk

Takte: 4

Alarm:

Sonstiges:

Adressenteil intern:

F = '--80'
2 = '--40'
E = '--20'
3 = '--10'
V = '--02'
M = '--01'
H = '--08'
T = '--04'

bei unzulässigen Spezifikationen:

Ergebnis: fehlerhafte Befehlsausführung

keine Spezifikation im Adressenteil:

s = leer

Ergebnis: wie Leerbefehl (NULL-Befehl)

LC	n

Lösche im Speicher

LC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle n \rangle \quad := t_n ; +0$

$\langle n \rangle_m \quad := \langle n \rangle_m$ nur bei $t_n = 0$ oder 1

Die Speicherzelle n wird auf +0 gelöscht. Die Typenkennung bleibt erhalten.

Bei Zahlwörtern (TK = 0 oder 1) bleibt auch die Markenstelle erhalten.

Externcode: LC

Interncode: '33'

belegt: Befehlswerk

Takte: 13

Alarm:

Sonstiges:

LEI	c p

Modifizieren mit Leitadressen

Adressenteil: c = Befehlscode des Zweitbefehls
p = Zahl (0...255)

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung: BEBY = L System- oder Spezialmodus

Ausführung:

op := c
adr := ⟨BL⟩ · 2^8 + p
Speicheradressen beim Zweitbefehl absolut
mod2 := mod2

Es wird ein Zweitbefehl erzeugt mit c als Operationsteil und mit dem Inhalt des Registers BL (Leitadressenregister) mal 2^8, erhöht um den Wert p als Adressenteil.

Das Register BL enthält die linken 16 Bits der Anfangsadresse des aktuellen Leitblocks. Mit dem Befehl LEI kann also der Adressenteil für den Zweitbefehl aus den ersten 256 Halbwörtern des Leitblocks geholt werden.

Holt der Zweitbefehl einen Operanden aus dem Speicher, so gilt für die Adresse, daß sie nicht über die Seitenadressenregister ersetzt wird. Die Adresse ist also eine absolute Adresse.

Es folgt die Ausführung des Zweitbefehls:

Zweitbefehl

c	⟨BL⟩ · 2^8 + p

Externcode: LEI

Internode: 'B5'

belegt: Befehlswerk

Takte: 10

Alarm:

Sonstiges:

bei Normal- oder Abwicklermodus:

BEBY ≠ L

Sprung ins Nanoprogramm "Makro"
Beschreibung siehe unter "Makro"

LMC	n

Lösche Marke im Speicher

LMC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle n \rangle_t$ = 0 oder 1

Ausführung:

$\langle n \rangle_m$:= 0

Das Markenbit in der Speicherzelle n wird auf 0 gesetzt. Sonst wird der Inhalt einschließlich Typenkennung nicht verändert.

Externcode: LMC

Interncode: '31'

belegt: Befehlswerk

Takte: 13

Alarm:

TK-Alarm: wenn $\langle n \rangle_t$ = 2 oder 3

Sonstiges:

bei falscher Typenkennung:

$\langle n \rangle_t$ = 2 oder 3

Ergebnis: wie Leerbefehl (NULL-Befehl)
TK-Alarm
1 Takt

LMT	n

Lösche markiert

LMT

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle n \rangle_t$ = 0 oder 1

Ausführung:

$\langle n \rangle$:= t_n ;+0

$\langle n \rangle_m$:= L

Die Speicherzelle n wird gelöscht und das Markenbit auf L gesetzt. Die Typenkennung = 0 oder 1 bleibt erhalten.

Externcode: LMT

Interncode: '32'

belegt: Befehlswerk

Takte: 13

Alarm:

TK-Alarm: wenn $\langle n \rangle_t$ = 2 oder 3

Sonstiges:

bei falscher Typenkennung:

$\langle n \rangle_t$ = 2 oder 3

Ergebnis: wie Leerbefehl (NULL-Befehl)

TK-Alarm

1 Takt

LR	s

Lösche Register

LR

Adressenteil: s_1 = Typenkennung 0,1,2 oder 3

s_2 = Rechenwerksregister A,Q,D oder H (mehrere in beliebiger Folge)

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle s_2 \rangle := s_1 ; +0$

Das bzw. die mit s_2 angegebenen Register werden auf Null gelöscht.
Das Register erhält die mit s_1 angegebene Typenkennung.

Externcode: LR

Interncode: '9A'

belegt: Rechenwerk

Takte: 3

Alarm:

Sonstiges:

Adressenteil intern:

s_1 : 0 = '--00'
1 = '--40'
2 = '--80'
3 = '--C0'

s_2 : A = '--20'
Q = '--10'
D = '--08'
H = '--04'

Es ist kein Register durch s_2 adressiert:

s_2 = leer

Ergebnis: wie Leerbefehl (NULL-Befehl)

LZL	s	Lösche und setze Merklichter	**LZL**

Adressenteil:

s_L = Merklichter, die auf L gesetzt werden sollen

s_R = Merklichter, die auf O gelöscht werden sollen

s = O bedeutet, kein Merklicht
(O muß angegeben werden)

Dezimalziffern O...8
Die Ziffern werden in beliebiger Reihenfolge ohne Leertaste hintereinander geschrieben; zwischen s_L und s_R muß eine Leertaste stehen.

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle K \rangle_{s_R} := O$

$\langle K \rangle_{s_L} := L$

Die durch s_R bezeichneten Merklichter werden auf Null gelöscht.

Anschließend werden die durch s_L bezeichneten Merklichter gesetzt.

Externcode: LZL

Interncode: '10'

belegt: Befehlswerk

Takte: 3

Alarm:

Sonstiges:

Adressenteil intern:

s_1 : 1 = '8000'
2 = '4000'
3 = '2000'
4 = '1000'
5 = '0800'
6 = '0400'
7 = '0200'
8 = '0100'

s_2 : 1 = '0080'
2 = '0040'
3 = '0020'
4 = '0010'
5 = '0008'
6 = '0004'
7 = '0002'
8 = '0001'

M	i

Modifiziere

M

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := ⟨i⟩ + mod2

mod2 := ⟨i⟩ + mod2

Der Befehl erzeugt einen Modifikator 2. Art. Er wird aus dem Inhalt der Indexzelle i gebildet.

Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert.

Der neue Modifikator 2. Art wird gleichzeitig in Register B gebracht.

Externcode: M

Interncode: '08 8-xx' (siehe auch Sonstiges)

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

Adressenteil intern:

'8-xx'

xx = Indexadresse

M2	m	Multipliziere Halbwort	**M2**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$|\langle m\rangle| < 2^{22}$

$\langle m\rangle_1 = \langle m\rangle_v$

$\langle A\rangle_1 = \langle A\rangle_v$

$|\langle A\rangle \cdot \langle m\rangle| < 2^{46}$

Ausführung:

$\langle A\rangle := t_A ; \langle A\rangle \cdot \langle m\rangle$

$\langle Q\rangle := t_A ; +0$

$\langle D\rangle := t_A ; +0$

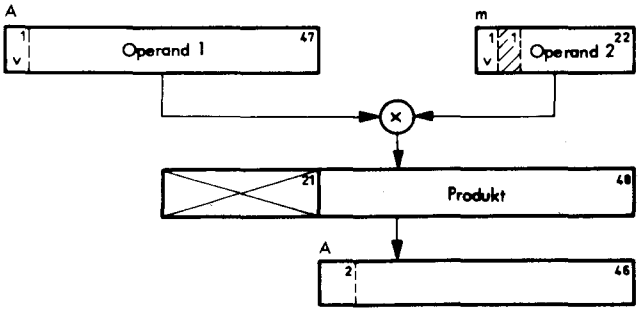
$\langle Y\rangle := +0$

Der Inhalt des Registers A wird mit dem Inhalt der Speicherzelle m multipliziert.

Der Inhalt des Registers A und der Speicherzelle m wird unabhängig von der Typenkennung als Festkommazahl betrachtet.

Das Ergebnis wird in das Register A gebracht; die linken 21 Binärstellen entfallen.

Ist das Ergebnis größer oder gleich $\pm 2^{46}$, so wird BÜ-Alarm gegeben.



Externcode: M2

Interncode: '7A'

belegt: Rechenwerk

Takte: 49.5

Alarm:

BÜ-Alarm: wenn $|\text{Ergebnis}| \geq 2^{46}$

Sonstiges:

bei BÜ-Alarm:

$|\text{Ergebnis}| \geq 2^{46}$

⟨A⟩ := falsches Ergebnis
⟨D⟩ := +0
⟨Q⟩ := +0
⟨Y⟩ := +0
BÜ-Alarm

Voraussetzung nicht erfüllt:

$|\langle m \rangle| \geq 2^{22}$

⟨A⟩ := falsches Ergebnis
⟨D⟩ := +0
⟨Q⟩ := +0
⟨Y⟩ := +0

M2N	m

Multipliziere Halbwort negativ

M2N

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$$|\langle m\rangle| < 2^{22}$$
$$\langle m\rangle_1 = \langle m\rangle_v$$
$$\langle A\rangle_1 = \langle A\rangle_v$$
$$|\langle A\rangle \cdot \langle m\rangle| < 2^{46}$$

Ausführung:

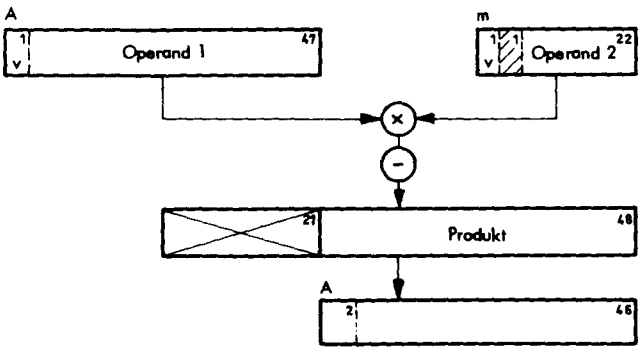
$\langle A\rangle := t_A ; -\langle A\rangle \cdot \langle m\rangle$
$\langle Q\rangle := t_A ; +0$
$\langle D\rangle := t_A ; +0$
$\langle Y\rangle := \quad +0$

Der Inhalt des Registers A wird mit dem Inhalt der Speicherzelle m multipliziert.

Der Inhalt des Registers A und der Speicherzelle m wird unabhängig von der Typenkennung als Festkommazahl betrachtet.

Das Ergebnis wird mit umgekehrtem Vorzeichen in das Register A gebracht; die linken 21 Binärstellen entfallen.

Ist das Ergebnis größer oder gleich $\pm 2^{46}$, so wird BÜ-Alarm gegeben.



Externcode: M2N

Interncode: '78'

belegt: Rechenwerk

Takte: 49,5

Alarm:

BÜ-Alarm: wenn Ergebnis $\geq 2^{46}$

Sonstiges:

Bei BÜ-Alarm:

$|\text{Ergebnis}| \geq 2^{46}$

⟨A⟩ := falsches Ergebnis
⟨D⟩ := +0
⟨Q⟩ := +0
⟨Y⟩ := +0
BÜ-Alarm

Voraussetzung nicht erfüllt:

$|\langle m \rangle| \geq 2^{22}$

⟨A⟩ := falsches Ergebnis
⟨D⟩ := +0
⟨Q⟩ := +0
⟨Y⟩ := +0

M2NR	m

Multipliziere Halbwort negativ mit Rundung **M2NR**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_1 = \langle A \rangle_2$

$|\langle m \rangle| < 2^{22}$

$\langle m \rangle_1 = \langle m \rangle_v$

Ausführung:

$\langle A \rangle := t_A ; -\langle A \rangle \cdot \langle m \rangle$ gerundet

$\langle Q \rangle := t_A ; +0$

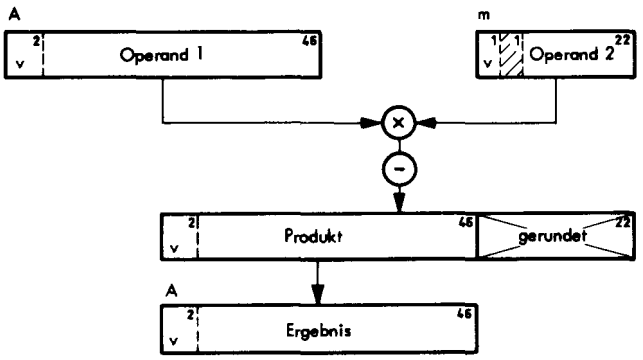
$\langle D \rangle := t_A ; +0$

$\langle Y \rangle := +0$

Der Inhalt des Registers A wird mit dem Inhalt der Speicherzelle m multipliziert.

Das Produkt wird in der 46. Stelle gerundet, die rechten 22 Binärstellen entfallen, das Vorzeichen wird umgekehrt.

Es wird unabhängig von der Typenkennung wie bei Festkommazahlen gearbeitet.



Externcode: M2NR

Interncode: '79'

belegt: Rechenwerk

Takte: 36,5

Alarm:

Sonstiges:

Bei über- oder untergelaufenem Operand:

$\langle A \rangle_1 \neq \langle A \rangle_2$

Ergebnis: siehe unter Ausführung
Das Ergebnis kann über- oder untergelaufen sein; in diesem Fall wird kein BÜ-Alarm gegeben.

Voraussetzung nicht erfüllt:

$|\langle m \rangle| \geq 2^{22}$

Ergebnis: siehe unter Ausführung
$\langle A \rangle$:= undefiniert

M2R	m

Multipliziere Halbwort mit Rundung

M2R

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$$\langle A\rangle_1 = \langle A\rangle_2$$
$$|\langle m\rangle| < 2^{22}$$
$$\langle m\rangle_1 = \langle m\rangle_v$$

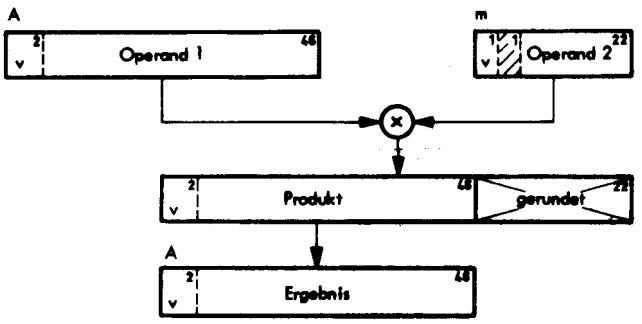
Ausführung:

$\langle A\rangle$:= t_A ; $\langle A\rangle \cdot \langle m\rangle$ gerundet
$\langle Q\rangle$:= t_A ; +0
$\langle D\rangle$:= t_A ; +0
$\langle Y\rangle$:= +0

Der Inhalt des Registers A wird mit dem Inhalt der Speicherzelle m multipliziert. Der Inhalt des Registers A und der Speicherzelle m wird unabhängig von der Typenkennung als Festkommazahl betrachtet.

Das Produkt wird in der 46. Stelle gerundet; die rechten 22 Binärstellen entfallen.

Das Ergebnis steht im Register A.



Externcode: M2R

Interncode: '7B'

belegt: Rechenwerk

Takte: 36,5

Alarm:

Sonstiges:

Bei über- oder untergelaufenem Operand:

$\langle A \rangle_1 \neq \langle A \rangle_2$

Ergebnis: siehe unter Ausführung
Das Ergebnis kann über- oder untergelaufen sein; in diesem Fall wird kein BÜ-Alarm gegeben.

Voraussetzung nicht erfüllt:

$|\langle m \rangle| \geq 2^{22}$

Ergebnis: siehe unter Ausführung
$\langle A \rangle$:= undefiniert

MA	z

Modifiziere mit Adressenteil

MA

Adressenteil: z = Zahl 0...65 535

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := z + mod2
mod2 := z + mod2

Der Befehl erzeugt einen Modifikator 2. Art. Er wird aus der Zahl z gebildet.

Ist vom Vorbefehl noch ein Modifikator 2. Art vorhanden, so wird er hinzuaddiert.

Der neue Modifikator wird gleichzeitig in das Register B gebracht.

Externcode: MA

Interncode: '03'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

MAB	c p	Modifiziere Adressenteil mit B	**MAB**

Adressenteil: c = Befehlscode des Zweitbefehls
p = Zahl ±0...±127

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

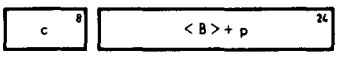
op := c
adr := ⟨B⟩ + p
⟨B⟩ := ⟨B⟩ + p

Es wird ein Zweitbefehl erzeugt mit c als Operationsteil und mit dem Inhalt des Registers B plus der Zahl p als Adressenteil.

Das Ergebnis steht gleichzeitig im Register B.

Es folgt die Ausführung des Zweitbefehls.

Zweitbefehl:



Externcode: MAB

Interncode: '20'

belegt: Befehlswerk

Takte: 9,5 (Mittelwert)

Alarm:

Sonstiges:

MABI	c p

Modifiziere Adressenteil mit B
bei Invarianz der Sprungadresse

MABI

Adressenteil: c = Befehlscode des Zweitbefehls
p = Zahl ±0...±127

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

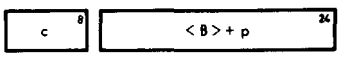
Ausführung:

op := c
adr := ⟨B⟩ + p
⟨B⟩ := ⟨B⟩ + p

Es wird ein Zweitbefehl erzeugt mit c als Operationsteil und mit dem Inhalt des Registers B plus der Zahl p als Adressenteil. Das Ergebnis steht gleichzeitig im Register B.

Es folgt die Ausführung des Zweitbefehls:

Zweitbefehl:



Ist der Zweitbefehl ein Sprungbefehl mit erfüllter Sprungbedingung, so werden alle 24 Bits in das Befehlszähler-Register F übertragen. Mit Hilfe dieser Befehle ist es möglich, von einer Großseite in eine andere zu springen.

In allen anderen Fällen werden bei Sprungbefehlen nur die rechten 16 Bits in das Register F übertragen und die linken 8 Bits bleiben erhalten. Es kann also nur innerhalb einer Großseite (65 535 Befehle) gesprungen werden.

Externcode: MABI

Interncode: '3F'

belegt: Befehlswerk

Takte: 9,5 (Mittelwert)

Alarm:

Sonstiges:

MAN	n

Multipliziere akkumulierend negativ

MAN

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle A\rangle_1 = \langle A\rangle_2$

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_u$

$\langle A,Q\rangle := 1,1;-\langle A\rangle \cdot \langle D\rangle + \langle H,Q\rangle$

$\langle A\rangle_v$ kann $\neq \langle Q\rangle_v$

$\langle Y\rangle := +0$

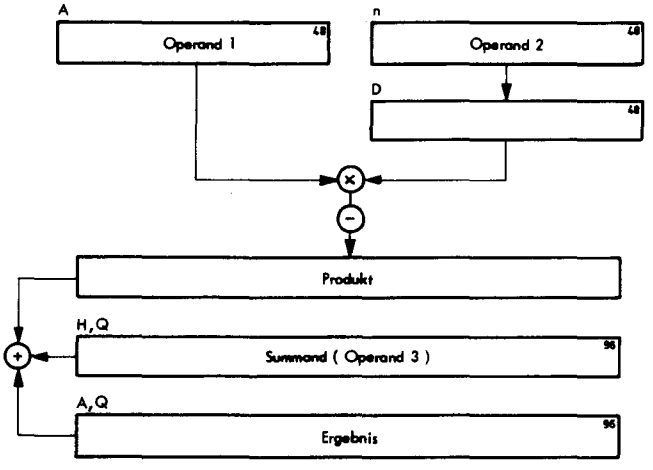
Addition als ob $\langle H\rangle_t = \langle Q\rangle_t = 1$

kein BÜ-Alarm bei: $\langle H\rangle_1 \neq \langle H\rangle_2$ oder $\langle Q\rangle_1 \neq \langle Q\rangle_2$

Beide Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Dieses Produkt erhält umgekehrte Vorzeichen und wird zum Inhalt des doppeltlangen Registers H,Q addiert. Die Addition erfolgt wie bei Festkommazahlen. Der Inhalt der beiden Teile des Registers H oder Q kann über- oder untergelaufen sein; es erfolgt kein BÜ-Alarm. Das Ergebnis steht im doppeltlangen Register A,Q und hat in beiden Teilen des Registers die Typenkennung = 1. Die Vorzeichen der beiden Teile des Registers A,Q brauchen nicht gleich zu sein.

Die Marke wird berücksichtigt.



Externcode: MAN

Interncode: '5A'

belegt: Rechenwerk

Takte: 66

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$
wenn $\langle H\rangle_t \neq 1$ oder $\langle Q\rangle_t \neq 1$

BÜ-Alarm: wenn beim Operand $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$
wenn beim Ergebnis $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_n\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

2 Takte

bei falscher Typenkennung:

$\langle H\rangle_t \neq 1$ oder $\langle Q\rangle_t \neq 1$

Ergebnis: siehe Ausführung

TK-Alarm

bei über- oder untergelaufenem Operand:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_n$

$\langle A\rangle$:= 1; undefiniert

$\langle Q\rangle$:= 1; undefiniert

$\langle Y\rangle$:= +0

evtl. BÜ-Alarm

bei $\langle H\rangle_t \neq 1$ oder $\langle Q\rangle_t \neq 1$:

TK-Alarm

66 Takte

bei über- oder untergelaufenem Ergebnis:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

Ergebnis: siehe Ausführung

BÜ-Alarm

66 Takte

-	-

Nanoprogramm Makro

MAKRO

Adressenteil: -

bei mod2: -	bei R: -

Voraussetzung:

a = Anfangsadresse des Leitblocks

Ausführung:

Warteschleife bis Rechenwerk frei

a := 0 wenn BEBY = L — System- oder Spezialmodus

a := ⟨BL⟩·2^8 wenn BEBY = 0 — Normal- oder Abwicklermodus

⟨a + 18⟩ := 3; ⟨B⟩, ⟨BA⟩
⟨a + 20⟩ := 3; ⟨F⟩, Steuerbits$_{28-48}$
} Sicherstellen von Registern und Steuerbits

⟨BIx⟩ zurückspeichern, wenn ⟨BIx⟩$_2$ = L
⟨BIx⟩$_1$:= 0 Indexregister ungültig setzen
⟨X⟩ := ⟨4⟩$_{1-24}$ absolut
} wenn BEBY = L

Steuerbits$_{25-45}$:= 0 — BEBO bleibt erhalten

BEBY := L wenn BEBY = 0 und BEBN = L
BEBN := 0 wenn BEBY = L und BEBN = L
} Umschaltung Modus:

Normal	→	Spezial
Abwickler	→	Spezial
Spezial	→	System
System	→	System

BEFE := L Eingriffssperre setzen

BEMP := L Sprung

BEMB := L Sprung in andere Großseite ist zugelassen

⟨B⟩$_{1-18}$:= 0
⟨B⟩$_{18-24}$:= Code des Befehls der das Makro angestoßen hat
⟨F⟩ := ⟨6⟩$_{28-48}$ Sprung ins Makroprogramm

Das Nanoprogramm Makro wird durchlaufen:

1. wenn bei den Befehlen HALT, LEI, SW, VMO, VPU, VSS, Y, ZDP eine Bedingung nicht erfüllt ist,
2. beim Befehl KB stets,
3. wenn ein Code erscheint, der nicht durch einen Befehl belegt ist.

Externcode: Makro

Interncode: –

belegt: Befehlswerk

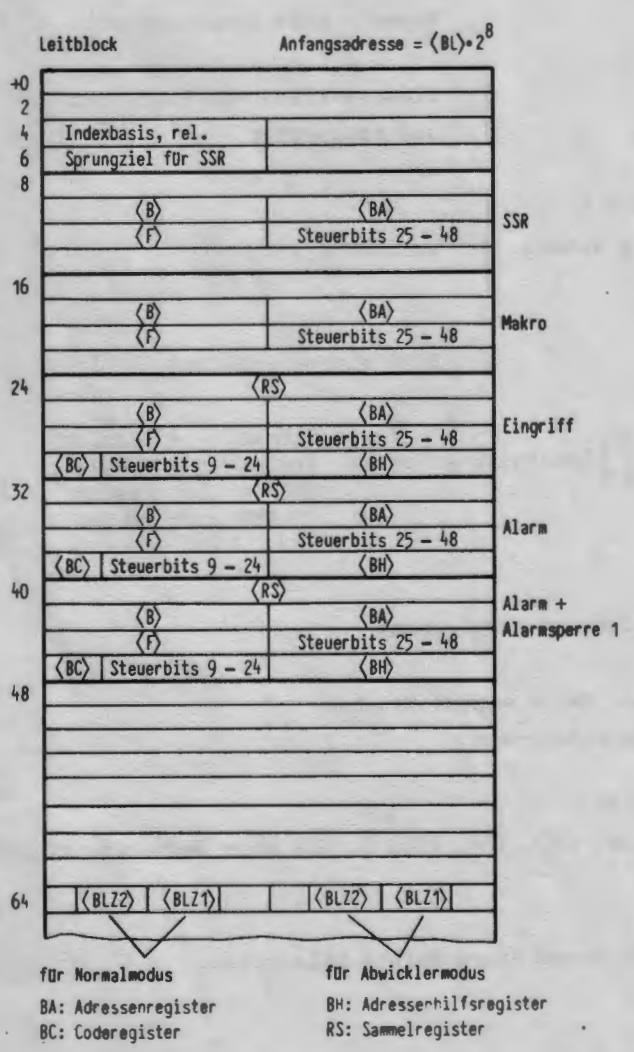
Takte: 28 wenn BEBY = 0

42 + 14 · n wenn BEBY = L

n = Anzahl der zurückzuspeichernden Indexregister BIx

Alarm:

Sonstiges:



Nr.	Name	Bedeutung der Steuerbits
10	BESP	Die Dreierprobe darf nicht ersetzt werden.
11	BE30	Kennzeichnung verschiedener Ansprungstellen aus dem Mikroprogramm der Ausführungsphase
12	BE20	
13	BE10	
14	BEAC	Der Befehl wurde im Abspeicher-Nanoprogramm unterbrochen.
15	BEAA	Der Befehl wurde am Anfang der Abrufphase unterbrochen.
16	BEAB	Der Befehl wurde in der Abrufphase bei Auslesen eines Operanden unterbrochen.
17	BEIC	Der Befehl wurde bei Bildung der absoluten Indexbasis unterbrochen.
18	BEMQ	Der Befehl ist nicht zu Ende
19	BEEH	Hauptalarm (Stromausfall bzw. -Abschaltung)
20	BEER4	Rechneralarm vom 4. Rechnerkern
21	BEER3	Rechneralarm vom 3. Rechnerkern
22	BEER2	Rechneralarm vom 2. Rechnerkern
23	BEER1	Rechneralarm vom 1. Rechnerkern
24	BEFT	Technischer Fehler
25	BEMA	Der Befehl MF, MCF oder MD geht vorher
26	BEMO	Der Befehl MFU oder MCFU geht vorher
27	BEMN	Der vorhergehende Befehl definiert mod2
28	BEMM	Der Befehl MM geht vorher (im Modus 16)
29	BEMU	Der Befehl MU geht vorher
30	BEMB	Der Befehl MABI geht vorher
31	BEML	Der Befehl LEI geht vorher
32	BEMP	Der anstehende Befehl ist ein Sprungbefehl
33	BEBE	Steuerbit: Stop vor Abrufphase
34	BEBA	Steuerbit: Modus 16
35	BEBT	Steuerbit: Wartungsmodus
36	BEBF	Steuerbit: Stop nach Abrufphase
37	REAL	Typenkennungsal-Alarm
38	REBUE	Arithmetischer Alarm
39	BEEC	Speicherschutz-Alarm
40	BEEU	Alarm: Überlauf des Registers U
41	BEEK	Befehls-Alarm
42	BEEF	Stop-Alarm
43	BEFE	Eingriffssperre
44	BEEW	Wecker-Alarm
45	BEED	Dreierproben-Alarm
46	BEBO	Abwicklermodus
47	BEBN	Normalmodus
48	BEBY	Systemmodus

MANR	n

Multipliziere akkumulierend
negativ mit Rundung

MANR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle H\rangle_t = \langle n\rangle_t = 1$
$\langle A\rangle_1 = \langle A\rangle_2$

Ausführung:

$\langle D\rangle := t_B ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

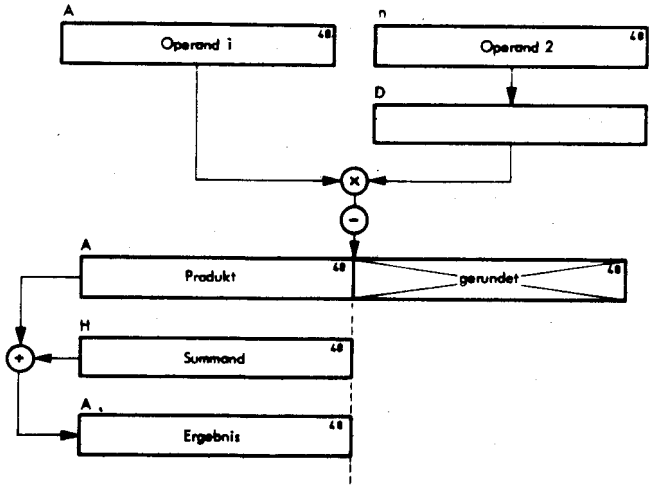
$\langle A,Q\rangle := \langle A\rangle \cdot \langle D\rangle$
$\langle A\rangle := 1;-\langle A,Q\rangle$ gerundet $+ \langle H\rangle$
$\langle Q\rangle := 1;+0$
$\langle Y\rangle := +0$

Alle Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Produkt im doppeltlangen Register A,Q erhält umgekehrte Vorzeichen; die rechten 48 Bits entfallen durch Rundung. Zu diesem gerundeten Produkt wird der Inhalt des Registers H addiert.

Das Ergebnis steht dann mit Typenkennung = 1 im Register A. Der Inhalt des Registers H bleibt erhalten.

Die Marke wird berücksichtigt.



Externcode: MANR

Interncode: '5B'

belegt: Rechenwerk

Takte: 67

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$
wenn $\langle H\rangle_t \neq 1$

BÜ-Alarm: wenn beim Operand $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$
wenn beim Ergebnis $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_n\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

2 Takte

bei falscher Typenkennung:

$\langle H\rangle_t \neq 1$

Ergebnis: siehe Ausführung

TK-Alarm

67 Takte

bei über- oder untergelaufenem Operand:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

Ergebnis: siehe Ausführung

$\langle A\rangle := 1$; undefiniert

evtl. BÜ-Alarm

bei $\langle H\rangle_t \neq 1$

TK-Alarm

67 Takte

bei über- oder untergelaufenem Ergebnis:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

Ergebnis: siehe Ausführung

67 Takte

MAR	n

Multipliziere akkumulierend mit Rundung

MAR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle H\rangle_t = \langle n\rangle_t = 1$

$\langle A\rangle_1 = \langle A\rangle_2$

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A,Q\rangle := \langle A\rangle \cdot \langle D\rangle$

$\langle A\rangle := 1;\langle A,Q\rangle$ gerundet $+ \langle H\rangle$

$\langle Q\rangle := 1;+0$

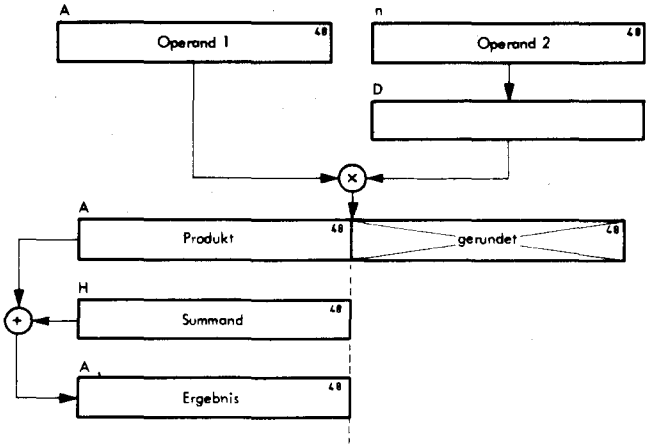
$\langle Y\rangle := +0$

Alle Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Vom Produkt im doppeltlangen Register A,Q entfallen die rechten 48 Bits durch Rundung. Zu diesem gerundeten Produkt wird der Inhalt des Registers H addiert.

Das Ergebnis steht dann mit Typenkennung = 1 im Register A. Der Inhalt des Registers H bleibt erhalten.

Die Marke wird berücksichtigt.



Externcode: MAR

Interncode: '57'

belegt: Rechenwerk

Takte: 67

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$
wenn $\langle H\rangle_t \neq 1$

BÜ-Alarm: wenn beim Operand $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$
wenn beim Ergebnis $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$

Sonstiges:

Bei falscher Typenkennung:
$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$
$\langle D\rangle := t_n ; \langle n\rangle$
$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$
} nur bei $t_n = 0$
TK-Alarm
2 Takte

bei falscher Typenkennung:
$\langle H\rangle_t \neq 1$
Ergebnis: siehe Ausführung
TK-Alarm
67 Takte

bei über- oder untergelaufenem Operand:
$\langle A\rangle_1 \neq \langle A\rangle_2$
$\langle A\rangle_t = \langle n\rangle_t = 1$
Ergebnis: siehe Ausführung
$\langle A\rangle$:= 1;undefiniert
evtl. BÜ-Alarm
bei $\langle H\rangle_t \neq$
TK-Alarm
67 Takte

bei über- oder untergelaufenem Ergebnis:
$\langle A\rangle_1 \neq \langle A\rangle_2$
$\langle A\rangle_t = \langle n\rangle_t = 1$
Ergebnis: siehe Ausführung
67 Takte

MC	m	Modifiziere aus Speicher	MC

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird speziell modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := ⟨m⟩
mod2 := ⟨m⟩ + mod2

Der Befehl erzeugt einen Modifikator 2. Art. Er wird aus dem Inhalt der Speicherzelle m plus dem vorhandenen Modifikator 2. Art gebildet und steht gleichzeitig im Register B.

Externcode: MC

Interncode: '14'

belegt: Befehlswerk

Takte: 2

Alarm:

Sonstiges:

MCE	m	Modifiziere aus Speicher nach Ersetzungen	MCE

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird speziell modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle\langle \ldots \langle m \rangle \ldots \rangle\rangle$
bis $\langle m \rangle_1 = L$

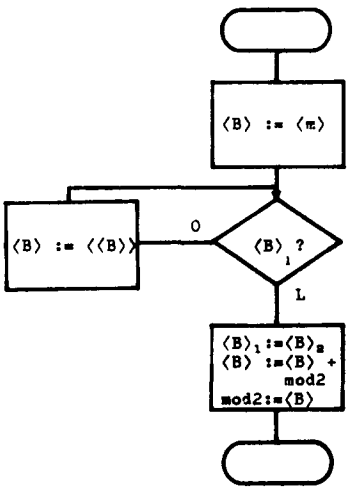
$\langle B \rangle_1 := \langle B \rangle_2$
$\langle B \rangle := \langle B \rangle$ + mod2
mod2 := $\langle B \rangle$

Dieser Befehl erzeugt einen Modifikator durch eine Reihe von Ersetzungen.

Die Ersetzungskette $\langle\langle \ldots \langle m \rangle \ldots \rangle\rangle$ wird abgebrochen, wenn ein Halbwort gefunden wird, dessen 1. Bit = L ist ($\langle m \rangle_1 = L$).

Ist dieses Halbwort gefunden, wird es in das Register B gebracht und das 1. Bit dem 2. Bit des Registers angeglichen. Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert.

Das Ergebnis im Register B ergibt den neuen Modifikator 2. Art.



Externcode: MCE

Interncode: '17'

belegt: Befehlswerk

Takte: 13x + 9 (x = Anzahl der Halbwörter in der Ersetzkette)

Alarm:

Sonstiges:

MCF	m

Modifiziere aus Speicher in jedem Fall (1. Art) **MCF**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird speziell modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

```
mod1 := ⟨m⟩ + mod2
⟨B⟩  := ⟨m⟩ + mod2
mod2 :=  0
```

Der Befehl erzeugt einen Modifikator 1. Art. Er wird aus dem Inhalt der Speicherzelle m gebildet.

Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert und anschließend gelöscht.

Gleichzeitig steht der so gebildetete Modifikator 1. Art im Register B.

Externcode: MCF

Interncode: '16'

belegt: Befehlswerk

Takte: 2

Alarm:

Sonstiges:

MCFU	m

Modifiziere aus Speicher in jedem Fall mit unverändertem B (1. Art)

MCFU

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird speziell modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

mod1 := ⟨m⟩ + mod2
mod2 := 0

Der Befehl erzeugt einen Modifikator 1. Art. Er wird aus dem Inhalt der Speicherzelle m gebildet.

Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert und anschließend gelöscht.

Externcode: MCFU

Interncode: '3D'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

MD	i_L i_R	Modifiziere doppelt (1. und 2. Art)	**MD**

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

mod1 := $\langle i_R \rangle$ + mod2
mod2 := $\langle i_L \rangle$
$\langle B \rangle$:= $\langle i_L \rangle$

Der Befehl erzeugt einen Modifikator 1. Art als auch 2. Art.

Der Modifikator 1. Art ergibt sich aus dem Inhalt der Speicherzelle i_R. Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert.

Der Modifikator 2. Art ergibt sich aus dem Inhalt der Indexzelle i_L.

Der Inhalt der Indexzelle i_L kommt gleichzeitig ins Register B.

Externcode: MD

Interncode: '09'

belegt: Befehlswerk

Takte: 5

Alarm:

Sonstiges:

MF	i

Modifiziere in jedem Fall (1. Art)

MF

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

```
mod1 := ⟨i⟩ + mod2
⟨B⟩  := ⟨i⟩ + mod2
mod2 :=  0
```

Der Befehl erzeugt einen Modifikator 1. Art. Er wird aus dem Inhalt der Indexzelle i gebildet.

Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert und anschließend gelöscht.

Der Modifikator 1. Art wird gleichzeitig ins Register B gebracht.

Externcode: MF

Interncode: 'OB'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

MFU	i

Modifiziere in jedem Fall mit unverändertem B (1. Art)

MFU

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

mod1 := ⟨i⟩ + mod2
mod2 := 0

Der Befehl erzeugt einen Modifikator 1. Art. Er wird aus dem Inhalt der Indexzelle i gebildet.

Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert und anschließend gelöscht.

Externcode: MFU

Interncode: '08 2-xx' (siehe auch Sonstiges)

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

Adressenteil intern:

'2-xx'

xx = Indexadresse

MH	p i

Modifiziere nach erhöhe (mit Parameter)

MH

Adressenteil: p = Zahl ±0...±127
i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

mod2 := ⟨i⟩ + p
⟨B⟩ := ⟨i⟩ + p
⟨i⟩ := ⟨i⟩ + p

Der Befehl erzeugt einen Modifikator 2. Art. Er wird gebildet, indem zum Inhalt der Indexadresse i die Zahl p addiert wird.

Das Ergebnis steht gleichzeitig im Register B und in der Indexadresse i.

Externcode: MH

Interncode: '2D'

belegt: Befehlswerk

Takte: 10,5

Alarm:

Sonstiges:

MHX	i_L i_R	Modifiziere nach Erhöhung um Indexzelle	**MHX**

Adressenteil: i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$mod2 := \langle i_L \rangle + \langle i_R \rangle$

$\langle B \rangle := \langle i_L \rangle + \langle i_R \rangle$

$\langle i_R \rangle := \langle i_L \rangle + \langle i_R \rangle$

Dieser Befehl erzeugt einen Modifikator 2. Art. Er wird gebildet, indem zum Inhalt der Indexzelle i_L der Inhalt der Indexzelle i_R addiert wird.

Das Ergebnis steht gleichzeitig im Register B und in der Indexzelle i_R.

Externcode: MHX

Interncode: 'OE'

belegt: Befehlswerk

Takte: 14,5

Alarm:

Sonstiges:

ML	n

Multipliziere

ML

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A,Q \rangle := \langle A \rangle \cdot \langle D \rangle$

$\langle A \rangle_t = \langle Q \rangle_t := 1$

$\langle Q \rangle_v := \langle A \rangle_v$

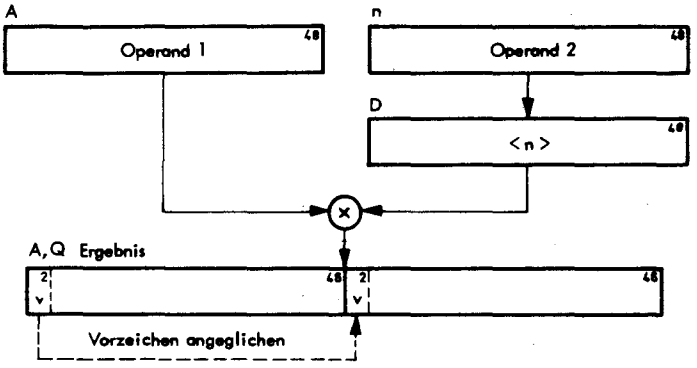
$\langle Y \rangle := +0$

Beide Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Ergebnis steht im doppeltlangen Register A,Q. Die Vorzeichen des Registers Q werden denen des Registers A angeglichen.

Beim Ergebnis haben beide Register die Typenkennung = 1.

Die Marke wird berücksichtigt.



Externcode: ML

Interncode: '54'

belegt: Rechenwerk

Takte: 55

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 1$ oder $\langle n \rangle_t \neq 1$

Sonstiges:

Bei falscher Typenkennung:

$\langle A \rangle_t \neq 1$ oder $\langle n \rangle_t \neq 1$

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_8 \end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

2 Takte

bei über- oder untergelaufenem Operand:

$\langle A \rangle_1 \neq \langle A \rangle_2$

$\langle A \rangle_t = \langle n \rangle_t = 1$

Ergebnis: siehe Ausführung

$\langle A \rangle = \langle Q \rangle := 1,1$; undefiniert

MLA	n

Multipliziere akkumulierend

MLA

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A,Q \rangle := \langle A \rangle \cdot \langle D \rangle + \langle H,Q \rangle$ Addition als ob $\langle H \rangle_t = \langle Q \rangle_t = 1$

$\langle A \rangle_t = \langle Q \rangle_t := 1$

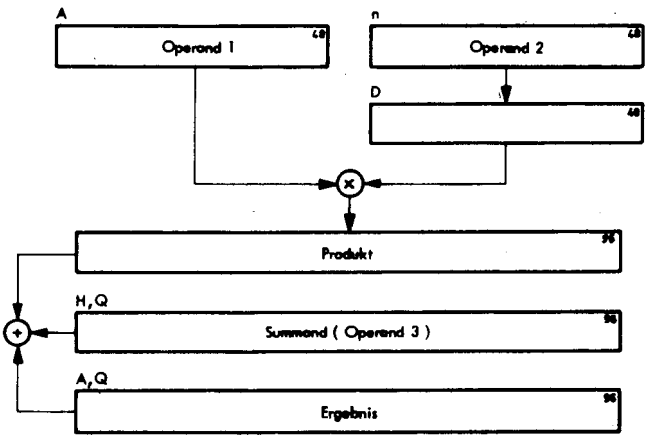
$\langle Q \rangle_v := \langle A \rangle_v$

$\langle Y \rangle := +0$

Alle Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Zu diesem Produkt wird der Inhalt des doppeltlangen Registers H,Q addiert. Die Addition erfolgt stets wie bei Festkommazahlen. Der Inhalt der beiden Teile des Registers H oder Q kann über- oder untergelaufen sein; es erfolgt kein BÜ-Alarm. Das Ergebnis steht im doppeltlangen Register A,Q und hat in beiden Teilen des Registers die Typenkennung = 1. Die Vorzeichen der beiden Teile des Registers A,Q brauchen nicht gleich zu sein.

Die Marke wird berücksichtigt.



Externcode: MLA

Interncode: '56'

belegt: Rechenwerk

Takte: 66

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$
wenn $\langle H\rangle_t \neq 1$ oder $\langle Q\rangle_t \neq 1$

BÜ-Alarm: wenn beim Operand $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$
wenn beim Ergebnis $\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_8\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

2 Takte

bei falscher Typenkennung:

$\langle H\rangle_t \neq 1$ oder $\langle Q\rangle_t \neq 1$

Ergebnis: siehe Ausführung

TK-Alarm

bei über- oder untergelaufenem Operand:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_8$

$\langle A\rangle$:= 1; undefiniert

$\langle Q\rangle$:= 1; undefiniert

$\langle Y\rangle := +0$

<u>evtl.</u> BÜ-Alarm

bei $\langle H\rangle_t \neq 1$ oder $\langle Q\rangle_t \neq 1$:

TK-Alarm
66 Takte

bei über- oder untergelaufenem Ergebnis:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

Ergebnis: siehe Ausführung

BÜ-Alarm
66 Takte

MLD	n

Gleitkomma: Multipliziere auf doppelte Genauigkeit **MLD**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird nicht modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 0$

$\langle A \rangle_1 = \langle A \rangle_2$

$\langle A \rangle$ und $\langle n \rangle$ normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A,Q \rangle := 0,1; (\langle A \rangle \cdot \langle D \rangle)$ normalisiert

$\langle D \rangle := 1; +0$

$\langle H \rangle := 1; +0$

$\langle Y \rangle$:= Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde

$\langle Y \rangle := +0$ wenn Ergebnis ± 0 oder bei Exponentenunterlauf

Beide Operanden müssen die Typenkennung = 0 haben und werden als normalisiert vorausgesetzt. Der Operand im Register A darf nicht über- oder untergelaufen sein.

Die beiden Gleitkommazahlen einfacher Länge werden miteinander multipliziert. Das Ergebnis ist eine normalisierte Gleitkommazahl doppelter Länge. Die Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde, steht im Register Y. Ist das Ergebnis ± 0 oder trat ein Exponentenunterlauf auf, so wird das Register Y auf + 0 gesetzt.

Die Register D und H wurden mit Typenkennung 1 auf + 0 gesetzt.

Die Marke wird berücksichtigt.

Externcode: MLD

Interncode: 'F3'

belegt: Rechenwerk und Befehlswerk

Takte: 64

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 0$
wenn $\langle n \rangle_t \neq 0$

BÜ-Alarm: wenn $\langle A \rangle_1 \neq \langle A \rangle_2$
wenn Exponent des Ergebnisses > +127

Sonstiges:

bei falscher Typenkennung:

$\langle A \rangle_t \neq 0$ oder $\langle n \rangle_t \neq 0$

$\langle D \rangle := 1; +0$
$\langle H \rangle := 1; +0$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$
$\langle Y \rangle := +0$

TK-Alarm

bei über- oder untergelaufenem Operanden:

$\langle A \rangle_1 \neq \langle A \rangle_2$

$\langle D \rangle := 1; +0$
$\langle H \rangle := 1; +0$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$
$\langle Y \rangle := +0$

BÜ-Alarm

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe unter Ausführung
$\langle A,Q \rangle := \langle A,Q \rangle \cdot 16^{-255}$

BÜ-Alarm

bei nicht normalisierten Operanden:

Das Ergebnis ist normalisiert und gerundet. Es kann jedoch von einer Stelle an (bis zu der es korrekt ist) nur noch vorzeichengleiche (statt berechnete) Stellen enthalten. (In diesen Fällen ist auch die Rundung ohne Bedeutung.)

MLN	n

Multipliziere negativ

MLN

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle A\rangle_1 = \langle A\rangle_2$

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A,Q\rangle := -\langle A\rangle \cdot \langle D\rangle$

$\langle A\rangle_t = \langle Q\rangle_t = 1$

$\langle Q\rangle_v := \langle A\rangle_v$

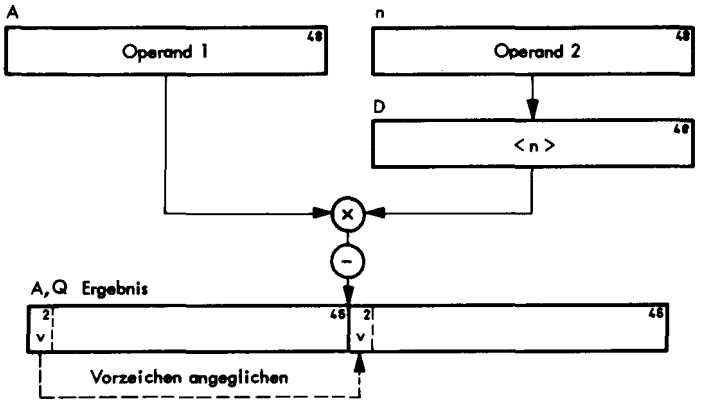
$\langle Y\rangle := +0$

Beide Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Ergebnis im doppeltlangen Register A,Q erhält umgekehrte Vorzeichen. Die Vorzeichen des Registers Q werden denen des Registers A angeglichen.

Beim Ergebnis haben beide Register die Typenkennung = 1.

Die Marke wird berücksichtigt.



Externcode: MLN

Interncode: '58'

belegt: Rechenwerk

Takte: 55

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_s\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

2 Takte

bei über- oder untergelaufenem Operand:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

Ergebnis: siehe Ausführung

$\langle A\rangle = \langle Q\rangle := 1{,}1$; undefiniert

MLR	n

Multipliziere mit Rundung

MLR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle A\rangle_1 = \langle A\rangle_2$

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_s$

$\langle A,Q\rangle := \langle A\rangle \cdot \langle D\rangle$

$\langle A\rangle := 1;\langle A,Q\rangle$ gerundet

$\langle Q\rangle := 1;+0$

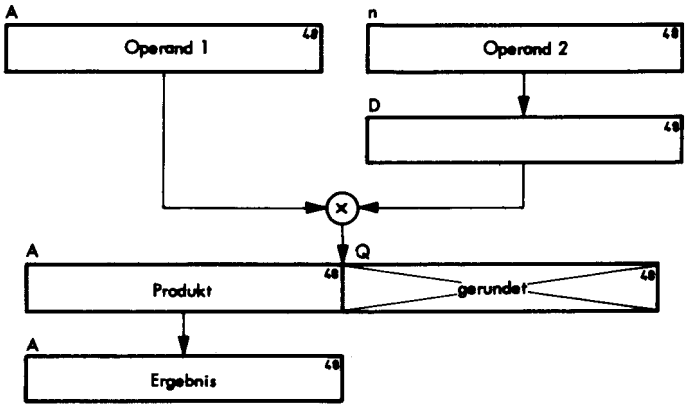
$\langle Y\rangle := +0$

Beide Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Vom Produkt im doppeltlangen Register A,Q entfallen die rechten 48 Bits.

Das Ergebnis steht gerundet mit Typenkennung = 1 im Register A.

Die Marke wird berücksichtigt.



Externcode: MLR

Interncode: '55'

belegt: Rechenwerk

Takte: 57

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_n\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

2 Takte

bei über- oder untergelaufenem Operand:

$\langle A\rangle_1 \neq \langle A\rangle_2$

$\langle A\rangle_t = \langle n\rangle_t = 1$

Ergebnis: siehe Ausführung

$\langle A\rangle := 1$; undefiniert

MNA	z

Modifiziere mit negativem Adressenteil

MNA

Adressenteil: z = Zahl 0...65 535

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := -z + mod2
mod2 := -z + mod2

Der Befehl erzeugt einen Modifikator 2. Art. Er wird aus der negativen Zahl z gebildet.

Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wird hinzuaddiert.

Der neue Modifikator wird gleichzeitig in das Register B gebracht.

Externcode: MNA

Interncode: '02'

belegt: Befehlswerk

Takte: 2

Alarm:

Sonstiges:

MNR	n

Multipliziere negativ mit Rundung

MNR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 1$

$\langle A\rangle_1 = \langle A\rangle_2$

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_n$

$\langle A,Q\rangle := -\langle A\rangle \cdot \langle D\rangle$

$\langle A\rangle := 1 ; \langle A,Q\rangle$ gerundet

$\langle Q\rangle := 1 ; +0$

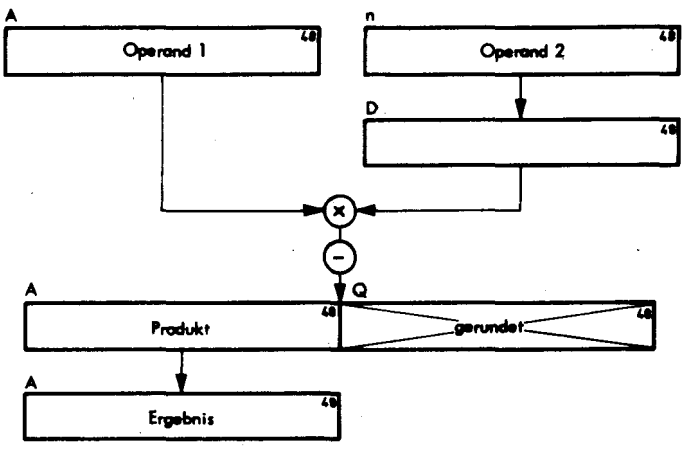
$\langle Y\rangle := +0$

Beide Operanden müssen die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und mit dem Inhalt des Registers A multipliziert. Das Produkt im doppeltlangen Register A,Q erhält umgekehrte Vorzeichen; die rechten 48 Bits entfallen durch Rundung.

Das Ergebnis steht gerundet mit Typenkennung = 1 im Register A.

Die Marke wird berücksichtigt.



Externcode: MNR

Interncode: '59'

belegt: Rechenwerk

Takte: 57

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 1$ oder $\langle n\rangle_t \neq 1$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_n\end{array}\right\}$ nur bei $t_n = 0$

TK-Alarm

2 Takte

bei über- oder untergelaufenem Operand:

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 1$

Ergebnis: siehe Ausführung

$\langle A\rangle := 1$; undefiniert

MRX	s i

Modifiziere mit Register und Indexzelle

MRX

Adressenteil:

s_1 = Register A,Q,D,H oder B
s_2 = leer = positiver Wert
N = negativer Wert
s_3 = leer = nicht zurückspeichern
C = zurückspeichern
i = Indexadresse (0...255)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

s_3 = leer	s_3 = C

Ausführung:

mod2 := ⟨i⟩ ± ⟨s_1⟩	mod2 := ⟨i⟩ ± ⟨s_1⟩
⟨B⟩ := ⟨i⟩ ± ⟨s_1⟩	⟨B⟩ := ⟨i⟩ ± ⟨s_1⟩
	⟨i⟩ := ⟨i⟩ ± ⟨s_1⟩

bei s_1 = A,Q,D,H wird $\langle s_1 \rangle_{25-48}$ addiert

Dieser Befehl erzeugt einen Modifikator 2. Art. Er wird gebildet, indem der Inhalt des Registers s_1 zum Inhalt der Indexzelle i addiert (bei s_2 = N subtrahiert) wird. Bei den Rechenwerksregistern wird die rechte Hälfte (24 Binärstellen) addiert bzw. subtrahiert. Der Modifikator 2. Art wird gleichzeitig ins Register B gebracht.

Ist s_3 = C, so wird die Summe außerdem in die Indexzelle i zurückgespeichert.

Externcode: MRX

Interncode: '8D' im Adressenteil muß das 15. Bit gesetzt sein (siehe auch Sonstiges)

belegt: Befehlswerk

Takte: 12

Alarm:

Sonstiges:

Adressenteil intern:

s_1 : A = '8000'
Q = '4000'
D = '2000'
H = '1000'
B = '0800'

s_2 : N = '4000'

s_3 : C = '0100'

i : = '02xx' ('02' muß gesetzt sein)
xx = Indexadresse

MU	c p	Modifiziere mit U	**MU**

Adressenteil: c = Befehlscode des Zweitbefehls
p = Zahl 0...±127

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

op := c
adr := ⟨⟨U⟩⟩ + p
⟨U⟩ := ⟨U⟩ - 1 nur wenn c ein Sprungbefehl und die Sprungbedingung erfüllt ist.
Zweitcode: siehe auch unter Sonstiges

Rücksprung in andere Großseite möglich

Dieser Befehl wird in Unterprogrammen verwendet, um Operanden zu holen, die in übergeordneten Programmen in der Nähe des Befehls SU (Unterprogrammsprung) stehen. ⟨⟨U⟩⟩ ist die Rücksprungadresse im übergeordneten Programm.

Mit diesem Befehl kann von einem Unterprogramm in das übergeordnete Programm zurückgesprungen werden. Es wird dann für c ein Sprungbefehl eingesetzt (nur sinnvoll bei Sprungbefehlen, die eine Halbwortadresse (m) haben). Beim Rücksprung wird das Unterprogrammregister um 1 erniedrigt. Es kann auch in eine andere Großseite gesprungen werden.

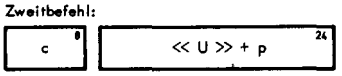
Es wird ein Zweitbefehl erzeugt mit c als Befehlscode und mit der Rücksprungadresse als Adressenteil. Zur Rücksprungadresse wird noch die Zahl p addiert (bzw. subtrahiert bei negativem p).

Ist der Zweitbefehl ein Sprungbefehl mit erfüllter Sprungbedingung, so werden alle 24 Bits in das Befehlszähler-Register F übertragen. Mit Hilfe dieser Befehle ist es möglich, von einer Großseite in eine andere zu springen.

In allen anderen Fällen werden bei Sprungbefehlen nur die rechten 16 Bits in das Register F übertragen und die linken 8 Bits bleiben erhalten. Es kann also nur innerhalb einer Großseite (65 535 Befehle) gesprungen werden.

Es folgt die Ausführung des Zweitbefehls.

Zweitbefehl:



Externcode: MU

Interncode: '05'

belegt: Befehlswerk

Takte: 14,5 (Mittelwert)

Alarm:

Sonstiges:

Zweitcode gleich R:

wenn R als Zweitcode angegeben wurde und durch R ein Zweitbefehl entsteht mit dem Code SE oder SUE, so wird das Register U um 1 erniedrigt.

NL	s

Negiere Merklichter

NL

Adressenteil: s = Merklichter, die invertiert werden sollen (Dezimalziffern 0...8 in beliebiger Reihenfolge)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle K \rangle_s := \langle K \rangle_s$ invertiert

Die mit s angegebenen Merklichter werden invertiert, d.h.
die nicht gesetzten Merklichter werden gesetzt (0 wird L)
die gesetzten Merklichter werden gelöscht (L wird 0)

Externcode: NL

Interncode: '12'

belegt: Befehlswerk

Takte: 3

Alarm:

Sonstiges:

Adressenteil intern:

s: 1 = '--80'
2 = '--40'
3 = '--20'
4 = '--10'
5 = '--08'
6 = '--04'
7 = '--02'
8 = '--01'

NRM	s

Normalisiere

Adressenteil:

s: Spezifikation
G = Gleitkommanormalisierung
andere Spezifikationen: FG auf NRM/2
F,F4 auf NRM/3
N,L auf NRM/4

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = 0$

Ausführung:

⟨A⟩ := ⟨A⟩ normalisiert
⟨Y⟩ := Anzahl der Normalisierungsschritte (Einerschritte)
⟨Y⟩ := 0 wenn ⟨A⟩ bereits normalisiert oder
wenn ⟨A⟩ = +0

Der Inhalt des Registers A wird normalisiert und die Anzahl der Normalisierungsschritte (Schritte um 1 Bit) werden in das Register Y gebracht.

Ist der Inhalt des Registers gleich Null, so entsteht die Gleitkommanull und das Register Y enthält Null.

Externcode: NRM/1

Interncode: '9F'

belegt: Rechenwerk

Takte: 2t + 3

t = Anzahl der Normalisierungsschritte

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 0$

Sonstiges:

bei falscher Typenkennung:

$\langle Y \rangle$:= +0

TK-Alarm

Adressenteil intern:

G = '--8-'

NRM	s

Normalisiere

Adressenteil:

s = Spezifikation
FG = Festkomma in Gleitkomma umwandeln

andere Spezifikationen: G auf NRM/1
F,F4 auf NRM/3
N,L auf NRM/4

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle Q \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2 = \langle Q \rangle_1 = \langle Q \rangle_2$

$\langle A,Q \rangle$ = Komma links

$\langle A,Q \rangle < 1$

Ausführung:

$\langle A \rangle_{\text{Gleitk.}} := (\langle A,Q \rangle_{\text{Festk.}})$ gerundet *

$\langle A \rangle_t := 0$

$\langle Q \rangle := 1;+0$

$\langle Y \rangle := +0$

Die Inhalte der beiden Register müssen die Typenkennung = 1 haben und die Vorzeichen müssen ebenfalls übereinstimmen.

Die Festkommazahl im doppeltlangen Register A,Q wird in eine Gleitkommazahl verwandelt und gerundet ins Register A gebracht. Der Inhalt des Registers A erhält die Typenkennung = 0. Das Register Q wird mit Typenkennung = 1 auf Null gelöscht.

Ist die Festkommazahl Null, so wird sie in die Gleitkommanull umgewandelt.

* Das Komma der Festkommazahl steht hinter den Vorzeichenstellen; steht das Komma an einer anderen Stelle, so muß zusätzlich eine Kommarechnung durchgeführt werden.

Externcode: NRM (Blatt 2)

Interncode: '9F'

belegt: Rechenwerk

Takte: 2t + 10

t = Anzahl der Tetraden, um die normalisiert wird

Alarm:

TK-Alarm: wenn $\langle A \rangle_t$ oder $\langle Q \rangle_t \neq 1$

BÜ-Alarm: wenn $\langle A \rangle_1 \neq \langle Q \rangle_1$

Sonstiges:

bei falscher Typenkennung:

$\langle A \rangle_t$ oder $\langle Q \rangle_t \neq 1$

$\langle Y \rangle$:= +0

TK-Alarm

bei ungleichen Vorzeichen:

$\langle A \rangle_1 \neq \langle Q \rangle_1$

$\langle Y \rangle$:= +0

BÜ-Alarm

bei ungleichen Vorzeichen im Register Q:

$\langle Q \rangle_1 \neq \langle Q \rangle_2$

Ergebnis: siehe unter Ausführung

$\langle A \rangle$:= undefiniert

bei ungleichen Vorzeichen im Register A:

$\langle A \rangle_1 \neq \langle A \rangle_2$

$\langle A \rangle$:= $\langle A \rangle$ gerundet, abhängig von $\langle A \rangle_{41}$

$\langle A \rangle$:= $\langle A \rangle$ um 4 Stellen nach rechts geschiftet
die 4 rechten Stellen gehen verloren, wenn
durch die Rundung $\langle A \rangle_1$ verändert wird.
Das Ergebnis ist falsch.

$\langle Y \rangle$:= +1

Adressenteil intern:

'--CO' = FG

NRM	s

Normalisiere

Adressenteil:

s = Spezifikation
F = Zählung der Einzelschifte
F4 = Zählung der Viererschifte

andere Spezifikationen: G auf NRM/1
F,G auf NRM/2
N,L auf NRM/4

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle Q \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2 = \langle Q \rangle_1 = \langle Q \rangle_2$

$\langle A,Q \rangle \neq 0$

$\langle A,Q \rangle \neq$ normalisiert

Ausführung:

$\langle A,Q \rangle := \langle A,Q \rangle$ um jeweils eine Tetrade (4 Binärstellen) nach links geschiftet bis eines der Bits 2 bis 6 im Register A nicht gleich dem 1. Bit ist.
$\langle Q \rangle_{1,2}$ werden umschiftet

$\langle Y \rangle$:= Anzahl der Viererschifte bei s = F4
Anzahl der Einerschifte bei s = F

Falls die linken 6 Bits im Register A alle gleich sind, wird der Inhalt des doppeltlangen Registers um jeweils 4 Stellen nach links geschiftet, bis sie nicht mehr alle gleich sind. Es werden vorzeichengleiche Stellen nachgezogen.

Die Anzahl der Viererschifte (bei s = F4) bzw. die Anzahl der Einerschifte (bei s = F) wird in das Register Y gebracht.

Die Vorzeichenstellen vom Register Q werden umschiftet.

Externcode: NRM/3

Interncode: '9F'

belegt: Rechenwerk

Takte: 2t + 7
t = Anzahl der Tetradenschifte

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 1$
wenn $\langle Q \rangle_t \neq 1$

BÜ-Alarm: wenn $\langle A \rangle_1 \neq \langle Q \rangle_1$

Sonstiges:

bei falscher Typenkennung:
$\langle A \rangle_t$ oder $\langle Q \rangle_t \neq 1$

$\langle Y \rangle$:= +0
TK-Alarm

bei ungleichen Vorzeichen:
$\langle A \rangle_1 \neq \langle Q \rangle_1$

$\langle Y \rangle$:= +0
BÜ-Alarm

bei über- oder untergelaufenem Zahlwort im Register A:
$\langle A \rangle_1 \neq \langle A \rangle_2$

$\langle Q \rangle_2$:= $\langle Q \rangle_1$
$\langle Y \rangle$:= +0

bei über- oder untergelaufenem Zahlwort im Register Q:
$\langle Q \rangle_1 \neq \langle Q \rangle_2$

$\langle Q \rangle_2$:= $\langle Q \rangle_1$
falsches Ergebnis, da $\langle Q \rangle_2$ umschiftet wird

bei Operand in A,Q gleich Null:
$\langle A,Q \rangle$ = 0

$\langle Q \rangle_2$:= $\langle Q \rangle_1$
$\langle Y \rangle$:= +0

bei Operand bereits normalisiert:
$\langle Q \rangle_2$:= $\langle Q \rangle_1$

$\langle Y \rangle$:= +0

NRM	s

Normalisiere

Adressenteil:

s = Spezifikation
N = führende "O"-Bits zählen
L = führende "L"-Bits zählen

andere Spezifikationen: G auf NRM/1
FG auf NRM/2
F,F4 auf NRM/3

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨A⟩ := ⟨A⟩ nach links geschiftet bis
$\langle A \rangle_1$ = L bei s = N
$\langle A \rangle_1$ = O bei s = L
Nullen werden nachgezogen

⟨Y⟩ := Anzahl der Schiftschritte (Einerschifte)
⟨Y⟩ ≤ 48

Der Inhalt des Registers A wird sooft um eine Binärstelle nach links geschiftet, bis das 1. Bit im Register A L ist (bei s = N) bzw. bis es Null ist (bei A = L).

Die Anzahl der Schiftschritte wird im Register Y gezählt; es sind maximal 48 Schiftschritte möglich.

Bei der Spezifikation N werden also alle links stehenden "O"-Bits (führende Nullen) gezählt und bei der Spezifikation L alle links stehen "L"-Bits (führendes L) gezählt.

Die links herausgeschifteten Bits gehen verloren, rechts werden Nullen nachgezogen.

Externcode: NRM/4

Interncode: '9F'

belegt: Rechenwerk

Takte: 2e + 3

e = Anzahl der Schiftschritte (Einerschifte)

Alarm:

Sonstiges:

Adressenteil intern:

N = '--0-'

L = '--2-'

NULL	z

NULL-Befehl

NULL

Adressenteil: z = ohne Bedeutung, muß aber angegeben werden

bei mod2:	wird nicht modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

Keine Wirkung

Es wird lediglich - wie bei allen Befehlen - der Befehlszähler erhöht.

Externcode: NULL

Interncode: '00'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

PDP	n

Prüfe Dreierprobe und springe wenn richtig

PDP

Adressenteil: n: Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

DP = Dreierprobenbits von ⟨n⟩
(Bits 49 und 50)

Ausführung:

⟨A⟩ := t_n ; ⟨n⟩

⟨Y⟩ := $\langle n \rangle_{5-6}$, OL, DP, $\langle n \rangle_t$

$\langle F \rangle_{9-24}$:= ⟨F⟩+2 wenn DP richtig
(wenn DP falsch: kein Dreierprobenalarm)

Der Inhalt der Speicherzelle n wird unverändert in das Register A gebracht. Gleichzeitig werden in das Register Y neben anderem die Dreierprobenbits (DP) und die Typenkennung vom Inhalt der Speicherzelle n gebracht.

Es wird geprüft, ob der Inhalt der Speicherzlle n die richtigen Dreierprobenbits (Prüfbits) hat. Die Dreierprobenbits ergänzen die 48 Informationsstellen des Wortes so, daß der binäre Wert der 50 Binärstellen (48 Informationsstellen und 2 DP-Stellen) durch drei ohne Rest teilbar ist (Modulo-3-Prüfung). Dem rechten Bit wird die Wertigkeit 2^0 und dem linken der Wert 2^{49} zugeordnet.

Ist die Dreierprobe richtig, so wird auf den übernächsten Befehl gesprungen. Bei falscher Dreierprobe erfolgt kein Dreierprobenalarm; der auf PDP folgende Befehl kommt zur Ausführung.

Externcode: PDP

Interncode: 'B9'

belegt: Befehlswerk und Rechenwerk

Takte: 21

Alarm:

Sonstiges:

QBR	n

Bequemes Bringen aller Register

QBR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

n = Endadresse des Speicherbereiches

Ausführung:

$\langle T \rangle := \langle n \rangle_{1-24}$ T: Prüfregister

$\langle H \rangle := t_{n-2}; \langle n - 2 \rangle$

$\langle D \rangle := t_{n-4}; \langle n - 4 \rangle$

$\langle Q \rangle := t_{n-6}; \langle n - 6 \rangle$

$\langle A \rangle := t_{n-8}; \langle n - 8 \rangle$

$\langle M \rangle := L$, falls $\langle n - 10 \rangle_t = 1$ oder 3

$\langle M \rangle := 0$, falls $\langle n - 10 \rangle_t = 0$ oder 2

$\langle B \rangle := \langle n - 10 \rangle_{1-24}$

$\langle K \rangle := \langle n - 10 \rangle_{25-32}$

$\langle Y \rangle := \langle n - 10 \rangle_{33-40}$

$\langle U \rangle := \langle n - 10 \rangle_{41-48}$

Die mit dem Befehl QCR abgespeicherten Registerinhalte werden wieder in die betreffenden Register zurückgebracht.

Die Registerinhalte werden unabhägig von der Typenkennung unverändert transportiert.

Die Adresse beim Befehl QBR muß um 10 größer sein als sie beim Befehl QCR war.

Externcode: QBR

Interncode: 'F4'

belegt: Befehlswerk und Rechenwerk

Takte: 84

Alarm:

Sonstiges:

QCR	n

Bequemes Speichern aller Register

QCR

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

n = Anfangsadresse des Speicherbereiches

Ausführung:

$\langle n \rangle_t := 2 + \langle M \rangle$

$\langle n \rangle_{1-24} := \langle B \rangle$
$\langle n \rangle_{25-32} := \langle K \rangle$
$\langle n \rangle_{33-40} := \langle Y \rangle$
$\langle n \rangle_{41-48} := \langle U \rangle$
$\langle n + 2 \rangle := t_A ; \langle A \rangle$
$\langle n + 4 \rangle := t_Q ; \langle Q \rangle$
$\langle n + 6 \rangle := t_D ; \langle D \rangle$
$\langle n + 8 \rangle := t_H ; \langle H \rangle$
$\langle n + 10 \rangle := 2 ; \langle T \rangle, 0$ T: Prüfregister

Es werden die angegebenen Registerinhalte in aufeinanderfolgenden Speicherzellen, beginnend bei der Adresse n, abgespeichert. Die Speicherung erfolgt unabhängig von der Typenkennung und unverändert.

Mit dem Befehl QCR kann der Zustand wieder hergestellt werden wie er vor der Anwendung des Befehls QBR war.

Die Adresse beim Befehl QBR muß um 10 größer sein als sie beim Befehl QCR ist.

Externcode: QCR

Interncode: 'FF'

belegt: Befehlswerk und Rechenwerk

Takte: 55

Alarm:

Sonstiges:

R	c s	Registeradressierung	**R**

Adressenteil: c = Befehlscode des Zweitbefehls; zugelassene Codes siehe Rückseite

s: s_1 = Register A,Q,D,H,Y,U,B oder F
s_2 = nur wenn der Zweitcode ein Halbwort adressiert:
L :für linke 24 Bits
leer:für rechte 24 Bits

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨A⟩, ⟨Q⟩, ⟨D⟩ oder ⟨H⟩ nicht über- oder untergelaufen

ro = Registeroperand, wird so aufgefaßt, als ob er im Speicher stünde

Ausführung:

adr := adr + mod2

mod2 := 0

op := c

operand := ro bei s_1 = A,Q,D oder H: ro := t_{s_1};⟨s_1⟩ bei ⟨s_1⟩$_t$ = 0 oder 1 gilt:
ro_1 = ro_m
ro_2 = ro_v

bei s_1 = Y oder U : ro := 1;0,⟨s⟩$_1$

bei s_1 = B : ro := 1;v,⟨B⟩

bei s_1 = F : ro := 1;v,⟨F⟩ + 1

Es wird ein Zweitbefehl erzeugt mit c als Operationscode. Es sind nur die auf der Rückseite aufgeführten Codes zulässig.

Der Operand wird nicht aus dem Speicher geholt, er wird aber so aufgefaßt, als ob er aus dem Speicher geholt würde. Als Operand wird das mit s_1 angegebene Register verwendet.

Bei Zweitcodes, die ein Halbwort adressieren, wird das rechte Halbwort verwendet. Soll das linke Halbwort benutzt werden, so muß für s_2 ein L angegeben werden. (Die Codes, welche Halbwörter adressieren, sind auf der Rückseite mit einem Stern gekennzeichnet.)

Beim Zweitbefehl gilt für den Registeroperanden die gleiche Wirkung als würde er aus der Speicherzelle (n oder m) geholt. Bei Zahlwörtern gilt die erste Binärstelle als Markenstelle; die zweite Binärstelle gilt als Vorzeichenstelle (negative Zahlwörter sind als Ganzwörter stets markiert).

Registeroperanden aus den Registern Y oder U werden auf jeden Fall als Festkommazahlen (TK = 1) aufgefaßt. Die linken Binärstellen werden auf Null gesetzt.

Kommt der Registeroperand aus dem Register B oder F, so wird er auf jeden Fall als ein Festkommazahlwort (TK = 1) aufgefaßt. Die linken Binärstellen werden vorzeichengleich gesetzt. Bei s_1 = F wird der Inhalt des Registers F um 1 erhöht.

Es folgt die Ausführung des Zweitbefehls.

Externcode: R

Interncode: '96'

belegt: Befehlswerk und Rechenwerk

Takte: 4 bei s_1 = A,Q,D,H,Y oder U
6 bei s_1 = B oder F

Alarm:

Sonstiges: zugelassene Codes:

A,A2*,AB,AQ,AT,AU,AUT,
B,B2*,B2V*,B2VN*,B3*,B3V*,BB,BD,BH,BN,BNR,BQ,BQB,BR,BT,BU,
DV,DVD,DVI,
ET,
GA,GAB,GDV,GDVI,GMAN,GML,GMLA,GMLN,GSB,GSBB,GSBD,GSBI,
HBC,
M2*,M2N*,M2NR*,M2R*,MAN,MANR,MAR,MC*,MCE*,MCF*,MCFU*,ML,MLA,MLD,MLN,MLR,MNR,
NULL,
REZ,
SB,SB2*,SBB,SBD,SBI,SBQ,SBT,SBU,SE,SUE,
T*,TCB,
VBC,VEL,
ZUS

* Diese Codes adressieren Halbwörter

Adressenteil intern:

c : 'xx00'

xx = Interncode des Zweitbefehls

s_1 : A = '0080'
Q = '0040'
D = '0020'
H = '0010'
F = '0088'
U = '0048'
B = '0028'
Y = '0018'

s_2 : leer = '0001'
L = '0000'

bei über- oder untergelaufenen Zahlwörtern (in den Registern A,Q,D oder H):
bei übergelaufenem (positivem) Zahlwort ergibt sich ein unmarkiertes negatives Zahlwort,
bei untergelaufenem (negativem) Zahlwort ergibt sich ein markiertes positives Zahlwort.

REZ	n	Bilde reziproken Wert	**REZ**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle n \rangle_t = 0$
$\langle n \rangle \neq 0$ und normalisiert

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$
$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_s$

$\langle D \rangle := 0;+0$
$\langle A \rangle := 0;1:\langle n \rangle$ normalisiert und gerundet
$\langle Q \rangle := 0;+0$
$\langle Y \rangle :=$ Anzahl der Binärstellen, um die das Ergebnis normalisiert wurde
$\langle Y \rangle := +0$ wenn Ergebnis = ± 0 oder bei Exponentenunterlauf

Der Inhalt der Speicherzelle soll normalisiert sein, er darf nicht Null sein und soll die Typenkennung = 0 haben.

Vom Inhalt der Speicherzelle n wird der Reziprokwert gebildet.
Das Ergebnis steht im Register A, ist normalisiert, gerundet und hat die Typenkennung = 0.

Die Register D und Q werden mit Typenkennung = 0 auf Null gelöscht.

Im Register Y steht die Anzahl der Binärstellen, um die das Ergebnis im Register A normalisiert wurde.

Die Marke wird berücksichtigt.

Externcode: REZ

Interncode: '65'

belegt: Rechenwerk

Takte: 213

Alarm:

TK-Alarm: wenn $\langle n \rangle_t \neq 0$

BÜ-Alarm: wenn $\langle n \rangle = \pm 0$
wenn Exponent des Ergebnisses > +127

Sonstiges:

Bei falscher Typenkennung:

$\langle n \rangle_t \neq 0$

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 1$

$\langle Q \rangle := t_A ; \langle Q \rangle$
TK-Alarm
2 Takte

bei Operand gleich Null:

$\langle n \rangle = \pm 0$ und $\langle n \rangle_t = 0$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle D \rangle := 0 ; \langle A \rangle$
$\langle A \rangle := 0 ; +0$
$\langle Q \rangle := 0 ; +0$
$\langle Y \rangle := +0$
BÜ-Alarm
3 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe Ausführung
BÜ-Alarm
213 Takte

RLR	c s	Relativ-Adressierung mit Registerinhalt	**RLR**

Adressenteil:

c = Befehlscode des Zweitbefehls
s = Angabe eines der Register
A,Q,D oder H rechte Hälfte
AL,QL,DL oder HL linke Hälfte
F oder B
} nur eines der Register darf angegeben sein
linkes Bit = Vorzeichen

U oder Y — werden als positive Zahlen ohne Vorzeichen aufgefaßt

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

```
op   :=  c
adr  := ⟨s⟩ + ⟨F⟩          nur wenn s ≠ F
adr  := ⟨F⟩ + ⟨F⟩ + 1      nur wenn s = F
mod2 := mod2
```

Dieser Befehl erzeugt einen Zweitbefehl. Zum Operationsteil dieses Zweitbefehls wird der für c stehende Befehlscode. Der Adressenteil des Zweitbefehls ergibt sich aus der Addition des mit s angegebenen Registers (Registerhälfte) und des Registers F.

Ist das Register F mit s angegeben, so wird außerdem noch eine 1 addiert.

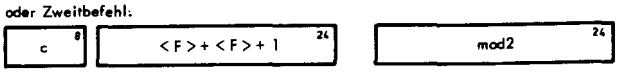
Der Inhalt der Register Y und U wird als ganze, positive und vorzeichenlose Zahl aufgefaßt. In den anderen Fällen ist das linke Bit (Bit 1 des Halbwortes) das Vorzeichen.

Ein vom vorhergehenden Befehl vorhandener Modifikator 2. Art wirkt nicht auf diesen Befehl, sondern auf den Zweitbefehl.

Es folgt die Ausführung des Zweitbefehls.

Zweitbefehl:

oder Zweitbefehl:



Externcode: RLR

Interncode: 'EO'

belegt: Befehlswerk und Rechenwerk

Takte: 10

Alarm:

Sonstiges:

Adressenteil intern:

s: A = 'xx81' AL = 'xx80'
Q = 'xx41' QL = 'xx40'
D = 'xx21' DL = 'xx20'
H = 'xx11' HL = 'xx10'
F = 'xx88'
U = 'xx48'
B = 'xx28'
Y = 'xx18'
xx = Zweitcode

Die Sedezimalen dürfen nicht addiert werden.
Genau eines der Register muß angegeben werden.
(Ausnahme: Falls c = Null, so darf auch s = Null sein)

RT	s

Registertausch

RT

Adressenteil: s_1 = Rechenwerksregister A,Q,D oder H
s_2 = Rechenwerksregister A,Q,D oder H

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$s_1 \neq s_2$

Ausführung:

$\langle s_1 \rangle :=: \langle s_2 \rangle$

Die Inhalte des mit s_1 und s_2 angegebenen Rechenwerksregisters A,Q,D oder H werden einschließlich Typenkennung gegeneinander ausgetauscht.

Externcode: RT

Interncode: '97'

belegt: Rechenwerk

Takte: 4

Alarm:

Sonstiges:

bei $s_1 = s_2$: fehlerhafte Ausführung

Adressenteil intern:

$s = s_1, s_2$: A = '0800'
Q = '0400'
D = '0200'
H = '0100'

RX	s i	Register um Indexzelle	**RX**

Adressenteil:

s_1 = Register A,Q,D,H (rechtes Halbwort)
oder B
s_2 = leer = positiver Wert
N = negativer Wert
s_3 = leer = nicht zurückspeichern
C = wird zurückgespeichert
i = Indexadresse (0...255)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

s_3 = leer	s_3 = C

Ausführung:

$\langle B \rangle := \langle i \rangle \pm \langle s_1 \rangle$	$\langle B \rangle := \langle i \rangle \pm \langle s_1 \rangle$ $\langle i \rangle := \langle i \rangle \pm \langle s_1 \rangle$

Der Inhalt des Registers s_1 wird zum Inhalt der Indexadresse i addiert (bei s_2 = N subtrahiert) und im Register B abgelegt.

Ist s_3 = C, so wird diese Summe außerdem in die Indexzelle i zurückgespeichert.

Bei s_1 = A,Q,D oder H wird das rechte Halbwort verwendet, der Einerrücklauf erfolgt über 24 Stellen.

Externcode: RX

Interncode: '8D' im Adressenteil darf das 15. Bit nicht gesetzt sein (siehe auch Sonstiges)

belegt: Befehlswerk

Takte: 12

Alarm:

Sonstiges:

Adressenteil intern:

s_1 : A = '8000'
Q = '4000'
D = '2000'
H = '1000'
B = '0800'

s_2 : N = '0400'

s_3 : C = '0100'

i : = '00xx'
xx = Indexadresse

'0200' ist nicht erlaubt, sonst Befehl MRX

S	m

Springe

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle F \rangle_{9-24} := m$

Es wird in jedem Fall ein Sprung ausgeführt.
Der nächste Befehl ist der in der Speicherzelle m.

Externcode: S

Interncode: '36'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SAA	m

Springe, wenn arithmetischer Alarm (BÜ-Alarm) **SAA**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

BÜ-Alarm

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24}$:= m
BÜ-Alarm wird gelöscht

Steht ein BÜ-Alarm an, so wird er gelöscht und als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SAA folgende Befehl zur Ausführung.

Mit diesem Befehl kann ein auftretender BÜ-Alarm abgefangen werden.

Externcode: SAA

Interncode: 'A9'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges: Steht ein BÜ-Alarm an, so führt das Fehlen dieses Befehls zum Abbruch beim nächsten Befehl der das Rechenwerk belegt, ausgenommen beim Befehl SAT. Befehle, die nur das Befehlswerk belegen, und der Befehl SAA werden noch ausgeführt.

SAT	m	Springe, wenn Alarm (Typenkennung)	**SAT**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

Typenkennungsalarm

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24}$:= m

TK-Alarm wird gelöscht

Steht ein Typenkennungsalarm an, so wird er gelöscht und als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SAT folgende Befehl zur Ausführung.

Mit diesem Befehl kann ein TK-Alarm abgefangen werden.

Externcode: SAT

Interncode: 'A8'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges: Steht ein TK-Alarm an, so führt das Fehlen dieses Befehls zum Abbruch beim nächsten Befehl der das Rechenwerk belegt, ausgenommen beim Befehl SAA. Befehle, die nur das Befehlswerk belegen, und der Befehl SAA werden noch ausgeführt.

SB	n	Subtrahiere	**SB**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_2 \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle A \rangle := t_{max} ; \langle A \rangle - \langle D \rangle$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und dort vom Inhalt des Registers A subtrahiert. Die Subtraktion erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis im Register A erhält die höhere der beiden Typenkennungen.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: SB

Interncode: '46'

belegt: Rechenwerk

Takte: 8

Alarm:

BÜ-Alarm: wenn beim Ergebnis:

$\langle A \rangle_t = 0$ oder 1 und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:

$\langle A \rangle_t$ oder $\langle n \rangle_t \neq 1$

Subtraktion erfolgt wie bei TK = 1

Ergebnis: siehe Ausführung

bei übergelaufenen Ergebnis:

$\langle A \rangle_1 \neq \langle A \rangle_2$

Ergebnis: siehe Ausführung

BÜ-Alarm

bei über- oder untergelaufenen Operanden:

$\langle A \rangle_1 \neq \langle A \rangle_2$

Wird mit über- oder untergelaufenen Operanden gerechnet, so kann das Ergebnis falsch sein, ohne daß ein BÜ-Alarm erfolgt.

SB2	m

Subtrahiere Halbwort

SB2

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$|\langle A\rangle| < 2^{22}$
$|\langle m\rangle| < 2^{22}$

Ausführung:

$\langle A\rangle_{25-48} := \langle A\rangle_{25-48} + \langle m\rangle$
$\langle A\rangle_{1-24} := \langle A\rangle_{25}$
$\langle A\rangle_{t} := \langle A\rangle_{t}$
$\langle D\rangle := t_A\ ;+0$

Der Inhalt der Speicheradresse m wird von den rechten 24 Bits des Registers A subtrahiert. Die Addition erfolgt unabhängig von der Typenkennung wie bei Festkommazahlen.

Das Ergebnis steht im Register A; die linken 24 Bits werden dem 25. Bit angeglichen.

Das Register D wird auf Null gelöscht und erhält die Typenkennung des Registers A.

Externcode: SB2

Interncode: '7D'

belegt: Rechenwerk

Takte: 14

Alarm:

BÜ-Alarm: wenn beim Ergebnis $\langle A \rangle_{25} \neq \langle A \rangle_{26}$

Sonstiges:

Bei über- oder untergelaufenem Ergebnis:

$\langle A \rangle_{25} \neq \langle A \rangle_{26}$

Ergebnis: siehe unter Ausführung

BÜ-Alarm

Voraussetzung nicht erfüllt:

$|\langle A \rangle| \geq 2^{22}$

$|\langle m \rangle| \geq 2^{22}$

Ergebnis: siehe unter Ausführung

Das Ergebnis kann um $\pm 2^{24} - 1$ falsch sein, ohne daß ein BÜ-Alarm erfolgt.

Der Einerrücklauf erfolgt von der 25. zur 48. Binärstelle.

SBA	z

Subtrahiere Adressenteil

SBA

Adressenteil: z = Zahl bei TK = 0: 0...±127
TK = 1: 0... 65 535
TK = 2: 0... 65 535
TK = 3: 0... 65 535

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = 0: |z| < 2^7$
$\langle A \rangle_t = 1: |z| < 2^{23}$
$\langle A \rangle_t = 2: z < 2^{16}$
$\langle A \rangle_t = 3: |z| < 2^{23}$

Ausführung:

Die Ausführung hängt von $\langle A \rangle_t$ ab

bei $\langle A \rangle_t = 0$: $\langle D \rangle := 1;0,z$
$\langle A \rangle := 0;\langle A \rangle \cdot 16^{-z}$

bei $\langle A \rangle_t = 1$: $\langle D \rangle := 1;v,z$
$\langle A \rangle := 1;\langle A \rangle - v,z$

bei $\langle A \rangle_t = 2$: $\langle D \rangle := 1;0,z$
$\langle A \rangle := 2;\langle A \rangle_{1-32},(\langle A \rangle_{33-46} 16^{-z})$

bei $\langle A \rangle_t = 3$: $\langle D \rangle := 1;v,z$
$\langle A \rangle := 3;\langle A \rangle - v,z$

Die Zahl z wird in das Register D gebracht und gleichzeitig vom Inhalt des Registers A subtrahiert.

Bei Gleitkommazahlen (Typenkennung = 0) wird die Zahl z vom Exponenten subtrahiert, dies bedeutet Division mit dem Faktor 16^{-z}.

Externcode: SBA

Interncode: '99'

belegt: Rechenwerk

Takte: 13

Alarm:

BÜ-Alarm: wenn bei $\langle A \rangle_t = 0$ beim Ergebnis der Exponent übergelaufen ist
wenn bei $\langle A \rangle_t = 1$ das Ergebnis über- oder untergelaufen ist

---

Sonstiges:

Bei über- oder untergelaufenem Ergebnis:

Ergebnis: siehe Ausführung
BÜ-Alarm
bei $\langle A \rangle_t = 0$: zusätzlich $\langle A \rangle := \langle A \rangle \cdot 16^{-127}$

Einerrücklauf: bei $\langle A \rangle_t = 0$ von $\langle A \rangle_{41}$ nach $\langle A \rangle_{48}$
bei $\langle A \rangle_t = 1$ von $\langle A \rangle_{1}$ nach $\langle A \rangle_{48}$
bei $\langle A \rangle_t = 2$ von $\langle A \rangle_{32}$ nach $\langle A \rangle_{48}$
bei $\langle A \rangle_t = 3$ von $\langle A \rangle_{1}$ nach $\langle A \rangle_{48}$

Voraussetzung nicht erfüllt: das Ergebnis wird falsch

Das Ergebnis wird deshalb falsch, weil nach der Modifizierung von z nur folgende Bits verwendet werden:

bei $\langle A \rangle_t = 0$: rechte 8 Bits; linke 40 Bits = 0
bei $\langle A \rangle_t = 1$: rechte 24 Bits; linke 24 Bits = vorzeichengleich *
bei $\langle A \rangle_t = 2$: rechte 16 Bits; linke 32 Bits = 0
bei $\langle A \rangle_t = 3$: rechte 24 Bits; linke 24 Bits = vorzeichengleich *

* Vorzeichen ist beim auf 24 Bits verlängerten Adressenteil das linke Bit.

SBB	n

Subtrahiere Betrag

SBB

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle A \rangle := t_{max} ; \langle A \rangle - |\langle D \rangle|$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Der Betrag vom Inhalt des Registers D wird vom Inhalt des Registers A subtrahiert. Die Subtraktion erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis im Register A erhält die höhere der beiden Typenkennungen.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: SBB

Interncode: '41'

belegt: Rechenwerk

Takte: 8

Alarm:

BÜ-Alarm: wenn beim Ergebnis:

$\langle A \rangle_t$ = 0 oder 1 und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:

$\langle A \rangle_t$ oder $\langle n \rangle_t \neq 1$

Subtraktion erfolgt wie bei TK = 1

Ergebnis: siehe Ausführung

bei über- oder untergelaufenem Ergebnis:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t$ = 0 oder 1

Ergebnis: siehe Ausführung

BÜ-Alarm

bei über- oder untergelaufenen Operanden:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t$ = 0 oder 1

Wird mit über- oder untergelaufenen Operanden gerechnet, so kann das Ergebnis falsch sein, ohne daß ein BÜ-Alarm erfolgt.

SBC	n

Subtrahiere im Speicher

SBC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle D \rangle := t_{max} ; \langle D \rangle - \langle A \rangle$

$\langle n \rangle := t_0 ; \langle D \rangle$

$\langle n \rangle_m := \langle n \rangle_m$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Vom Inhalt dieses Registers wird der Inhalt des Registers A subtrahiert. Die Subtraktion erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis im Register D erhält die höhere der beiden Typenkennungen und wird in die Speicherzelle n abgespeichert. Die ursprüngliche Marke in der Speicherzelle bleibt erhalten.

Der Inhalt des Registers A bleibt unverändert.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: SBC

Interncode: '47'

belegt: Befehlswerk und Rechenwerk

Takte: 17

Alarm:

BÜ-Alarm: wenn beim Ergebnis:

$\langle D \rangle_t = 0$ oder 1 und $\langle D \rangle_1 \neq \langle D \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:

$\langle A \rangle_t$ oder $\langle n \rangle_t \neq 1$

Subtraktion erfolgt wie bei TK = 1

Ergebnis: siehe Ausführung

bei über- oder untergelaufenem Ergebnis:

$\langle D \rangle_1 \neq \langle D \rangle_2$ und $\langle D \rangle_t = 0$ oder 1

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle D \rangle := t_{max} ; \langle D \rangle - \langle A \rangle$

BÜ-Alarm

Das im Register D stehende Ergebnis wird nicht in die Speicherzelle n abgespeichert.

bei über- oder untergelaufenem Operand:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t = 0$ oder 1

Wird mit einem über- oder untergelaufenem Operand gerechnet, so kann das Ergebnis falsch sein, ohne daß ein BÜ-Alarm erfolgt.

SBD	n

Subtrahiere von D

SBD

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle D \rangle_t = \langle n \rangle_t = 1$

$\langle D \rangle_1 = \langle D \rangle_2$

Ausführung:

$\langle A \rangle := t_n ; \langle n \rangle$

$\langle A \rangle_1 := \langle A \rangle_2$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

$\langle A \rangle := t_{max} ; \langle D \rangle - \langle A \rangle$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers D darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register A gebracht und vom Inhalt des Registers D subtrahiert. Die Subtraktion erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis im Register A erhält die höhere der beiden Typenkennungen.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: SBD

Interncode: '45'

belegt: Rechenwerk

Takte: 11

Alarm:

BÜ-Alarm: wenn beim Ergebnis
$\langle A \rangle_t$ = 0 oder 1 und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:
$\langle D \rangle_t$ oder $\langle n \rangle_t \neq 1$

Subtraktion erfolgt wie bei TK = 1
Ergebnis: siehe Ausführung

bei über- oder untergelaufenem Ergebnis:
$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t$ = 0 oder 1

Ergebnis: siehe Ausführung
BÜ-Alarm

bei über- oder untergelaufenem Operand:
$\langle D \rangle_1 \neq \langle D \rangle_2$

Wird mit über- oder untergelaufenem Operand gerechnet, so kann das Ergebnis falsch sein, ohne daß ein BÜ-Alarm erfolgt.

SBI	n

Subtrahiere invers

SBI

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A \rangle_t = \langle n \rangle_t = 1$

$\langle A \rangle_1 = \langle A \rangle_2$

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ nur bei $t_n = 0$ oder 1

$\langle A \rangle := t_{max} ; \langle D \rangle - \langle A \rangle$

Beide Operanden sollten die Typenkennung = 1 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht. Vom Inhalt dieses Registers wird der Inhalt des Registers A subtrahiert. Die Subtraktion erfolgt, auch bei Typenkennung $\neq 1$, wie bei Festkommazahlen.

Das Ergebnis im Register A erhält die höhere der beiden Typenkennungen.

Bei Zahlwörtern wird die Marke berücksichtigt.

Externcode: SBI

Interncode: '44'

belegt: Rechenwerk

Takte: 8

Alarm:

BÜ-Alarm: wenn beim Ergebnis:

$\langle A \rangle_t$ = 0 oder 1 und $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

Bei Typenkennung ungleich 1:

$\langle A \rangle_t$ oder $\langle n \rangle_t \neq 1$

Subtraktion erfolgt wie bei TK = 1

Ergebnis: siehe Ausführung

bei übergelaufenem Ergebnis:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t$ = 0 oder 1

Ergebnis: siehe Ausführung

BÜ-Alarm

bei über oder untergelaufenen Operanden:

$\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle A \rangle_t$ = 0 oder 1

Wird mit über- oder untergelaufenen Operanden gerechnet, so kann das Ergebnis falsch sein, ohne daß ein BÜ-Alarm erfolgt.

SBII	p s	Springe wenn Bit gesetzt	**SBIT**

Adressenteil:

p: Zahl in der angegeben wird, um wieviel Befehle weitergesprungen wird (±0... ±127)

s_1: Nummer des Bits (1... 48)

s_2: Register A, Q, D oder H

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$s_1 \neq 0$

$s_1 \leq 48$

Ausführung:

Sprungbedingung:

$\langle s_2 \rangle_{s_1} = L$

Sprungsbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

Hat das Bit Nummer s_1 im Register s_2 den Wert L, so wird zur Adresse des Befehls die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Bei positivem p erfolgt ein Vorwärtssprung, bei negativem p ein Rückwärtssprung um p Befehle.

Im anderen Fall wird der auf SBIT folgende Befehl ausgeführt.

Externcode: SBIT

Interncode: 'F9'

belegt: Befehlswerk und Rechenwerk

Takte: 7+2q

Tabelle für Zeitberechnung:

s_1	1	2	...	14	15	16	17	...	31	32	33	...	47	48
q	16	15	...	3	2	1	16	...	2	1	16	...	2	1

Alarm:

Sonstiges:

bei nicht erfüllter Voraussetzung:
Wirkung wie Befehl NULL = 9 Takte

Adressenteil intern:

s_1 : '00xx'
xx = Nummer

s_2 : A = '0000'
Q = '0040'
D = '0080'
H = '00C0'

p : 'yy00'
yy = Sprungweite

SBQ	n	Subtrahiere in A, Q	**SBQ**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle Q\rangle_t = \langle n\rangle_t = 1$

$\left.\begin{array}{l}\langle A\rangle_1 = \langle A\rangle_2 \\ \langle Q\rangle_1 = \langle Q\rangle_2\end{array}\right\}\ \langle A\rangle_v = \text{oder} \neq \langle Q\rangle_v$

Ausführung:

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle m$

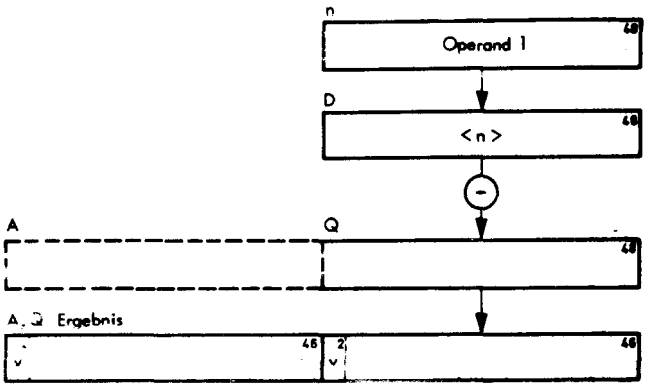
$\langle A,Q\rangle := 1,1; \langle A,Q\rangle - \langle n\rangle$

$\langle A\rangle_v$ kann $\neq \langle Q\rangle_v$

Die Typenkennung der Register A und Q sowie der Speicherzelle n muß = 1 sein; die Register dürfen nicht über- oder untergelaufen sein, sie können aber verschiedene Vorzeichen haben.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und vom Inhalt des doppeltlangen Registers A,Q subtrahiert. Das Ergebnis kann in jedem der beiden Register verschiedene Vorzeichen haben, da die Register getrennt verarbeitet werden.

Die Marke wird berücksichtigt.



Externcode: SBQ

Interncode: '7F'

belegt: Rechenwerk

Takte: 17

Alarm:

TK-Alarm: wenn $\langle A \rangle_t$ oder $\langle Q \rangle_t$ oder $\langle n \rangle_t \neq 1$

BÜ-Alarm: wenn beim Ergebnis $\langle A \rangle_1 \neq \langle A \rangle_2$

Sonstiges:

Bei falscher Typenkennung:

$\langle A \rangle_t$ oder $\langle Q \rangle_t$ oder $\langle n \rangle_t \neq 1$

$\langle D \rangle := t_n ; \langle n \rangle$

$\langle D \rangle_1 := \langle D \rangle_2$ nur bei $t_n = 0$

$\langle M \rangle := \langle M \rangle \vee \langle n \rangle_m$

TK-Alarm

6 Takte

bei über- oder untergelaufenen Operanden:

$\langle A \rangle_1 \neq \langle A \rangle_2$ oder $\langle Q \rangle_1 \neq \langle Q \rangle_2$

Die Befehlsausführung wird falsch.

bei über- oder untergelaufenem Ergebnis:

$\langle A \rangle_1 \neq \langle A \rangle_2$

Ergebnis: siehe Ausführung

BÜ-Alarm

17 Takte

SBT	n	Subtrahiere Teilwort	**SBT**

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

⟨Q⟩ = Maske

⟨A⟩ ≥ ⟨nr⟩

Ausführung:

⟨D⟩ := t_Q ;⟨Q⟩

Hilfsgröße qr := ⟨Q⟩ um p Stellen nach rechts im Kreis geschiftet

Hilfsgröße nr := ⟨n⟩ um p Stellen nach rechts geschiftet

p: Anzahl der L-Bits, die rechtsbündig im Register Q stehen
Falls ⟨Q⟩ = LL...L, dann p = 0

nr_x := 0 falls qr_x = L

$⟨A⟩_x$:= 0 falls qr_x = L

⟨A⟩ := t_A ;$⟨A⟩_x$ - nr_x

x: 1,2,...48
⟨A⟩ und nr werden als zusammenhängende, positive und vorzeichenlose Festkommazahlen aufgefaßt

Im Register Q steht eine Maske. Es wird aus dem Inhalt des Registers Q eine Hilfsgröße gebildet. Diese Hilfsgröße wird qr genannt. Sie ergibt sich aus dem Inhalt des Registers Q um p Stellen nach rechts im Kreis geschiftet, wobei p die Anzahl der im Register Q rechtsbündig stehenden L-Bits angibt. Falls der Inhalt des Registers Q nur aus L-Bits besteht, wird p zu Null. Aus dem Inhalt der Speicherzelle n wird eine Hilfsgröße nr gebildet, die sich aus dem um p Stellen nach rechts geschifteten Inhalt ergibt.

Durch das Nullfeld der Maske in der Hilfsgröße qr werden aus dem Inhalt des Registers A und aus dem Inhalt der Hilfsgröße nr Teilwörter ausgeschnitten.

Der so veränderte Inhalt der Hilfsgröße nr wird von dem gleichfalls veränderten Inhalt des Registers A subtrahiert. Gleichgültig welche Bits aus den Ganzwörtern ausgeschnitten werden, erfolgt die Operation immer so, als handele es sich um zusammenhängende, positive und vorzeichenlose Festkommazahlen. Das Ergebnis steht mit der ursprünglichen Typenkennung im Register A. Binärstellen, die dem L-Feld der Maske entsprechen, werden zu Null.

Der Inhalt des Registers Q steht einschließlich Typenkennung im Register D.

Externcode: SBT

Interncode: 'F5'

belegt: Rechenwerk

Takte: 21 + 2p

Alarm:

BÜ-Alarm: wenn ⟨A⟩ < ⟨nr⟩

Sonstiges:

Bei negativem Ergebnis:

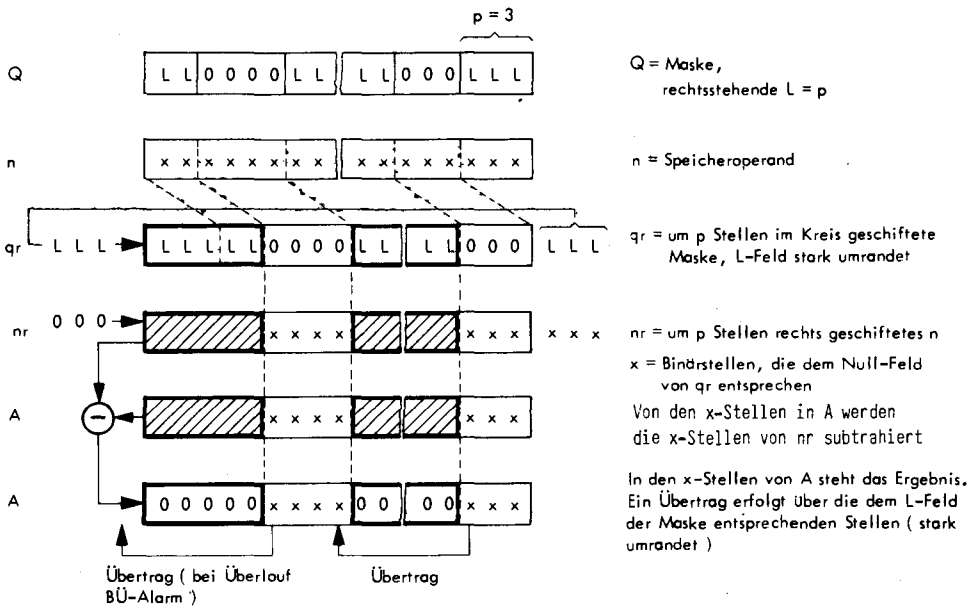
⟨A⟩ < ⟨nr⟩

Ergebnis: siehe Ausführung

⟨A⟩ := t_A ;⟨A⟩ + 2^{48} - ⟨nr⟩

BÜ-Alarm

Das Ergebnis ist eine negative Zahl im B-Komplement.



SBU	n

Subtrahiere unnormalisiert

SBU

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle A\rangle_1 = \langle A\rangle_2$

Ausführung:

$\langle D\rangle := t_n ;\langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$

$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle := 0;\langle A\rangle - \langle D\rangle$

$\langle Q\rangle := 0;+0$

$\langle Y\rangle := +0$

Beide Operanden müssen die Typenkennung = 0 haben. Der Inhalt des Registers A darf nicht über- oder untergelaufen sein.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht und vom Inhalt des Registers A subtrahiert. Das Ergebnis steht mit Typenkennung = 0 im Register A und ist weder normalisiert noch gerundet.

Das Register Q wird mit Typenkennung = 0 auf Null gelöscht. Der Inhalt des Registers Y ergibt sich zu Null.

Die Marke wird berücksichtigt.

Externcode: SBU

Interncode: '4D'

belegt: Rechenwerk

Takte: 26

Alarm:

TK-Alarm: wenn $\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

BÜ-Alarm: wenn $\langle A\rangle_1 \neq \langle A\rangle_2$ (nicht unbedingt BÜ-Alarm)
wenn Exponent des Ergebnisses > +127

Sonstiges:

Bei falscher Typenkennung:

$\langle A\rangle_t \neq 0$ oder $\langle n\rangle_t \neq 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\left.\begin{array}{l}\langle D\rangle_1 := \langle D\rangle_2 \\ \langle M\rangle := \langle M\rangle \vee \langle n\rangle_m\end{array}\right\}$ nur bei $t_n = 1$

TK-Alarm

2 Takte

bei übergelaufenem Operanden:

$\langle A\rangle_1 \neq \langle A\rangle_2$ und $\langle A\rangle_t = \langle n\rangle_t = 0$

$\langle D\rangle := t_n ; \langle n\rangle$

$\langle D\rangle_1 := \langle D\rangle_2$
$\langle M\rangle := \langle M\rangle \vee \langle n\rangle_m$

$\langle A\rangle$:= 0; undefiniert
$\langle Q\rangle$:= 0;+0
$\langle Y\rangle$:= undefiniert
evtl. BÜ-Alarm
26 Takte

bei übergelaufenem Ergebnis:

Exponent > +127

Ergebnis: siehe Ausführung
$\langle A\rangle := \langle A\rangle \cdot 16^{-255}$
BÜ-Alarm
26 Takte

SE	m

Springe nach Ersetzung

SE

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird speziell modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

⟨F⟩ := ⟨m⟩ + mod2

Es wird in jedem Fall ein Sprung ausgeführt.
Die Adresse des nächsten Befehls befindet sich in der Speicherzelle m.

Ein vom Vorbefehl vorhandener Modifikator wird auf die Adresse, die in der Speicherzelle m ist, addiert (nicht auf den Adressenteil des Befehls SE).

Mit SE kann in eine andere Großseite gesprungen werden.

Externcode: SE

Interncode: 'BC'

belegt: Befehlswerk

Takte: 2

Alarm:

Sonstiges:

SEGG	p_L p_R

Springe, wenn Exponent größer gleich

SEGG

Adressenteil:

p_L = Zahl, in der angegeben wird, um wieviel Befehle weitergesprungen wird (0...±127)

p_R = Zahl, mit der der Exponent verglichen wird
0...127 wenn positiv (entspricht +0...+127)
N0...N127 wenn negativ (entspricht -0...-127) *

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = 0$

Ausführung:

Sprungbedingung:

$\langle A \rangle_{41-48} \geq p_R$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p_L$

$\langle D \rangle := t_A ; \langle A \rangle$

Sprungbedingung nicht erfüllt:

$\langle D \rangle := t_A ; \langle A \rangle$

Der Exponent der Gleitkommazahl im Register A wird mit der Zahl p_R verglichen.

Ist der Exponent gleich (+0 = -0) oder ist er größer, so wird zur Adresse des Befehls SEGG die Zahl p_L addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Bei positivem p erfolgt ein Vorwärtssprung, bei negativem p ein Rückwärtssprung um p Befehle.

Im anderen Fall wird der auf SEGG folgende Befehl ausgeführt.

In jedem Fall wird der Inhalt des Registers A einschließlich Typenkennung auch ins Register D gebracht.

*Die Angabe eines Vorzeichens bei p_R ist für den TAS-Assembler nicht erlaubt. Aus diesem Grund sind positive Werte ohne Vorzeichen zu schreiben und negative Werte durch Vorausstellen des Buchstabens N zu kennzeichnen.

Externcode: SEGG

Interncode: '93'

belegt: Befehlswerk und Rechenwerk

Takte: 10 bei erfüllter Sprungbedingung
4 bei nicht erfüllter Sprungbedingung

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 0$

Sonstiges:

bei falscher Typenkennung:

$\langle A \rangle_t \neq 0$

$\langle D \rangle := t_A ; \langle A \rangle$
TK-Alarm
2 Takte

SFB	m

Springe und bringe ⟨F⟩ + 1 nach B

SFB

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle F \rangle + 1$

$\langle F \rangle_{9-24} := m$

Es wird in jedem Fall ein Sprung ausgeführt.
Der nächste Befehl ist der in der Speicherzelle m.

Gleichzeitig wird die Adresse des auf den Befehl SFB folgenden Befehls in das Register B gebracht (Rücksprungadresse).

Externcode: SFB

Interncode: '3A'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SFBE	m

Springe nach Ersetzung und bringe ⟨F⟩+1 nach B

SFBE

Adressenteil: m: Adresse eines Halbwortes

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := ⟨F⟩+1
⟨F⟩ := ⟨m⟩+ mod2

Es wird in jedem Fall ein Sprung ausgeführt. In der Speicherzelle m steht die Adresse des Befehls, der als nächster zur Ausführung kommt.

Gleichzeitig wird die Adresse des auf SFBE folgenden Befehls in das Register F gebracht.

Mit SFBE kann in eine andere Großseite gesprungen werden.

Externcode: SFBE

Interncode: 'FA'

belegt: Befehlswerk und Rechenwerk

Takte: 15

Alarm:

Sonstiges:

SG	m	Springe, wenn größer	**SG**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle H \rangle_1 = \langle H \rangle_2$ bei $t_H = 0$

$\langle A \rangle_1 = \langle A \rangle_2$ bei $t_A = 0$

Ausführung:

Die Art des Vergleichs wird durch die höhere der beiden Typenkennungen ($\langle A \rangle_t$, $\langle H \rangle_t$) bestimmt.

$\langle A \rangle$:= $\langle A \rangle$ normalisiert
$\langle H \rangle$:= $\langle H \rangle$ normalisiert
Falls $\langle A \rangle$ Exponent = +0, wird er -0

} nur bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Sprungbedingung:

$\langle A \rangle > \langle H \rangle$

Sprungbedingung erfüllt:

$\langle F \rangle_{9\_24}$:= m

Bei Gleitkommazahlen darf die Mantisse der Operanden nicht über- oder untergelaufen sein.

Es wird der Inhalt des Registers A mit dem Inhalt des Register H auf größer verglichen.

Sind beides Gleitkommazahlen, so wird der Inhalt der Register A und H vorher normalisiert.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SG folgende Befehl zur Ausführung.

Externcode: SG

Interncode: 'AB'

belegt: Befehlswerk und Rechenwerk

Takte: 6

7 bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Alarm:

Sonstiges:

Bei Gleitkomma-Operanden, deren Mantisse über- oder untergelaufen ist:

$\langle H \rangle_1 \neq \langle H \rangle_2$ oder $\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle H \rangle_t = \langle A \rangle_t = 0$

Ergebnis: im allgemeinen fehlerhafter Vergleich

SGO	m	Springe, wenn größer 0	**SGO**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle > 0$ bei t_A = 0: Mantisse ±0,
1: ±0,
2: 0,
3: 0,

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers A mit Null auf größer verglichen. Der Vergleich ist selbstverständlich von der Typenkennung abhängig.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt; im anderen Fall wird der auf SGO folgende Befehl ausgeführt.

Externcode: SGO

Interncode: 'D4'

belegt: Rechenwerk und Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SGG	m	Springe, wenn größer oder gleich	**SGG**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle H \rangle_1 = \langle H \rangle_2$ bei $t_H = 0$

$\langle A \rangle_1 = \langle A \rangle_2$ bei $t_A = 0$

Ausführung:

Die Art des Vergleichs wird durch die höhere der beiden Typenkennungen ($\langle A \rangle_t$, $\langle H \rangle_t$) bestimmt.

$\langle A \rangle$:= $\langle A \rangle$ normalisiert
$\langle H \rangle$:= $\langle H \rangle$ normalisiert
Falls $\langle A \rangle$ Exponent +0, wird er -0

} nur bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Sprungbedingung:

$\langle A \rangle \geq \langle H \rangle$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-z4}$:= m

Bei Gleitkommazahlen darf die Mantisse der Operanden nicht über- oder untergelaufen sein.

Es wird der Inhalt des Registers A mit dem Inhalt des Register H auf gleich oder größer verglichen.

Sind beides Gleitkommazahlen, so wird der Inhalt der Register A und H vorher normalisiert.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SGG folgende Befehl zur Ausführung.

Externcode: SGG

Internode: 'AF'

belegt: Befehlswerk und Rechenwerk

Takte: 6

7 bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Alarm:

Sonstiges:

Bei Gleitkomma-Operanden, deren Mantisse über- oder untergelaufen ist:

$\langle H \rangle_1 \neq \langle H \rangle_2$ oder $\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle H \rangle_t = \langle A \rangle_t = 0$

Ergebnis: im allgemeinen fehlerhafter Vergleich

SGGO	m

Springe, wenn größer gleich 0

SGGO

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle \geq 0$

bei t_A = 0: Mantisse ±0,
1: ±0,
2: } immer erfüllt,
3: }

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Ist der Inhalt des Registers A ein Zahlwort, so wird er mit Null auf größer oder gleich verglichen.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SGGO folgende Befehl zur Ausführung.

Ist der Inhalt des Registers A ein Nichtzahlwort (TK = 2 oder 3), so wird in jedem Fall als nächster der Befehl in der Speicherzelle m ausgeführt.

Externcode: SGGO

Interncode: 'A6'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

SH	s p	Schifte	**SH**

Adressenteil:

s_1 : A = Register A
Q = Register Q
AQ = Register A und Q getrennt
Z = doppeltlangen Register A und Q zusammen
leer, es erfolgt kein Schift

s_2 : leer,L= Rechtsschift, Linksschift

s_3 : leer,K= gestreckter Schift,Kreisschift

s_4 : leer,R = ohne Rundung, mit Rundung

s_5 : leer,U = abhängig von TK, unabh. von TK

s_6 : leer,B = nicht zählen, zählen der aus Reg. A geschifteten L-Bits

p : Anzahl der Schiftschritte ±0...±127

bei mod2: wird modifiziert

bei R: nicht zugelassen

Voraussetzung:

$\langle A\rangle_1 = \langle A\rangle_2$ nur bei t_A = 0 oder 1 wenn s_1 = A,AQ oder Z
$\langle Q\rangle_1 = \langle Q\rangle_2$ nur bei t_Q = 0 oder 1 wenn s_1 = Q,AQ oder Z
$\langle A\rangle_1 = \langle Q\rangle_1$ nur bei t_A und t_Q = 0 oder 1 wenn s_1 = Z
$\langle A\rangle_r = \langle Q\rangle_r$

nur bei $s_5 \neq U$, bei s_5= U bedeutungslos. Zahlwörter dürfen nicht über- oder untergelaufen sein. Gleitkommazahlen werden wie Festkommaz. verarbeitet. bei s_1 = Z bestimmt die höhere der beid. Typenkennungen die Art des Schiftes

p = 0...±127 (sinnvoll nur 0...±96)

Die Spezifikationen s_1 bis s_6 können kombiniert werden, wobei einige Kombinationen nicht sinnvoll sind und nicht ausgeführt werden.

Ausführung:

Schiften gemäß Spezifikation s um p Stellen

⟨A⟩	:= ⟨A⟩	geschiftet	bei s_1 : A	bei s_2 = leer: nach rechts geschiftet
⟨Q⟩	:= ⟨Q⟩	geschiftet	Q	L : nach links geschiftet
⟨A⟩,⟨Q⟩	:= ⟨A⟩,⟨Q⟩	geschiftet	AQ	bei s_3 = leer: gestreckter Schift; bei TK ≤ 1 werden vorzeichengleiche Stellen, bei TK ≥ 2 werden Nullen nachgezogen.
⟨A,Q⟩	:= ⟨A,Q⟩	geschiftet	Z	K : Kreisschift; herausgeschiftete Stellen werden auf der anderen Seite nachgezogen.

Runden bei s_4 = R, nach dem Schiften von Zahlwörtern

⟨A⟩	:= ⟨A⟩	gerundet	bei s_1 : A	nur bei TK = 0 oder 1, sonst s_4 = R bedeutungslos
⟨Q⟩	:= ⟨Q⟩	gerundet	Q	
⟨A⟩,⟨Q⟩	:= ⟨A⟩,⟨Q⟩	gerundet	AQ	die Reg. A und Q werden getrennt gerundet
⟨A,Q⟩	:= ⟨A,Q⟩	gerundet	bei s_1 = Z und bei s_2 = leer	Rechtsschift, das doppeltlange Reg. wird zusammen gerundet
⟨A⟩ := ⟨A,Q⟩ ⟨Q⟩ := +0		gerundet	bei s_1 = Z und bei s_2 = L	Linksschift, das doppeltlange Register wird so gerundet, daß der Inhalt des Registers Q zu Null wird

Schifte bei s_5 = U, unabhängig von der Typenkennung

Das zu schiftende Wort wird unabhängig von der Typenkennung als Bitmuster aufgefaßt. Vorzeichen haben keinen Einfluß auf den Schift.

TK ist bedeutungslos

s_4 = R ist bedeutungslos es wird bei s_5 = U nicht gerundet.

Zählen der besetzten Bits bei s_6 = B

⟨Y⟩ := Anzahl der aus ⟨A⟩ links oder rechts herausgeschifteten L-Bits

bei s_6 = leer: wird der Inhalt des Registers Y nicht verändert

Gemäß der Spezifikation s erfolgt der Schift immer um p Binärstellen. Es ist aber nur sinnvoll, in p soviel Stellen anzugeben, wie Binärstellen in dem in s angegebenen Registern enthalten sind. Ist p = ±0, so erfolgt kein Schift.

Die Typenkennung der Register wird nicht mitgeschiftet. Gleitkommazahlen werden wie Festkommazahlen verarbeitet. Bei s_1 = AQ oder Z und $s_5 \neq$ U können die Register verschiedene Typenkennungen haben, die Art des Schiftes wird von der höheren der beiden Typenkennungen bestimmt.

Bei s_1 = Z und s_2 = leer oder L, sowie s_3 = leer, wird ein gestreckter Langschift nach rechts oder links ausgeführt, wobei die Vorzeichenstellen des Registers Q ($\langle Q\rangle_{1,2}$) umschiftet werden als sei $\langle A\rangle_{48}$ mit $\langle Q\rangle_3$ verbunden.

Nähere Erläuterungen zum Schiften sind dem TAS-Handbuch zu entnehmen.

Externcode: SH

Interncode: '9B'

belegt: Rechenwerk

Takte: 2 (q + r) + 5 ohne Rundung
zusätzlich 5 Takte, wenn Rundung ausgeführt wurde

p : 4 = q Rest r
q = Anzahl der Viererschifte
r = Anzahl der Einerschifte

Alarm:

<u>BÜ-Alarm nur bei Zahlwörtern TK = 0 oder 1 (bei s_1 = Z maximale TK = 0 oder 1) und $s_5 \neq$ U:</u>

bei BÜ-Alarm wird der Befehl zu Ende geführt; Ergebnis: siehe unter Ausführung, es muß mit einem falschen Ergebnis gerechnet werden;

bei einem durch Aufrundung über- oder untergelaufenem Zahlwort:
wenn s_1 = A,Q,AQ oder Z und s_4 = R, wobei durch Aufrundung $\langle A\rangle_1 \neq \langle A\rangle_2$ wurde;

während oder nach einem Linksschift bei einem über- oder untergelaufenem Zahlwort:
wenn s_1 = A,Q oder AQ und s_2 = L, wobei nach irgendeinem Schiftschritt $\langle A\rangle_2 \neq$ dem ursprünglichen $\langle A\rangle_1$ wurde;

bei einem Zahlwort, welches schon vor dem Schift untergelaufen war und dann um 5 Stellen nach links geschiftet wurde:
wenn vor dem Schift $\langle A\rangle_1 \neq \langle A\rangle_2$ oder $\langle Q\rangle_1 \neq \langle Q\rangle_2$ war und s_1 = A,Q,AQ oder Z und s_2 = L, wobei das untergelaufene Zahlwort um 5 Stellen geschiftet wurde;

bei Zahlwörtern im doppeltlangen Register A,Q, die während oder nach dem Linksschift ungleiche Vorzeichen haben:
wenn s_1 = Z und s_2 = L, wobei nach irgendeinem Schiftschritt $\langle A\rangle_2 \neq \langle Q\rangle_1$ wurde.

Sonstiges:

bei s_1 = Z und s_2 = L und s_4 = R und p = 0 (SH ZLR 0), bei $\langle A\rangle_t$ = 0 oder 1 und $s_5 \neq$ U:
wenn $\langle A\rangle$ maximal über- oder untergelaufen ist und auf Grund von $\langle Q\rangle_3$ aufgerundet wurde;

es erfolgt kein BÜ-Alarm, das Ergebnis erhält umgekehrte Vorzeichen

bei s_1 = Z und $\langle A\rangle_t$ = 0 oder 1 und $s_5 \neq$ U:
wenn vor dem Schift $\langle A\rangle_1 \neq \langle Q\rangle_2$ bei s_2 = L : $\langle A\rangle := -\langle A\rangle$
bei s_2 = leer: $\langle Q\rangle := -\langle Q\rangle$
} das Zahlwort ändert seinen Wert durch Invertierung aller Bits

Adressenteil intern:

s_1 : A = '8000'
Q = '4000'
Z = '0800'
} AQ = 'C000'

s_2 : L = '2000'

s_3 : K = '1000'

s_4 : R = '0400'

s_5 : U = '0200'

s_6 : B = '0100'

p : = '00xx'

xx = p = Anzahl der Binärstellen, um die geschiftet wird

SHB	s p

Schifte in B

SHB

Adressenteil:

s = Spezifikation
R = Schift nach rechts
L = Schift nach links

p = Zahl der Stellen, die geschiftet werden sollen (0...255)
Der Schift wird um höchstens 24 Stellen durchgeführt.

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := ⟨B⟩ geschiftet um p Binärstellen nach rechts oder links

Die Spezifikation s gibt die Schiftrichtung im Register B an. Es werden Nullen nachgezogen.

Externcode: SHB

Interncode: '21'

belegt: Befehlswerk

Takte: (4 • p + 6) p = Anzahl der Binärstellen,um die geschiftet wird
Es wird höchstens solange geschiftet bis ⟨B⟩ := +O

Alarm:

Sonstiges:

Adressenteil intern:

leer = '00xx'

L = '80xx'

xx = Anzahl der Schiftschritte

SI	m	Springe, wenn identisch	**SI**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung: $\langle A\rangle_1 = \langle A\rangle_2$ bei $t_A = 0$
$\langle H\rangle_1 = \langle H\rangle_2$ bei $t_H = 0$

Ausführung:

Die Art des Vergleichs wird durch die höhere der beiden Typenkennungen ($\langle A\rangle_t$, $\langle H\rangle_t$) bestimmt.

$\langle H\rangle$:= $\langle H\rangle$ normalisiert
$\langle A\rangle$:= $\langle A\rangle$ normalisiert
falls $\langle A\rangle$ Exponent = +0, wird er -0

} bei $\langle A\rangle_t = \langle H\rangle_t = 0$

Sprungbedingung:

$\langle A\rangle = \langle H\rangle$

Sprungbedingung erfüllt:

$\langle F\rangle_{9-24}$:= m

Bie Gleitkommazahlen darf die Mantisse der Operanden nicht über- oder untergelaufen sein.

Es wird der Inhalt des Registers A mit dem Inhalt des Registers H verglichen, ob der Inhalt beider Register identisch ist.

Sind beides Gleitkommazahlen, so wird vorher der Inhalt der Register A und H normalisiert.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt; im anderen Fall kommt der auf SI folgende Befehl zur Ausführung.

Externcode: SI

Internode: 'AC'

belegt: Befehlswerk und Rechenwerk

Takte: 6

7 bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Alarm:

Sonstiges:

Bei Gleitkomma-Operanden, deren Mantisse über- oder untergelaufen ist:

$\langle H \rangle_1 \neq \langle H \rangle_2$ oder $\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle H \rangle_t = \langle A \rangle_t = 0$

$\langle A \rangle$:= $\langle A \rangle$ normalisiert
$\langle A \rangle_{40-48}$:= -0 wenn vorher +0
} wenn $\langle A \rangle_t - \langle H \rangle_t = 0$ und $\langle A \rangle_1 = \langle A \rangle_2$

$\langle H \rangle$:= normalisiert wenn $\langle H \rangle_t = 0$ und $\langle H \rangle_1 = \langle H \rangle_2$

Ergbnis: im allgemeinen fehlerhafter Vergleich

SIO	m

Springe, wenn identisch 0

SIO

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle = 0$ bei t_A = 0: Mantisse ±0,
1: ±0,
2: 0,
3: 0,

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers A auf identisch Null verglichen. Der Vergleich ist selbstverständlich von der Typenkennung abhängig.

Bei erfüllter Sprungbedingung wird als nächstes der Befehl in der Speicherzelle m ausgeführt; im anderen Fall kommt der auf SIO folgende Befehl zur Ausführung.

Externcode: SIO

Internode: 'A4'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

SK	m	Springe, wenn kleiner	**SK**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle H \rangle_1 = \langle H \rangle_2$ bei $t_H = 0$

$\langle A \rangle_1 = \langle A \rangle_2$ bei $t_A = 0$

Ausführung:

Die Art des Vergleichs wird durch die höhere der beiden Typenkennungen ($\langle A \rangle_t$, $\langle H \rangle_t$) bestimmt.

$\langle A \rangle$:= $\langle A \rangle$ normalisiert
$\langle H \rangle$:= $\langle H \rangle$ normalisiert
Falls $\langle A \rangle$ Exponent = +0, wird er -0

} nur bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Sprungbedingung:

$\langle A \rangle < \langle H \rangle$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24}$:= m

Bei Gleitkommazahlen darf die Mantisse der Operanden nicht über- oder untergelaufen sein.

Es wird der Inhalt des Registers A mit dem Inhalt des Registers H auf kleiner verglichen.

Sind beides Gleitkommazahlen, so wird der Inhalt der Register A und H vorher normalisiert.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SK folgende Befehl zur Ausführung.

Externcode: SK

Interncode: 'AA'

belegt: Befehlswerk und Rechenwerk

Takte: 6

7 bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Alarm:

Sonstiges:

Bei Gleitkomma-Operanden, deren Mantisse über- oder untergelaufen ist:

$\langle H \rangle_1 \neq \langle H \rangle_2$ oder $\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle H \rangle_t = \langle A \rangle_t = 0$

Ergebnis: im allgemeinen fehlerhafter Vergleich

SKO	m	Springe, wenn kleiner 0	**SKO**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle < 0$

bei t_A = 0: Mantisse ±0
1: ±0
2: } nie erfüllt
3: }

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Ist der Inhalt des Registers A ein Zahlwort, so wird der Inhalt mit 0 auf kleiner verglichen.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt; im anderen Fall wird der auf SKO folgende Befehl ausgeführt.

Ist der Inhalt des Registers A ein Nichtzahlwort (TK = 2 oder 3), so wird in jedem Fall der auf SKO folgende Befehl ausgeführt.

Edierncode: SKO

Interncode: 'D5'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

SKO	m

Springe, wenn kleiner 0

SKO

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle < 0$

bei t_A = 0: Mantisse ±0
1: ±0
2: } nie erfüllt
3: }

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Ist der Inhalt des Registers A ein Zahlwort, so wird der Inhalt mit 0 auf kleiner verglichen.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt; im anderen Fall wird der auf SKO folgende Befehl ausgeführt.

Ist der Inhalt des Registers A ein Nichtzahlwort (TK = 2 oder 3), so wird in jedem Fall der auf SKO folgende Befehl ausgeführt.

Externcode: SKO

Interncode: 'D5'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

SKG	m	Springe, wenn kleiner oder gleich	**SKG**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle H\rangle_1 = \langle H\rangle_2$ bei $t_H = 0$
$\langle A\rangle_1 = \langle A\rangle_2$ bei $t_A = 0$

Ausführung:

Die Art des Vergleichs wird durch die höhere der beiden Typenkennungen ($\langle A\rangle_t$, $\langle H\rangle_t$) bestimmt.

$\langle A\rangle$:= $\langle A\rangle$ normalisiert
$\langle H\rangle$:= $\langle H\rangle$ normalisiert
Falls $\langle A\rangle$ Exponent = +0, wird er -0

} nur bei $\langle A\rangle_t = \langle H\rangle_t = 0$

Sprungbedingung:

$\langle A\rangle \leq \langle H\rangle$

Sprungbedingung erfüllt:

$\langle F\rangle_{9-24}$:= m

Bei Gleitkommazahlen darf die Mantisse der Operanden nicht über- oder untergelaufen sein.

Es wird der Inhalt des Registers A mit dem Inhalt des Registers H auf gleich oder kleiner verglichen.

Sind beides Gleitkommazahlen, so wird der Inhalt der Register A und H normalisiert.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SKG folgende Befehl zur Ausführung.

Externcode: SKG

Interncode: 'AE'

belegt: Befehlswerk und Rechenwerk

Takte: 6

7 bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Alarm:

Sonstiges:

Bei Gleitkomma-Operanden, deren Mantisse über- oder untergelaufen ist:

$\langle H \rangle_1 \neq \langle H \rangle_2$ oder $\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle H \rangle_t = \langle A \rangle_t = 0$

Ergebnis: im allgemeinen fehlerhafter Vergleich

SKGO	m

Springe, wenn kleiner oder gleich 0

SKGO

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle \leq 0$

bei t_A = 0: Mantisse ±0
1: ±0
2: 0,
3: 0,

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers A mit Null auf kleiner oder gleich verglichen. Der Vergleich ist selbstverständlich von der Typenkennung abhängig.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt; im anderen Fall kommt der auf SKGO folgende Befehl zur Ausführung.

Externcode: SKGO

Interncode: 'A5'

belegt: Rechenwerk und Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SL	p s	Springe, wenn Merklicht gesetzt	**SL**

Adressenteil: p = Zahl, in der angegeben wird, um wieviel Befehle weitergesprungen wird (0...±127)

s = Angabe der Merklichter 1 - 8

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

eines der $\langle K \rangle_s = L$ s = Merklichter = 1,2,...,8

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

Ist mindestens eines der genannten Merklichter gesetzt, so wird zur Adresse des Befehls SL die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Bei positivem p erfolgt ein Vorwärtssprung, bei negativem p ein Rückwärtssprung um p Befehle.

Ist keines der angegebenen Merklichter gesetzt, so kommt der auf SL folgende Befehl zur Ausführung.

Externcode: SL

Internode: '1E'

belegt: Befehlswerk

Takte: 9 bei erfüllter Sprungbedingung
1 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

SLL	p s

Springe, wenn Merklicht gesetzt und lösche

SLL

Adressenteil: p = Zahl, in der angegeben wird, um wieviel Befehle weitergesprungen wird (0...±127)

s = Angabe der Merklichter 1 - 8

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

eines der $\langle K \rangle_s = L$ s = Merklichter = 1,2,...,8

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

$\langle K \rangle_s := 0$

Ist mindestens eines der angegebenen Merklichter gesetzt, so wird zur Adresse des Befehls SLL die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Bei positivem p erfolgt ein Vorwärtssprung, bei negativem p ein Rückwärtssprung um p Befehle. Die angegebenen Merklichter werden gelöscht.

Ist keines der angegebenen Merklichter gesetzt, so kommt der auf SLL folgende Befehl zur Ausführung.

Externcode: SLL

Interncode: '1F'

belegt: Befehlswerk

Takte: 9 bei erfüllter Sprungbedingung
1 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

SLN	p s	Springe, wenn Merklichter nicht gesetzt	**SLN**

Adressenteil: p = Zahl, in der angegeben wird, um wieviel Befehle weitergesprungen wird (0...±127)

s = Angabe der Merklichter 1 - 8

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

alle $\langle K \rangle_s = 0$ s = Merklichter = 1,2,...,8

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

Sind alle angegebenen Merklichter nicht gesetzt, so wird zur Adresse des Befehls SLN die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Ein positives p bedeutet einen Vorwärtssprung, ein negatives p einen Rückwärtssprung um p Befehle.

Ist mindestens eines der angegebenen Merklichter gesetzt, so kommt der auf SLN folgende Befehl zur Ausführung.

Externcode: SLN

Interncode: '1C'

belegt: Befehlswerk

Takte: 9 bei erfüllter Sprungbedingung
1 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

SM	m

Springe, wenn Markenregister gesetzt

SM

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle M \rangle = L$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

$\langle M \rangle := 0$

Ist das Markenregister M auf "L" gesetzt, so wird als nächster der in der Speicherzelle m stehende Befehl ausgeführt und das Markenregister M auf Null gesetzt.

Im anderen Fall wird der auf SM folgende Befehl ausgeführt.

Externcode: SM

Interncode: '34'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SMN	m

Springe, wenn Markenregister nicht gesetzt

SMN

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle M \rangle = 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Sprungbedingung nicht erfüllt:

$\langle M \rangle := 0$

Ist das Markenregister M auf "0" gesetzt, so wird als nächster der in der Speicherzelle m stehende Befehl ausgeführt.

Im anderen Fall wird der auf SMN folgende Befehl ausgeführt und das Markenregister M auf Null gesetzt.

Externcode: SMN

Interncode: '35'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SN	m	Springe, wenn nicht identisch	**SN**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle H \rangle_1 = \langle H \rangle_2$ bei $t_H = 0$

$\langle A \rangle_1 = \langle A \rangle_2$ bei $t_A = 0$

Ausführung:

Die Art des Vergleichs wird durch die höhere der beiden Typenkennungen ($\langle A \rangle_t$, $\langle H \rangle_t$) bestimmt.

$\langle H \rangle$:= $\langle H \rangle$ normalisiert
$\langle A \rangle$:= $\langle A \rangle$ normalisiert
falls $\langle A \rangle$ Exponent = +0, wird er -0

} bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Sprungbedingung:

$\langle A \rangle \neq \langle H \rangle$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24}$:= m

Bei Gleitkommazahlen darf die Mantisse der Operanden nicht über- oder untergelaufen sein.

Es wird der Inhalt des Registers A mit dem Inhalt des Registers H verglichen ob der Inhalt beider Register nicht identisch ist.

Sind beides Gleitkommazahlen, so wird vorher der Inhalt der Register H und A normalisiert.

Bei erfüllter Sprungbedingung wird als nächster der Befehl in der Speicherzelle m ausgeführt; im anderen Fall kommt der auf SN folgende Befehl zur Ausführung.

Externcode: SN

Interncode: 'AD'

belegt: Befehlswerk und Rechenwerk

Takte: 6

7 bei $\langle A \rangle_t = \langle H \rangle_t = 0$

Alarm:

Sonstiges:

Bei Gleitkomma-Operanden, deren Mantisse über- oder untergelaufen ist:

$\langle H \rangle_1 \neq \langle H \rangle_2$ oder $\langle A \rangle_1 \neq \langle A \rangle_2$ und $\langle H \rangle_t = \langle A \rangle_t = 0$

Ergebnis: im allgemeinen fehlerhafter Vergleich

SNO	m

Springe, wenn nicht 0

SNO

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle \neq 0$ bei t_A = 0: Mantisse ±0,
1: ±0,
2: 0,
3: 0,

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers A auf nicht identisch Null verglichen. Der Vergleich ist selbstverständlich von der Typenkennung abhängig.

Bei erfüllter Sprungbedingung wird als nächstes der Befehl aus der Speicherzelle m ausgeführt; im anderen Fall kommt der auf SNO folgende Befehl zur Ausführung.

Externcode: SNO

Interncode: 'A7'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

SNL	p s

Springe, wenn Merklichter nicht gesetzt, sonst lösche

SNL

Adressenteil:

p = Zahl, in der angegeben wird, um wieviel Befehle weitergesprungen wird (0...±127)

s = Angabe der Merklichter 1 - 8

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

alle $\langle K \rangle_s = 0$ s = Merklichter = 1,2,...,8

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

Sprungbedingung nicht erfüllt:

$\langle K \rangle_s := 0$

Sind alle angegebenen Merklichter nicht gesetzt, so wird zur Adresse des Befehls SNL die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Bei positivem p erfolgt ein Vorwärtssprung, bei negativem p ein Rückwärtssprung um p Befehle.

Ist mindestens eines der angegebenen Merklichter gesetzt, so werden die angegebenen Merklichter gelöscht und der auf SNL folgende Befehl kommt zur Ausführung.

Externcode: SNL

Interncode: '1D'

belegt: Befehlswerk

Takte: 9 bei erfüllter Sprungbedingung
1 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

SR	m

Springe, wenn rechtes Bit in A gesetzt

SR

Adressenteil: m = Adresse einer Speicherzelle

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle_{48} = L$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Wenn das rechte Bit des Registers A "L" ist, wird als nächstes der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall wird der auf SR folgende Befehl ausgeführt.

Externcode: SR

Interncode: 'B8'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

SRN	m

Springe, wenn rechtes Bit in A nicht gesetzt

SRN

Adressenteil: m = Adresse einer Speicherzelle

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle A \rangle_{48} = 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9\_24} := m$

Wenn das rechte Bit des Registers A "0" ist, wird als nächstes der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall wird der auf SRN folgende Befehl ausgeführt.

Externcode: SRN

Interncode: 'BA'

belegt: Befehlswerk und Rechenwerk

Takte: 1

Alarm:

Sonstiges:

SSR	p_L p_R	Springe ins System und reserviere	**SSR**

Adressenteil:

p_L : 0... 255
p_R : 0... 255

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Normalmodus eingestellt

$a = \langle BL \rangle \cdot 2^8$ Anfangsadresse des Leitblockes

Ausführung:

$\langle BPx \rangle_1 := 0$ Seitenadreßregister ungültig setzen x=1,2,3,4

$\langle BLZ1 \rangle := \langle a+64 \rangle_{32-36}$
$\langle BLZ2 \rangle := \langle a+64 \rangle_{27-31}$
} Speicherbereiche des Abwicklers einstellen

$\langle a+18 \rangle := 3; \langle B \rangle, \langle BA \rangle$
$\langle a+20 \rangle := 3; \langle F \rangle+1$, Steuerbits 25-48
} Register und Steuerbits abspeichern

$\langle B \rangle := adr$ Adressenteil von SSR ins Register B

$\langle F \rangle := \langle a+6 \rangle_{1-24}$ Sprung in den Abwickler

Steuerbits 25-42 := 0
Steuerbits 44-45 := 0
} Eingriffssperre BEFE bleibt erhalten

BEB0 := L Umschalten auf Abwicklermodus

BEMP := L SSR ist Sprungbefehl
BEMB := L Sprung in beliebige Großseite möglich

BE10 := 0
BE20 := 0
BE30 := 0
} Steuerbits 11, 12 und 13 löschen

Es wird auf den Abwickler umgestellt, indem der Abwicklermodus eingestellt wird, die Steuerbits auf einen definierten Zustand gesetzt werden und mit den Registern BLZ1 und BLZ2 der Speicherbereich des Abwicklers eingestellt wird.

Dem Abwickler wird der Adressenteil im Register B hinterlassen. Im Adressenteil ist angegeben, welche Dienstleistung der Abwickler erbringen soll (siehe besondere Schrift über die Dienstleistungen des Abwicklers).

Es erfolgt ein Sprung in den Abwickler. Die Einsprungstelle ist im Leitblock vorgegeben.

März 71

Externcode: SSR

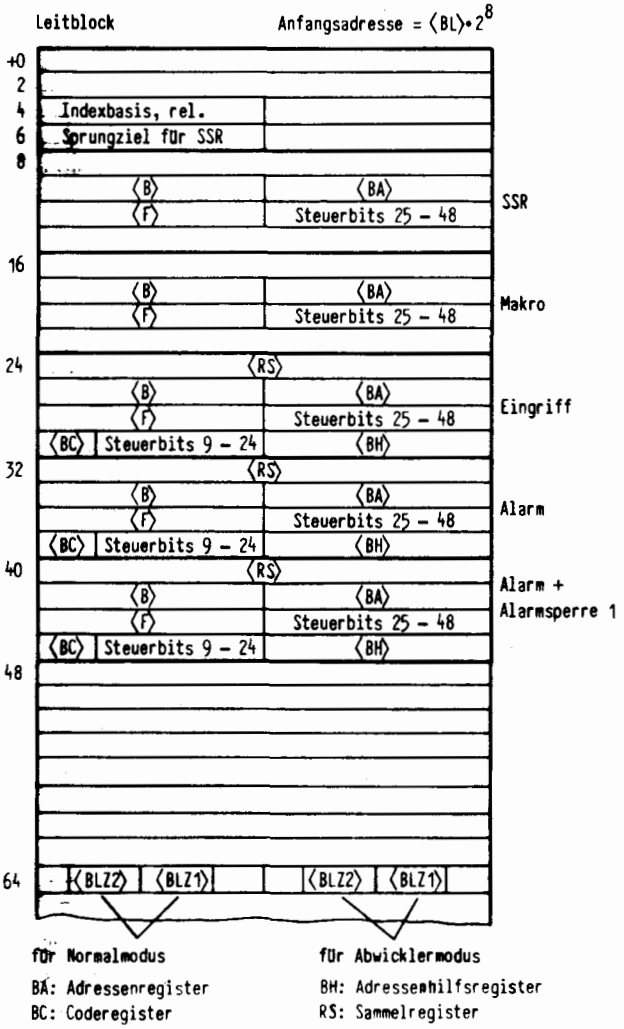
Interncode: 'BB'

belegt: Befehls- und Rechenwerk

Takte: 48

Alarm:

Sonstiges: Voraussetzung nicht erfüllt: im Abwickler-, Spezial- und Systemmodus ergibt sich eine andere Ausführung des Befehls



Nr.	Name	Bedeutung der Steuerbits
9	BEVA	In der Ausführungsphase wird ein Hauptspeicheroperand benötigt.
10	BESP	Die Dreierprobe darf nicht ersetzt werden.
11	BE30	Kennzeichnung verschiedener Ansprungstellen aus dem Mikroprogramm der Ausführungsphase
12	BE20	
13	BE10	
14	BEAC	Der Befehl wurde im Abspeicher-Nanoprogramm unterbrochen.
15	BEAA	Der Befehl wurde am Anfang der Abrufphase unterbrochen.
16	BEAB	Der Befehl wurde in der Abrufphase bei Auslesen eines Operanden unterbrochen.
17	BEIC	Der Befehl wurde bei Bildung der absoluten Indexbasis unterbrochen.
18	BEMQ	Der Befehl ist nicht zu Ende
19	BEEH	Hauptalarm (Stromausfall bzw. -Abschaltung)
20	BEER4	Rechneralarm vom 4. Rechnerkern
21	BEER3	Rechneralarm vom 3. Rechnerkern
22	BEER2	Rechneralarm vom 2. Rechnerkern
23	BEER1	Rechneralarm vom 1. Rechnerkern
24	BEFT	Technischer Fehler
25	BEMA	Der Befehl MF, MCF oder MD geht vorher
26	BEMO	Der Befehl MFU oder MCFU geht vorher
27	BEMN	Der vorhergehende Befehl definiert mod2
28	BEMM	Der Befehl MM geht vorher (im Modus 16)
29	BEMU	Der Befehl MU geht vorher
30	BEMB	Der Befehl MABl geht vorher
31	BEML	Der Befehl LEI geht vorher
32	BEMP	Der anstehende Befehl ist ein Sprungbefehl
33	BEBE	Steuerbit: Stop vor Abrufphase
34	BEBA	Steuerbit: Modus 16
35	BEBT	Steuerbit: Wartungsmodus
36	BEBF	Steuerbit: Stop nach Abrufphase
37	REAL	Typenkennungsal-Alarm
38	REBUE	Arithmetischer Alarm
39	BEEC	Speicherschutz-Alarm
40	BEEU	Alarm: Überlauf des Registers U
41	BEEK	Befehls-Alarm
42	BEEF	Stop-Alarm
43	BEFE	Eingriffssperre
44	BEEW	Wecker-Alarm
45	BEED	Dreierproben-Alarm
46	BEBO	Abwicklermodus
47	BEBN	Normalmodus
48	BEBY	Systemmodus

SSR	p_L p_R	Springe ins System und reserviere	SSR

Adressenteil: p_L: 0... 255
p_R: 0... 255

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

a = 8	wenn	BEBY = L,	BEBN = 0	Sytemmodus
a = 8	wenn	BEBY = L,	BEBN = L	Spezialmodus
a = ⟨BL⟩·2^8 + 8	wenn	BEBY = 0,	BEBN = L, BEBO = L	Abwicklermodus
a = ⟨BL⟩·2^8 + 16	wenn	BEBY = 0,	BEBN = L, BEBO = 0	Normalmodus

Ausführung:

⟨BIx⟩ zurückspeichern wenn $\langle BIx\rangle_2$ = L
$\langle BIx\rangle_1$:= 0 Indexregister ungültig setzen } x = 1,2,3,4
⟨X⟩ := $\langle 4\rangle_{1-24}$
⟨BXBZ⟩ := $\langle\langle X\rangle + 256\rangle_{1-11}$ } Indexbasis des Systems setzen
} nur bei System- oder Spezialmodus

$\langle BPx\rangle_1$:= 0 Seitenadreßregister ungültig setzen
⟨BLZ1⟩ := $\langle\langle BL\rangle\cdot 2^8 + 64\rangle_{32-36}$ } Speicherbereich des Abwicklers einstellen
⟨BLZ2⟩ := $\langle\langle BL\rangle\cdot 2^8 + 64\rangle_{27-31}$ }
} nur bei Normalmodus

BEFE := L wenn Abwicklermodus Eingriffssperre setzen

⟨a + 2⟩ := 3; ⟨B⟩, ⟨BA⟩
⟨a + 4⟩ := 3; ⟨F⟩ + 1, Steuerbits 25 - 48 } Register und Steuerbits abspeichern
⟨B⟩ := adr Adressenteil von SSR im Register B

⟨F⟩ := $\langle\langle BL\rangle\cdot 2^8 + 6\rangle_{1-24}$ Sprung bei Normalmodus
⟨F⟩ := $\langle 6\rangle_{1-24}$ Sprung bei Abwickler-, Spezial- und Systemmodus

Steuerbits 25 - 42 := 0
Steuerbits 44 - 45 := 0 } Steuerbits löschen (Eingriffssperre BEFE bleibt erhalten)

BEBO := L wenn Normalmodus
BEBY := L wenn Abwicklermodus
BEBN := 0 wenn Spezial- oder Systemmodus

BEMP := L SSR ist Sprungbefehl
BEMB := L Sprung in beliebige Großseite ist möglich

BE10 := 0
BE20 := 0 } Steuerbits 11, 12 und 13 löschen.
BE30 := 0

SSR kann durch Alarm oder Eingriff nicht unterbrochen werden. Bei MU SSR wird das Register U nicht verändert.

Externcode: SSR

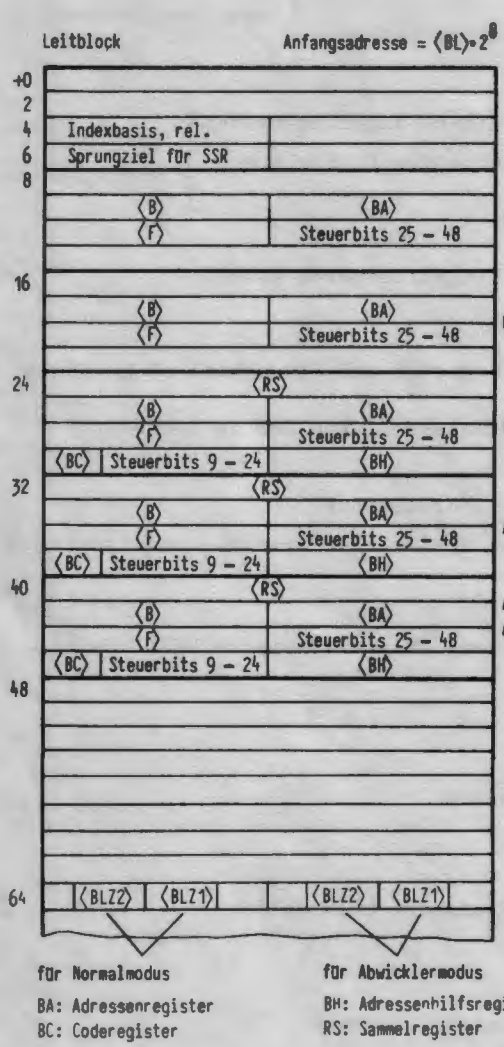
Interncode: 'BB'

belegt: Befehls- und Rechenwerk

Takte:
48 wenn Normalmodus
32 wenn Abwicklermodus
53 + 17 n wenn Spezial- oder Systemmodus
n= Anzahl der Indexregister, die zurückgespeichert wurden

Alarm:

Sonstiges:



BA: Adressenregister
BC: Coderegister
BH: Adressenhilfsregister
RS: Sammelregister

Nr.	Name	Bedeutung der Steuerbits
9	BEVA	In der Ausführungsphase wird ein Hauptspeicheroperand benötigt.
10	BESP	Die Dreierprobe darf nicht ersetzt werden.
11	BE30	Kennzeichnung verschiedener Ansprungstellen aus dem Mikroprogramm der Ausführungsphase
12	BE20	
13	BE10	
14	BEAC	Der Befehl wurde im Abspeicher-Nanoprogramm unterbrochen.
15	BEAA	Der Befehl wurde am Anfang der Abrufphase unterbrochen.
16	BEAB	Der Befehl wurde in der Abrufphase bei Auslesen eines Operanden unterbrochen.
17	BEIC	Der Befehl wurde bei Bildung der absoluten Indexbasis unterbrochen.
18	BEMQ	Der Befehl ist nicht zu Ende
19	BEEH	Hauptalarm (Stromausfall bzw. -Abschaltung)
20	BEER4	Rechneralarm vom 4. Rechnerkern
21	BEER3	Rechneralarm vom 3. Rechnerkern
22	BEER2	Rechneralarm vom 2. Rechnerkern
23	BEER1	Rechneralarm vom 1. Rechnerkern
24	BEFT	Technischer Fehler
25	BEMA	Der Befehl MF, MCF oder MD geht vorher
26	BEMO	Der Befehl MFU oder MCFU geht vorher
27	BEMN	Der vorhergehende Befehl definiert mod2
28	BEMM	Der Befehl MM geht vorher (im Modus 16)
29	BEMU	Der Befehl MU geht vorher
30	BEMB	Der Befehl MABI geht vorher
31	BEML	Der Befehl LEI geht vorher
32	BEMP	Der anstehende Befehl ist ein Sprungbefehl
33	BEBE	Steuerbit: Stop vor Abrufphase
34	BEBA	Steuerbit: Modus 16
35	BEBT	Steuerbit: Wartungsmodus
36	BEBF	Steuerbit: Stop nach Abrufphase
37	REAL	Typenkennungsal-Alarm
38	REBUE	Arithmetischer Alarm
39	BEEC	Speicherschutz-Alarm
40	BEEU	Alarm: Überlauf des Registers U
41	BEEK	Befehls-Alarm
42	BEEF	Stop-Alarm
43	BEFE	Eingriffssperre
44	BEEW	Wecker-Alarm
45	BEED	Dreierproben-Alarm
46	BEBO	Abwicklermodus
47	BEBN	Normalmodus
48	BEBY	Systemmodus

ST	p s

Springe, wenn Typenkennung

ST

Adressenteil:

p = Zahl, in der angegeben wird, um wieviel Befehle weitergesprungen wird (0...±127)

s_1 = Typenkennung 0,1,2 oder 3

s_2 = Register A,Q,D oder H

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle s_2 \rangle_t = s_1$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

Hat das mit s_2 angegebene Register die mit s_1 angegebene Typenkennung, so wird zur Adresse des Befehls die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Bei positivem p erfolgt ein Vorwärtssprung, bei negativem p ein Rückwärtssprung um p Befehle.

Im anderen Fall wird der auf ST folgende Befehl ausgeführt.

Externcode: ST

Interncode: '90'

belegt: Befehlswerk und Rechenwerk

Takte: 10 bei erfüllter Sprungbedingung
3 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

Adressenteil intern:

s_1 : 0 = '0000'
1 = '0040'
2 = '0080'
3 = '00C0'

s_2 : A = '0020'
Q = '0010'
D = '0008'
H = '0004'
'0002' darf nicht gesetzt sein, sonst wird auf jeden Fall gesprungen

p : = 'xx00'
xx = Anzahl

STN	p s

Springe, wenn Typenkennung nicht

STN

Adressenteil:

p = Zahl in der angegeben wird, um wieviel Befehle weitergesprungen wird (0...±127)

s_1 = Typenkennung 0,1,2 oder 3

s_2 = Register A,Q,D oder H

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle s_2 \rangle_t \neq s_1$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

Hat das mit s_2 angegebene Register nicht die mit s_1 angegebene Typenkennung, so wird zur Adresse des Befehls die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zur Ausführung kommt. Bei positivem p erfolgt ein Vorwärtssprung, bei negativem p ein Rückwärtssprung um p Befehle.

Im anderen Fall wird der auf STN folgende Befehl ausgeführt.

Externcode: STN

Interncode: '91'

belegt: Befehlswerk und Rechenwerk

Takte: 10 bei erfüllter Sprungbedingung
3 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

Adressenteil intern:

s_1 : 0 = '0000'
1 = '0040'
2 = '0080'
3 = '00C0'

s_2 : A = '0020'
Q = '0010'
D = '0008'
H = '0004'
'0002' darf nicht gesetzt sein, sonst wird in jedem Fall gesprungen

p : = 'xx00'
xx = Anzahl

SU	m

Springe ins Unterprogramm

SU

Adressenteil: m = Adresse eines Halbwortes (Anfangsadresse des Unterprogramms)

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨U⟩ ≠ 254
⟨U⟩ = 0...253

⟨U⟩ = 255

Ausführung:

⟨U⟩ := ⟨U⟩ + 1
⟨⟨U⟩⟩ := ⟨F⟩ + 1
$\langle F \rangle_{9-24}$:= m

⟨U⟩ := 0
⟨⟨U⟩⟩ := ⟨F⟩ + 1
$\langle F \rangle_{9-24}$:= m

Durch diesen Befehl erfolgt ein Sprung in ein Unterprogramm mit gleichzeitiger Sicherung der Rücksprungadresse.

Die Speicheradresse m wird in das Register F gebracht und gibt die Adresse des ersten Befehls im Unterprogramm an.

Der Inhalt des Registers U wird um 1 erhöht und gibt die neue Indexadresse an. In dieser neuen Indexadresse steht die Adresse des auf SU folgenden Befehls (Rücksprungadresse).

Ein Rücksprung ist mit dem Befehl MU möglich.

Ist der Inhalt des Registers U = 255, so wird (nach Erhöhung) mit dem Wert 0 fortgefahren.

Externcode: SU

Interncode: '38'

belegt: Befehlswerk

Takte: 8

Alarm:

Alarm an das Betriebssystem (BEEU): wenn ⟨U⟩ = 254

Sonstiges:

Bei ⟨U⟩ = 254: $\langle U \rangle := \langle U \rangle_{+1}$

$\langle\langle U \rangle\rangle := \langle F \rangle_{+1}$

$\langle F \rangle_{0-24} := m$

es erfolgt ein BEEU-Alarm

SUE	m

Springe in Unterprogramm nach Ersetzung

SUE

Adressenteil: m = Adresse eines Halbwortes (Anfangsadresse des Unterprogramms)

bei mod2:	wird speziell modifiziert	bei R:	zugelassen

Voraussetzung:

⟨U⟩ ≠ 254	⟨U⟩ = 255
⟨U⟩ = 0...253	

Ausführung:

⟨U⟩ := ⟨U⟩ + 1	⟨U⟩ := 0
⟨⟨U⟩⟩ := ⟨F⟩ + 1	⟨⟨U⟩⟩ := ⟨F⟩ + 1
⟨F⟩ := ⟨m⟩ + mod2	⟨F⟩ := ⟨m⟩ + mod2

Durch diesen Befehl erfolgt ein Sprung in ein Unterprogramm mit gleichzeitiger Sicherung der Rücksprungadresse.

Der Inhalt der Speicherzelle m wird in das Register F gebracht und gibt die Adresse des ersten Befehls im Unterprogramm an.

Der Inhalt des Registers U wird um 1 erhöht und gibt die neue Indexadresse an. In dieser neuen Indexadresse steht die Adresse des auf SUE folgenden Befehls (Rücksprungadresse).

Ein Rücksprung ist mit dem Befehl MU möglich.

Im Gegensatz zum Befehl SU kann mit dem Befehl SUE in eine andere Großseite gesprungen werden.

Ist der Inhalt des Registers U = 255, so wird (nach Erhöhung) mit dem Wert 0 fortgefahren.

Externcode: SUE

Interncode: 'BD'

belegt: Befehlswerk

Takte: 8

Alarm:

Alarm an das Betriebssystem (BEEU): wenn ⟨U⟩ = 254

Sonstiges:

Bei ⟨U⟩ = 254: ⟨U⟩ := ⟨U⟩ + 1
⟨⟨U⟩⟩ := ⟨F⟩ + 1
⟨F⟩ := ⟨m⟩ + mod2

bei ⟨U⟩ = 255:
Die Indexzelle 255 wird um 1 erhöht, dadurch
wird wieder mit der Indexzelle 0 weitergerechnet.

Ergebnis: siehe unter Ausführung

SW	p s

Springe wenn Wahlschalter gesetzt

SW

Adressenteil: p: Zahl mit der angegeben wird, um wieviel Befehle weitergesprungen wird (±0... ±127)

s: Angabe der Wahlschalter (1, 2, 3, 4, 5, 6, 7 und 8) oder 0

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

BEBY = L — System- oder Spezialmodus

oder BEBT1 = L
oder BEBT = BEWA = L
oder BEBT = BEWH = L
} Wartungsvariante

Ausführung:

Sprungbedingung:

eines der $\langle K \rangle_{s+8}$ = L s = Wahlschalter = 1, 2, 3, 4, 5, 6, 7, 8

Sprungbedingung erfüllt:

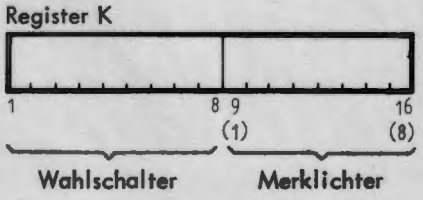
$\langle F \rangle_{8-24} := \langle F \rangle + p$

Ist mindestens ein Wahlschalter angegeben und ist mindestens einer der angegebenen Wahlschalter gesetzt, so erfolgt ein Sprung um p.

Ist kein Wahlschalter angegeben oder ist keiner der angegebenen Wahlschalter gesetzt, kommt der nächste Befehl zur Ausführung.

Die Wahlschalter können durch Tasten auf dem Bedienfeld gesetzt werden.

Die linken 8 Binärstellen des Registers K entsprechen den Wahlschaltern, die rechten den Merklichtern.



Externcode: SW

Interncode: '1B'

belegt: Befehlswerk

Takte: 9 bei erfüllter Sprungbedingung
1 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

Voraussetzung nicht erfüllt:
Sprung in das Nanoprogramm "Makro"
Beschreibung siehe unter "Makro"

SXG	m

Springe, wenn Index größer 0

SXG

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle_1 = \langle B \rangle_v$

Ausführung:

Sprungbedingung:

$\langle B \rangle > 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers B mit Null auf größer verglichen. Die erste Binärstelle des Registers B ist die Vorzeichenstelle.

Ist die Sprungbedingung erfüllt, so wird der Befehl in der Speicherzelle m als nächster ausgeführt.

Im anderen Fall kommt der auf SXG folgende Befehl zur Ausführung.

Externcode: SXG

Interncode: 'CE'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SXGG	m

Springe, wenn Index größer oder gleich O

SXGG

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle_1 = \langle B \rangle_v$

Ausführung:

Sprungbedingung:

$\langle B \rangle \geq \pm 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers B mit Null auf größer oder gleich verglichen. Die erste Binärstelle des Registers B ist die Vorzeichenstelle.

Ist die Sprungbedingung erfüllt, so wird der Befehl in der Speicherzelle m als nächster ausgeführt.

Im anderen Fall kommt der auf SXGG folgende Befehl zur Ausführung.

Externcode: SXGG

Interncode: '25'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SXI	m

Springe, wenn Index identisch 0

SXI

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle_1 = \langle B \rangle_v$

Ausführung:

Sprungbedingung:

$\langle B \rangle = \pm 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers B auf identisch Null verglichen. Die erste Binärstelle des Registers B ist die Vorzeichenstelle.

Ist die Sprungbedingung erfüllt, so wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SXI folgende Befehl zur Ausführung.

Externcode: SXI

Interncode: '24'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SXK	m

Springe, wenn Index kleiner 0

SXK

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle_1 = \langle B \rangle_v$

Ausführung:

Sprungbedingung:

$\langle B \rangle < 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Der Inhalt des Registers B wird mit Null auf kleiner verglichen. Die erste Binärstelle des Registers B ist die Vorzeichenstelle der im Register B stehenden Zahl.

Ist die Sprungbedingung erfüllt, so wird der Befehl in der Speicherzelle m als nächster ausgeführt.

Im anderen Fall kommt der auf SXK folgende Befehl zur Ausführung.

Externcode: SXK

Interncode: 'CF'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SXKG	m

Springe, wenn Index kleiner oder gleich 0

SXKG

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle_1 = \langle B \rangle_v$

Ausführung:

Sprungbedingung:

$\langle B \rangle \leq \pm 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Der Inhalt des Registers B wird mit Null auf kleiner oder gleich verglichen. Die erste Binärstelle des Registers B ist die Vorzeichenstelle.

Ist die Sprungbedingung erfüllt, so wird der Befehl in der Speicherzelle m als nächster ausgeführt.

Im anderen Fall kommt der auf SXKG folgende Befehl zur Ausführung.

Externcode: SXKG

Interncode: '26'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SXN	m	Springe, wenn Index nicht identisch 0	**SXN**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle_1 = \langle B \rangle_v$

Ausführung:

Sprungbedingung:

$\langle B \rangle \neq \pm 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird der Inhalt des Registers B auf nicht identisch Null verglichen. Die erste Binärstelle des Registers B ist die Vorzeichenstelle.

Ist die Sprungbedingung erfüllt, so wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall kommt der auf SXN folgende Befehl zur Ausführung.

Externcode: SXN

Interncode: '27'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SXR	m

Springe, wenn Indexgröße rechtes Bit = L

SXR

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle B \rangle_{24} = L$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird das rechte Bit des Registers B mit "L" verglichen.

Ist die Sprungbedingung erfüllt, so wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall wird der auf SXR folgende Befehl ausgführt.

Externcode: SXR

Internecode: 'B2'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SXRN	m

Springe, wenn Indexgröße rechtes Bit nicht L **SXRN**

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

Sprungbedingung:

$\langle B \rangle_{24} = 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := m$

Es wird das rechte Bit des Registers B mit "0" verglichen.

Ist die Sprungbedingung erfüllt, so wird als nächster der Befehl in der Speicherzelle m ausgeführt.

Im anderen Fall wird der auf SXRN folgende Befehl ausgeführt.

Externcode: SXRN

Interncode: 'B3'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

SZX	p i

Springe und zähle wenn Index kleiner 0

SZX

Adressenteil: p = Zahl (0...±127)
i = Adresse einer Indexzelle

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle i \rangle_1 = \langle i \rangle_v$

Ausführung:

Sprungbedingung:

$\langle i \rangle < \pm 0$

Sprungbedingung erfüllt:

$\langle F \rangle_{9-24} := \langle F \rangle + p$

$\langle i \rangle := \langle i \rangle + 1$

Es wird der Inhalt der Indexzelle i mit dem Wert +0 verglichen. Die erste Binärstelle in der Indexzelle ist das Vorzeichen. Ist die Sprungbedingung erfüllt - Inhalt von i kleiner als +0 - so wird der Inhalt der Indexzelle um 1 erhöht. Zu der Adresse des Befehls SZX wird die Zahl p addiert. Das Ergebnis gibt die Adresse des Befehls an, der als nächster zu Ausführung kommt.

Im anderen Fall wird der auf SZX folgende Befehl ausgeführt.

Externcode: SZX

Interncode: 'OA'

belegt: Befehlswerk

Takte: 11 bei erfüllter Sprungbedingung
2 bei nicht erfüllter Sprungbedingung

Alarm:

Sonstiges:

T	m

Tue! (Führe den Befehl in der Zelle m aus)

T

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird speziell modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle m \rangle_t$ = beliebig

Ausführung:

op := $\langle m \rangle_{1-8}$
adr := 0,$\langle m \rangle_{9-24}$
mod2 := mod2

Es wird ein Zweitbefehl erzeugt. Es ist der in der Speicherzelle m stehende Befehl. Er braucht nicht die Typenkennung = 2 haben.

Ein Modifikator 2. Art wirkt auf den Zweitbefehl.

Es folgt die Ausführung des Zweitbefehls.

Zweitbefehl:

$\langle m \rangle_{1-8}$ (8)	0	$\langle m \rangle_{9-24}$ (24)	mod2 (24)

Externcode: T

Interncode: 'CC'

belegt: Befehlswerk

Takte: 5

Alarm:

Sonstiges:

TBC	m

Transport aus Register B nach Speicher

TBC

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle m \rangle := \langle B \rangle$

$\langle m \rangle_t$ bleibt erhalten

Der Inhalt des Registers B wird in die Speicherzelle m gebracht.

Externcode: TBC

Interncode: '07'

belegt: Befehlswerk

Takte: 7

Alarm:

Sonstiges:

TCB	m

Transport aus Speicher nach B

TCB

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := ⟨m⟩

Der Inhalt der Speicherzelle m wird in das Register B gebracht.

Externcode: TCB

Interncode: '39'

belegt: Befehlswerk

Takte: 2

Alarm:

Sonstiges:

TDM	n	Tabelle durchsuchen mit Dehnung und Maske	**TDM**

Adressenteil: n = Adresse eines Ganzwortes (Anfangsadresse der Tabelle)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨D⟩ = Suchwort

⟨H⟩ = Maske

⟨B⟩ = Dehnungswert = Adressenabstand der Wörter

⟨B⟩ ≠ ±0

Suchbedingung:

$\langle n + k\langle B\rangle\rangle_x = \langle D\rangle_x$	für $\langle H\rangle_x = 0$	k = 0,1,2... x = 1...48 entspr. 0-Feld im Register H bei t_n = 0 oder 1 wird vor dem Vergleich das 1. Bit dem 2. Bit angeglichen. Das Wort in der Speicherzelle bleibt unverändert.

Abbruchkriterien:

$\langle n + k\langle B\rangle\rangle_x = \langle D\rangle_x$	für $\langle H\rangle_x = 0$	Suchwort gefunden
$\langle n + k\langle B\rangle\rangle_t \neq \langle D\rangle_t$		Suchwort nicht gefunden, Abbruch wenn ein Wort auftritt, dessen TK ungleich $\langle D\rangle_t$ ist (TK-Alarm).

Ausführung:

⟨B⟩ := n *		n* = Adresse des Wortes, das zum Abbruch führte
$\langle A\rangle_x := \langle n^*\rangle_x$	für $\langle H\rangle_x = L$	L-Feld der Maske
$\langle A\rangle_x := 0$	für $\langle H\rangle_x = 0$	0-Feld der Maske
$\langle A\rangle_t := \langle H\rangle_t$		
TK-Alarm		nur wenn Suchwort nicht gefunden

Im Register D steht ein Suchwort. Der Dehnungswert, d.h. der Abstand zwischen den Adressen zweier aufeinanderfolgender Wörter, steht im Register B. Ist dieser Dehnungswert positiv, beginnt die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt vorwärts. Bei negativem Dehnungswert endet die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt rückwärts. Der Dehnungswert muß ungleich Null sein.

Die Tabelle wird auf jedes ⟨B⟩-te Wort durchsucht, welches in den Binärstellen, an denen in der Maske (Register H) das Nullfeld steht, identisch ist mit dem Suchwort im Register D.
Ist das erste gesuchte Wort gefunden, wird der Suchvorgang abgebrochen und die Adresse dieses Wortes n* in das Register B gebracht.

Wird kein Wort gefunden und tritt ein Wort auf, dessen Typenkennung ungleich der Typenkennung des Registers D ist, so wird der Suchvorgang ebenfalls abgebrochen und die Adresse dieses Wortes n* auch in das Register B gebracht. In diesem Fall erfolgt TK-Alarm.

In das Register A wird der Inhalt des Wortes n*, der dem L-Feld der Maske entspricht, in die Binärstellen gebracht, die ebenfalls dem L-Feld entsprechen. Die Binärstellen, die dem Null-Feld der Maske entsprechen, werden im Register A auf Null gelöscht. Die Typenkennung des Registers A ergibt sich aus der Typenkennung des Registers H.

Stand in der Speicherzelle n ein Zahlwort (TK = 0 oder 1), so ist die Markenstelle des Wortes ohne Bedeutung. Vor dem Vergleich im Rechenwerk wird das 1. Bit dem 2. Bit angeglichen. In der Speicherzelle n bleibt dieses Wort unverändert.

Beispiel umseitig

Externcode: TDM

Interncode: 'EA'

belegt: Befehlswerk und Rechenwerk

Takte: 14,5p + 2,5

p = Anzahl der untersuchten Wörter in der Tabelle

Alarm:

TK-Alarm: wenn $\langle n + k\langle B\rangle\rangle_t \neq \langle D\rangle_t$

Sonstiges:

Bei ungleicher Typenkennung:

$\langle n + k\langle B\rangle\rangle_t \neq \langle D\rangle_t$

Ergebnis: siehe Ausführung

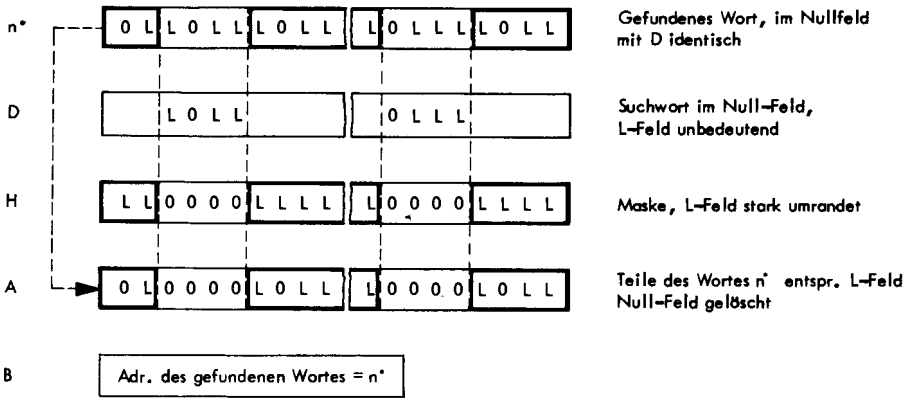
TK-Alarm

14,5p + 2,5 Takte

bei Dehnungswert gleich Null:

$\langle B\rangle = \pm 0$

Wirkung wie Leerbefehl (Nullbefehl)



TLD	n

Tabelle durchsuchen (auf gleich oder größer) mit Dehnung

TLD

Adressenteil: n = Adresse eines Ganzwortes (Anfangsadresse der Tabelle)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨D⟩ = Suchwort

$\langle D\rangle_1 = \langle D\rangle_2$ bei $\langle D\rangle_t = 0$ oder 1

⟨B⟩ = Dehnungswert = Adressenabstand der Wörter

$\langle B\rangle \neq \pm 0$

⟨a⟩ = Vergleichsoperand aus der Tabelle

Ausführung:

⟨a⟩ := ⟨n+k⟨B⟩⟩ $\langle a\rangle_1 := \langle a\rangle_2$	k = 0,1,2,... nur bei $t_a = 0$ oder 1 (im Speicher unverändert)
⟨a⟩ := ⟨a⟩ normalisiert ⟨D⟩ := ⟨D⟩ normalisiert	nur bei $t_b = 0$
Suchbedingung:	
⟨a⟩ ≥ ⟨D⟩	Die Art des Vergleichs wird durch t_b bestimmt
Abbruchkriterien:	
⟨a⟩ ≥ ⟨D⟩	Suchwort gefunden
$\langle a\rangle_t \neq \langle D\rangle_t$	Suchwort nicht gefunden, Abbruch wenn ein Wort auftritt, dessen TK ungleich der des Suchwortes ist. (TK-Alarm)
⟨B⟩ := n+k⟨B⟩	Adresse des Wortes, das zum Abbruch führte
⟨A⟩ := t_b ; ⟨D⟩	
TK-Alarm	nur wenn Suchwort nicht gefunden

Im Register D steht ein Suchwort. Enthält das Register D ein Zahlwort (TK = 0 oder 1), so darf es nicht über- oder untergelaufen sein. Bei Typenkennung = 0 wird der Inhalt des Registers D normalisiert.

Im Register B steht der Dehnungswert, d.h. der Abstand zwischen den Adressen zweier aufeinanderfolgender Wörter. Ist dieser Dehnungswert positiv, beginnt die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt vorwärts. Bei negativem Dehnungswert endet die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt rückwärts. Der Dehnungswert muß ungleich Null sein.

Die Tabelle wird auf jedes ⟨B⟩-te Wort durchsucht, das gleich oder größer ist als das Suchwort im Register D. Die Art des Vergleichs wird durch die TK des Suchwortes bestimmt. Der Suchvorgang wird abgebrochen, sobald ein Wort auftritt, welches der Suchbedingung genügt. Die Adresse des gefundenen Wortes steht im Register B.

Wird kein Wort gefunden und tritt ein Wort auf, dessen Typenkennung ungleich der Typenkennung des Registers D ist, so wird der Suchvorgang ebenfalls abgebrochen und die Adresse dieses Wortes auch in das Register B gebracht. In diesem Fall erfolgt TK-Alarm.

Im Register A steht einschließlich Typenkennung der Inhalt des Registers D (Suchwort).

Stand in der Speicherzelle n ein Zahlwort (TK = 0 oder 1), so ist die Markenstelle des Wortes ohne Bedeutung. Vor dem Vergleich im Rechenwerk wird das 1. Bit dem 2. Bit angeglichen. In der Speicherzelle n bleibt dieses Wort unverändert.

Externcode: TLD

Interncode: 'EB'

belegt: Befehlswerk und Rechenwerk

Takte: 14,5p + 10,5
p = Anzahl der untersuchten Wörter in der Tabelle

Alarm:

TK-Alarm: wenn $\langle a \rangle_t \neq \langle D \rangle_t$

Sonstiges:

Bei ungleicher Typenkennung:
$\langle a \rangle_t \neq \langle D \rangle_t$

Ergebnis:
$\langle B \rangle$:= n+k$\langle B \rangle$
$\langle A \rangle$:= t_0 ;$\langle D \rangle$
TK-Alarm
14,5p + 10,5 Takte

bei übergelaufenem Operand:
$\langle D \rangle_1 \neq \langle D \rangle_2$ bei $\langle D \rangle_t$ = 0 oder 1
falsche Befehlsausführung

bei Dehnungswert gleich Null:
$\langle B \rangle = \pm 0$
Wirkung wie Leerbefehl (Nullbefehl)

TLI	n

Tabelle durchsuchen auf Identität

TLI

Adressenteil: n = Adresse eines Ganzwortes (Anfangsadresse der Tabelle)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨D⟩ = Suchwort

$\langle D\rangle_1 = \langle D\rangle_2$ bei $\langle D\rangle_t = 0$ oder 1

⟨a⟩ = Vergleichsoperand aus der Tabelle

Ausführung:

⟨a⟩ := ⟨n+2k⟨B⟩⟩	k = 0,1,2...
$\langle a\rangle_1 := \langle a\rangle_2$	nur bei $t_a = 0$ oder 1 (im Speicher unverändert)
⟨a⟩ := ⟨a⟩ normalisiert ⟨D⟩ := ⟨D⟩ normalisiert	nur bei $t_b = 0$
Suchbedingung:	
⟨a⟩ = ⟨D⟩	Die Art des Vergleichs wird durch t_b bestimmt
Abbruchkriterien:	
⟨a⟩ = ⟨D⟩	Suchwort gefunden
$\langle a\rangle_t \neq \langle D\rangle_t$	Suchwort nicht gefunden, Abbruch wenn ein Wort auftritt, dessen TK ungleich der des Suchwortes ist (TK-Alarm)
⟨B⟩ := n+2k⟨B⟩	Adresse des Wortes, das zum Abbruch führte
⟨A⟩ := t_b ;⟨D⟩	
TK-Alarm	nur wenn Suchwort nicht gefunden

Im Register D steht ein Suchwort. Enthält das Register D ein Zahlwort (TK = 0 oder 1), so darf es nicht über- oder untergelaufen sein. Bei Typenkennung = 0 wird der Inhalt des Registers D normalisiert.

Die bei der Speicherzelle n beginnende Liste wird auf jedes Wort durchsucht, das mit dem Suchwort im Register D identisch ist. Die Art des Vergleichs wird durch die TK des Suchwortes bestimmt.

Der Suchvorgang wird abgebrochen, sobald ein Wort gefunden wird, welches mit dem Suchwort identisch ist. Die Adresse des gefundenen Wortes steht im Register B.

Wird kein Wort gefunden und tritt ein Wort auf dessen Typenkennung ungleich der Typenkennung des Registers D ist, wird der Suchvorgang ebenfalls abgebrochen. Die Adresse dieses Wortes wird auch in das Register B gebracht. in diesem Fall erfolgt TK-Alarm.

Im Register A steht einschließlich Typenkennung der Inhalt des Registers D (Suchwort).

Stand in der Speicherzelle n ein Zahlwort (TK = 0 oder 1), so ist die Markenstelle des Wortes ohne Bedeutung. Vor dem Vergleich im Rechenwerk wird das 1. Bit dem 2. Bit angeglichen. In der Speicherzelle n bleibt dieses Wort unverändert.

Externcode: TLI

Interncode: 'EC'

belegt: Befehlswerk und Rechenwerk

Takte: 15p + 10

p = Anzahl der untersuchten Wörter in der Tabelle

Alarm:

TK-Alarm: wenn $\langle a \rangle_t \neq \langle D \rangle_t$

Sonstiges:

Bei ungleicher Typenkennung:

$\langle a \rangle_t \neq \langle D \rangle_t$

Ergebnis:

$\langle B \rangle := n+2k\langle B \rangle$

$\langle A \rangle := t_p ; \langle D \rangle$

TK-Alarm

15p + 10 Takte

bei übergelaufenem Operand:

$\langle D \rangle_1 \neq \langle D \rangle_2$ bei $\langle D \rangle_t = 0$ oder 1

falsche Befehlsausführung

TLOG	n

Tabelle durchsuchen logarithmisch

TLOG

Adressenteil: n = Adresse eines Ganzwortes (Anfangsadresse der Tabelle)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Die Tabelle muß in ihrem Aufbau folgenden Forderungen genügen:

$\langle n\rangle_x < \langle D\rangle_x$ für $\langle H\rangle_x = 0$ Das erste Wort der Tabelle muß kleiner als das Suchwort sein.

$\langle n+2k\rangle_x \geq \langle n+2k\rangle_x$ für $\langle H\rangle_x = 0$ Die folgenden Wörter müssen gleich oder größer als das vorhergehende sein.

Tabelle mit Dehnung: siehe Rückseite

$\langle A\rangle$ = Tabellenlänge in Ganzwörtern ($2 \leq \langle A\rangle \leq 2^{20}$)
$\langle D\rangle$ = Suchwort
$\langle H\rangle$ = Maske

Suchbedingung:

$\langle n+2k\rangle_x \geq \langle D\rangle_x$	für $\langle H\rangle_x = 0$	k = 0,1,2,...$\langle A\rangle$ - 1 x = 1...48 entspr. Null-Feld im Register H

Abbruchkriterien:

$\langle n+2k\rangle_x \geq \langle D\rangle_x$	für $\langle H\rangle_x = 0$	Wort gefunden
alle $\langle n+2k\rangle_x < \langle D\rangle_x$	für $\langle H\rangle_x = 0$	Wort nicht gefunden. Alle durchsuchten Wörter in der Tabelle sind kleiner als das Suchwort.

Ausführung:

$\langle B\rangle := n^+$		n^+ = Adresse des gefundenen Wortes oder des ersten Wortes hinter der Tabelle.
$\langle D\rangle := t_0;\langle D\rangle_x$	für $\langle H\rangle_x = 0$	0-Feld der Maske
$\langle D\rangle_x := 0$	für $\langle H\rangle_x = L$	L-Feld der Maske
$\langle A\rangle_x := \langle n^+\rangle_x$	für $\langle H\rangle_x = L$	nur wenn Wort gefunden
$\langle A\rangle_x := 0$	für $\langle H\rangle_x = 0$	
$\langle A\rangle_t := \langle n^+\rangle_t$		
$\langle Q\rangle := t_A;\langle A\rangle$		
$\langle F\rangle := \langle F\rangle + 2$		nur wenn gefundenes Wort $\langle n^+\rangle$ größer als Suchwort $\langle D\rangle$
$\langle A\rangle := 3;+0$		nur wenn kein Wort gefunden wurde
$\langle Q\rangle := 3;+0$		
$\langle F\rangle := \langle F\rangle + 2$		
BÜ-Alarm		

Bei der Speicheradresse n beginnt eine Tabelle, deren erstes Wort kleiner als das Suchwort im Register D sein muß. Die folgenden Wörter müssen gleich oder größer als die vorhergehenden sein. Die Länge der Tabelle steht im Register A ($2 \leq \langle A\rangle \leq 2^{20}$). Im Register D steht das Suchwort. Das Register H enthält die Maske. Die Typenkennung der Register und der Tabellenwörter ist beliebig.

Die Tabelle wird mit Hilfe der Maske im Register H nach dem Intervallschachtelverfahren (logarithmisch) durchsucht. Es wird ein Bitmuster verglichen. Das zu suchende Teilwort wird als vorzeichenlose, positive und zusammenhängende Festkommazahl aufgefaßt. Das gesuchte Wort soll in den Binärstellen, an denen in der Maske Null steht, gleich oder größer sein, wie das Suchwort im Register D.

Der Suchvorgang wird abgebrochen, sobald das gesuchte Wort gefunden ist. Die Adresse des gefundenen Wortes n^+ wird in das Register B gebracht. Werden mehrere gleiche oder größere Wörter gefunden, so wird das kleinste Wort mit der niedrigsten Adresse ausgesucht und die Adresse n^+ in das Register B gebracht.

In das Register D werden in die Binärstellen, die dem Null-Feld der Maske entsprechen, die Binärstellen von n^+ gebracht, welche ebenfalls dem Null-Feld entsprechen. Die übrigen, dem L-Feld der Maske entsprechenden Binärstellen, werden im Register D auf Null gelöscht. Die ursprüngliche Typenkennung bleibt erhalten.

Externcode: TLOG

Interncode: 'ED'

belegt: Befehlswerk und Rechenwerk

Takte: $23q + 46$ $q = ({}_2\log\langle A\rangle)-1 \leq q < {}_2\log\langle A\rangle$

q = ganzzahlig

Alarm: BÜ-Alarm: wenn alle $\langle n+2k\rangle_x < \langle D\rangle_x$ für $\langle H\rangle_x = 0$

Ergebnis: siehe unter Ausführung
BÜ-Alarm

BÜ-Alarm: wenn $1 \leq \langle A\rangle > 2^{20}$
Die Tabelle wird nicht durchsucht
$\langle B\rangle$:= +0
sonst. Ergebnis: siehe unter Ausführung, kein Wort gefunden
BÜ-Alarm

Sonstiges: Fortsetzung von der Vorderseite:

In die Register A und Q wird in die Binärstellen, die dem L-Feld der Maske entsprechen, die Stellen des gefundenen Wortes n^+ gebracht, die ebenfalls dem L-Feld entsprechen. Die Binärstellen, die dem Null-Feld der Maske entsprechen, werden auf Null gelöscht. Beide Register A und Q erhalten die Typenkennung des gefundenen Wortes n^+.

Ist das gefundene Wort mit dem Suchwort des Registers D in den Binärstellen, die dem Null-Feld der Maske im Register H entsprechen, identisch, wird als nächster der auf TLOG folgende Befehl ausgeführt.

Wenn das gefundene Wort n^+ nicht gleich, sondern größer als das Suchwort ist, wird außerdem der Inhalt des Registers F (Befehlsfolgeregister) um 2 erhöht und gibt so die Adresse des übernächsten Befehls an. Der auf TLOG folgende Befehl wird übersprungen (Skip).

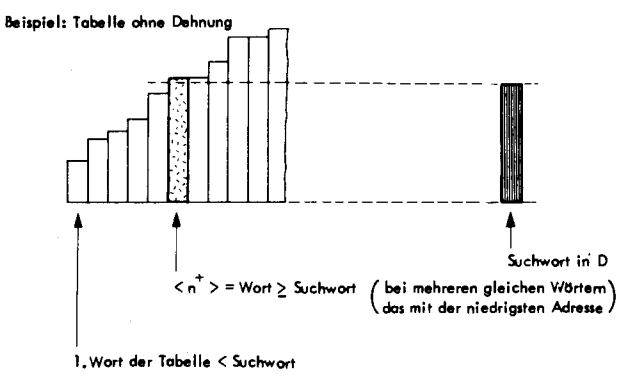
Wird kein Wort gefunden, wird der Suchvorgang abgebrochen. Die Register A und Q erhalten beide die Typenkennung = 3 und werden auf Null gelöscht. Das Register F wird um 2 erhöht (Skip). In diesem Fall erfolgt BÜ-Alarm.

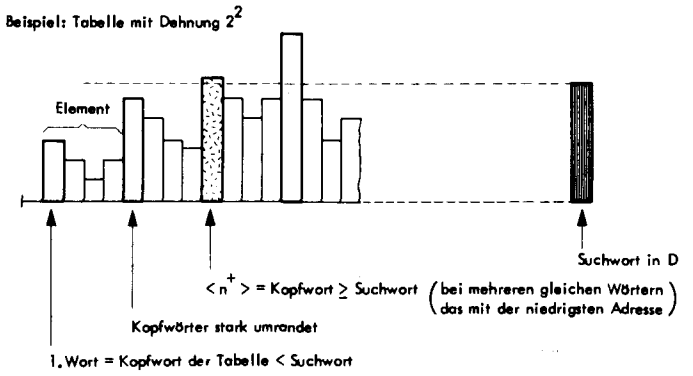
Dehnung der Tabelle:

Mit nachstehendem Tabellenaufbau kann die Tabelle auch mit Dehnung durchsucht werden:

Tabelle wird in gleichlange Elemente aufgeteilt,
Länge eines Elementes 2^e $e = 0,1,2,\ldots,$
Länge der Tabelle = $\langle A\rangle = 2^e \cdot$ Anzahl der Kopfwörter.
Das erste Wort im Element wird Kopfwort genannt.
Das Kopfwort muß gleich oder größer sein als die folgenden Wörter des Elementes.
Das erste Kopfwort muß kleiner als das Suchwort sein.
Die folgenden Kopfwörter müssen gleich oder größer als das jeweils vorhergehende Kopfwort sein.

Nur die Kopfwörter werden durch den Befehl nach der Suchbedingung logarithmisch durchsucht.
Der Abstand der untersuchten Kopfwörter ist eine Potenz von 2 (2^e).





TMAX	n

Tabelle durchsuchen auf Maximum

TMAX

Adressenteil: n = Adresse eines Ganzwortes (Anfangsadresse der Tabelle)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle H \rangle$ = Maske

$\langle B \rangle$ = Dehnungswert (Adressenabstand der Wörter)

$\langle B \rangle \neq \pm 0$

$\langle n + k\langle B \rangle \rangle_t = \langle n \rangle_t$

Suchbedingung:

$\langle n + k\langle B \rangle \rangle_x$ = Maximum	für $\langle H \rangle_x = 0$	k= 0,1,2... Die Tabelle besteht nur aus Worten $\langle n + k\langle B \rangle \rangle_t = \langle n \rangle_t$ x = 1...48 entspr. O-Feld im Register H

Abbruchkriterien:

$\langle n + k\langle B \rangle \rangle_x$ = Maximum	für $\langle H \rangle_x = 0$	Suchwort gefunden
$\langle n + k\langle B \rangle \rangle_t \neq \langle n \rangle_t$		Tabelle endet wenn ein Wort auftritt, dessen TK ungleich der TK in der Anfangsadresse der Tabelle ist. Dieses Wort gehört nicht mehr zur Tabelle.

Ausführung:

$\langle B \rangle := n^*$		n* = Adresse des zuerst gefund. Wortes
$\langle D \rangle_x := t_n ; \langle n^* \rangle_x$	für $\langle H \rangle_x = 0$	O-Feld der Maske
$\langle D \rangle_x := 0$	für $\langle H \rangle_x = L$	L-Feld der Maske
$\langle A \rangle_x , \langle Q \rangle_x := \langle n^* \rangle_x$	für $\langle H \rangle_x = L$	L-Feld der Maske
$\langle A \rangle_x , \langle Q \rangle_x := 0$	für $\langle H \rangle_x = 0$	O-Feld der Maske
$\langle A \rangle_t , \langle Q \rangle_t := \langle n^* \rangle_t$		

Im Register H steht eine Maske. Der Dehnungswert, d.h. der Abstand zwischen den Adressen zweier aufeinanderfolgender Wörter, steht im Register B. Ist dieser Dehnungswert positiv, beginnt die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt vorwärts. Bei negativem Dehnungswert endet die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt rückwärts. Der Dehnungswert muß ungleich Null sein.

Die Tabelle ist nur so lang wie Wörter auftreten, deren Typenkennung gleich der Typenkennung der Anfangsadresse n ist. Tritt ein Wort mit ungleicher Typenkennung auf, so gehört dieses Wort nicht mehr zur Tabelle und der Suchvorgang wird abgebrochen.

Die Tabelle wird auf jedes $\langle B \rangle$-te Wort durchsucht, welches in den Binärstellen, an denen in der Maske (Register H) das Null-Feld steht, das größte Bitmuster (Maximum) aufweist . Dieses Teilwort wird als positive Festkommazahl (ohne Vorzeichen) aufgefaßt. Gibt es mehrere solcher Wörter, so wird das erste gefundene Wort (mit der ersten Adresse in der Reihenfolge des Durchsuchens) ausgesucht und die Adresse n* in das Register B gebracht.

Der Inhalt des gefundenen Wortes $\langle n^* \rangle$ wird einschl. Typenkennung an den Binärstellen, die dem Null-Feld der Maske (Reg. H) entsprechen, in das Reg. D gebracht. Die dem L-Feld der Maske entsprechenden Binärstellen werden im Reg. D zu Null gelöscht.

In die Register A und Q wird der Inhalt des gefundenen Wortes $\langle n^* \rangle$, der dem L-Feld der Maske entspricht, einschließlich Typenkennung in die Binärstellen gebracht, die ebenfalls dem L-Feld entsprechen. Die dem Null-Feld entsprechenden Stellen werden zu Null gelöscht.

Beispiel umseitig

Externcode: TMAX

Interncode: 'EF'

belegt: Befehlswerk und Rechenwerk

Takte: $17p + 4$

p = Anzahl der untersuchten Wörter in der Tabelle

Alarm:

Sonstiges:

Bei ungleicher Typenkennung:

$\langle n + k\langle B\rangle\rangle_t \neq \langle n\rangle_t$

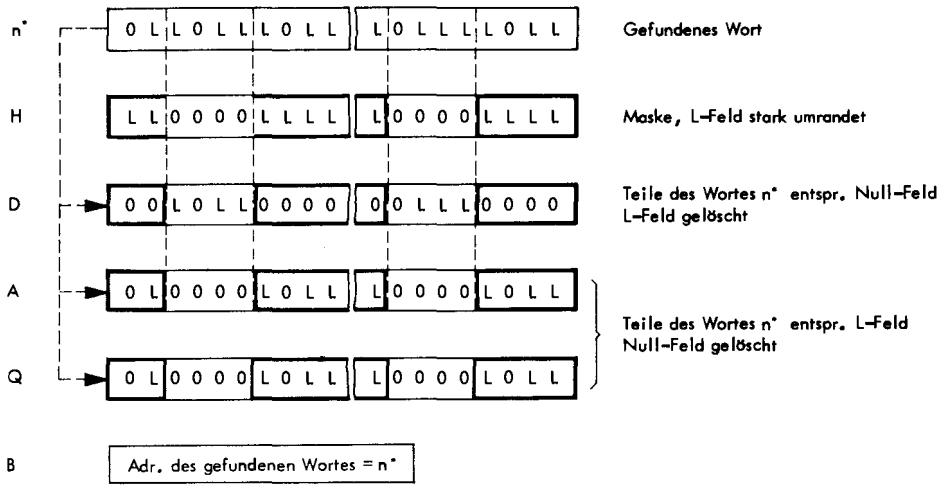
Ergebnis: siehe Ausführung

$17p + 4$ Takte

bei Dehnungswert gleich Null:

$\langle B\rangle = \pm 0$

Wirkung wie Leerbefehl (Nullbefehl)



TMIN	n

Tabelle durchsuchen auf Minimum

TMIN

Adressenteil: n = Adresse eines Ganzwortes (Anfangsadresse der Tabelle)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨H⟩ = Maske

⟨B⟩ = Dehnungswert (Adressenabstand der Wörter)

⟨B⟩ ≠ ±0

$\langle n + k\langle B\rangle\rangle_t = \langle n\rangle_t$

Suchbedingung:

$\langle n + k\langle B\rangle\rangle_x$ = Minimum	für $\langle H\rangle_x$ = 0	k = 0,1,2... Die Tabelle besteht nur aus Worten $\langle n + k\langle B\rangle\rangle_t = \langle n\rangle_t$ x = 1...48 entspr. 0-Feld im Register H

Abbruchkriterien:

$\langle n + k\langle B\rangle\rangle_x$ = Minimum	für $\langle H\rangle_x$ = 0	Suchwort gefunden
$\langle n + k\langle B\rangle\rangle_t \neq \langle n\rangle_t$		Tabelle endet wenn ein Wort auftritt, dessen TK ungleich der TK in der Anfangsadresse der Tabelle ist. Dieses Wort gehört nicht mehr zur Tabelle.

Ausführung:

⟨B⟩ := n *		n* = Adresse des zuerst gefund. Wortes
$\langle D\rangle_x := t_n ; \langle n^*\rangle_x$	für $\langle H\rangle_x$ = 0	0-Feld der Maske
$\langle D\rangle_x := 0$	für $\langle H\rangle_x$ = L	L-Feld der Maske
$\langle A\rangle_x, \langle Q\rangle_x := \langle n^*\rangle_x$	für $\langle H\rangle_x$ = L	L-Feld der Maske
$\langle A\rangle_x, \langle Q\rangle_x := 0$	für $\langle H\rangle_x$ = 0	0-Feld der Maske
$\langle A\rangle_t, \langle Q\rangle_t := \langle n^*\rangle_t$		

Im Register H steht eine Maske. Der Dehnungswert, d.h. der Abstand zwischen den Adressen zweier aufeinanderfolgender Wörter, steht im Register B. Ist dieser Dehnungswert positiv, beginnt die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt vorwärts. Bei negativem Dehnungswert endet die Tabelle bei der Speicherzelle n und die Durchsuchung erfolgt rückwärts. Der Dehnungswert muß ungleich Null sein.

Die Tabelle ist nur so lang wie Wörter auftreten, deren Typenkennung gleich der Typenkennung der Anfangsadresse n ist. Tritt ein Wort mit ungleicher Typenkennung auf, so gehört dieses Wort nicht mehr zur Tabelle und der Suchvorgang wird abgebrochen.

Die Tabelle wird auf jedes ⟨B⟩- Wort durchsucht, welches in den Binärstellen, an denen in der Maske (Register H) das Null-Feld steht, das kleinste Bitmuster (Minimum) aufweist. Dieses Teilwort wird als positive Festkommazahl (ohne Vorzeichen) aufgefaßt. Gibt es mehrere solcher Wörter, so wird das erste gefundene Wort (mit der ersten Adresse in der Reihenfolge des Durchsuchens) ausgesucht und die Adresse n* in das Register B gebracht.

Der Inhalt des gefundenen Wortes ⟨n*⟩ wird einschl. Typenkennung an den Binärstellen, die dem Null-Feld der Maske (Reg. H) entsprechen, in das Reg. D gebracht. Die dem L-Feld der Maske entsprechenden Binärstellen werden im Reg. D zu Null gelöscht.

In die Register A und Q wird der Inhalt des gefundenen Wortes ⟨n*⟩, der dem L-Feld der Maske entspricht, einschließlich Typenkennung in die Binärstellen gebracht, die ebenfalls dem L-Feld entsprechen. Die dem Null-Feld entsprechenden Stellen werden zu Null gelöscht.

Beispiel umseitig

Externcode: TMIN

Interncode: 'EE'

belegt: Befehlswerk und Rechenwerk

Takte: 17p + 4

p = Anzahl der untersuchten Wörter in der Tabelle

Alarm:

Sonstiges:

Bei ungleicher Typenkennung:

$\langle n + k\langle B\rangle\rangle_t \neq \langle n\rangle_t$

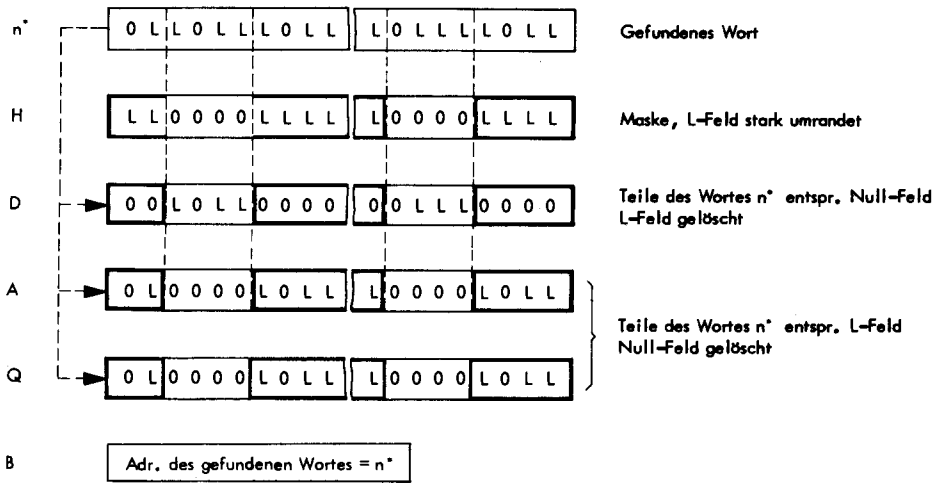
Ergebnis: siehe Ausführung

17p + 4 Takte

bei Dehnungswert gleich Null:

$\langle B\rangle = \pm 0$

Wirkung wie Leerbefehl (Nullbefehl)



TOK	z

Transportiere Oktaden

TOK

Adressenteil: z = Anzahl der zu transportierenden Oktaden (0 ... 65 535)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:
$z \neq 0$
$a = \langle H \rangle_{1-24}$: Oktadenadresse Zielgebiet
$q = \langle H \rangle_{25-48}$: Oktadenadresse Quellgebiet

Ganzwortadresse n
Ganzwort TK
Oktadenadresse a= 3n+0 3n+1 3n+2 3n+3 3n+4 3n+5

$\langle H \rangle_t = 3$: $(q+6) \geq a$ (Überlappung der Gebiete)
$\langle H \rangle_t = 2$: $q = 3n+0$ (Adresse der ersten Oktade im Ganzwort)

Typenbenennung des Ganzwortes im Speicher = 2 oder 3

Ausführung:
$\langle a+x \rangle := \langle q+x \rangle$ wenn $\langle H \rangle_t = 3$ Quellgebiet z Oktaden
$\langle a+x \rangle := \langle q+y \rangle$ wenn $\langle H \rangle_t = 2$ Quellgebiet 6 Oktaden in einem Ganzwort

Jedes betroffene Speicherwort erhält TK = 3

x = 0, 1, 2, ..., z-1
y = 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, ...

$\langle A \rangle$:= 3; Inhalt des letzten Ganzwortes im Zielgebiet
$\langle D \rangle$:= 3; $\langle A \rangle$

$\langle H \rangle$:= 3; $\langle A \rangle$ wenn $\langle H \rangle_t = 3$
$\langle H \rangle$:= 2; (a+z)q, wenn $\langle H \rangle_t = 2$: Oktadenadresse { des Quellgebietes $\langle H \rangle_{25-48}$ / des Zielgebietes+z $\langle H \rangle_{1-24}$

$\langle Q \rangle$:= 3; 0
$\langle Y \rangle$:= +0

$\langle B \rangle$:= Halbwortadresse des letzten Wortes im Zielgebiet

Dieser Befehl transportiert eine Oktadenfolge im Speicher vom Quellgebiet (Anfangsadresse = q) in das Zielgebiet (ab Anfangsadresse = a). Im Adressenteil des Befehls wird die Anzahl der zu transportierenden Oktaden (z) angeben.

Vor Ausführung des Befehls muß in das linke Halbwort des Registers H die Oktadenadresse des Zielgebietes (a) und in das rechte Halbwort die Oktadenadresse des Quellgebietes (q) gebracht werden. Die Typenkennung des Registers H (2 oder 3) gibt die Art des Transportes an. Bei $\langle H \rangle_t = 0$ oder 1 erfolgt undefinierte Befehlsausführung.

Bei $\langle H \rangle_t = 3$

werden z Oktaden aus dem Quellgebiet (beginnend mit der Anfangsadresse = q) in das Zielgebiet transportiert und dort (bei der Anfangsadresse = a beginnend) abgespeichert. Bei Überlappung von Quell- und Zielgebiet wird das Quellgebiet nur dann unverändert im Zielgebiet abgespeichert, wenn die Bedingung $(q+6) \geq a$ erfüllt ist.

Bei $\langle H \rangle_t = 2$

besteht das Quellgebiet nur aus einem Ganzwort mit 6 Oktaden. Die Anfangsadresse q des Quellgebietes ist die Adresse der ersten (linken) Oktade in diesem Ganzwort ($q = 3n+0$); q muß durch 6 teilbar sein. Der Transport beginnt mit der ersten Oktade (0), danach werden z Oktaden in das Zielgebiet (beginnen mit der Anfangsadresse = a) abgespeichert. Sind alle 6 Oktaden der "Quelle" aufeinanderfolgend abgespeichert und ist $z > 6$, werden fortlaufend, (wieder beginnend mit der Oktade 0), soviel Oktaden im Zielgebiet abgespeichert wie in z angegeben sind (0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, ...).

Nach Befehlsausführung steht in den Registern A und D, bei $\langle H \rangle_t = 3$ auch im Register H, der Inhalt des letzten Ganzwortes im Zielgebiet (in dem die letzte Oktade abgespeichert wurde). Die Typenkennung dieser Register ist =3. War $\langle H \rangle_t = 2$, so steht im Register H jedoch mit TK=2, im linken Halbwort die Oktadenadresse des Zielgebietes+z ($\langle H \rangle_{1-24}$:= a+z), im rechten Halbwort die Oktadenadresse des Quellgebietes ($\langle H \rangle_{25-48}$:= q).

Das Register Q wird mit Typenkennung = 3 auf Null gelöscht. Der Inhalt des Registers Y ergibt sich zu +0. Im Register B steht die Adresse des Halbwortes im Zielgebiet, in dem die letzte Oktade abgespeichert wurde.

Externcode: TOK

Interncode: 'FD'

belegt: Befehlswerk und Rechenwerk

Takte:

bei z = 0: 4 Takte

wenn im Zielgebiet nur in 1 Ganzwort abgespeichert wurde
- bei $\langle H \rangle_t = 3$: 191 Takte
- bei $\langle H \rangle_t = 2$: 209 Takte

wenn im Zielgebiet in mehr als 1 Ganzwort abgespeichert wurde
- bei $\langle H \rangle_t = 3$: $(229+30(\frac{z-1}{6}-1))$ Takte
- bei $\langle H \rangle_t = 2$: $(231+12(\frac{z-1}{6}-1))$ Takte

---

Alarm:

Sonstiges: Bei z = 0:

keine Veränderung im Speicher (Transport unterbleibt)
Register A, B und Y undefiniert
Abbruch nach 4 Takten

bei $\langle H \rangle_t = 3$ und $(q+6) < a$ (bei Überlappung von Ziel- und Quellgebiet):
Ergebnis undefinierter Transport

bei $\langle H \rangle_t = 2$ und $q \neq 3+0$ (q nicht durch 6 teilbar):
Ergebnis undefinierte Befehlsausführung

bei $\langle H \rangle_t = 0$ oder 1:
Ergebnis undefinierte Befehlsausführung

bei TK ≠ 2 oder 3 des Ganzwortes im Speicher:

Wurde aus dem Quellgebiet ein Zahlwort (TK 0 oder 1) ausgelesen, so wird während des Transportes (wie beim Befehl B) das 1. Bit dem 2. Bit angeglichen (Vorzeichenangleich). Das Markenregister wird bei einem markierten Zahlwort auf L gesetzt. Das Wort im Speicher bleibt unverändert.

TRX	s i	Transport aus Rechenwerk in Indexzelle	**TRX**

Adressenteil:

s_1 = Register A,Q,D,H oder leer
s_2 = leer = positiver Wert
N = negativer Wert
i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle s_1 \rangle_1 = \langle s_1 \rangle_v$

s_1 = A,Q,D oder H s_1 = leer

Ausführung:

$\langle i \rangle := \pm\langle s \rangle_{25-48}$ $\langle i \rangle := \pm 0$

$\langle B \rangle := \pm\langle s \rangle_{25-48}$ $\langle B \rangle := \pm 0$

Das rechte Halbwort des Rechenwerksregisters, welches durch die Spezifikation s angegeben wird, wird in die Indexzelle i und ins Register B gebracht.

Durch die zusätzliche Spezifikation N wird der Inhalt des angegebenen Registers mit umgekehrtem Vorzeichen in die Indexzelle i und ins Register B gebracht.

Wird keines der Register angegeben (s_1 = leer), so wird die Indexzelle i und das Register B auf Null gelöscht.

Externcode: TRX

Interncode: '96'

belegt: Befehlswerk und Rechenwerk

Takte: 6

Alarm:

Sonstiges:

Adressenteil intern:

s_1 : A = '0080'
Q = '0040'
D = '0020'
H = '0010'

} nur eine der Spezifikationen darf angegeben werden

s_2 : N = '0008'

i : = 'xx00'

xx = Indexadresse

TTX	i_L i_R	Tausch-Transport in Indexzellen	**TTX**

Adressenteil: i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle i_R \rangle$:=: $\langle i_L \rangle$
$\langle B \rangle$:= $\langle i_L \rangle$

Der Inhalt der Indexzelle i_R wird mit dem Inhalt der Indexzelle i_L vertauscht. Gleichzeitig wird in das Register B der Inhalt der Indexzelle i_L gebracht.

Externcode: TTX

Interncode: 'OD'

belegt: Befehlswerk

Takte: 11

Alarm:

Sonstiges:

TXR	s i	Transport aus Indexzelle nach Rechenwerk	**TXR**

Adressenteil:

s_1 = Register A,Q,D und H (mehrere in beliebiger Reihenfolge)

s_2 = leer = positiver Wert

N = negativer Wert

i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle i \rangle_1 = \langle i \rangle_v$

Ausführung:

$\langle s \rangle_1 := 1;v,\pm\langle i \rangle$

$\langle B \rangle := \pm\langle i \rangle$

Das 1. Bit in der Indexzelle wird als Vorzeichen betrachtet.

Der Inhalt der Indexzelle i wird in die im Spezifikationsteil bezeichneten Rechenwerksregister und ins Register B transportiert. Die vorderen 24 Bits in den Rechenwerksregistern werden dem Vorzeichen angeglichen.

Durch die Spezifikation N erfolgt der Transport mit umgekehrten Vorzeichen in die bezeichneten Rechenwerksregister.

Die Typenkennung in den Registern wird auf 1 gesetzt.

Externcode: TXR

Interncode: '8C'

belegt: Befehlswerk und Rechenwerk

Takte: 5

Alarm:

Sonstiges:

Adreßteil intern:

s_1 : A = '8000'
Q = '4000'
D = '2000'
H = '1000'

s_2 : N = '0800'

i : = '00xx'

xx = Indexadresse

TXX	i_L i_R	Transport aus Indexzelle in Indexzelle	**TXX**

Adressenteil: i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle i_L \rangle := \langle i_R \rangle$
$\langle B \rangle := \langle i_R \rangle$

Der Inhalt der Indexzelle i_R wird in die Indexzelle i_L und gleichzeitig in das Register B gebracht.

Externcode: TXX

Interncode: 'OC'

belegt: Befehlswerk

Takte: 4

Alarm:

Sonstiges:

US	s i	Umschlüsseln von Zeichen	**US**

Adressenteil:

s_1 = Bitlänge der Zeichen 6,8 oder C (12 Bits)

s_2 = E = ein Zeichen (rechtsbündig)
G = alle Zeichen (8,6 oder 4)

i = Adresse einer Indexzelle

bei mod2:	wird speziell modifiziert	bei R:	nicht zugelassen

Voraussetzung:

⟨A⟩ = Umschlüsselnde Zeichen

⟨i⟩ + mod2 = Anfangsadresse der Umschlüsselungstabelle (Halbwortadresse)

Tabelle = neuer Code in aufeinanderfolgenden Viertelwörtern rechtsbündig in aufsteigender Wertigkeit des alten Codes, beginnend beim Wert 0

Ausführung:

⟨B⟩ := ⟨i⟩ + mod2 — Anfangsadresse der Umschlüsselungstabelle

bei s_2 = E:

⟨A⟩ := ⟨A⟩ umgeschlüsselt
$\langle A\rangle_{1-x}$:= +0
$\langle A\rangle_t$:= $\langle n\rangle_t$

in A rechtsbündig stehendes Zeichen umgeschlüsselt,
A wird in den x-Stellen links mit Nullen aufgefüllt, x = 42,40 oder 36
$\langle n\rangle_t$ = TK des Tabellenwortes, in dem das umzuschlüsselnde Zeichen steht.

bei s_2 = G:

⟨A⟩ := ⟨A⟩ umgeschlüsselt
$\langle A\rangle_t$:= $\langle n\rangle_t$
⟨Q⟩ := t_A ;⟨A⟩
⟨Y⟩ := Anzahl der umgeschlüsselt. Zeichen

alle in A stehenden Zeichen sind umgeschlüsselt

$\langle n\rangle_t$ = TK des Tabellenwortes, welches das in A links stehende Zeichen enthält.

Die Umschlüsselungstabelle beginnt bei einem Halbwort. Das erste Viertelwort enthält rechtsbündig den neuen Code der zum alten Code mit der binären Wertigkeit 0 gehört, das zweite den, der zur Wertigkeit 1 gehört usw. Die Tabelle muß so lang sein, daß sie den Code mit der höchsten Wertigkeit aufnehmen kann, also maximal 64 Viertelwörter bei 6 Bits/Zeichen, 256 Viertelwörter bei 8 Bits/Zeichen und 4096 Viertelwörter bei 12 Bits/Zeichen. Die Typenkennung der Tabelle ist beliebig.

Bei s_2 = E wird nur ein Zeichen, das rechtsbündig im Register steht, durch den neuen Code überschrieben. Der Rest des Registers wird auf +0 gelöscht. Der Inhalt des Registers erhält die Typenkennung des Wortes, in dem das umgeschlüsselte Zeichen stand.

Bei s_2 = G werden alle Zeichen im Register A durch den neuen Code überschrieben. Der Inhalt des Registers erhält die Typenkennung des Wortes, in dem das linke Zeichen stand. Das Register Y wird auf den Wert 8,6 oder 4 gesetzt (unabhängig von der Bitlänge).

In beiden Fällen enthält das Register B die Anfangsadresse der Umschlüsselungstabelle.

Externcode: US

Interncode: 'E3'

belegt: Befehlswerk und Rechenwerk

Takte: s_2 = E: 32 s_2 = G: s_1 = 6: 176
8: 136
C: 96

Alarm:

Sonstiges:

Adressenteil intern:

s_1: 6 = '80xx'
8 = '40xx'
C = '20xx'

s_2: E = '0000'
G = '1000'

xx = Indexadresse

VAQ	z

Vorzeichenangleich zwischen A und Q

VAQ

Adressenteil: z = Zahl Der Adressenteil ist ohne Bedeutung, muß aber angegeben werden.

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle A \rangle_t = 1$

$\langle Q \rangle_1 = \langle Q \rangle_2$

Ausführung:

$\langle A,Q \rangle := \langle A \rangle + \langle Q \rangle \cdot 2^{-48}$

$\langle Q \rangle_{1,2} := \langle A \rangle_1$

$\langle Q \rangle_t := 1$

1. Erweitern einer Festkommazahl im Register A auf doppelte Wortlänge im Register A,Q. Dabei muß $\langle Q \rangle = 0$ sein.

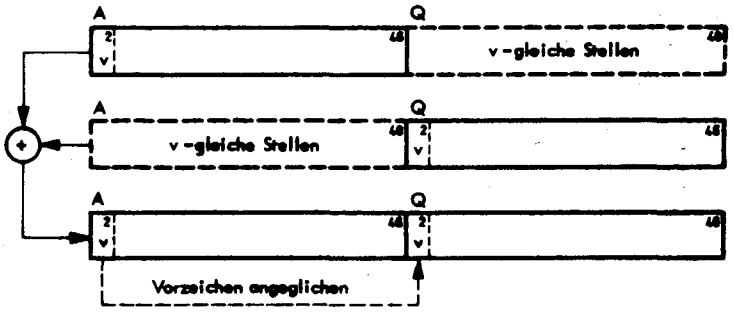
2. Erweitern einer Festkommazahl im Register Q auf doppelte Wortlänge im Register A,Q. Dabei muß $\langle A \rangle = 0$ sein.

3. Nach Operationen mit doppelter Wortlänge können die Vorzeichen des doppeltlangen Registers A,Q unterschiedlich sein, da beide Register getrennt verarbeitet werden.

VAQ gleicht die Vorzeichen an und berichtigt entsprechend das Ergebnis.

Die Kommastellung bleibt immer erhalten.

Das Register Q erhält die Typenkennung = 1.



Externcode: VAQ

Interncode: '63'

belegt: Rechenwerk

Takte: 2 bei $\langle A \rangle_1 = \langle Q \rangle_{1,2}$
17 bei $\langle A \rangle_1 \neq \langle Q \rangle_{1,2}$

Alarm:

TK-Alarm: wenn $\langle A \rangle_t \neq 1$

Sonstiges:

bei falscher Typenkennung:

$\langle A \rangle_t \neq 1$

Ergebnis: siehe unter Ausführung
TK-Alarm

bei $\langle Q \rangle$ über- oder untergelaufen:

$\langle Q \rangle_1 \neq \langle Q \rangle_2$ bei $\langle A \rangle_t = 1$

$\langle A,Q \rangle$:= undefiniert
$\langle Q \rangle_t$:= 1

VBA	z

Vermindere B um Adressenteil

VBA

Adressenteil: z: Zahl = 0...65 535

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle B \rangle - z$

Der Inhalt des Registers B wird um die Zahl z vermindert.

Externcode: VBA

Interncode: '13'

belegt: Befehlswerk

Takte: 8,5

Alarm:

Sonstiges:

VBC	m

Vermindere B um Speicher

VBC

Adressenteil: m = Adresse eines Halbwortes

bei mod2:	wird nicht modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle B \rangle - \langle m \rangle$

Der Inhalt des Registers B wird um den Inhalt der Speicheradresse m vermindert.

Externcode: VBC

Interncode: '15'

belegt: Befehlswerk

Takte: 8,5

Alarm:

Sonstiges:

VEL	n

VEL ("ODER"-Verknüpfung - Disjunktion)

VEL

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_m \end{array}\right\}$ bei t_n = 0 oder 1

$\langle A \rangle := t_{max} ; \langle A \rangle \vee \langle D \rangle$

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht.

Der Inhalt des Registers A wird mit dem Inhalt des Registers D durch die Boolesche Operation "ODER" verknüpft.

Das Ergebnis steht im Register A und erhält die größere der beiden Typenkennungen der Register A oder D.

Bei Zahlwörtern wird die Marke berücksichtigt.

Verknüpfung durch "ODER"

a := b ∨ c		
0	0	0
L	0	L
L	L	0
L	L	L

Externcode: VEL

Interncode: '68'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

VLA	z

VEL-Adressenteil ("ODER"-Verknüpfung)

VLA

Adressenteil: z: Zahl = 0...65 535

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨D⟩ := 1;0,z
⟨A⟩ := t_H ;⟨H⟩ ∨ ⟨D⟩

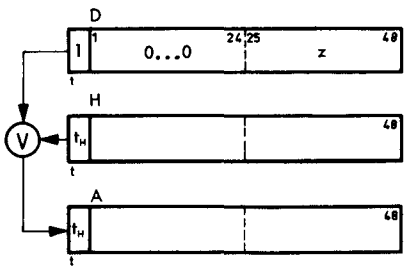
In die rechte Hälfte des Registers D (Binärstellen 25 - 48) wird der Wert z gebracht; die linke Hälfte des Registers (Binärstellen 1 - 24) ergibt sich zu 0, die Typenkennung zu 1.

Der Inhalt des Registers D wird mit dem Inhalt des Registers H durch die Boolesche Operation "ODER" (Disjunktion) verknüpft.

Das Ergebnis steht im Register A mit der Typenkennung des Registers H.

Verknüpfung durch "ODER"

a := b ∨ c		
0	0	0
L	0	L
L	L	0
L	L	L



Externcode: VLA

Interncode: '88'

belegt: Rechenwerk

Takte: 8

Alarm:

Sonstiges:

VMO	s_L s_R	Verändere Modus	VMO

Adressenteil:

s_L : Bits a_9 bis a_{16}
s_R : Bits a_{17} bis a_{24}

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

BEBY = L — System- oder Spezialmodus
oder BEBT1 = L
oder BEBT = BEWA = L
oder BEBT = BEWH = L
} Wartungsvariante

Ausführung:

a_{24} = L : ⟨F⟩ := ⟨B⟩

a_{23} = L : Das Eingriffswerk meldet solche anstehenden Eingriffe, die infolge der Stellung der Bits in den Kanalzuordnungsstellen bisher nicht durchkommen konnten, jetzt aber (d.h. nachdem eine Änderung der Bits der Kanal-Zuordnungsbits vorgenommen wurde) gemeldet werden können.

a_{22} = L : BEFE := L — Eingriffssperre setzen

a_{21} = L : ⟨B⟩ := ⟨BG⟩ — Uhrregister BG nach Register B

a_{20} = L : ⟨BTV⟩ := $\langle B \rangle_{3-24}$ — Prüfregister BTV setzen

a_{19} = L : ⟨BW⟩ := $\langle B \rangle_{9-24}$ — Weckerregister BW setzen

a_{18} = L : ⟨BT⟩ := ⟨B⟩ — Prüfregister BT setzen

a_{17} = L : ⟨BLZ2⟩ := $\langle B \rangle_{3-7}$
⟨BLZ1⟩ := $\langle B \rangle_{9-12}$
} Register BLZ1 und BLZ2 setzen (Δ-Werte ändern)

$\langle BPx \rangle_1$:= 0 — Seitenadreßregister ungültig setzen
x = 1, 2, 3, 4

a_{16} = L : ⟨BL⟩ := $\langle B \rangle_{1-16}$ — Leitadressenregister BL setzen

$\langle BPx \rangle_1$:= 0 — Seitenadreßregister ungültig setzen

$\langle BIx \rangle_1$:= 0 — Indexregister ungültig setzen
x = 1, 2, 3, 4

a_{15} = L : BEBO := a_{10} Normal- oder Abwicklermodus

a_{13} = L, : BEBY := L ; BEBN := 0 Systemmodus

a_{13} = 0, a_{12} = L : BEBY := L ; BEBN := L Spezialmodus

a_{13} = 0, a_{12} = 0, a_{11} = L : BEBY := 0 ; BEBN := L Normal-oder Abwicklermodus

a_{14} = L : BEBA := a_9 — Modus 24 (0) oder Modus 16 (L)

a_{13} bis a_{10} siehe unter a_{15}

a_9 siehe unter a_{14}

a_8 bis a_1 bedeutungslos

Alle Spezifikationen sind unabhängig voneinander. Die Spezifikation a_{21} = L wird zeitlich nach den übrigen Spezifikationen ausgeführt. Die Spezifikation a_{24} = L bedeutet, daß VMO als Sprungbefehl wirkt; dabei ist ein Sprung in eine andere Großseite möglich.

März 70

Externcode: VMO

Interncode: 'BF'

belegt: Befehlswerk

Takte: 3 wenn a_{21} = 0
10 wenn a_{21} = L

Alarm:

Sonstiges: Betriebsmodi:

BEBY	BEBN	BEBO	
0	L	0	Normalmodus
0	L	L	Abwicklermodus
L	L		Spezialmodus
L	0		Systemmodus

BEBA = 0 : Modus 24
= L : Modus 16

Voraussetzung nicht erfüllt:
Sprung in Nanoprogramm "Makro"
Beschreibung siehe unter "Makro"

VPU	s p

Verlasse Programmunterbrechung

VPU

Adressenteil:

p: 0... 255 Ort im Leitblock

s: Bits s_1 bis s_8

- s_1: Leitadressenregister BL neu setzen
- s_2: Leitblock des Systems/des Prozesses
- s_3: Rückkehr nach Eingriff oder Alarm/nach SSR oder Makro
- s_4: Indexbasisregister X neu setzen.
- s_5: Sperren lockern wie bei Befehl VSS
- s_6 bis s_8: bedeutungslos

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

BEBY = L System- oder Spezialmodus

bei der Ausführung von VPU tritt kein Speicherschutz-Alarm auf

a = Anfangsadresse des Leitblocks

Ausführung:

$\langle BIx \rangle$ zurückspeichern, wenn $\langle BIx \rangle_8$ = L
$\langle BIx \rangle_1$:= 0 Indexregister ungültig setzen
$\langle BPx \rangle_1$:= 0 Seitenadreßregister ungültig setzen
} x = 1, 2, 3, 4

$\langle BL \rangle$:= $\langle B \rangle_{1-16}$ wenn s_1 = L Leitadressenregister setzen

a := 0 wenn s_2 = 0 Leitblock des Systems
a := $\langle BL \rangle \cdot 2^8$ wenn s_2 = L Leitblock des Prozesses

Ⓐ

$\langle BLZ1 \rangle$:= $\langle \langle BL \rangle \cdot 2^8 + 64 \rangle_{8-12}$
$\langle BLZ2 \rangle$:= $\langle \langle BL \rangle \cdot 2^8 + 64 \rangle_{3-7}$
} wenn Steuerbit 46* = 0 { Δ-Werte des Operators wenn neuer Modus = Normalmodus

$\langle BLZ1 \rangle$:= $\langle \langle BL \rangle \cdot 2^8 + 64 \rangle_{32-36}$
$\langle BLZ2 \rangle$:= $\langle \langle BL \rangle \cdot 2^8 + 64 \rangle_{27-31}$
} wenn Steuerbit 46* = L { Δ-Werte des Abwicklers wenn neuer Modus = Abwicklermodus

$\langle X \rangle$:= $\langle 4 \rangle$ absolut wenn neuer Modus* = Systemmodus
$\langle X \rangle$:= $\langle a + 4 \rangle$ absolut wenn neuer Modus* ≠ Systemmodus
Berechnung der absoluten Indexbasisadresse ($\langle X \rangle$) erfolgt über:
Δ-Werte des Abwicklers wenn $\langle a+4 \rangle_t$ = 0 oder 2
Δ-Werte des Operators wenn $\langle a+4 \rangle_t$ = 1 oder 3
} wenn s_4 = L

Ⓑ

$\langle B \rangle$, $\langle BA \rangle$:= $\langle a+p+2 \rangle$
$\langle F \rangle$, Steuerbits 25-42, 44-48 := $\langle a+p+4 \rangle$
} Register und Steuerbits auf alten Wert setzen, BEFE bleibt unverändert

$\langle S \rangle$:= $\langle a+p \rangle$
$\langle BC \rangle$, Steuerbits 9-24, $\langle BH \rangle$:= $\langle a+p+6 \rangle$
wenn s_3 = 0 Rückkehr nach Eingriff oder Alarm

BEFE := 0 wenn BEFB = 0 und BEFA = 0 Eingriffssperre aufheben
BEFA := 0 wenn BEFB = 0 Alarmsperre 1 aufheben
BEFB := 0 Alarmsperre 2 aufheben
} wenn s_5 = L

wenn s_2 = L: Es wird an die Unterbrechungsstelle zurückgekehrt
Ausnahme: Nach Speicherschutzalarm in der Ausführungsphase einer der Befehle VPU, BCI, ZI, BL und PDP werden diese Befehle wiederholt

BEBF, BEBE, BEEF } bleiben bis zum Ende der Ausführungsphase des Befehls, der unterbrochen war und durch VPU fortgesetzt wird, unberücksichtigt.

Wenn s_3 = L ist, ist PVU ein Sprungbefehl auf den Befehl, der unmittelbar nach dem SSR oder Makros steht. Sprung in andere Großseite möglich.

* Für diese Bedingung gelten die Werte der Steuerbits, wie sie in der Speicherzelle $\langle a+p+4 \rangle$ stehen.

März 70

Externcode: VPU

Interncode: B1

belegt: Befehlswerk

Takte:

Sonstiges: bei der Ausführung tritt Speicherschutzalarm beim Berechnen der Indexbasis auf:

Ergebnis: siehe unter Ausführung bis Ⓐ

$\langle BLZ1\rangle := \langle\langle BL\rangle \cdot 2^8 + 64\rangle_{8-12}$
$\langle BLZ2\rangle := \langle\langle BL\rangle \cdot 2^8 + 64\rangle_{3-7}$
} wenn $\langle 4\rangle_t$ = 0 oder 2

$\langle BLZ1\rangle := \langle\langle BL\rangle \cdot 2^8 + 64\rangle_{32-36}$
$\langle BLZ2\rangle := \langle\langle BL\rangle \cdot 2^8 + 64\rangle_{27-31}$
} wenn $\langle 4\rangle_t$ = 1 oder 3

$\langle XB\rangle$:= undefiniert
$\langle XBZ\rangle$:= undefiniert
$\langle S\rangle_{1-24}$:= 3; $\langle 4\rangle_{1-24}$ relativ wenn neuer Modus* = Systemmodus
$\langle S\rangle_{1-24}$:= 3;$\langle a+4\rangle_{1-24}$ relativ wenn neuer Modus* ≠ Systemmodus
$\langle S\rangle_{25-48}$:= $\langle BA\rangle$
BEBO := 0 wenn $\langle a+4\rangle_t$ = 0 oder 2
BEBO := L wenn $\langle a+4\rangle_t$ = 1 oder 3

BEIC := L
BEEC := L
BEMB := 0
BEMU := 0

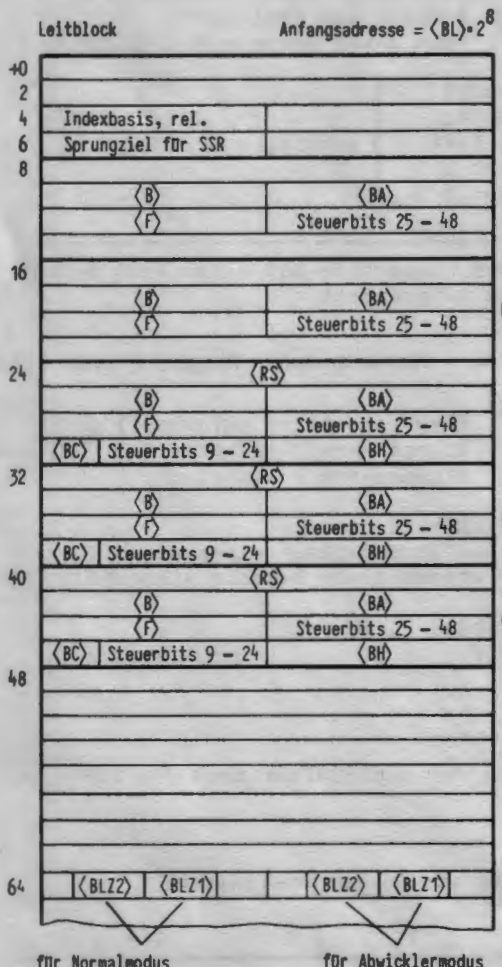
* Für diese Bedingung gelten die Werte der Steuerbits, wie sie in der Speicherzelle $\langle a+p+4\rangle$ stehen.

bei der Ausführung tritt Speicherschutzalarm beim Auslesen des rechten Befehlshalbwortes auf:

Ergebnis siehe unter Ausführung bis Ⓑ

$\langle S\rangle_{1-24}$:= 3; $\langle a+p+4\rangle_{1-24}$
$\langle S\rangle_{25-48}$:= $\langle BA\rangle$

BEBO := $\langle a+p+4\rangle_{46}$
BEIC := 0
BEEC := L
BEMB := 0
BEMU := 0
BEML := 0



BA: Adressenregister
BC: Coderegister
BH: Adressenhilfsregister
RS: Sammelregister

Nr.	Name	Bedeutung der Steuerbits
9	BEVA	In der Ausführungsphase wird ein Hauptspeicheroperand benötigt.
10	BESP	Die Dreierprobe darf nicht ersetzt werden.
11	BE30	Kennzeichnung verschiedener Sprungstellen aus dem Mikroprogramm der Ausführungsphase
12	BE20	
13	BE10	
14	BEAC	Der Befehl wurde im Abspeicher-Nanoprogramm unterbrochen.
15	BEAA	Der Befehl wurde am Anfang der Abrufphase unterbrochen.
16	BEAB	Der Befehl wurde in der Abrufphase bei Auslesen eines Operanden unterbrochen.
17	BEIC	Der Befehl wurde bei Bildung der absoluten Indexbasis unterbrochen.
18	BEMQ	Der Befehl ist nicht zu Ende
19	BEEH	Hauptalarm (Stromausfall bzw. -Abschaltung)
20	BEER4	Rechneralarm vom 4. Rechnerkern
21	BEER3	Rechneralarm vom 3. Rechnerkern
22	BEER2	Rechneralarm vom 2. Rechnerkern
23	BEER1	Rechneralarm vom 1. Rechnerkern
24	BEFT	Technischer Fehler
25	BEMA	Der Befehl MF, MCF oder MD geht vorher
26	BEMO	Der Befehl MFU oder MCFU geht vorher
27	BEMN	Der vorhergehende Befehl definiert mod2
28	BEMM	Der Befehl MM geht vorher (im Modus 16)
29	BEMU	Der Befehl MU geht vorher
30	BEMB	Der Befehl MABI geht vorher
31	BEML	Der Befehl LEI geht vorher
32	BEMP	Der anstehende Befehl ist ein Sprungbefehl
33	BEBE	Steuerbit: Stop vor Abrufphase
34	BEBA	Steuerbit: Modus 16
35	BEBT	Steuerbit: Wartungsmodus
36	BEBF	Steuerbit: Stop nach Abrufphase
37	REAL	Typenkennungsal-Alarm
38	REBUE	Arithmetischer Alarm
39	BEEC	Speicherschutz-Alarm
40	BEEU	Alarm: Überlauf des Registers U
41	BEEK	Befehls-Alarm
42	BEEF	Stop-Alarm
43	BEFE	Eingriffssperre
44	BEEW	Wecker-Alarm
45	BEED	Dreierproben-Alarm
46	BEBO	Abwicklermodus
47	BEBN	Normalmodus
48	BEBY	Systemmodus

VSS	m	Verändere Sperre und springe

VSS

Adressenteil: m Adresse eines Halbworts

bei mod2:	wird modifziert	bei R:	nicht zugelassen

Voraussetzung: BEBY = L System- oder Spezialmodus

Ausführung:

BEFE := 0 wenn BEFB = 0 und BEFA = 0 Eingriffssperre aufheben
BEFA := 0 wenn BEFB = 0 Alarmsperre 1 aufheben
BEFB := 0 Alarmsperre 2 aufheben
$\langle F \rangle_{9-24}$:= m

Ist eines der drei Steuerbits gesetzt, so wird es gemäß vorstehenden Bedingungen gelöscht. Es wird in jedem Fall ein Sprung ausgeführt.

Der nächste Befehl ist der in der Speicherzelle m.

März 70

Externcode: VSS

Interncode: '37'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges: Steuerbits:

BEFA = L : Alarmsperre 1 gesetzt
BEFB = L : Alarmsperre 2 gesetzt
BEFE = L : Eingriffssperre gesetzt

Voraussetzung nicht erfüllt:

Sprung in das Nanoprogramm "Makro"
Beschreibung siehe unter "Makro"

VSS	m

Verändere Sperre und springe

VSS

Adressenteil: m Adresse eines Halbworts

bei mod2:	wird modifziert	bei R:	nicht zugelassen

Voraussetzung: BEBY = L System- oder Spezialmodus

Ausführung:

BEFE := 0 wenn BEFB = 0 und BEFA = 0 Eingriffssperre aufheben
BEFA := 0 wenn BEFB = 0 Alarmsperre 1 aufheben
BEFB := 0 Alarmsperre 2 aufheben
$\langle F \rangle_{0-23}$:= m

Ist eines der drei Steuerbits gesetzt, so wird es gemäß vorstehenden Bedingungen gelöscht. Es wird in jedem Fall ein Sprung ausgeführt.

Der nächste Befehl ist der in der Speicherzelle m.

Externcode: VSS

Interncode: '37'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges: Steuerbits:

BEFA = L : Alarmsperre 1 gesetzt
BEFB = L : Alarmsperre 2 gesetzt
BEFE = L : Eingriffssperre gesetzt

Voraussetzung nicht erfüllt:

Sprung in das Nanoprogramm "Makro"
Beschreibung siehe unter "Makro"

VXX	i_L i_R	Vermindere Indexzelle um Indexzelle	**VXX**

Adressenteil: i = Adresse einer Indexzelle

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle B \rangle := \langle i_R \rangle - \langle i_L \rangle$
$\langle i_R \rangle := \langle i_R \rangle - \langle i_L \rangle$

Der Inhalt der Indexzelle i_R wird um den Inhalt der Indexzelle i_L vermindert. Das Ergebnis kommt gleichzeitig ins Register B.

Externcode: VXX

Interncode: '2F'

belegt: Befehlswerk

Takte: 14,5

Alarm:

Sonstiges:

WB	z	Wartebefehl	**WB**

Adressenteil: z: Anzahl der Uhrimpulse
(0...65 535)

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

keine Wirkung

Dieser Befehl hat nur eine veränderliche Ausführungszeit, die von der Größe z im Adressenteil des Befehls abhängt.

In der Ausführungsphase wird bei jedem Uhrtakt (alle 10 µs) der Wert z der im Adressenteil angegeben ist, um 1 vermindert. Nach jeder Subtraktion von 1 kann der Befehl durch Eingriff oder Alarm unterbrochen werden.

Okt.

Externcode: WB

Interncode: 'F8'

belegt: Befehls- und Rechenwerk

Takte: für $z = 0$: 3 Takte
für $z \neq 0$: $(10z-5)\mu s$

Alarm:

Sonstiges:

WTR	i_L i_R

Wortgruppentransport rückwärts

WTR

Adressenteil: i_L = Adresse der Indexzelle, welche die Endadresse des Zielbereiches enthält

i_R = Adresse der Indexzelle, welche die Endadresse des Quellenbereiches enthält

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle$ = Länge der Wortgruppe in Ganzwörtern

$\langle i_L \rangle$ = Zielbereich-Endadresse

$\langle i_R \rangle$ = Quellenbereich-Endadresse

} größte Adressen der Bereiche

Ausführung:

$$\langle \langle i_L \rangle - 2k \rangle := \langle \langle i_R \rangle - 2k \rangle \qquad k = 0,1,2,\ldots(\langle B \rangle - 1)$$

$$\langle B \rangle := \langle i_L \rangle - 2(\langle B \rangle - 1)$$

Es werden alle Worte des Quellenbereiches - beginnend am Ende der Bereiche - in den Zielbereich gebracht.

Die Länge der Wortgruppe muß vor der Ausführung im Register B sein und die beiden Endadressen der Bereiche in den Indexzellen i_L und i_R.

Nach der Ausführung enthält das Register B die Anfangsadresse des Zielbereiches.

Externcode: WTR

Interncode: '23'

belegt: Befehlswerk

Takte: 21a + 10

a = Anzahl der Ganzwörter in der Wortgruppe

Alarm:

Sonstiges:

WTV	i_L i_R	Wortgruppentransport vorwärts	**WTV**

Adressenteil:

i_L = Adresse der Indexzelle, welche die Anfangsadresse des Zielbereiches enthält

i_R = Adresse der Indexzelle, welche die Anfangsadresse des Quellenbereiches enthält

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle B \rangle$ = Länge der Wortgruppe in Ganzwörtern

$\langle i_L \rangle$ = Zielbereich-Anfangsadresse

$\langle i_R \rangle$ = Quellenbereich-Anfangsadresse

} kleinste Adressen der Bereiche

Ausführung:

$$\langle \langle i_L \rangle + 2k \rangle := \langle \langle i_R \rangle + 2k \rangle \qquad k = 0,1,2,\ldots(\langle B \rangle - 1)$$

$$\langle B \rangle := \langle i_L \rangle + 2(\langle B \rangle - 1)$$

Es werden alle Worte des Quellenbereiches in den Zielbereich gebracht - beginnend am Anfang der Bereiche.

Die Länge der Wortgruppe muß vor der Ausführung im Register B sein und die beiden Anfangsadressen der Bereiche in den Indexzellen i_L und i_R.

Nach der Ausführung enthält das Register B die Endadresse des Zielbereiches.

Externcode: WTV

Internode: '22'

belegt: Befehlswerk

Takte: 21a + 10

a = Anzahl der Ganzwörter in der Wortgruppe

Alarm:

Sonstiges:

XB	i

Index: Bringe

XB

Adressenteil: i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := ⟨i⟩

Der Inhalt der Indexzelle i wird in das Register B gebracht.

Externcode: XB

Interncode: '08' Adreßteil: '40xx' (siehe unten)

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

Adreßteil intern:

'40xx'

xx = Indexadresse

XBA	z

Index: Bringe Adressenteil

XBA

Adressenteil: z = Zahl 0...65 535

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := z

Die Zahl z wird in das Register B gebracht.

Externcode: XBA

Interncode: '01'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

XBAN	z

Index: Bringe Adressenteil negativ

XBAN

Adressenteil: z = Zahl 0...65 535

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨B⟩ := -z

Die Zahl z wird mit negativem Vorzeichen in das Register B gebracht.

Externcode: XBAN

Interncode: '19'

belegt: Befehlswerk

Takte: 2

Alarm:

Sonstiges:

XC	i

Index: Speichere

XC

Adressenteil: i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨i⟩ := ⟨B⟩

Der Inhalt des Registers B wird in die Indexzelle i gebracht.

Externcode: XC

Internode: '18' Adreßteil: '00xx' (siehe unten)

belegt: Befehlswerk

Takte: 3

Alarm:

Sonstiges:

Adreßteil intern:

'00xx'

xx = Indexadresse

XCN	i

Index: Speichere negativ

XCN

Adressenteil: i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨i⟩ := -⟨B⟩

Der Inhalt des Registers B wird mit umgekehrtem Vorzeichen in die Indexzelle i gebracht.

Externcode: XCN

Interncode: '18' Adreßteil: '80xx' (siehe unten)

belegt: Befehlswerk

Takte: 5

Alarm:

Sonstiges:

Adreßteil intern:

'80xx'

xx = Indexadresse

Y	z

Y

Adressenteil:

z = Bitmuster, siehe unter Ausführung

bei modi:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

z : Bits a_9 bis a_{15}

BEBY = L — System- oder Spezialmodus

oder BEBT1 = L
oder BEBT = BEWA = L
oder BEBT = BEWH = L
} Wartungsvariante

Ausführung:

a_9 = 0: xBSAK := L — Anstoß des mit s_R angegebenen EA-Kanals
x = s_R

für s_R > 31 : keine Ausführung

a_{13} = L: BEYRn := L im Rechnerkern 1
a_{14} = L: BEYRn := L im Rechnerkern 2
a_{15} = L: BEYRn := L im Rechnerkern 3
[a_{16} = L: BEYRn := L im Rechnerkern 4]

n = Rechnerkern, in dem man sich befindet (n = 1, 2, 3, [4])

a_9 = L: Spezifikation siehe gesonderte Beschreibung der Prüfein- und ausgabe
andere Bits bedeutungslos

Externcode: Y

Interncode: 'B4'

belegt: a_9 = 0: Befehlswerk a_9 = L:Befehls und Rechenwerk

Takte: a_9 = 0: 2 a_9 = L;Zeit, die für die Prüfein- und -ausgabe benötigt wird

---

Alarm:

Sonstiges:

Voraussetzung nicht erfüllt:

Sprung ins Nanoprogramm "Makro"
Beschreibung siehe unter "Makro"

ZDP	n	Setze Dreierprobe	ZDP

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

DP = Dreierprobenbits von ⟨n⟩

BEBY = L — System- oder Spezialmodus

oder BEBT1 = L

oder BEBT = BEWA = L

oder BEBT = BEWH = L — Wartungsvariante

Ausführung:

$\langle n \rangle := t_A ; \langle A \rangle$

$\langle n \rangle_{DP} :=$ LL

Die Dreierprobe (= LL) wird beim Abspeichern nicht wie üblich ersetzt sondern geprüft!

Die Dreierprobe ergänzt ein Wort derart, daß es ohne Rest durch drei teilbar ist. Bei Befehlen, bei denen die Dreierprobe nicht mitgerechnet wird, wird die Dreierprobe auf den Wert LL gesetzt. Dies bedeutet, daß die Dreierprobe neu zu bilden ist.

Beim Befehl ZDP wird die Dreierprobe des Wortes im Register A auf den Wert LL gesetzt. Die Dreierprobe wird nicht neu gebildet, sondern beim Abspeichern geprüft, wobei LL mit 00 gleichzusetzen ist. Das Wort wird mit LL abgespeichert.

Ist LL nicht die richtige Dreierprobe, so wird ein Dreierprobenalarm erzeugt, und zwar beim Abspeichern und auch bei einem späteren Auslesen.

Achtung!

Durch diesen Befehl kann ein Dreierprobenalarm erzeugt werden, der sonst nur bei defektem Rechner auftritt. Bei der Anwendung dieses Befehls ist es auf jeden Fall ratsam, sich vorher über die Folgen eines Dreierprobenalarms zu informieren.

Externcode: ZDP

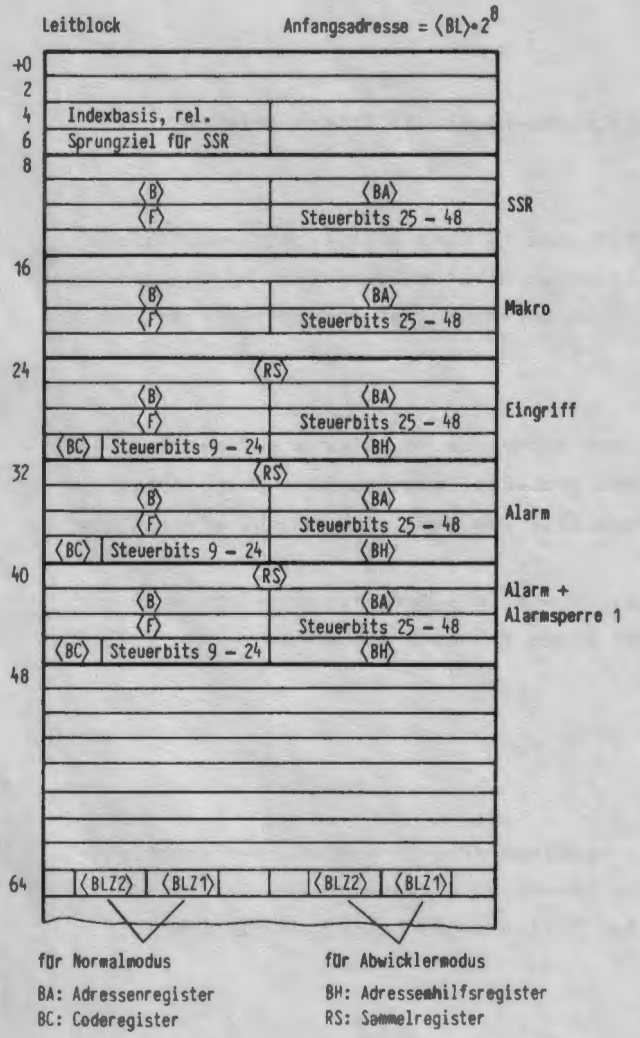
Interncode: 'D2'

belegt: Rechenwerk und Befehlswerk

Takte: 9

Alarm: DP-Alarm: wenn Dreierprobe LL (entspricht 00) für das Wort im Register A nicht zutrifft
BEED := L

Sonstiges:



BA: Adressenregister
BC: Coderegister
BH: Adressenhilfsregister
RS: Sammelregister

Nr.	Name	Bedeutung der Steuerbits
9	BEVA	In der Ausführungsphase wird ein Hauptspeicheroperand benötigt.
10	BESP	Die Dreierprobe darf nicht ersetzt werden.
11	BE30	Kennzeichnung verschiedener Ansprungsstellen aus dem Mikroprogramm der Ausführungsphase
12	BE20	
13	BE10	
14	BEAC	Der Befehl wurde im Abspeicher-Nanoprogramm unterbrochen.
15	BEAA	Der Befehl wurde am Anfang der Abrufphase unterbrochen.
16	BEAB	Der Befehl wurde in der Abrufphase bei Auslesen eines Operanden unterbrochen.
17	BEIC	Der Befehl wurde bei Bildung der absoluten Indexbasis unterbrochen.
18	BEMQ	Der Befehl ist nicht zu Ende
19	BEEH	Hauptalarm (Stromausfall bzw. -Abschaltung)
20	BEER4	Rechneralarm vom 4. Rechnerkern
21	BEER3	Rechneralarm vom 3. Rechnerkern
22	BEER2	Rechneralarm vom 2. Rechnerkern
23	BEER1	Rechneralarm vom 1. Rechnerkern
24	BEFI	Technischer Fehler
25	BEMA	Der Befehl MF, MCF oder MD geht vorher
26	BEMO	Der Befehl MFU oder MCFU geht vorher
27	BEMN	Der vorhergehende Befehl definiert mod2
28	BEMM	Der Befehl MM geht vorher (im Modus 16)
29	BEMU	Der Befehl MU geht vorher
30	BEMB	Der Befehl MABI geht vorher
31	BEML	Der Befehl LEI geht vorher
32	BEMP	Der anstehende Befehl ist ein Sprungbefehl
33	BEBE	Steuerbit: Stop vor Abrufphase
34	BEBA	Steuerbit: Modus 16
35	BEBT	Steuerbit: Wartungsmodus
36	BEBF	Steuerbit: Stop nach Abrufphase
37	REAL	Typenkennungsal-Alarm
38	REBUE	Arithmetischer Alarm
39	BEEC	Speicherschutz-Alarm
40	BEEU	Alarm: Überlauf des Registers U
41	BEEK	Befehls-Alarm
42	BEEF	Stop-Alarm
43	BEFE	Eingriffssperre
44	BEEW	Wecker-Alarm
45	BEED	Dreierproben-Alarm
46	BEBO	Abwicklermodus
47	BEBN	Normalmodus
48	BEBY	Systemmodus

ZI	m

Setze Indexbasis

ZI

Adressenteil: m = Adresse eines Halbwortes in dem die Indexbasisadresse steht

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle m \rangle$ = Anfangsadresse des Indexbereiches

Ausführung:

$\langle X \rangle := \langle m \rangle_{3-24}$

In dem Halbwort mit der Adresse m muß die gewünschte Anfangsadresse des Indexbereiches stehen.

Die Anfangsadresse des Indexbereiches wird in das Indexbasisregister X gebracht.

Externcode: ZI

Interncode: 'B7'

belegt: Befehlswerk

Takte: 17,5x + 58 x = Anzahl der Register, welche zurückgespeichert werden

---

Alarm:

Sonstiges:

Adressenteil:

s_1 : B = Bringen (leer = Speichern siehe ZK/2)
s_2 : leer oder I = Adresse nicht erhöhen oder Adresse erhöhen um p (Inkrement)
s_3 : leer oder V = nicht verändern oder verändern
s_4 : leer oder R = Oktadenadresse in Indexzelle oder Register
s_5 : p = Anzahl der Oktaden (1...12)
i : Indexadresse; wenn s_4 = leer
Register D, H oder B; wenn s_4 = R

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

a = Oktadenadresse
= ⟨i⟩ wenn s_4 = leer
= $\langle D \rangle_{25-48}$ wenn s_4 = R und i = D
= $\langle H \rangle_{25-48}$ wenn s_4 = R und i = H
= ⟨B⟩ wenn s_4 = R und i = B

Ganzwortadresse n
Ganzwort TK
Oktadenadresse a= 3n+0 3n+1 3n+2 3n+3 3n+4 3n+5

Typenkennung des Ganzwortes im Speicher, in dem die Oktaden gem. a stehen = 2 oder 3
p = 1 ... 12
v = Vorzeichenbits: O wenn bei der ersten Oktade die linken Bits ≠ LLLL
L wenn bei der ersten Oktade die linken Bits = LLLL

Ausführung:

⟨A⟩ := 3; v, p Oktaden gemäß a
⟨A⟩ := 3; v, ⟨A⟩ Oktaden in Tetraden verwandelt gem. Tabelle
} wenn s_3 = V wenn p ≤ 6

⟨A,Q⟩ := 3, 3; v, p Oktaden gemäß a
⟨A⟩ := 3; v, ⟨A,Q⟩ Oktaden in Tetraden verwandelt gem. Tabelle
} wenn s_3 = V wenn p > 6

⟨Q⟩ := t_A ; ⟨A⟩

a := a + p wenn s_2 = I

⟨D⟩ := 3; O,a
⟨H⟩ := 3; O,a
⟨B⟩ := a
⟨i⟩ := a wenn s_4 = leer
⟨Y⟩ := +O

Gemäß Oktadenadresse a werden p Oktaden rechtsbündig in das Register A bzw. in das doppelt lange Register AQ gebracht.

Sind die linken 4 Bits der linken Oktade L, so wird der Rest des Registers mit L aufgefüllt. Dies ermöglicht es, Gruppen von 2 Tetraden zu bringen, wobei die linke Tetrade als Vorzeichen aufgefaßt wird und der Rest des Registers dem Vorzeichen angeglichen wird. (Für Oktaden bedeutet dies, daß entweder mit dem Oktadenwert 0 oder 255 aufgefüllt wird.)

Wird die Spezifikation V angegeben, so werden die p Oktaden gemäß den Tabellen A, B und C zu Tetraden verkürzt. Bei nicht in der Tabelle angegebenen Oktaden erfolgt fehlerhafte Ausführung. Der Rest des Registers wird mit "0" aufgefüllt. Gehörte die rechte Oktade zur Tabelle C, so wird anschließend das Register in allen Stellen invertiert. Die Tetradenfolge stellt eine in Tetraden codierte Dezimalzahl dar. Die negative Zahl wird durch das B-1-Komplement dargestellt. Die linke Tetrade ist das Vorzeichen; es ist dann p mit maximal 11 anzugeben.

Wird die Spezifikation I angegeben, so wird die Oktadenadresse um p erhöht.

Die Oktadenadresse wird, ggf. um p erhöht, in die Register D, H und B gebracht.

War s_4 = leer, steht die Oktadenadresse in der Indexzelle. Das Register Y wird auf +O gelöscht.

Tabelle auf der Rückseite

Externcode: ZK/1 (Bringen)

Interncode: 'FC'

belegt: Rechen- und Befehlswerk

Takte:

- p = 0: 3 Takte
- bei s_1 = B:
  - und p ≤ 6:
    - wenn s_3 = leer: 122 Takte
    - wenn s_3 = V: 158 Takte
  - und p > 6:
    - wenn s_3 = leer: 146 Takte
    - wenn s_3 = V: 207 Takte
- hinzu kommen:
  - für s_4 = leer und s_2 = leer: 3 Takte
  - für s_4 = leer und s_2 = I: 10 Takte

Alarm:

Sonstiges: Bei TK ≠ 2 oder 3 des Ganzwortes im Speicher (in dem die Oktaden gemäß a stehen):

Erfolgt aus einem markiertem Zahlwort (TK 0 oder 1) ein Transport, so wird (wie beim Befehl B) das 1. Bit dem 2. Bit angeglichen (Vorzeichenangleich). Ist das Zahlwort markiert, so wird das Markenregister auf L gesetzt. Das Wort im Speicher bleibt unverändert.

bei p = 0: Register A, Q und B undefiniert
Abbruch nach 3 Takten

bei p > 12: undefinierte Befehlsausführung

sonstige Voraussetzung nicht erfüllt:
undefinierte Befehlsausführungen

Adressenteil intern:

s_1 : B = '8000'
s_2 : I = '4000'
s_3 : V = '2000'
s_4 : R = '1000'
s_5 : p = '0x00' x = p = Anzahl der Oktaden 1...12

i : '00yy' wenn s_4 ≠ R; yy = Indexadresse
i : D '0020' } wenn s_4 = R
H '0010' }
B '0008' }

Tabelle →

Tetrade	A		B		C	
	Oktade		Oktade		Oktade	
	ZC1	Karte	ZC1	Karte	ZC1	Karte
'0'	0	0	PZ	+0	MZ	-0
'1'	1	1	A	+1	J	-1
'2'	2	2	B	+2	K	-2
'3'	3	3	C	+3	L	-3
'4'	4	4	D	+4	M	-4
'5'	5	5	E	+5	N	-5
'6'	6	6	F	+6	O	-6
'7'	7	7	G	+7	P	-7
'8'	8	8	H	+8	Q	-8
'9'	9	9	I	+9	R	-9
'0'	SP	leer				

ZC1 = Zentralcode

Dezimalziffern - Tetraden

	positive Zahlen	negative Zahlen
Vorzeichen	0000	LLLL
0	0000	LLLL
1	000L	LLLO
2	00LO	LLOL
3	00LL	LLOO
4	0LOO	LOLL
5	0LOL	LOLO
6	0LLO	LOOL
7	0LLL	LOOO
8	LOOO	0LLL
9	LOOL	0LLO

ZK	s i	Zeichenkettenverarbeitung (Speichern)	ZK/2

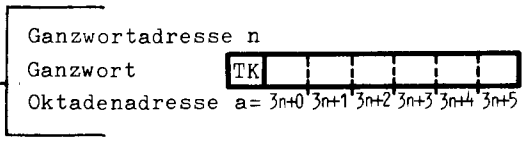
Adressenteil:

s_1 : leer = Speichern (B = Bringen siehe ZK/1)
s_2 : leer oder I = Adresse nicht erhöhen oder Adresse erhöhen um p (Inkrement)
s_3 : leer oder V = nicht verändern oder verändern
s_4 : leer oder R = Oktadenadresse in Indexzelle oder in Register
s_5 : p = Anzahl der Oktaden (1...12)
i : Indexadresse; wenn s_4 = leer
Register D, H oder B; wenn s_4 = R

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

a = Oktadenadresse = ⟨i⟩ wenn s_4 = leer
= $\langle D \rangle_{25-48}$ wenn s_4 = R und i = D
= $\langle H \rangle_{25-48}$ wenn s_4 = R und i = H
= ⟨B⟩ wenn s_4 = R und i = B
p = 1...12
v = Vorzeichenbits: 0 wenn linke Tetrade ≠ LLLL
L wenn linke Tetrade = LLLL
⟨A⟩ = t; v, p Tetraden wenn s_3 = V
Bei p < 12 muß das Reg. A links mit vorzeichengleichen Tetraden aufgefüllt sein.



Ausführung:

⟨A⟩ := -⟨A⟩
⟨A⟩ := ⟨A⟩ wenn p ≤ 6
⟨A,Q⟩ := ⟨A⟩ wenn p > 6

} umgewandelt von Tetraden in Oktaden gem. Tabelle A, rechte Tetrade gem. Tabelle C } Wenn s_3 = V und v = LLLL

⟨a + ⟩ := Oktaden x von ⟨A⟩
jedes betroffene Speicherwort erhält TK = 3 } wenn p ≤ 6

⟨a + x⟩ := Oktaden x von ⟨A,Q⟩
jedes betroffene Speicherwort erhält TK = 3 } wenn p > 6

} x: 0, 1, 2, ... p-1

a := a+p wenn s_2 = I

⟨A⟩ := 3; 0
⟨Q⟩ := 3; 0
⟨D⟩ := 3; 0,a
⟨H⟩ := 3; 0,a
⟨B⟩ := a
⟨i⟩ := a wenn s_4 = leer
⟨Y⟩ := +0

Wird die Spezifikation V angegeben, so wird im Register A eine in Tetraden codierte Dezimalzahl erwartet. Die linke, erste Tetrade ist das Vorzeichen. Negative Zahlen werden im B-1-Komplement dargestellt. Ist die Zahl positiv, so wird sie gemäß Tabelle A zu Oktaden erweitert; ist sie negativ, so wird sie zuerst invertiert und dann nach Tabelle A und die rechte Tetrade nach Tabelle C zu Oktaden erweitert (negative Ziffer). Das Ergebnis der Erweiterung steht im Register A bzw. im doppelt langen Register AQ.

Die im Register A bzw. im doppelt langen Register AQ stehenden rechten p Oktaden werden, beginnend mit der linken Oktade, bei der Oktadenadresse a und folgende abgespeichert. Die Wörter in denen Oktaden abgespeichert werden, erhalten die Typenkennung 3. Der Rest des Wortes bleibt unverändert.

Wird die Spezifikation I angegeben, so wird die Oktadenadresse um p erhöht.

Die Oktadenadresse wird ggf. um p erhöht in die Register D, H und B gebracht.

Nach Ausführung des Befehls werden die Register A und Q mit Typenkennung = 3 auf Null gelöscht. War s_4 = leer, steht die Oktadenadresse in der Indexzelle. Das Register Y wird auf +0 gelöscht.

Tabelle auf der Rückseite

Externcode: ZK/2 (Speichern)

Interncode: 'FC'

belegt: Rechen- und Befehlswerk

Takte:

- p = 0 — 3 Takte
- bei s_1 = leer
  - und s_3 = leer (122+3p) Takte
  - und s_3 = V (168+3p) Takte
- hinzu kommen:
  - für s_4 = leer und s_2 = leer 3 Takte
  - für s_4 = leer und s_2 = I 10 Takte

Alarm:

Sonstiges: Bei p = 0:

Register A, Q und B undefiniert
Abbruch nach 3 Takten

sonstige Voraussetzungen nicht erfüllt:
undefinierte Befehlsausführung

Adressenteil intern:

s_1 : B = '8000' (b = Bringen, siehe ZK/1;)
s_2 : I = '4000'
s_3 : V = '2000'
s_4 : R = '1000'
s_5 : p = '0x00' x = p = Anzahl der Oktaden 1...12

i : '00yy' wenn $s_4 \neq$ R; yy = Indexadresse
i : D '0020'
H '0010'
B '0008'
} wenn s_4 = R

Tabelle →	A		B		C	
Tetrade	Oktade		Oktade		Oktade	
	ZC1	Karte	ZC1	Karte	ZC1	Karte
'0'	0	0	PZ	+0	MZ	-0
'1'	1	1	A	+1	J	-1
'2'	2	2	B	+2	K	-2
'3'	3	3	C	+3	L	-3
'4'	4	4	D	+4	M	-4
'5'	5	5	E	+5	N	-5
'6'	6	6	F	+6	O	-6
'7'	7	7	G	+7	P	-7
'8'	8	8	H	+8	Q	-8
'9'	9	9	I	+9	R	-9
'0'	SP	leer				

ZC1 = Zentralcode

Dezimalziffern - Tetraden

	positive Zahlen	negative Zahlen
Vorzeichen	OOOC	LLLL
0	OOOO	LLLC
1	OOOL	LLCO
2	OOLO	LLCL
3	OOLL	LLOO
4	OLOO	LOLL
5	OLOL	LOLO
6	OLLO	LOOL
7	OLLL	LOOO
8	LOOO	OLLL
9	LOOL	OLLO

ZMC	n

Setze Marke im Speicher

ZMC

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

$\langle n \rangle_t$ = 0 oder 1

Ausführung:

$\langle n \rangle_m$:= L

Das Markenbit in der Speicherzelle n wird auf L gesetzt. Sonst wird der Inhalt einschließlich Typenkennung nicht verändert.

Externcode: ZMC

Interncode: '30'

belegt: Befehlswerk

Takte: 13

Alarm:

TK-Alarm: wenn $\langle n \rangle_t$ = 2 oder 3

Sonstiges:

bei falscher Typenkennung:

$\langle n \rangle_t$ = 2 oder 3

Ergebnis: wie Leerbefehl (NULL-Befehl)

TK-Alarm

1 Takt

ZTO	n

Setze Typenkennung 0 (im Speicher)

ZTO

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle n \rangle_t := 0$

Das Wort in der Speicherzelle n erhält die Typenkennung = 0.

Wenn das erste Bit der Speicherzelle n gesetzt war, entsteht eine markierte Gleitkommazahl.

Externcode: ZTO

Interncode: 'C8'

belegt: Befehlswerk

Takte: 13

Alarm:

Sonstiges:

ZT1	n

Setze Typenkennung 1 (im Speicher)

ZT1

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle n \rangle_t := 1$

Das Wort in der Speicherzelle n erhält die Typenkennung = 1.

Wenn das erste Bit der Speicherzelle n gesetzt war, entsteht eine markierte Festkommazahl.

Externcode: ZT1

Interncode: 'C9'

belegt: Befehlswerk

Takte: 13

Alarm:

Sonstiges:

ZT2	n

Setze Typenkennung 2 (im Speicher)

ZT2

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle n \rangle_t := 2$

Das Wort in der Speicherzelle n erhält die Typenkennung = 2.

Externcode: ZT2

Interncode: 'CA'

belegt: Befehlswerk

Takte: 13

Alarm:

Sonstiges:

ZT3	n

Setze Typenkennung 3 (im Speicher)

ZT3

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

$\langle n \rangle_t := 3$

Das Wort in der Speicherzelle n erhält die Typenkennung = 3.

Externcode: ZT3

Interncode: 'CB'

belegt: Befehlswerk

Takte: 13

Alarm:

Sonstiges:

ZTR	s

Setze Typenkennung im Register

ZTR

Adressenteil:

s_1 = Typenkennung 0,1,2,3 oder leer

s_2 = Register A,Q,D,H oder leer

s_3 = Register M oder leer

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

s_2 = A,Q,D oder H $\qquad$ s_3 = M

Ausführung:

$\langle s_2 \rangle_t := s_1$ $\qquad$ $\langle M \rangle := L$

Das Wort in dem durch s_2 angegebenen Register erhält die Typenkennung s_1.

Ist s_3 = M, so wird das Markenregister M auf L gesetzt.

Wenn durch die Änderung der Typenkennung ein über- oder untergelaufenes Zahlwort (TK = 0 oder 1) entsteht, wird kein BÜ-Alarm gegeben.

Externcode: ZTR

Interncode: '92'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

Adressenteile intern:

s_1 : 0 = '--00'
1 = '--40'
2 = '--80'
3 = '--C0'

s_2 : A = '--20'
Q = '--10'
D = '--08'
H = '--04'
M = '--02'

ZU	i	Setze Unterprogrammregister	ZU

Adressenteil: i = Indexadresse
(Die auf i folgenden Indexadressen werden für Unterprogrammsprünge belegt.)

bei mod2:	wird modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨U⟩ := i

Die Indexadresse i wird in das Unterprogrammregister gebracht. (Beim Sprung in ein Unterprogramm wird die Rücksprungadresse nach i + 1 gebracht.)

Externcode: ZU

Interncode: '3E'

belegt: Befehlswerk

Takte: 1

Alarm:

Sonstiges:

ZUS	n

Setze zusammen

ZUS

Adressenteil: n = Adresse eines Ganzwortes

bei mod2:	wird modifiziert	bei R:	zugelassen

Voraussetzung:

$\langle H \rangle$ = Maske

Ausführung:

$\langle D \rangle := t_n ; \langle n \rangle$

$\left.\begin{array}{l} \langle D \rangle_1 := \langle D \rangle_2 \\ \langle M \rangle := \langle M \rangle \vee \langle n \rangle_n \end{array}\right\}$ nur bei t_n = 0 oder 1

x = 1...48 Maske im Register H

$\langle A \rangle_x := \langle A \rangle_x$	für $\langle H \rangle_x$ = 0	0-Feld der Maske
$\langle A \rangle_x := \langle D \rangle_x$	für $\langle H \rangle_x$ = L	L-Feld der Maske
$\langle A \rangle_1 := t_{n+x}$	von $\langle A \rangle_t$ oder $\langle D \rangle_t$	

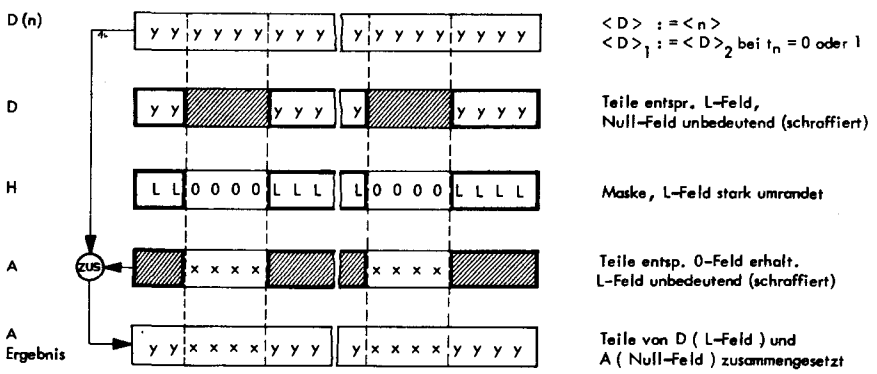
Im Register H steht eine Maske.

Der Inhalt der Speicherzelle n wird einschließlich Typenkennung in das Register D gebracht.

Der Inhalt des Registers A und des Registers D wird in der Art zusammengesetzt, daß alle Binärstellen des Registers A erhalten bleiben, die dem Null-Feld der Maske im Register H entsprechen. Die Binärstellen des Registers D, die dem L-Feld der Maske entsprechen, werden in das Register A eingesetzt.

Das Ergebnis im Register A erhält die größere der beiden Typenkennungen der Register A oder D.

Bei Zahlwörtern wird die Marke berücksichtigt.



Externcode: ZUS

Interncode: '6B'

belegt: Rechenwerk

Takte: 5

Alarm:

Sonstiges:

ZX	p i

Setze Index

ZX

Adressenteil: p = Zahl 0...±127
i = Indexadresse

bei mod2:	wird nicht modifiziert	bei R:	nicht zugelassen

Voraussetzung:

Ausführung:

⟨i⟩ := v,p
⟨B⟩ := v,p

Die Zahl p wird mit ihrem Vorzeichen in die Indexzelle i und in das Register B gebracht (jeweils in die Binärstellen 17 - 24).

Die linken Stellen werden dem Vorzeichen (Binärstelle 17) angeglichen.

Externcode: ZX

Interncode: '1A'

belegt: Befehlswerk

Takte: 6

Alarm:

Sonstiges:

Alphabetische Liste der Befehle

Code	Int.	Seite	mod2	R	Op
A	42	7	+	x	+
A2	7C	9	+	x	+
AA	98	7,8	+		
AB	40	7	+	x	+
AC	43	7	+		+
AQ	7E	7	+	x	+
AT	F4	9	+	x	+
ATA	89	9	+		
AU	49	8	+	x	+
AUT	69	9	+	x	+
B	70	4	+	x	+
B2	6E	4	+	x	+
B2V	6F	4	+	x	+
B2VN	67	4	+	x	+
B3	6C	4	+	x	+
B3V	6D	4	+	x	+
BA	8E	10	+		
BAN	DF	10	+		
BANR	DD	10	+		
BAR	DC	10	+		
BB	74	4	+	x	+
BC	A3	5	+		+
BCI	B6	5			+
BCL	06	6			+
BD	71	4	+	x	+
BH	73	4	+	x	+
BL	B0	5	+		
BLEI	BE	4	+		
BN	75	4	+	x	+
BNR	77	4	+	x	+
BNZ	E6	4	sp		
BQ	72	4	+	x	+
BQB	DA	4	+	x	+
BR	76	4	+	x	+
BSS	FB	4			
BT	F6	4	+	x	+
BU	D3	4	+	x	+
BZ	D9	4	+		+
BZ2	D8	4	+		+
BZN	D1	4	+		+
C	80	5	+		
C2	A0	5	+		+
C3	A1	5	+		+
CB	85	5	+		
CD	86	5	+		
CH	8F	5	+		
CMC	A2	5	+		+
CMR	83	5	+		
CMT	82	5	+		
CN	84	5	+		
CNZ	E7	5	sp		
CQ	87	5	+		
CR	81	5	+		
CT	F7	5	+		+
CU	D0	5	+		
CZ	DB	5	+		
DA	F0	8			+
DML	F2	8			+
DSB	F1	8			+
DV	60	7	+	x	+
DVD	61	7	+	x	+
DVI	62	7	+	x	+
E	29	15	sp		
EMB	28	15			
EMU	04	15	sp		
ENZ	2A	15	sp		
ET	6A	9	+	x	+
ETA	8A	9	+		
EZ	2B	15	sp		
GA	4B	8	+	x	+
GAB	52	8	+	x	+
GAC	4A	8	+		+
GDV	64	8	+	x	+
GDVI	66	8	+	x	+
GMAN	5D	8	+	x	+
GML	5E	8	+	x	+
GMLA	5F	8	+	x	+
GMLN	5C	8	+	x	+
GSB	4F	8	+	x	+
GSBB	53	8	+	x	+
GSBC	4E	8	+		+
GSBD	4C	8	+	x	+
GSBI	48	8	+	x	+
HBA	11	10			
HBC	3C	10		x	+
HBPX	0F	10			
HXP	2C	10			
HXX	2E	10			
IR	E1	16			
KDFR	94	17	+		
KFLD	95	17	+		
LA	8B	10	+		
LC	33	11	+		+
LMC	31	11	+		+
LMT	32	11	+		+
LR	9A	10	+		
LZL	10	11			
M	08	14	sp		
M2	7A	9	+	x	+
M2N	78	9	+	x	+
M2NR	79	9	+	x	+
M2R	7B	9	+	x	+
MA	03	14	sp		
MAB	20	15			
MABI	3F	15			
MAN	5A	7	+	x	+
MANR	5B	7	+	x	+
MAR	57	7	+	x	+
MC	14	14	sp	x	+
MCE	17	14	sp	x	+
MCF	16	14	sp	x	+
MCFU	3D	14	sp	x	+
MD	09	14	sp		
MF	0B	14	sp		
MFU	08	14	sp		
MH	2D	14			
MHX	0E	14			
ML	54	7	+	x	+
MLA	56	7	+	x	+
MLD	F3	8			+
MLN	58	7	+	x	+
MLR	55	7	+	x	+
MNA	02	14	sp		
MNR	59	7	+	x	+
MRX	8D	14			
MU	05	15			
NL	12	11			
NRM	9F	16	+		
NULL	00	12		x	
PDP	B9	13	+		+
QBR	FE	6			+
QCR	FF	6			
R	96	15	+		
REZ	65	8	+	x	+
RLR	E0	15	sp		
RT	97	6	+		
RX	8D	10			
S	36	12	+		
SAA	A9	13	+		
SAT	A8	13	+		
SB	46	7	+	x	+
SB2	7D	9	+	x	+
SBA	99	7,8	+		
SBB	41	7	+	x	+
SBC	47	7	+		+
SBD	45	7	+	x	+
SBI	44	7	+	x	+
SBIT	F9	13			
SBQ	7F	7	+	x	
SBT	F5	9	+	x	+
SBU	4D	8	+	x	+
SE	BC	12	sp	x	+
SEGG	93	13	+		
SFB	3A	12	+		
SFBE	FA	12	sp		
SG	AB	12	+		
SG0	D4	12	+		
SGG	AF	12	+		
SGG0	A6	12	+		
SH	9B	16	+		
SHB	21	16			
SI	AC	12	+		
SI0	A4	12	+		
SK	AA	12	+		
SK0	D5	12	+		
SKG	AE	12	+		
SKG0	A5	12	+		
SL	1E	13			
SLL	1F	13			
SLN	1C	13			
SM	34	13	+		
SMN	35	13	+		
SN	AD	12	+		
SN0	A7	12	+		
SNL	1D	13			
SR	B8	12	+		
SRN	BA	12	+		
SSR	BB	12	+		
ST	90	13	+		
STN	91	13	+		
SU	38	12	+		
SUE	BD	12	sp	x	+
SXG	CE	13			
SXGG	25	13			
SXI	24	13			
SXK	CF	13			
SXKG	26	13			
SXN	27	13			
SXR	B2	13			
SXRN	B3	13			
SZX	0A	13			
T	CC	15	sp	x	+
TBC	07	6			+
TCB	39	6	+	x	+
TDM	EA	17			+
TLD	EB	17			+
TLI	EC	17			+
TLOG	ED	17			+
TMAX	EF	17			+
TMIN	EE	17			+
TOK	FD	6			
TRX	9C	6			
TTX	0D	6			
TXR	8C	6			
TXX	0C	6			
US	E3	16	sp		
VAQ	63	16	+		
VBA	13	10			
VBC	15	10		x	+
VEL	68	9	+	x	+
VLA	88	9	+		
VXX	2F	10			
WB	F8	12			
WTR	23	6			
WTV	22	6			
XB	08	6			
XBA	01	11			
XBAN	19	11			
XC	18	6			
XCN	18	6			
ZI	B7	11			+
ZK	FC	6			
ZMC	30	11	+		+
ZT0	C8	11	+		+
ZT1	C9	11	+		+
ZT2	CA	11	+		+
ZT3	CB	11	+		+
ZTR	92	11	+		
ZU	3E	11	+		
ZUS	6B	9	+	x	+
ZX	1A	11			

Code: Befehlscode

Int.: Interncode in 2 Sedezimalen

Seite: Angabe der Seite in der "Großen Befehlsliste"

mod2: + = Modifizierung 2. Art

mod2: sp = spezielle Modifizierung 2. Art

R: x = als Zweitcode beim Befehl R zugelassen

Op: Dieser Befehl holt in der Abrufphase einen Operanden aus dem Speicher

Konvertierungstafel

sedezimal	dezimal
100 000	1 048 576
200 000	2 097 152
300 000	3 145 728
400 000	4 194 304

010 000	65 536
020 000	131 072
030 000	196 608
040 000	262 144
050 000	327 680
060 000	393 216
070 000	458 752
080 000	524 288
090 000	589 824
0A0 000	655 360
0B0 000	720 896
0C0 000	786 432
0D0 000	851 968
0E0 000	917 504
0F0 000	983 040

001 000	4 096
002 000	8 192
003 000	12 288
004 000	16 384
005 000	20 480
006 000	24 576
007 000	28 672
008 000	32 768
009 000	36 864
00A 000	40 960
00B 000	45 056
00C 000	49 152
00D 000	53 248
00E 000	57 344
00F 000	61 440

000 100	256
000 200	512
000 300	768
000 400	1 024
000 500	1 280
000 600	1 536
000 700	1 792
000 800	2 048
000 900	2 304
000 A00	2 560
000 B00	2 816
000 C00	3 072
000 D00	3 328
000 E00	3 584
000 F00	3 840

000 010	16
000 020	32
000 030	48
000 040	64
000 050	80
000 060	96
000 070	112
000 080	128
000 090	144
000 0A0	160
000 0B0	176
000 0C0	192
000 0D0	208
000 0E0	224
000 0F0	240

000 001	1
000 002	2
000 003	3
000 004	4
000 005	5
000 006	6
000 007	7
000 008	8
000 009	9
000 00A	10
000 00B	11
000 00C	12
000 00D	13
000 00E	14
000 00F	15

dezimal	sedezimal
1 000 000	0F4 240
2 000 000	1E8 480
3 000 000	2DC 6C0
4 000 000	3D0 900

100 000	018 6A0
200 000	030 D40
300 000	049 3E0
400 000	061 A80
500 000	07A 120
600 000	092 7C0
700 000	0AA E60
800 000	0C3 500
900 000	0DB BA0

10 000	002 710
20 000	004 E20
30 000	007 530
40 000	009 C40
50 000	00C 350
60 000	00E A60
70 000	011 170
80 000	013 880
90 000	015 F90

1 000	000 3E8
2 000	000 7D0
3 000	000 BB8
4 000	000 FA0
5 000	001 388
6 000	001 770
7 000	001 B58
8 000	001 F40
9 000	002 328

100	000 064
200	000 0C8
300	000 12C
400	000 190
500	000 1F4
600	000 258
700	000 2BC
800	000 320
900	000 384

10	000 00A
20	000 014
30	000 01E
40	000 028
50	000 032
60	000 03C
70	000 046
80	000 050
90	000 05A

=

Interncode-Externcode

0000	000L	00L0	00LL	0L00	0L0L	0LL0	0LLL	L000	L00L	L0L0	L0LL	LL00	LL0L	LLL0	LLLL	2^7 1. Sedezimale	2^0 2. Sedezimale
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
NULL	LZL	MAB	ZMC	AB		DV	B	C	ST	C2	BL		CU	RLR	DA	0	0000
XBA	HBA	SHB	LMC	SBB		DVD	BD	CR	STN	C3	VPU *		BZN	IR	DSB	1	000L
MNA	NL	WTV	LMT	A	GAB	DVI	BQ	CMT	ZTR	CMC	SXR		ZDP		DML	2	00L0
MA	VBA	WTR	LC	AC	GSBB	VAQ	BH	CMR	SEGG	BC	SXRN		BU	US	MLD	3	00LL
EMU	MC	SXI	SM	SBI	ML	GDV	BB	CN	KDFR	SI0	Y *		SG0		AT	4	0L00
MU	VBC	SXGG	SMN	SBD	MLR	REZ	BN	CB	KFLD	SKG0	LEI *		SK0		SBT	5	0L0L
BCL	MCF	SXKG	S	SB	MLA	GDVI	BR	CD	R	SGG0	BCI			BNZ	BT	6	0LL0
TBC	MCE	SXN	VSS *	SBC	MAR	B2VN	BNR	CQ	RT	SN0	ZI			CNZ	CT	7	0LLL
MFU * M, XB	XC ** XCN	EMB	SU	GSBI	MLN	VEL	M2N	VLA	AA	SAT	SR	ZT0	BZ2		WB	8	L000
MD	XBAN	E	TCB	AU	MNR	AUT	M2NR	ATA	SBA	SAA	PDP	ZT1	BZ		SBIT	9	L00L
SZX	ZX	ENZ	SFB	GAC	MAN	ET	M2	ETA	LR	SK	SRN	ZT2	BQB	TDM	SFBE	A	L0L0
MF	SW *	EZ		GA	MANR	ZUS	M2R	LA	SH	SG	SSR	ZT3	CZ	TLD	BSS	B	L0LL
TXX	SLN	HXP	HBC	GSBD	GMLN	B3	A2	TXR	TRX	SI	SE	r	BAR	TLI	ZK	C	LL00
TTX	SNL	MH	MCFU	SBU	GMAN	B3V	SB2	RX ** MRX	HALT *	SN	SUE		BANR	TLOG	TOK	D	LL0L
MHX	SL	HXX	ZU	GSBC	GML	B2	AQ	BA		SKG	BLEI	SXG		TMIN	QBR	E	LLL0
HBPX	SLL	VXX	MABI	GSB	GMLA	B2V	SBQ	CH	NRM	SGG	VMO *	SXK	BAN	TMAX	QCR	F	LLLL

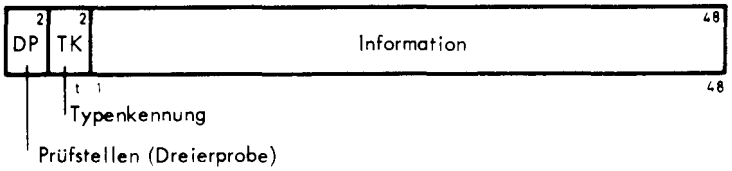
GR 1

*nicht für die Programmierung von Operatoren

** Unterscheidung im Adressenteil
(siehe Internspezifikationen)

Wortstruktur (im Speicher)

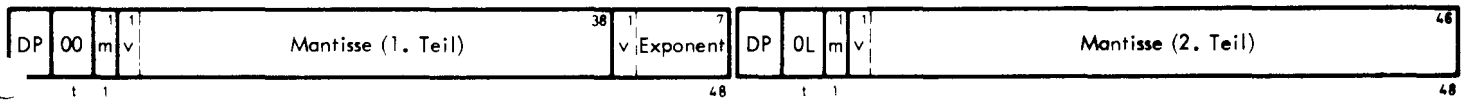
ALLGEMEIN



GLEITKOMMAZAHL (Basis 16)

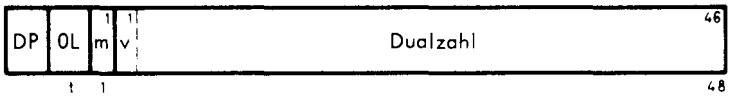
Einfache Länge

Doppelte Länge

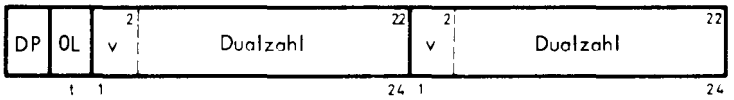


FESTKOMMAZAHL

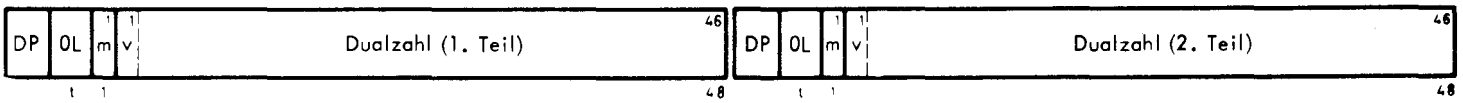
Einfache Länge



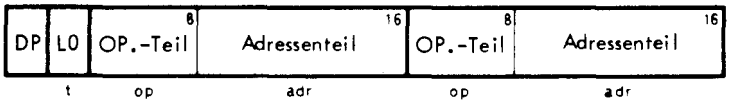
Halbe Länge (zwei Zahlen pro Wort)



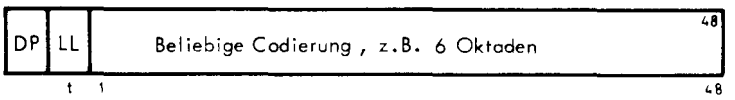
Doppelte Länge



BEFEHLE (zwei Befehle pro Wort)



ALPHATEXT



DP = Bits für Prüfzwecke (Dreierprobe)

TK = Bits für Typenkennung

t = Typenkennung

m = Marke (nur im Speicher, im Register gleich der v-Stelle)

v = Vorzeichen

ER 1711

Blockschaltbild