TA 1000

Mikroinstruktionshandbuch

Entwurf von EE34

Inhaltsverzeichnis

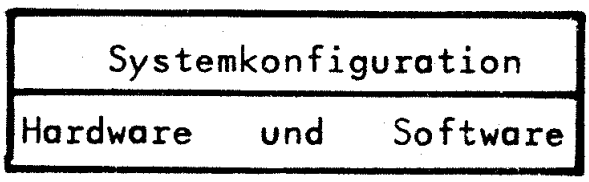
1 Vorwort

Das vorliegende Mikroinstruktionshandbuch enthält einige allgemeine Erläuterungen, eine Einführung mit nachfolgender Beschreibung des Mikroinstruktions-Satzes.

Diese Beschreibung ist als Nachschlagwerk des Mikroprogrammierers gedacht und dient gleichzeitig als Einarbeitungshilfe.

2 Allgemeines

2.1 Hardware und Software:



2.2 Hardware

Unter Hardware wird die Gesamtheit aller realtechnischen Einrichtungen verstanden, die für die maschinelle Abwicklung von Datenverarbeitungsprozessen notwendig sind.

Dazu gehören

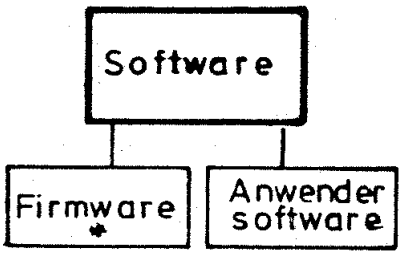
- die Zentraleinheit
- die Ein-Ausgabe-Einheit (alle peripheren Geräte).

Die Aggregate bestehen aus einer Vielzahl von unterschiedlichen logischen Verknüpfungen integrierter Schaltkreise, Transistoren, Bausteinen, Kabel, Stecker und andere technische Elemente.

2.3 Software

Software ist der Sammelbegriff für alle in Datenverarbeitungsanlagen einsetzbare Programme.

Gruppierung der Software:



* oder auch:
Systemsoftware,
Betriebssystem.

Firmware

Die Firmware gewährleistet die Steuerung und Überwachung des Programmablaufes mit optimaler Ausnutzung der Hardware und ermöglicht die Bedienung der Anlage in einfacher Weise und gibt Hilfestellung für die Entwicklung der Anwender-Software.

Anwendersoftware

Sie enthält eine logisch geordnete Folge von Arbeitsvorschriften die zur Auslösung ganz bestimmter Programme, die direkt für die vom Anwender zu lösenden Datenverarbeitungsaufgaben verwendet werden. (in standardisierter Form für die verschiedensten Unternehmensbereiche).

2.4 Mikroprogramm

Das Mikroprogramm bestimmt das Können des Prozessors. Ihm kommt deshalb besondere Bedeutung zu.

Gliederung in:

- Standardmikro
- Gerätemikros

Das Standardmikro enthält:

- Betriebs-, Fehler-, Steuer-, Unterbrechungsroutinen
- das wesentliche interpretative Mikro

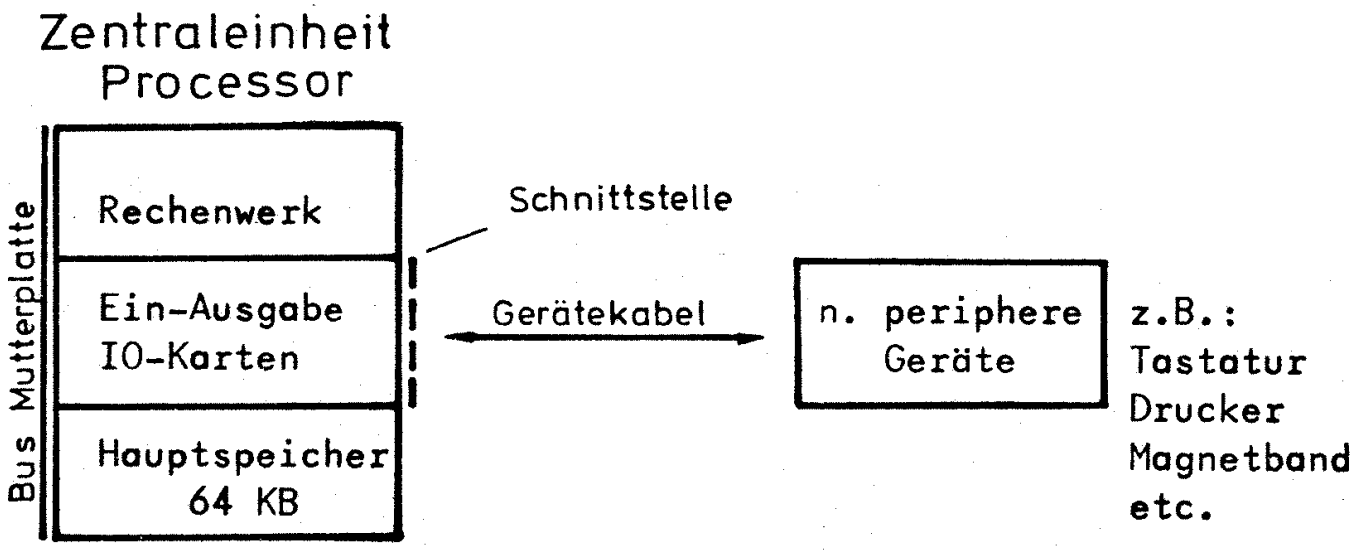
Die Gerätemikros dienen der Steuerung der angeschlossenen peripheren Geräte.

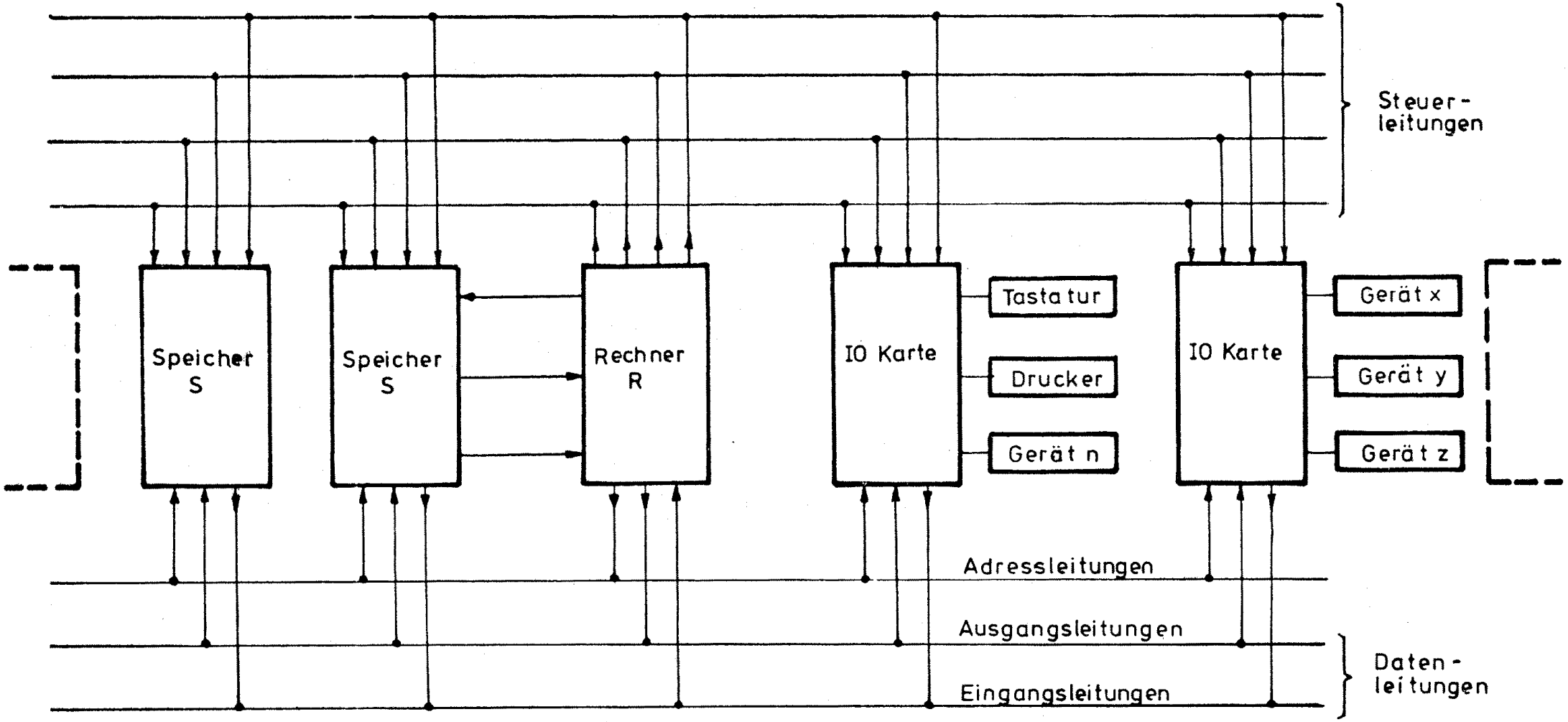
3 Einführung in die Komponenten der Zentraleinheit

3.1 Zentraleinheit ZE

Das Kernstück jedes Computers ist die Zentraleinheit oder auch Prozessor genannt.

Die Zentraleinheit hat die Aufgabe den Datenfluß zu steuern, zu überwachen und den Komunikationsverkehr mit den peripheren Geräten zu erledigen.





3.2 Processor Blockschaltbild

Prozessor Blockschaltbild Beschreibung

S = Speicher, R = Rechner, IO = Ein-Ausgabe bzw. IO-Karte

Der Prozessor ist modular im Baukastenprinzip aufgebaut. Die nötigen Verbindungsleitungen zwischen den Baugruppen stellt ein Bussystem (Mutterplatte) her.

Es ist eine Rückwandverdrahtung in gedruckter Schaltung.

Sie besteht aus:

Adress-, Daten-, Steuer-, und Versorgungsleitungen.

Über die Adreßleitungen werden Speichermodule, Speicher-Zellen, IO-Karten und IO-Zeilen adressiert.

Die Datenleitungen teilt man in Eingangs- und Ausgangs-Leitungen ein. Über diese Leitungen erfolgt der Datenfluß des Rechners von oder zum Speicher und zu den peripheren Geräten.

Die Steuerleitungen dienen zur Taktung der Speicher und IO-Karten etc.

3.3 Der Rechner

Als Rechenwerk wird der Teil des Prozessors bezeichnet mit dem es möglich ist, arithmetische und/oder logische Operationen durchzuführen.

Rechnergliederung

- Mikroinstruktions Decoder, Steuerwerk bzw. Leitwerk
- Arithmetische logische Einheit, ALU
- Register (A-, BA-, BR-, E-, I-, P-, u. das R Register)
- Unterbrechungen

Der Mikroinstruktions Decoder und Steuerwerk, beaufsichtigt den Datenfluß.

Der anliegende Mikrobefehlscode wird analysiert und entsprechend dem Befehl ausgeführt. Es gibt die Instruktions-, Subtitutions- und die Ausführungsphase.
Dabei werden die Takte an den Rechnereinheiten so gesteuert, daß die Ausführung des Befehls richtig abläuft.

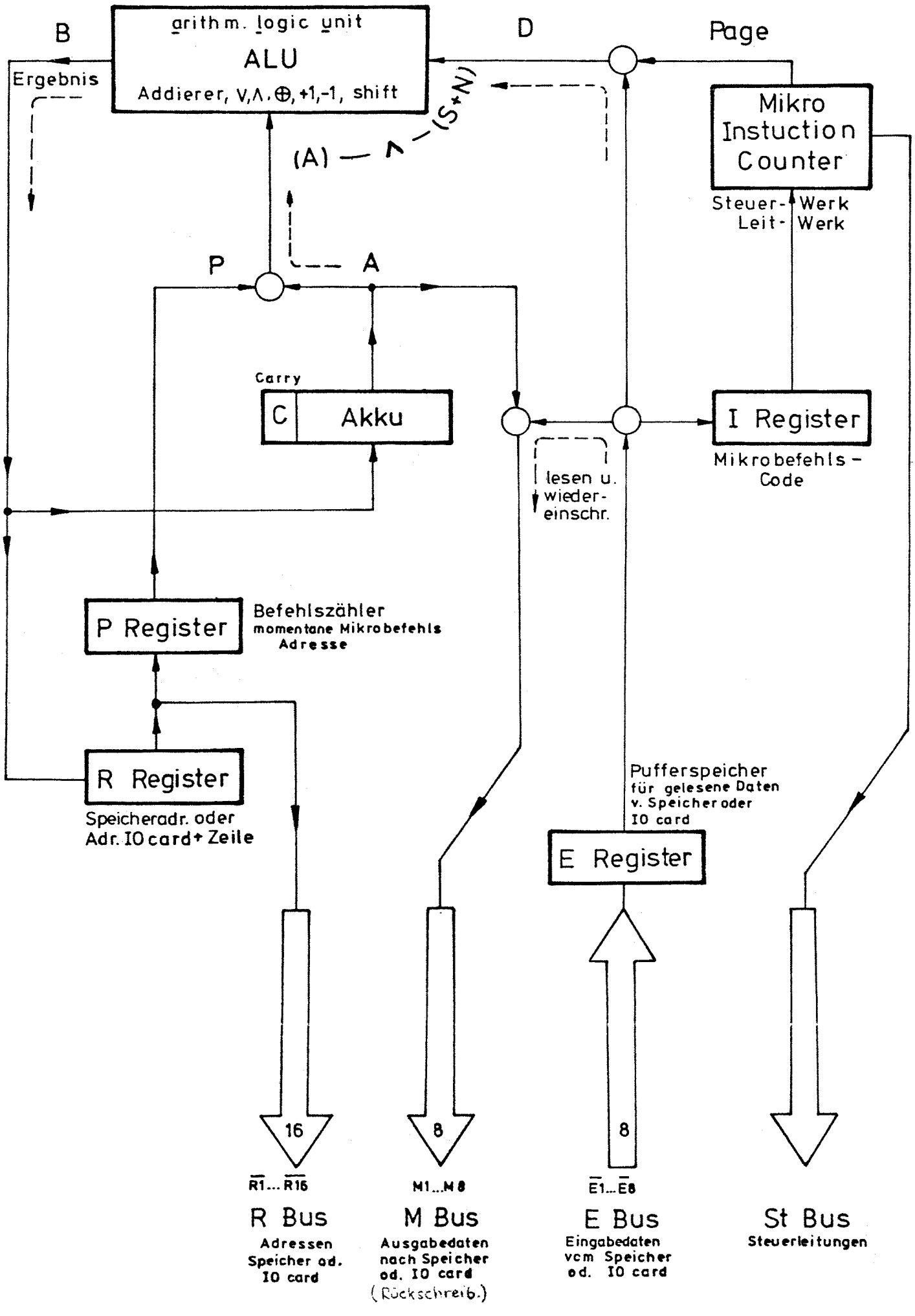
Die Arithmetische logische Einheit (arithmetic logic unit ALU)

enthält

UND, ODER, EXOR, SHIFT, +1 Zunahme (incrementing), -1 Abnahme (decrementing) und andere allgemeine Funktionen wie Addieren ect.

Register: schnelle Zwischenspeicher für Daten-, Speicher-, Instruktions-, und Befehlsadressen.

Rechner Blockschaltbild



Kurzbeschreibung der Register:

A - Register, (16 bit) Zentrales Arbeitsregister, Akkumulator kurz Akku genannt.
Alle logischen-, arithmetischen und Ein/Ausgabebefehle arbeiten mit dem Akku.
Am Ende jeder Rechenoperation steht das Ergebnis im Akku.

Im E-Register, (8 bit) werden die Eingabedaten vom Speicher oder von den peripheren Geräte für den gesamten Zyklus zwischengespeichert.

Im I-Register, (8 bit) steht der Mikrobefehlscode an und ist dort während der ganzen Befehlsverarbeitung gespeichert. Er wird anschließend vom Mikroinstruktions Decoder analysiert.

Das P-Register, (16 bit) beinhaltet die Adresse des momentan zu bearbeitenden Mikrobefehls. Es dient zur schrittweisen Verfolgung der einzelnen Operationen.
Aus dem hohem Byte ist die Page (Seite) ersichtlich in der sich der Befehl befindet.

Das R-Register, (16 bit) enthält die jeweils aktuelle Speicher oder IO-Adresse.

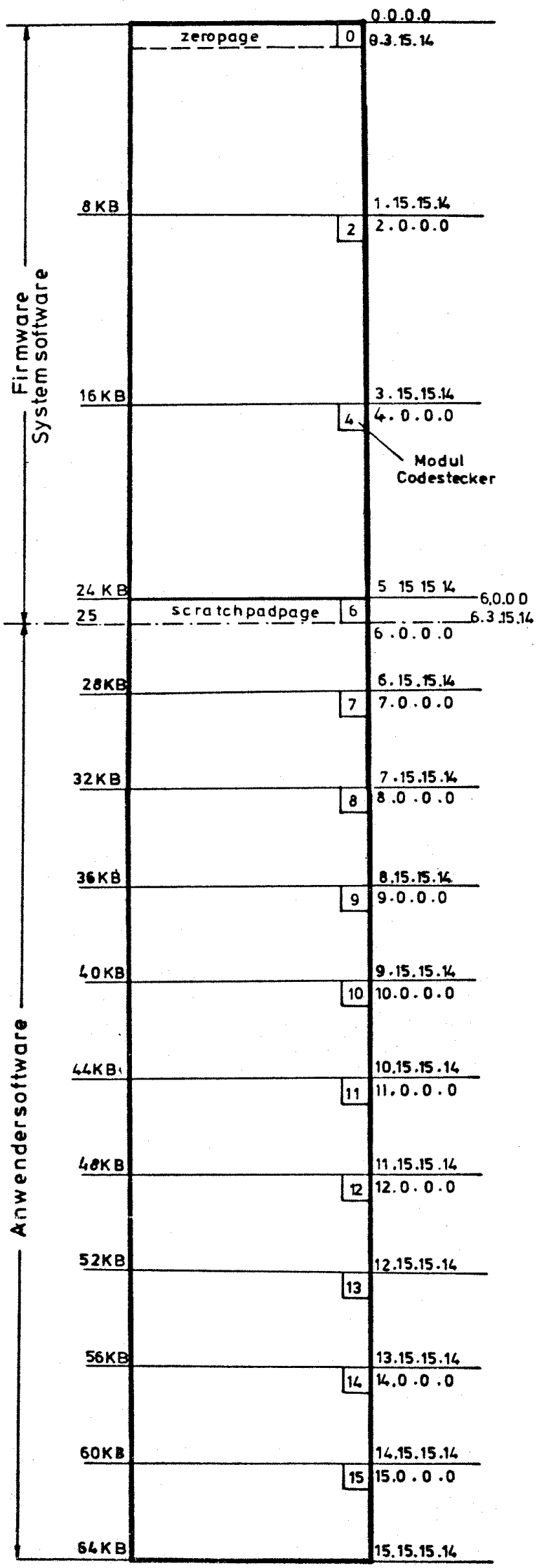
Das C-Register, 1 bitiges Überlaufregister, carry
Das C-Register ist dem Akkumulator zugeordnet.
Falls bei Addition oder Subtraktionsbefehlen (die in ihrer Mikrobefehlsstruktur das C enthalten) die Rechenoperation durchgeführt wird, erfolgt gleichzeitig eine Überprüfung des C-Registers.
Bei Überlauf des Akkumulators wird das C-Register auf logisch 1 gesetzt, andernfalls wird es gelöscht.

Unterbrechungen, interrupts

Netzausfallroutine
Uhrunterbrechungsprogramm 1 ms Uhr
E/A Unterbrechungsprogramm
Unterbrechung durch schnelle Geräte
DMA direct memory access
Siehe Unterbrechungsprogramm Beschreibung Seite.....

3.4 Der Hauptspeicher

Der Hauptspeicher dient der Aufnahme der Daten und Programme und kann beliebig aus Festspeichern und Lebendspeichern zusammengesetzt werden. Für den Rechner ist es gleichgültig, ob die Adresse einen Fest- oder Lebendspeicher ansteuert. Durch die Möglichkeit einer beliebigen Aufteilung im Programm und Datenbereich kann der Lebendspeicher optimal ausgenützt werden. Der Hauptspeicher ist bytweise organisiert.



Beispiel einer
Speicheraufteilung

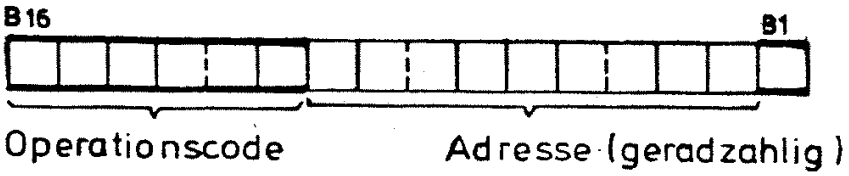
Speicheraufteilungsbeschreibung

Der Adressteil des Mikrobefehlswortes ist ein 10 bites Dualwort.

Die volle Kapazität des Adressteils beträgt 1024 Bytes = 1 KB. Entsprechend dieser Festlegung ist die Speicherstruktur KB-weise gegliedert.

Die maximale Speicherkapazität beträgt 64 KB.

Mikrobefehlswort



Die Zero- und die Scratchpadpage nehmen im Mikroinstruktionssatz eine gewisse Sonderstellung ein.

Zeropage "Z" Nullseite 1. Speicherseite

0.0.0.0....0.3.15.15 = 1 KB

Diesen Speicherbereich kann man von jeder beliebigen Mikroprogrammadresse innerhalb der 64 KB direkt ansprechen.

Hardwaremäßig wird bei einem Sprung in die Zeropage der Adreßteil des Mikrobefehlswortes in den Befehlszähler geladen, Im Befehlszähler werden dabei die Bits 11 - 16 und das Bit 1 auf Null gesetzt.

Der Zugriff zu wichtigen Werten ist somit direkt gegeben. In diesem Speicherbereich sind alle oft benötigten Konstanten Tabellen Adressen etc. unterzubringen.

Scratchpad "S" 1 KB - Arbeitsbereich im Lebendspeicher

Veränderliche Daten eines Mikrobereiches benötigen einen Lebendspeicher.
Wie aus dem Mikroinstruktionssatz ersichtlich, werden die meisten Rechenoperationen über die Scratchpad abgewickelt.
Die Plazierung der Scratchpad ist in Abständen von 4 KB frei wählbar.
Die Festlegung der Scratchpad ist durch Rangierung auf der Mutterplatte über die Punke SC1 - SC4 in codierter Form möglich (0...15).

In dem Scratchpadbereich sollten Arbeitszellen, Pufferfelder spezielle Gerätezellen etc. untergebracht werden.

In der folgenden Beschreibung erkennt man eine solche Scratchpadzelle am "X" z.B. X0, X1, XI0 usw.

Currentpage "P", laufende Seite

Unter Currentpage wird die Page bezeichnet in der momentan der Befehlszähler steht.

Speicheradressierung

Der Hauptspeicher wird dual adressiert, die Speicherworte sind fortlaufend von 0...65535, entsprechend dem Dualwert einer 16-bitigen Dualzahl adressiert.

3.6 IO-Karte

Der Datenfluß zu oder von den peripheren Geräten zum Rechner erfolgt über die IO-Karte.
Das Gerät ist entsprechend seiner Aufgabe über diese Karte an den Rechner anzupassen.
Die IO-Karte stellt die Elektronik-Schnittstelle zum Gerät dar.

Bei bestimmten Geräten ist weitere Elektronik auf der Karte notwendig für:

Interfacestufen
Datenzwischenspeicher
Serien - Parallelwandler
Parallel - Serienwandler

Die IO-Karte besitzt mehrere Eingabe-/Ausgabezeilen und einen Standardteil für die Karten- u. Zeilenauswahl.

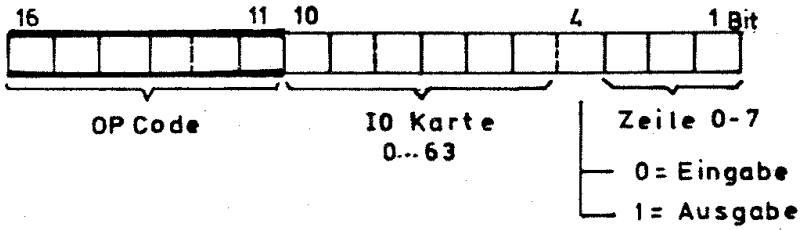
Datenleitungen vom Gerät werden kurzzeitig durch IO-Befehle auf den Eingabedatenbus des Rechners durchgeschaltet.

Datenleitungen zum Gerät werden kurzzeitig durch IO-Befehle vom Ausgabedatenbus des Rechners zum Gerät durchgeschaltet.

Die kleinste adressierbare Einheit auf der IO-Karte ist 1 Byte ≙ 8 Bit, d.h. es werden mindestens 8 Leitungen geschaltet.

Pro Karte können, je nach Bestückung mehrere Geräte angeschlossen werden. Abhängig von dem Platzbedarf eventuleller weiterer Elektronik auf der IO-Karte und von dem Zeilenbedarf der Geräte.

IO - Befehlsformat



Bit 4 entscheidet ob eine Eingabe oder eine Ausgabe erfolgen soll. Bit 1 - 3 wählt die Zeile aus.
Mit der Kartenadresse wird die IO-Karte entsprechend dem peripheren Gerät angesprochen.

Durch den OP-Code des IO-Befehls wird die Wahl folgender Befehle bestimmt.

Direkte IO-Befehle

	mathem. Schreibw.	OP Code	Assembler Schreibw
1.	IO(N)L⟶A	12.8	IOL

Der Inhalt des niederwertigen Akkumulatorbytes wird auf die im Befehl angegebene IO-Karte und Zeile ausgegeben, bzw. dort werden die Daten in das niederwertige Byte des Akkumulators übertragen.

2.	IO(N) ⟵ A	12.12	IOX

Entsprechende Übertragung eines Doppelbytes Bit 1 wird für die Zeilenadresse zu logisch 0 angenommen.

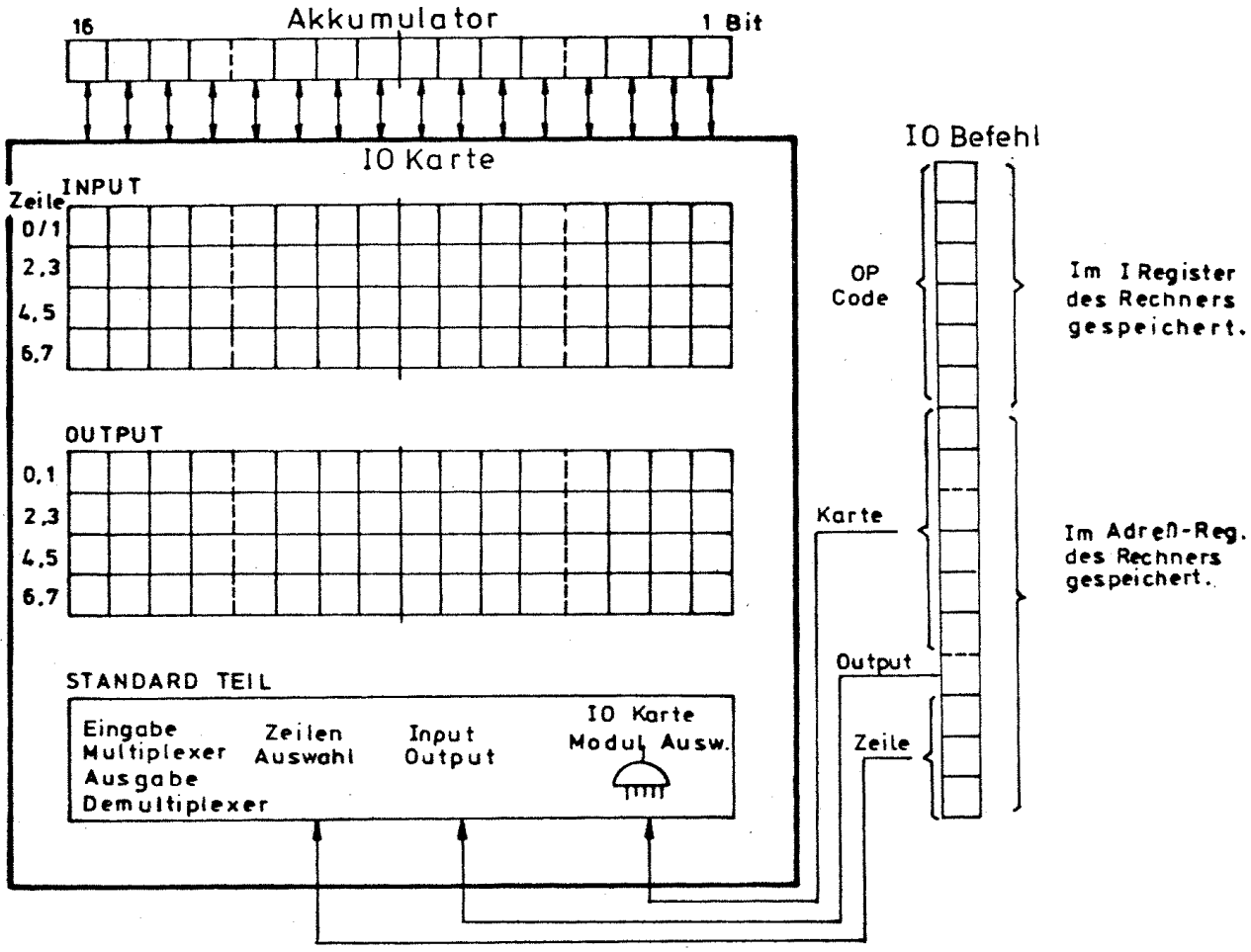
Indirekte IO-Befehle

3.	IO((S+N))B⟶A	12.8 u.Bit 1	IOB

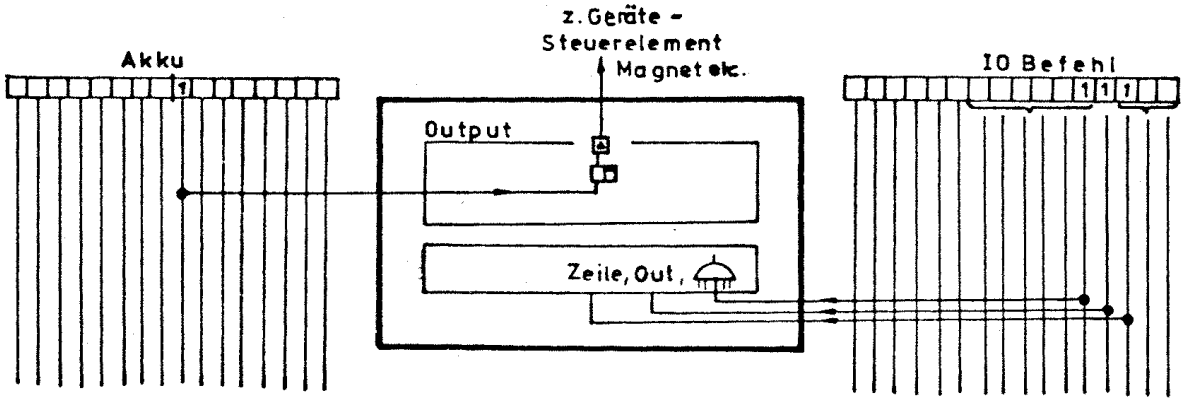
Wie 1., aber die Karten- u. Zeilenadresse steht in der Scratchpad S+N.

4.	IO((S+N)) ⟵ A	12.12 u.Bit 1	IOI

Wie 2., aber die Karten- u. Zeilenadresse steht in der Scratchpad S+N.



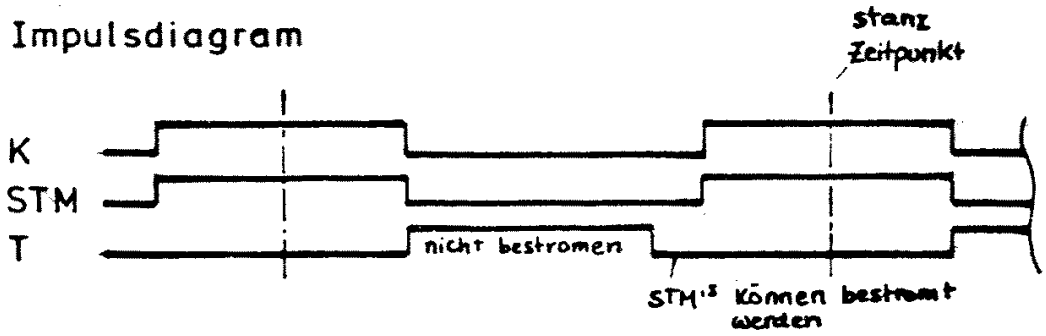
zum Beispiel:



Schematische Darstellung
IO Kartenauswahl
Geräteansteuerung

Lochstreifenanschluß (Prinzip)

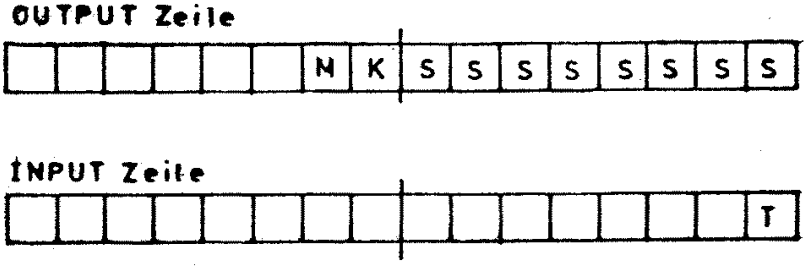
Impulsdiagram



K = Kupplung , Stanzvorgangauslösung.

STM = Stanzmagnete

T = Taktgeber ⟶ z.B. mechanischer Flügel m. Lichtschranken ⧖ auf der Antriebsachse montiert Abtastung.



S = Stanzinformation
K = Kupplung
M = Motor

Gerät ein, Motoreinschalten
warten bis Motor volle Drehzahl erreicht
Taktgeber muß in Grundstellung stehen (Fehler)
┌→Information ausgeben (STM bestromen)
│ Kupplung freigeben
│ warten auf Taktgeber
│ löschen Kupplung und Stanzmagnete
│ warten bis Taktgeber in Grundstellung
└ neues Zeichen
Ende

Simples IO Beispiel

3.7 Unterbrechungen

Uhrunterbrechung 1 ms Uhr
I/O Unterbrechung für schnelle Geräte (< 1 ms)
Direkter Speicherzugriff DMA
Netzausfallunterbrechung
Paritäts - Unterbrechung

Tritt eine Unterbrechung auf, so wird vom laufenden Mikroprogramm in die Unterbrechungsroutine verzweigt. Hierbei ist die Rückkehradresse, der Akkuinhalt und der Übertrag (C) sicherzustellen. Anschließend wird die Art der Unterbrechung geprüft und in das entsprechende Unterbrechungsprogramm verzweigt.

Beispiel:

Uhr Unterbrechung

Die einzelnen peripheren Geräte sollen zueinander und zu dem Makro-Programm simultan betrieben werden.

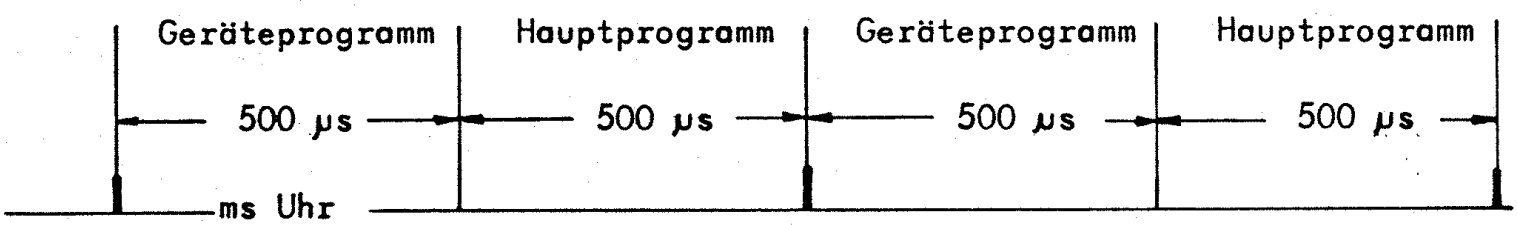
Sie sollen mit so wenig wie möglich Hardware ausgestattet sein. Die Ansteuerung und das Timing geschieht vom Mikroprogramm aus. Die Simulatanzeit soll möglichst wenig Rechnerzeit benötigen.

Für den Betrachter hat das den Anschein, als wenn alle Geräte parallel betrieben würden, in Wirklichkeit werden aber die einzelnen peripheren Geräte seriell alle ms angeschlossen.

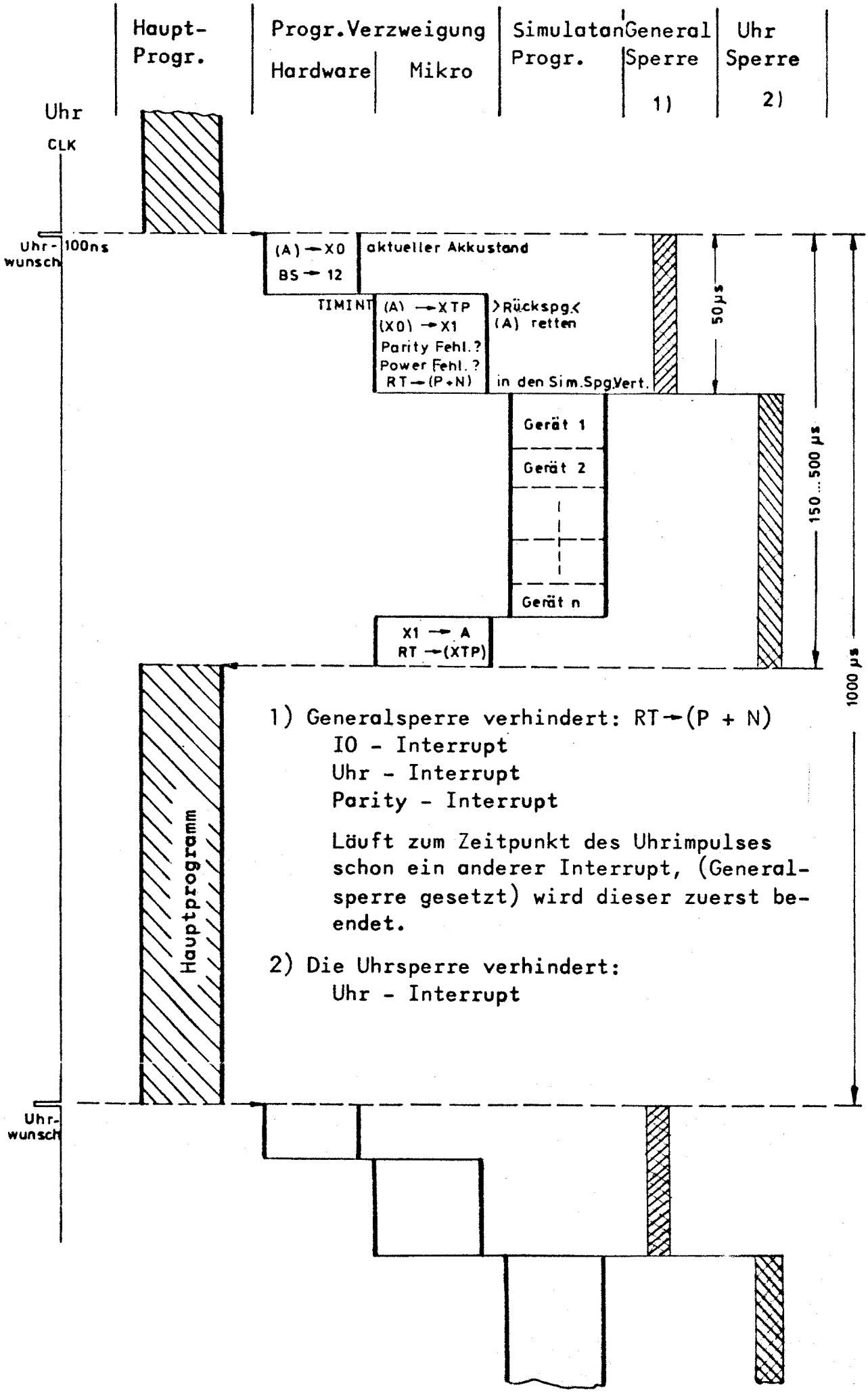
Die Millisekunden-Uhr

Der Rechner besitzt einen Taktgeber, der alle Millisekunden einen Impuls erzeugt.(Zwangsweise werden innerhalb dieser ms alle Geräte bedient, sowie die Fortsetzung des laufenden Programmes sichergestellt).

Beispiel einer Zeiteinteilung



Uhr Interrupt

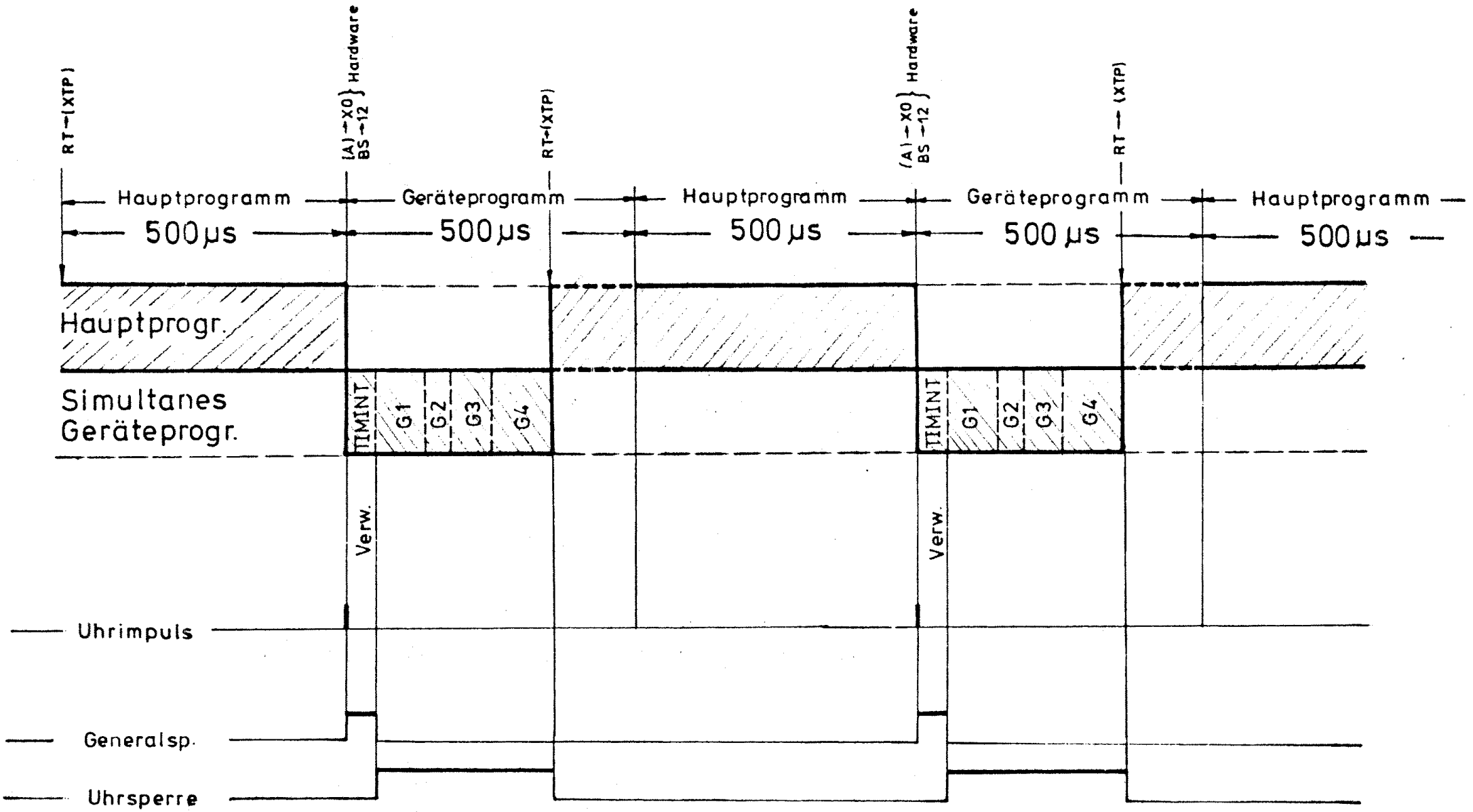


1) Generalsperre verhindert: RT → (P + N)
   IO - Interrupt
   Uhr - Interrupt
   Parity - Interrupt

   Läuft zum Zeitpunkt des Uhrimpulses schon ein anderer Interrupt, (Generalsperre gesetzt) wird dieser zuerst beendet.

2) Die Uhrsperre verhindert:
   Uhr - Interrupt

Beispiel:



G1... G4 z.B. Periph.Geräte
Verw. Uhrverwaltung
XTP namentl. frei gewählte scratchpad Zelle

Uhr Interrupt

Beschreibung der Uhr Unterbrechung

Der laufendene Mikrobefehl wird noch abgearbeitet.

Im Rechner wird hardwaremäßig der Inhalt vom Akku- nach Register X0 gerettet. Durch einen Unterprogrammsprung wird in der Zeropage auf die Adresse 0.0.0.12 gesprungen.

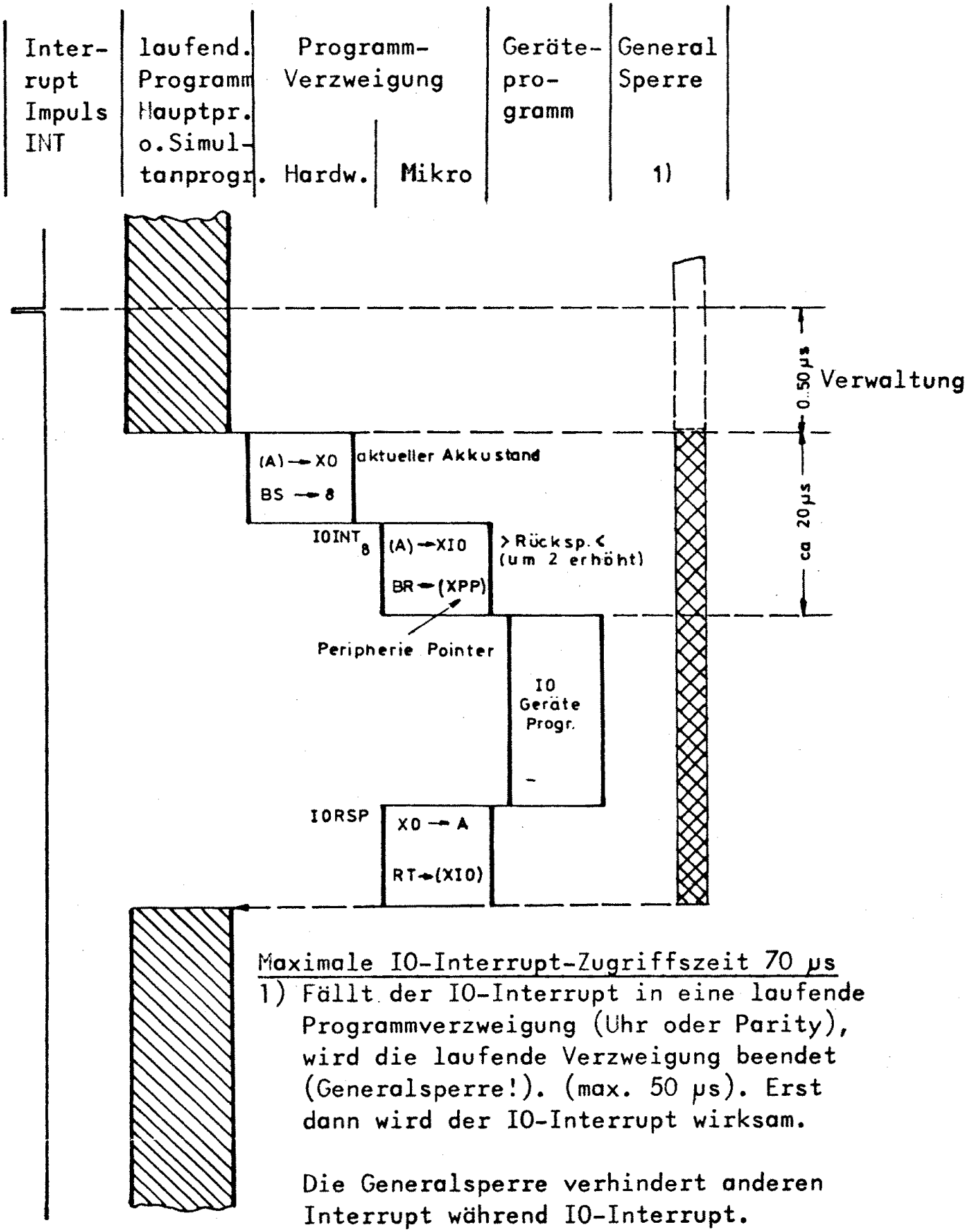
Bei diesem vom Rechner vollführten branch subroutine wird auch hier wie üblich, vor Ausführung des Sprungs der alte Befehlszählerstand um + 2 erhöht und in den Akku geladen. Dort steht jetzt die Rücksprungadresse für das unterbrochene Hauptprogramm. Diese Adresse muß in einer Scratchpadzelle gerettet werden.

Dieser Vorgang läuft unter der Generalsperre ab, d.h. jegliche Unterbrechungen sind untersagt (außer DMA).

Die Zeit die benötigt wird den Akku zu retten, in die Zeropage auf 0.0.0.12 zu springen, die Rücksprung-Adresse ins Hauptprogramm zu retten, auf Netzausfall und auf Parity zu prüfen, nennt man die Verwaltungszeit. Der letzte Befehl dieser Routine ist der RT→(P+N). Er löscht die Generalsperre und setzt die Uhrsperre.

Nach Abschluß der Simultanarbeit wird der gerettete Akku Wert wieder nach A gespeichert, über den Befehl RT→(S+N) wird die Uhrsperre wieder gelöscht und in das Hauptprogramm verzweigt. Die Uhrunterbrechungssperre war während der gesamten Simultanarbeit gesetzt und verhindert fehlerhafte Uhrunterbrechungen.

IO Interrupt



Maximale IO-Interrupt-Zugriffszeit 70 µs

1) Fällt der IO-Interrupt in eine laufende Programmverzweigung (Uhr oder Parity), wird die laufende Verzweigung beendet (Generalsperre!). (max. 50 µs). Erst dann wird der IO-Interrupt wirksam.

Die Generalsperre verhindert anderen Interrupt während IO-Interrupt.

IOINT = IO-Interrupt
IORSP = IO-Rücksprung
XIO, XPP, sind namentlich frei gewählte Scratchpadzellen

EA Unterbrechung, IO Interrupt

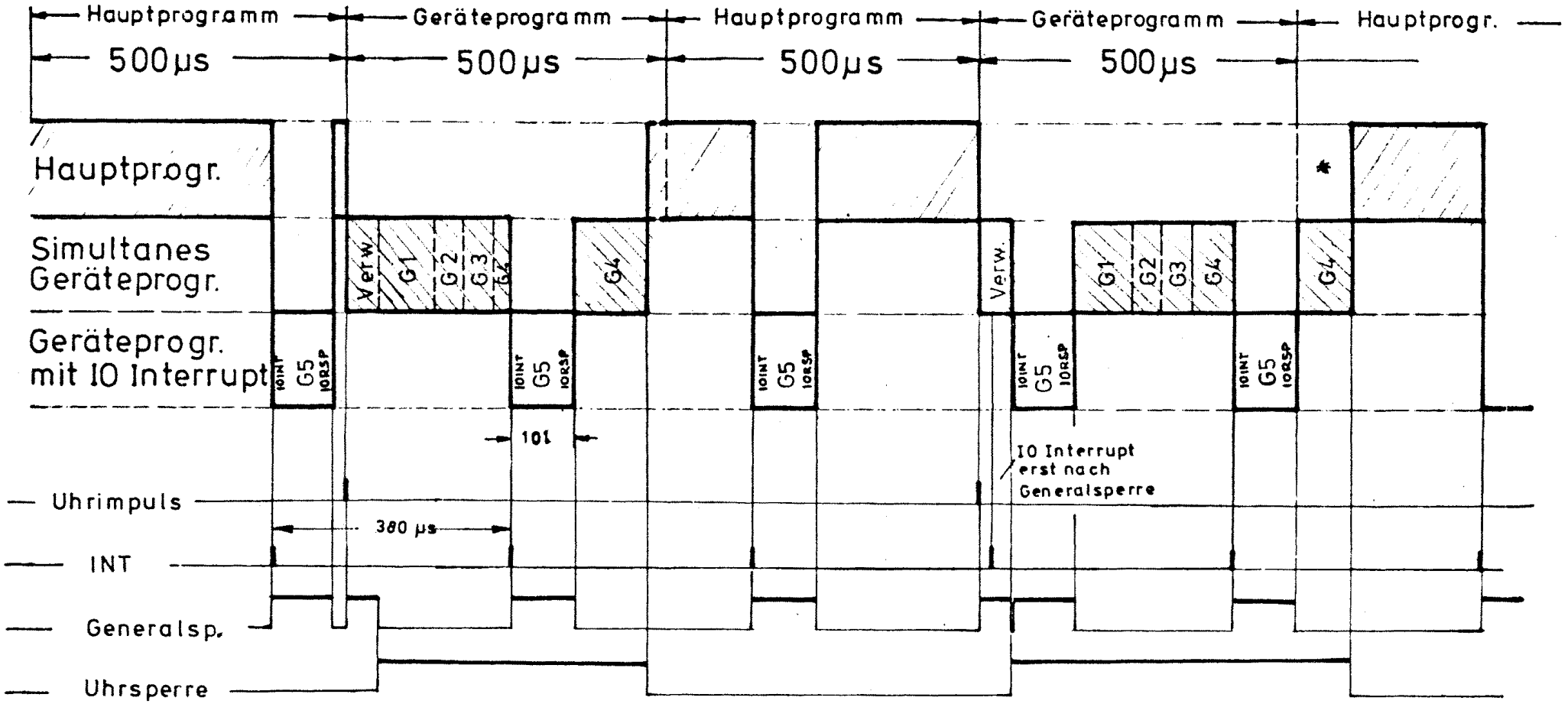
Diese Unterbrechungsart wird für schnelle periphere Geräte notwendig, bei denen ein Datenverkehr öfters als alle Millisekunde stattfindet.

Die Unterbrechung wird vom Gerät selbst über eine Signalleitung INT ausgelöst, sobald Bearbeitungsvorgänge notwendig werden.

Der IO Interrupt besitzt eine zusätzliche Verwaltungszeit von ca 20 µs. Vom Rechner wird hardwaremäßig der Inhalt vom Akku-Wert nach Register X0 gerettet und zum Konnektor IOINT, IO Interrupt durch BS 0.0.0.8 gesprungen. Bei dem vom Rechner vollführten branch subroutine, wird auch wie üblich, vor Ausführung des Sprungs der alte Befehlszählerstand um + 2 erhöht und in den Akku geladen. Dort steht jetzt die Rücksprungadresse für das unterbrochene Haupt- bzw. Geräteprogramm. Diese Adresse wird in die Scratchpadzelle XIO gespeichert und anschließend zum Inhalt der Zelle XPP, peripherie pointer, verzweigt.

Am Schleifenende wird der Akku-Wert wieder nach A geladen und über den Befehl RT→(XIO) rückgesprungen.

Beispiel: Gerät G5 mit einer angenommenen Transfergeschwindigkeit von 2,63 KHz (380 µs Taktzeit) und einer maximalen Zeitschleife von 81 µs plus 20 µs IO Interrupt Verwaltungszeit ergibt 100 µs.



* statt Hauptpr. Simult. Gerätepr.

G1...G5	z.B. Periphere Geräte
Verw.	Uhr Verwaltungszeit
INT	Interrupt Impuls
IOINT	IO Interrupt Verw.
IORSP	IO Interrupt Rücksprung

IO Interrupt

DMA Unterbrechung, "direkter Speicherzugriff", direct memory access

---

Periphere Geräte mit großem Datenanfall, benötigen einen sehr schnellen Datenaustausch.

Daten und Steuersignale werden dabei vom Gerät in einem fest zugeteilten Speicherbereich geholt oder in diesen abgelegt.

Ein Zeitvergleich soll den Unterschied zwischen dem direkten Speicherzugriff und den anderen Unterbrechungsarten veranschaulichen.

Beispiel: Datenaustausch von 100 Bytes

Bei normaler Übertragung, die byteweise über eine Mikrobefehlsschleife durchgeführt wird, benötigt man z.B. 6 Befehle um ein Byte auszugeben bzw. abzuholen. Die Durchschnittsverarbeitungszeit pro Mikrobefehl beträgt 4 µs.

6 Befehle mal 4 µs = 24 µs pro Byte
24 µs mal 100 Bytes ergibt
2,4 ms Übertragungszeit

Die DMA Unterbrechung benötigt für 1 Byte 1,2 µs Zykluszeit

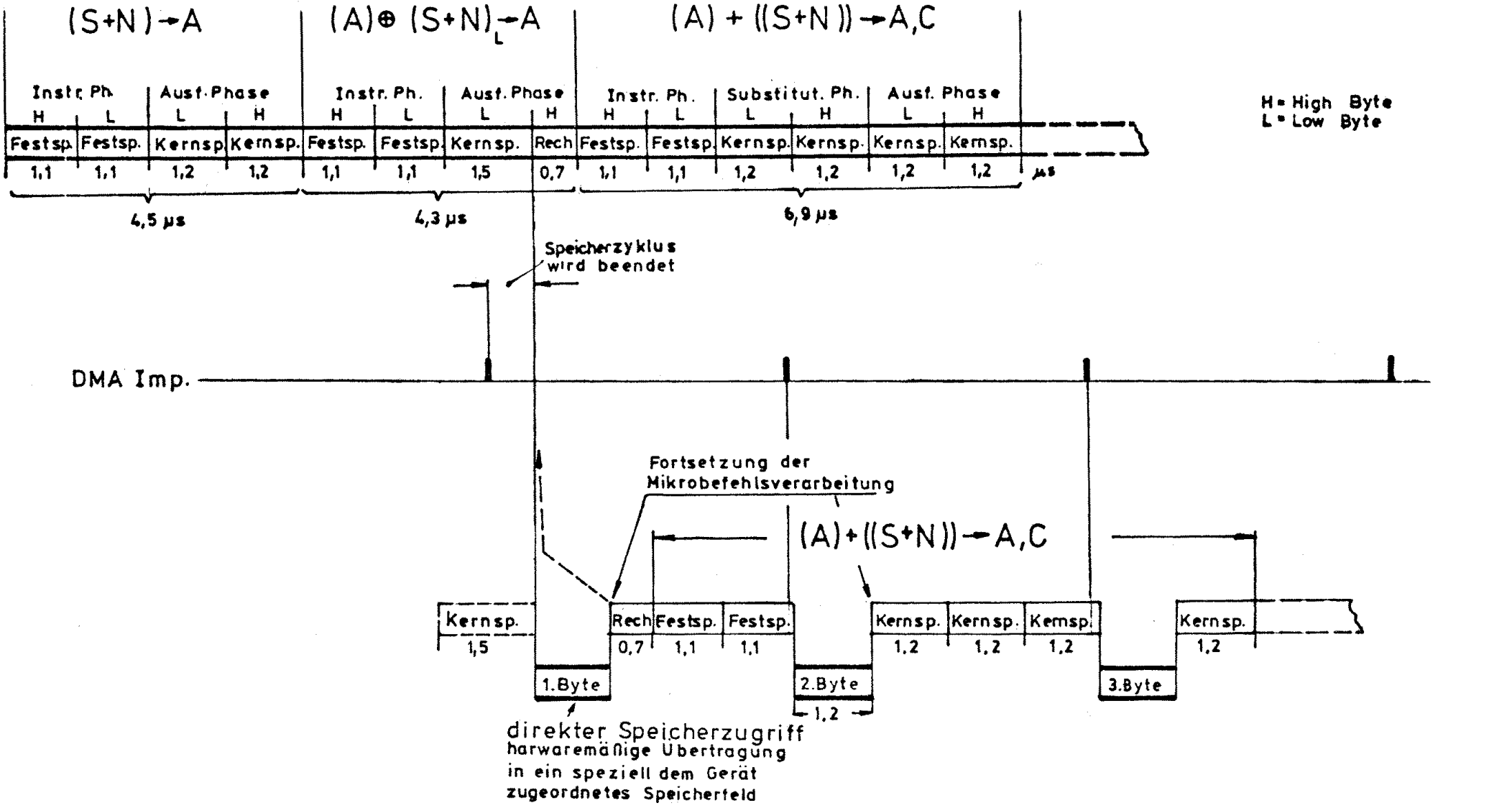
1,2 µs pro Byte mal 100 Bytes ergibt 120 µs Übertragungszeit

Der DMA ist in der Lage, selbst bei gesetzter Generalsperre innerhalb von max. 1,5 µs * den Interrupt zu veranlassen. Die Hardware übernimmt die Datenverkehrssteuerung.

Während des Zugriffs wird der Rechner angehalten und dabei der momentane in Bearbeitung befindliche Befehlszyklus beendet.

Die notwendigen Steuerungen im Gerät werden im Uhrunterbrechungsprogramm (im normalen Simultanen Geräteprogramm) vollzogen. Anschließend erfolgt erst der direkte Speicherzugriff.

* 1,5 µs = längster Befehlszyklus beim Ändern.



DMA Interrupt

Direkter Speicherzugriff

Netzausfallunterbrechung

Das Netzteil gewährleistet, daß nach einer möglichen Netzausfallerkennung noch eine ausreichenden Spannungsversorgung für mehr als 1 ms sichergestellt ist. Auf diese Weise ist eine Sonderunterbrechung nicht notwendig. Die Prüfung auf Netzausfall übernimmt die Uhrunterbrechung. Diese verzweigt gegebenenfalls auf den Befehl "Rechner Löschen".

Dieser Befehlt bewirkt unter anderem eine Löschung aller Ausgabezeilen. Somit werden alle Geräte definiert abgeschaltet und laufen mechanisch aus. Es ist bei dem Aufbau weiterer Peripherie darauf zu achten, daß die gelöschten Ausgaben für das Gerät die Grundstellung bedeutet.

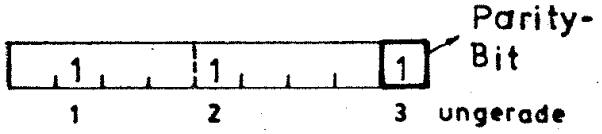
Läuft der Rechner wieder an, wird über die Simultananzeiger der Unterbrechungszeitpunkt abgefragt. Für jedes Gerät wird dieser Zeiger auf einen Neustartpunkt gesetzt, damit die Simultanarbeit mit einem neuen Gesamtarbeitszyklus fortgesetzt werden kann.

Paritäts - Unterbrechung

Diese Unterbrechung ist eine Option. Sie ist im Rechner und im Mikroprogramm bereits enthalten, ist aber nur in Verbindung mit einem 9-Bit-Speicher (8 Bit Information und 1 Bit Parität) wirksam. Tritt während des Mikroprogrammablaufes ein Paritätsfehler im Rechner auf, so verzweigt das Mikroprogramm über Adresse 12 in die Paritätsfehlerroutine, die eine Verzweigung nach INTF 14 vorbereitet. Die Mikroadresse, bei der der Fehler auftrat, wird nach X 0 abgespeichert. Liegt die Generalsperre vor, so bleibt der Paritätsfehler-Unterbrechungswunsch bis zur Behebung der Generalsperre anstehen (z.B. während einer I/O-Interrupt Schleife). Die Paritätsunterbrechung läuft zunächst unter Generalsperre. Die 1-ms-Taktung wird durch die Paritätsfehlerunterbrechung nicht gestört, d.h. ist die Unterbrechung im Simultanprogramm erfolgt, so wird anschließend in dieses zurückgekehrt und erst nach Ablauf der Simultanarbeit auf INTF 14 verzweigt.

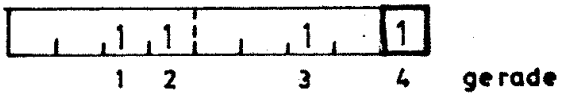
Bei Paritätskontrolle findet die Parity-Prüfung jeweils byteweise statt.

1 byte



Das Parity-bit hat auf ungerade Parität ergänzt (kein Fehler, wenn auf ungerade Parität geprüft wurde).

1 byte



Das Parity-bit hat auf gerade Parität ergänzt (Fehler, wenn auf ungerade Parität geprüft wurde).

Mikroinstruktionssatz

Symbole: → nach

Beispiel

N → A

Ein Festwert (Konstante) N wird nach dem Akkumulator gespeichert.

/ wenn, if

Beispiel

BR/ (A) = Ø → N_Z

Verzweige, wenn der Akkumulatorinhalt Null ist nach N_Z.

= gleich, equal

≠ nichtgleich, ungleich, unequal

Beispiel

BR/ (A) ≠ Ø → N_Z

Verzweige, wenn der Akku-Inhalt ungleich Null ist nach N_Z.

(A1)=1 Bit1 im Akkumulator =1

Beispiel

BR/ (A1)= 1 → P+N

Verzweige, wenn der Inhalt vom Bit 1 im Akku gleich Eins ist nach P+N.

(C) =Ø Übertrags-Flipflop, Carry (siehe Rechnerregister)

Beispiel

BR/ (C) = Ø → N_Z

Verzweige, wenn der Inhalt des 1bitigen C-Registers Null ist nach N_Z.

(C) =1 Verzweige, wenn der Inhalt des C-Registers Eins ist nach N_Z.

(A1...A9) odd Parity — (siehe Paritäts-Unterbrechung)

Beispiel

BR/ (A1...A9) odd Parity ⟶ P+N

Verzweige, wenn im A-Register in A1...A9 eine ungerade Anzahl von "Einsen" stehen.

odd = ungerade, even = gerade

+	Addition	siehe logische Operationen
+1	Zunahme um 1, incrementing	
-1	Abnahme um 1, decrementing	
∧	logisch UND	
V	logisch ODER	
⊕	Exclusiv ODER (EXOR)	
RS	Rechts shiften um 1 Bit	

Ø	Null, um eine Verwechslung mit dem Buchstaben O zu vermeiden, wird die Null mit einem zusätzlichen Schrägstrich versehen.
>...<	bedeutet Adresse z. B. > ANL < = Adresse Anlauf

Vorschlag über die symbolische Schreibweise der Mikroinstruktionen, falls der Ausdruck über ein Druckwerk erfolgen soll.

Schreibweisen:

links= Schreibweise Mikroprogrammierer	rechts=Schreibweise des Mikroprogramm-Ausdruckes beim Assemblieren
⟶	:=
$(S+N)_L$	(S+N).L
$(S+N)_B$	(S+N).B
⊕	*
∧	&
V	!
≠	^=;¬=
$\bar{R}$	-R

Begriffe: (alphabetisch geordnet)

A

Akkumulator, kurz Akku genannt, ist ein 16bitiges Arbeitsregister im Rechner. Alle logischen -, arithmetischen - und Ein/Ausgabebefehle arbeiten mit dem Akku. Am Ende dieser Rechenoperationen steht das Ergebnis im Akkumulator.

Abfrage

Der Begriff Abfrage wird im Mikroprogramm verwendet, etwa zur Feststellung, welcher von zwei Zahlenwerten größer oder kleiner ist.

Adresse

Speicherwort zum Adressieren einer bestimmten Speicherstelle. Z. B. In dem entsprechenden Mikrobefehlswort wird neben den Angaben über die auszuführende Funktion auch die Adresse der vorgesehenen Speicherstelle angegeben.

Befehlszähleradresse ist die Adresse des momentan zu bearbeitenden Befehls.

Assembler

Der Mikroassembler übersetzt bzw. codiert die, auf Lochkarten gestanzte symbolische Mikrobefehlsfolge.

Ausführungsphase

In dieser Ausführungsphase wird die eigentliche Operation des vorliegenden Mikrobefehlswortes vollzogen.
siehe Mikrobefehlsverarbeitung

Ausgabedaten
Ausgabe-Einheit

Als Ausgabedaten werden alle Daten bezeichnet, die über die I/O-Karte und über ein peripheres Gerät auf Datenträger oder auf einen Bildschirm ausgegeben werden.
siehe Ein/Ausgabe-Konzept

B S

Branch subroutine, Unterprogr.-Sprung

Der Unterprogrammsprung dient dazu, an verschiedenen Stellen eines Mikroprogramms zu einem allgemeinen Programmabschnitt, einem Unterprogramm zu verzweigen. Im Unterpro-

gramm stehen immer wiederkehrende, gleiche Mikrobefehlsfolgen, die öfters im normalen Mikroprogrammablauf benötigt werden. Nach Abarbeitung des Unterprogramms wird unmittelbar nach dem Absprung-Befehl (auf den nächstfolgenden Befehl), d.h. von der Unterprogrammebene in das normale Mikroprogramm, rückgesprungen.

C
carry

1bitiges Überlaufregister

Das C-Register ist dem Akkumulator zugeordnet (gewissermaßen als 17. Bit).
Falls bei Addition oder Subtraktionsbefehlen, die in ihrer Befehlsstruktur das C enthalten, die Rechenoperation durchgeführt wird, erfolgt gleichzeitig eine Überprüfung des C-Registers. Bei Überlauf des Akkumulators wird das C-Register auf logisch 1 gesetzt, andernfalls wird es gelöscht.
Bei BS wird (C)→A1 und bei RT wird (A1)→C geladen.

D M A

Direct memory acces, direkter Speicherzugriff

Bei Eingabedaten, die in völlig ungeordneter Reihenfolge und in großer Menge ankommen und sofort verarbeitet werden sollen, ist der Einsatz des direkten Speicherzugriffs erforderlich.
siehe Unterbrechungen

Eingabedaten

Als Eingabedaten werden alle Daten bezeichnet, die über die Ein/Ausgabe (I/O-Karte) von einem peripheren Gerät kommend, in den Rechner eingelesen werden.
siehe Ein/Ausgabe-Konzept

Entscheidungen

Im Mikroprogramm werden Entscheidungen, d. h. Abwägungen zur Auswahl einer zu vollziehenden Alternative eingesetzt.
siehe auch unter "Abfrage"

Befehl
Mikrobefehl
Mikroinstruktion

Der Befehl ist das kleinste Element einer Verarbeitungsaufgabe. Er ist zwei Byte lang.
Im Mikrobefehl ist jeweils ein technisch realisierbarer Verarbeitungsschritt enthalten. Er enthält die Aufgabe über die zu vollziehende

Operation im Operationsteil bzw. im Operationscode und die Angabe über den Speicherplatz oder Direktoperant im Adreßteil des Befehls.

B R
branch

Branch, verzweigen - siehe Sprungbefehle

Die Mikrobefehle werden normalerweise seriell, d. h. in der Reihenfolge ausgeführt, in der sie abgespeichert sind. Diese lineare Verarbeitungsfolge kann durch Sprungbefehle unterbrochen werden. In diesen Fällen wird das Mikroprogramm mit dem Befehl fortgesetzt, der durch den Sprungbefehl gekennzeichnet ist.
Es gibt bedingte und unbedingte Sprungbefehle.
Bei unbedingten Sprungbefehlen wird stets verzweigt; bei bedingten Sprungbefehlen wird nur dann verzweigt, wenn die Sprungbedingung erfüllt ist.

Befehlssatz

Mikrobefehlssatz

Mikroinstruktionssatz

Unter dem Mikrobefehlssatz wird die Gesamtmenge der unterschiedlichen Mikrobefehle verstanden. Der nachfolgend beschriebene Mikrobefehlssatz enthält 132 Mikrobefehle. Die Zusammensetzung des Befehlsvorrates und seine absolute Größe ist für die Leistungsfähigkeit der Anlage von Bedeutung.

Befehlszähler

Instruktionszähler

Um dem Rechner eindeutig vorgeben zu können, welcher der jeweils nächste Befehl ist, der in den Rechner eingelesen werden muß, ist dort ein Befehlszähler des P-Registers vorhanden. Er dient zur schrittweisen Verfolgung der einzelnen Operationen.

Flußdiagramm

Ablaufplan

Flußdiagramme sind graphische Übersichten eines Mikroprogramms mit Sinnbildern für unterschiedliche Abläufe. Sie sind mit kurzen Texterläuterungen versehen, die sich auf den Inhalt bestimmter Teile des Flußdiagramms beziehen.

Instruktion,

Anweisung

siehe unter Befehl

Instruktionsphase

Die Instruktionsphase ist der erste Arbeitsschritt bei der Mikrobefehlsverarbeitung und besteht wieder aus mehreren Teilschritten:

- Abspeichern des aktuellen Befehlszählerstandes
- Feststellen der N-Adresse und des OP-Codes aus dem Mikrobefehlswort
- Ermitteln der neuen Sprungadresse

I/O- Karte

Ein/Ausgabe-Karte
siehe Ein/Ausgabe-Konzept

Komplement

siehe Anhang

Konstante Festwert

Jedes Mikroprogramm kann, je nach Aufgabenstellung, mehrere Konstanten beinhalten, die immer gleich bleiben. Sie dienen z. B. als Vergleichsmaßstab.

N

Das "N" kann, je nach Befehlsart, einen Festwert oder eine Adresse darstellen.

N = Festwert, Direktoperant, Literalbefehl
z.B. N→A, (A) + N→A, (A) + N + HB→A,C

(N)= Direkte Adresse
z.B. (P+N)→A, $(A)_V$ (S+N)→A

((N))= Indirekte Adresse
z.B. BR→(P+N), ((S+N))→A

siehe Adressierungsmöglichkeiten und Befehlsverarbeitung

N_Z

Adresse Zeropage, Nullseite, 1 Speicherseite

Operationen

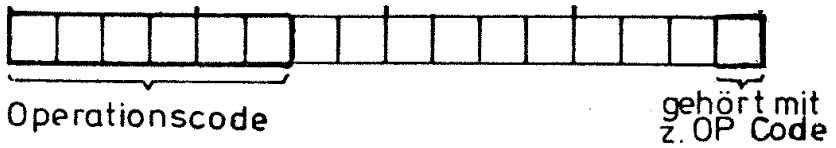
Die Operationen lassen sich in folgende Gruppen einteilen:

- Arithmetische Operationen
  Ihre Aufgabe ist es, die erforderlichen Rechenprozesse durchzuführen.

- Logische Operationen
  Sie dienen zum Vergleich zweier Tatbestände und Treffen eine Feststellung der Abweichung, wie z.B. größer, kleiner.
  Sie sind die Grundlage für Verzweigungen.
- Übertragungsoperationen
  Z. B. Übertragung einer Speicherzelle in den Akkumulator.
- Ein/Ausgabe-Operationen
  Sie stellen den Datenverkehr zwischen den peripheren Geräten und dem Rechner her.

Operationscode

Im Operationscode des Mikrobefehls wird die Auswahl der 132 verschiedenen Instruktionen getroffen.



P + N

P = page, die laufende Seite und die Adresse N des Mikrobefehlswortes.
Currentpage bedeutet laufende Seite. Es ist die Seite, in der der momentane Befehlszähler steht.
siehe Adressierungsmöglichkeiten und Mikroinstruktionsverarbeitung

Programm-Ablaufplan

siehe Flußdiagramm

Programmierer Mikroprogramm

Tätigkeits- bzw. Berufsbezeichnung. Die Aufgabe des Mikroprogrammierers ist es, entsprechend der Aufgabenstellung, die Befehlsfolge zu erstellen und zu prüfen. Dabei wird er ein Flußdiagramm erstellen, damit die Struktur der Aufgabe übersichtlich, in Teilen deutlich erkennbar, wird.

Rechner

Rechenwerk

Ein wichtiger Teil der Zentraleinheit ist der Rechner. Mit ihm ist es möglich, arithmetische, als auch logische Operationen zu vollziehen.
Er gliedert sich in:

- den Mikroinstruktions-Decoder (Steuerwerk)
- die Arithm. logische Einheit
- die Rechner-Register

Register

Rechnerinterne Speicherstellen besonderer Art
siehe Rechnerbeschreibung

R T

Return, Rücksprungbefehle vom Unterbrechungsprogramm ins Hauptprogramm

In der 1. Reihe, Gruppe 3, des Mikroinstruktionssatzes befindet sich der RT→(S+N) RTX. In der Gruppe 4 ist der RT→(P+N) RTL aufgeführt.
siehe Sprungbefehle

Schnittstelle

Die technische Grenze zwischen zwei oder auch mehreren Geräten, Einheiten wird als Schnittstelle (interface) bezeichnet.

S + N

S = scratchpad page und die Adresse N des Mikrobefehls

$S + N_L$

$_L$ bedeutet linkes oder auch hohes Byte. Hier wird nur das linke Byte der 2 Byte-langen Scratchpad-Adresse angesprochen.

$((S + N))_B$

$_B$ bedeutet Byte bzw. Einzelbyte. Ist die indirekte Adresse geradzahlig, wird das linke, ist dieselbe ungeradzahlig, wird das rechte Byte angesprochen.

Startadresse

Die Speicherstelle, an der der erste Befehl zu stehen kommt und von wo aus alle Befehle fortlaufend gespeichert werden, nennt man Startadresse.

Stelle

Die Stelle kennzeichnet einmal die Lage einer Ziffer oder eine Bit-Position (Speicherposition).

Steuerwerk

Der, im Rechner befindliche Mikroinstruktions-Decoder wird auch Steuerwerk genannt.

Substitutionsphase

Der zweite Hauptschritt bei der indirekten Mikrobefehlsverarbeitung ist die Substitutionsphase. In ihr wird die absolute Adresse erzeugt und angesprungen.

Tetrade

Als Tetrade wird eine Folge von jeweils 4 Speicherelementen bezeichnet; z. B. ein Adresswort Bit 1...Bit 16 hat vier Tetraden.

Verzweigung

siehe unter BR branch

Wahlfreier Zugriff

random acces, Direktzugriff

Bestimmte Speichermedien sind technisch so konstruiert, daß entweder jeder Speicherplatz oder jede Gruppe, von Speicherstellen unmittelbar angesprochen werden kann.

Wort

Kennzeichnung einer Zusammenfassung jeweils mehrerer bits; Mikrobefehlswort, Adresswort.

Zugriffszeit

Die Zugriffszeit kennzeichnet den Zeitbedarf, der von Beginn einer Operation zum Lesen des Inhaltes einer bestimmten Speicherstelle (eines Speicherbereiches) bis zum Ende der Übertragung des Inhaltes erforderlich ist.

Operationscode Bit 16 15 14 13 12	0 0 Assembler-schreibweise	0 0 Mathematische Schreibweise	0 1 Assembler-schreibweise	0 1 Mathematische Schreibweise	1 0 Assembler-schreibweise	1 0 Mathematische Schreibweise	1 1 Assembler-schreibweise	1 1 Mathematische Schreibweise
0 0 0 0 0	BS	BS → N_Z (0) (3,4) 1.)	BS	BS → P + N (1) (3,4) 1.)	BSX	BS → (S + N) (5,1) 1.)	BSI	BS → (P + N) (5,4) 1.)
0 0 0 1 0	BR	BR → N_Z (4096)	BR	BR → P + N (4097)	BRX	BR → (S + N)	BRI	BR → (P + N) (5121)
0 0 1 0 0	BE	BR / (A) = 0 → N_Z (8192)	BE	BR / (A) = 0 → P + N	BEX	BR / (A) = 0 → (S + N)	BEI	BR / (A) = 0 → (P + N)
0 0 1 1 0	BU	BR / (A) ≠ 0 → N_Z	BU	BR / (A) ≠ 0 → P + N	BUX	BR / (A) ≠ 0 → (S + N)	BUI	BR / (A) ≠ 0 → (P + N)
0 1 0 0 0	BP	BR/(A1...A9) odd Parity → N_Z 2.)	BP	BR/(A1...A9) odd Parity → P+N 2.)	RTX	RT → (S + N) 3.)	RTI	RT → (P + N) 4)
0 1 0 1 0	BL	BR / (A1) = 1 → N_Z	BL	BR / (A1) = 1 → P + N	BLX	BR / (A1) = 1 → (S + N)	BLI	BR / (A1) = 1 → (P + N)
0 1 1 0 0	BZ	BR / (C) = 0 → N_Z	BZ	BR / (C) = 0 → P + N	BZX	BR / (C) = 0 → (S + N)	BZI	BR / (C) = 0 → (P + N)
0 1 1 1 0	BC	BR / (C) = 1 → N_Z	BC	BR / (C) = 1 → P + N	BCX	BR / (C) = 1 → (S + N)	BCI	BR / (C) = 1 → (P + N)
1 0 0 0 0	LAD	N → A (32768) (3,4)		(3,4)	LA	$(N)_Z$ → A (5,4)	LA	(P + N) → A (33793) (5,4)
1 0 0 1 0	MAD	(A) ∧ N → A			MA	(A) ∧ $(N)_Z$ → A	MA	(A) ∧ (P + N) → A
1 0 1 0 0	EAD	(A) ⊕ N → A			EA	(A) ⊕ $(N)_Z$ → A	EA	(A) ⊕ (P+N) → A (41985)
1 0 1 1 0	OAD	(A) ∨ N → A			OA	(A) ∨ $(N)_Z$ → A	OA	(A) ∨ (P+N) → A
1 1 0 0 0	SY	Sonderbefehl 5.)			IA	$(N)_Z$ + 1 → A	IA	(P + N) + 1 → A
1 1 0 1 0	AFD	(A) + N + M.B. → A, C 6.)			DA	$(N)_Z$ − 1 → A	DA	(P + N) − 1 → A
1 1 1 0 0	CAD	(A, C) + N → A, C			CA	(A, C) + $(N)_Z$ → A, C	CA	(A, C) + (P+ N) → A, C
1 1 1 1 0	AAD	(A) + N → A, C (65 044)			AA	(A) + $(N)_Z$ → A, C	AA	(A) + (P+ N) → A, C
0 0 0 0 1	SML	(A) → S + N (4,2)	SMB	(A) → $(S + N)_B$ (7,3)	SMX	(A) → S + N (5,7)	SMI	(A) → (S + N) (3073) (8,7)
0 0 0 1 1	MML	(A) ∧ $(S + N)_L$ → A, S + N_L	MMB	(A) ∧ $((S+N))_B$ → A $(S+N)_B$	MMX	(A) ∧ (S+N) → A, S+N	MMI	(A) ∧ ((S+N)) → A, (S+N)
0 0 1 0 1	EML	(A) ⊕ $(S+N)_L$ → A, S + N_L	EMB	(A) ⊕ $((S+N))_B$ → A $(S+N)_B$	EMX	(A) ⊕ (S+N) → A, S+N (11264)	EMI	(A) ⊕ ((S+N)) → A, (S+N)
0 0 1 1 1	OML	(A) ∨ $(S+N)_L$ → A, S + N_L	OMB	(A) ∨ $((S+N))_B$ → A $(S+N)_B$	OMX	(A) ∨ (S+N) → A, S+N (15360)	OMI	(A) ∨ ((S+N)) → A, (S+N)
0 1 0 0 1	IML	$(S+N)_L$ + 1 → A, S+N_L	IMB	$((S+N))_B$ + 1 → A $(S+N)_B$	IMX	(S+N) + 1 → A, S+N	IMI	((S+N)) + 1 → A, (S+N)
0 1 0 1 1	DML	$(S+N)_L$ − 1 → A, S+N_L	DMB	$((S+N))_B$ − 1 → A $(S+N)_B$	DMX	(S+N) − 1 → A, S+N	DMI	((S+N))−1 → A, (S+N)
0 1 1 0 1	CML	(A,C) + $(S+N)_L$ → A,C, S+N_L	CMB	(A,C) + $((S+N))_B$ → A,C, $(S+N)_B$	CMX	(A,C) + (S+N) → A,C, S+N	CMI	(A,C) + ((S+N)) → A,C, (S+N)
0 1 1 1 1	AML	(A) + $(S+N)_L$ → A,C, S+N_L	AMB	(A) + $((S+N))_B$ → A,C, $(S+N)_B$	AMX	(A) + (S+N) → A,C, S+N (31744)	AMI	(A) + ((S+N)) → A,C, (S+N)
1 0 0 0 1	LAL	$(S+N)_L$ → A	LAB	$((S+N))_B$ → A (4,6) (7,0)	LAX	(S+N) → A (5,7)	LAI	$((S+N))$ → A (35841) (8,7)
1 0 0 1 1	MAL	(A) ∧ $(S+N)_L$ → A	MAB	(A) ∧ $((S+N))_B$ → A	MAX	(A) ∧ (S+N) → A (39936)	MAI	(A) ∧ ((S+N)) → A
1 0 1 0 1	EAL	(A) ⊕ $(S+N)_L$ → A	EAB	(A) ⊕ $((S+N))_B$ → A	EAX	(A) ⊕ (S+N) → A	EAI	(A) ⊕ ((S+N)) → A
1 0 1 1 1	OAL	(A) ∨ $(S+N)_L$ → A	OAB	(A) ∨ $((S+N))_B$ → A	OAX	(A) ∨ (S+N) → A	OAI	(A) ∨ ((S+N)) → A
1 1 0 0 1	IOL 8.)	IO $(N)_L$ → A (4,1) 7.)	IOB	IO $((S+N))_B$ → A (51201) (6,1) 7.)	IOX 8.)	IO (N) → A (6,5) 7.)	IOI	IO $((S+N))$ → A (4,1) 7.)
1 1 0 1 1	DAL	$(S+N)_L$ − 1 → A	DAB	$((S+N))_B$ − 1 → A	DAX	(S+N) − 1 → A	DAI	$((S+N))$ − 1 → A
1 1 1 0 1	CAL	(A,C) + $(S+N)_L$ → A, C	CAB	(A,C) + $((S+N))_B$ → A,C	CAX	(A,C) + (S+N) → A,C	CAI	(A,C) + ((S+N)) → A,C
1 1 1 1 1	AAL	(A) + $(S+N)_L$ → A, C	AAB	(A) + $((S+N))_B$ → A,C	AAX	(A) + (S+N) → A,C	AAI	(A) + ((S+N)) → A,C

L = Linkes Byte

B = Einzelbyte

Bemerkung:

1. Beim Unterprogrammsprung BS wird vor der Ausführung des Sprungs der um +2 erhöhte alte Befehlszähler P in das A-Register A2 bis A16 und das C-Register nach A1 geladen
   (P) + 2 → A2 ... A16, (C) → A1

2. Dieser Sprungbefehl wird dann ausgeführt, wenn im A-Register in A1 bis A9 eine ungerade Anzahl von "Einsen" stehen.

3. Rücksprung vom Unterbrechungsprogramm ins Hauptprogramm:
   Der Inhalt von B1 der Rücksprungadresse wird nach C geladen.
   Die Generalunterbrechungssperre wird rückgesetzt und wenn die Generalunterbrechungssperre vorher nicht gesetzt war, wird die Uhrunterbrechungssperre gelöscht.

4. Rücksprung vom Unterbrechungsprogramm ins Hauptprogramm:
   Der Inhalt von B1 der Rücksprungadresse wird nach C geladen.
   Die Generalunterbrechungssperre wird rückgesetzt und wenn die Generalunterbrechungssperre vorher gesetzt war, wird die Uhrunterbrechungssperre gesetzt und der Uhrunterbrechungswunsch gelöscht.

5. Sonderbefehl

   Der Sonderbefehl bezieht sich nur auf das A-Register. Im Adreßteil sind nebenstehende Befehle codiert. Kombinationen beider Befehlsgruppen sind möglich.

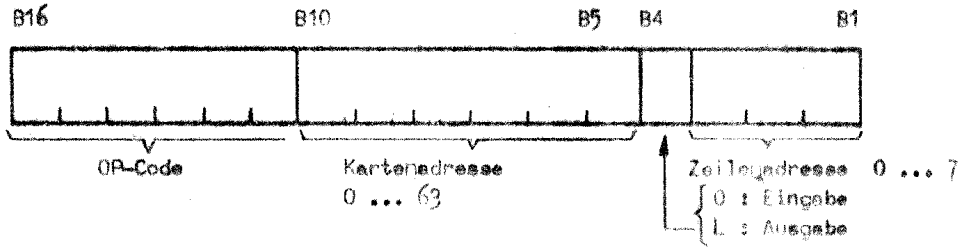
*	B10	B9	
RS	1	0	(A) Rechtsshift um 1 Bit; 0→A16
RC	1	1	(A) Rechtsshift um 1 Bit;(C)→A16
GR	0	1 B1	Löschen Rechner
PR	1	x B3	Löschen Parityfehler (Option)

* Assembler-Schreibweise

*	B3	B2	B1	
CC	0	0	0	: (C) → C
SC	0	0	1	: 1 → C
LC	0	1	0	: (A1) → C
HC	0	1	1	: (A16) → C
TC	1	0	0	: Uhrunterbrechungssperre → C
FC	1	0	1	: Netzausfall → C
PC	1	1	0	: Paritätsfehler → C
DC	1	1	1	: IO Unterbrechungswunsch → C

6. Dem Direktoperanden N werden die "Hohen Bits H.B." 11 ... 16 = 1 hinzugefügt.

7. IO-Befehlsformat:



Mikroinstruktionssatz Bl. 2
Ausführungsstand 1.9.71

Programmunterbrechung: 1 ms Uhr und Parityfehler (als Option) bewirkt (A) → SO, BS → 12, IO-Unterbrechung bewirkt (A) → SO, BS → 8.

Gruppeneinteilung der Mikrobefehle

B6,B12 \ B11, B1	0 0	0 1	1 0	1 1
0 0	Gruppe 1 0.0.0.0 → N_Z Direkter Sprung 1-Seite	Gruppe 2 0.0.0.1 → P+N Direkter Sprung laufende Seite	Gruppe 3 0.4.0.0 → (S+N) Indirekter Sprung laufende Seite	Gruppe 4 0.4.0.1 → (P+N) Indirekter Sprung laufende Seite
1 0	Gruppe 5 8.0.0.0 B1 - B10 → A Laden ger.Werte Direktoperand	Gruppe 5a 8.0.0.1 B1 - B10 → A Laden unger.Werte Direktoperand	Gruppe 6 8.4.0.0 $(N)_Z$ → A Laden 2 Byte direkt 1-Seite	Gruppe 7 8.4.0.1 (P+N) → A Laden 2 Byte direkt laufende Seite
0 1	Gruppe 8 0.8.0.0 (A) → $S+N_L$ Absp. 1 B dir. scratchpad Seite	Gruppe 9 0.8.0.1 (A) → $(S+N)_B$ Absp. 1 B ind. (scratchpad- *	Gruppe 10 0.12.0.0 (A) → S+N Absp. 2 B dir. scratchpad-Seite	Gruppe 11 0.12.0.1 (A) → (S+N) Absp. 2 B indir. (scrachtpad- *
1 1	Gruppe 12,8.8.0.0 $(S+N)_L$ → A Laden 1 Byte dir. scratchpad-Seite	Gruppe 13,8.8.0.1 $((S+N))_B$ → A Laden 1 Byte ind. (scratchpad- *	Gruppe 14 8.12.0.0 (S+N) → A Laden 2 Byte dir. scratchpad-Seite	Gruppe 15 8.12.0.0 ((S+N)) → A Laden 2 Byte indir. scratchpad- *

* Über die Scratchpad adressiert

Mikroinstruktionen, Mikrobefehle

1. Reihe Sprungbefehle (Branchbefehle)

Diese sind unterteilt in:

bedingte Sprünge z.B. BR/(A) = $\emptyset \rightarrow N_Z$

unbedingte Sprünge z.B. BR $\longrightarrow$ N_Z

direkte Sprünge z.B. BR $\longrightarrow$ P+N

z.B. BS $\longrightarrow$ N_Z (Unterprogr. Spr.)

indirekte Sprünge z.B. BR $\longrightarrow$ (S+N)

z.B. RT $\longrightarrow$ (S+N) RTX (Rückspr. v. Unterpr. ins Hauptpr.)

2. Reihe
1. Hälfte Der Oberbegriff für die nachfolgenden Instruktionen lautet Akkumulatorbefehle.

Direktoperandenbefehle (Literalbefehl)

Diese sind unterteilt in:

Akkumulatorbefehle N $\longrightarrow$ A

logische Befehle z.B.(A) $\wedge$ N $\longrightarrow$ A

arithmetische Befehle z.B.(A) + N $\longrightarrow$ A, C

Sonderbefehle

z.B. RS $\emptyset$ $\longrightarrow$ A16 SY (Rechtsshiften)

2. Hälfte

Zeropagebefehle z.B. $(N)_Z$ $\longrightarrow$ A

Currentpagebefehle z.B. (P+N) $\longrightarrow$ A

3. und 4. Reihe Scrathchpad-Befehle

Diese sind unterteile in 2 Hauptgruppen, wobei die 1. Gruppe (3.Reihe) nach Akku und Scratchpad läuft, während die 2. Gruppe nur nach Akku läuft. Diese beiden Hauptgruppen unterteilen sich wieder in

direkte Befehle z.B. $(A) \longrightarrow S+N_L$
indirekte Befehle z.B. $(A) \longrightarrow (S+N)$
logische Befehle z.B. $(A) \wedge (S+N)_L \longrightarrow A, S+N_L$
arithmetische Befehle z.B. $(A,C)+(S+N)_L \longrightarrow A,C,S+N_L$

IO - Befehle in der 4. Reihe
diese sind unterteile in:

direkte Befehle z.B. IO $(N)_L \longleftrightarrow A$
indirekte Befehle z.B. IO $((S+N)) \longleftrightarrow A$
siehe Ein- Ausgabe Konzept

1. Reihe des Mikroinstruktionssatzes bzw. Gruppe 1 ... 4

Unterprogrammsprung BS branch subroutine

Immer wiederkehrende gleiche Mikrobefehlsfolgen, wie z.B. Berechnungs- oder Laderoutinen, sollten zum Unterprogramm deklariert werden. Sie sind nur einmal im gesamten Programm aufzuführen.

Man springt gewissermaßen von der normalen Mikroprogrammebene in eine Unterprogrammebene, dabei muß die um +2 erhöhte Absprungadresse (die Rücksprungadresse) in einer Zelle gerettet werden. (C) wird dabei nach A 1 geladen.

Die Rettung der Rücksprungadresse ist immer der 1. Befehl im Unterprogramm.

Der Akkumulatorinhalt vor dem Absprung wird zerstört. Der Absprung ist so zu wählen, daß der Akku-Inhalt ohne Bedeutung ist.

Im Akku steht dann die Rücksprungadresse.

Am Ende des Unterprogramms wird zum Inhalt dieser Adresse in das normale Mikroprogramm rückgesprungen.

Sprungbefehle BR branch, verzweigen

Gliederung in bedingte und unbedingte Sprungbefehle:

Bedingte Sprungbefehle:

Der Sprung wird nur dann ausgeführt, falls eine bestimmte Bedingung erfüllt ist. Im anderen Fall wird im Programm fortgefahren.

Verzweige wenn:
- -(A)=Ø Inhalt Akku ist gleich Null
- -(A)≠Ø Inhalt Akku ist ungleich Null
- -(A1)=1 Inhalt Bit 1 vom Akku ist Eins
- -(C)=0 Inhalt C ist gleich Null
- -(C)=1 Inhalt C ist gleich Eins
- -(A1...A9) odd Parity. Dieser Sprungbefehl wird dann ausgeführt, wenn im Akkumulator in Bit A1...A9 eine ungerade Anzahl von "Einsen" stehen (siehe unter Paritätsunterbrechung).

Unbedingte Sprungbefehle:

Der Sprung ist von keiner Bedingung abhängig, es wird sofort verzweigt.

Rücksprungbefehle RT

RT → (S+N) Assemblerschreibweise RTX, (Gruppe 3)

Rücksprung vom Unterbrechungsprogramm ins Hauptprogramm. Die Generalunterbrechungssperre wird rückgesetzt und wenn die Generalunterbrechungssperre vorher nicht gesetzt war, wird die Uhrunterbrechungssperre gelöscht.

RT → (P+N) Assemblerschreibweise RTI, (Gruppe 4)

Rücksprung vom Unterbrechungsprogramm ins Hauptprogramm. Die Generalunterbrechungssperre wird rückgesetzt und wenn die Generalunterbrechungssperre vorher gesetzt war, wird die Uhrunterbrechungssperre gesetzt und der Uhrunterbrechungswunsch gelöscht.

Bei beiden Befehlen wird dabei der Inhalt von Bit 1 (Rücksprungadresse) nach C geladen.

Sprungbefehle, branch, in die Zeropage "Z", Nullseite

(0.0.0.0 - 0.3.15.14)

Direkte, bedingte und unbedingte Sprungbefehle nach der 1. Speicherseite, Nullseite.

Diesen Speicherbereich kann man von jeder beleibigen Adresse innerhalb der 64 KB, direkt ansprechen. Deshalb sollten in der Zeropage alle oft benötigten Unterprogramme, Konstanten und Adressen untergebracht werden.

Ist die Sprungbedingung erfüllt, wird der Adressteil des Befehlswortes Bit 2 - Bit 10 in den Befehlszähler P geladen, dabei wird im P-Register Bit 1, Bit 11...16 auf Null gesetzt.

Beim Unterprogrammsprung BS wird vor Ausführung des Sprunges der um +2 erhöhte alte Befehlszähler P in das A-Register A2...A16 und (C) nach A1 geladen. (P) +2 → A2...A16, (C) → A1.

	Mikrobefehlscode	> Sprungziel <	
BS → N_Z	0.0		BS
BR → N_Z	1.0	1	BR
BR/(A) = 0 → N_Z	2.0	1	BE
BR/(A) ≠ 0 → N_Z	3.0	1 1	BU
BR/(A1...A9) odd Parity → N_Z	4.0	1	BP
BR/(A1) = 1 → N_Z	5.0	1 1	BL
BR/(C) = 0 → N_Z	6.0	1 1	BZ
BR/(C) = 1 → N_Z	7.0	1 1 1	BC

Ist die Sprungbedingung nicht erfüllt wird der Befehlszähler (alt) um +2 erhöht.

Sprungbefehle, branch, in der Currentpage (laufende Seite)

Direkte bedingte und unbedingte Sprungbefehle in der Seite in der sich momentan der Befehlszähler befindet, auch Befehlszählerseite genannt. Beim Sprung bleibt man in dieser Page.

Wenn die Sprungbedingung erfüllt ist, wird der Adressteil des Befehlswortes Bit 2 - Bit 10 in den Befehlszähler P übernommen. Die Stellen Bit 11 - Bit 16 des Befehlszählers bleiben erhalten und Bit 1 wird automatisch auf Null gesetzt.

Beim Unterprogrammsprung BS wird vor Ausführung des Sprunges der um +2 erhöhte alte Befehlszähler P in das A-Register A2 - A16 und (C) nach A1 geladen. (P) +2 → A2...A16, (C)→A1

		Mikrobefehlscode						> Sprungziel <									
BS → P+N	0.0+1															1	BS
BR → P+N	1.0+1				1											1	BR
BR/(A)=0 → P+N	2.0+1			1												1	BE
BR/(A)≠0 → P+N	3.0+1			1	1											1	BU
BR/(A1...A9)odd Parity → P+N	4.0+1		1													1	BP
BR/(A1)=1 → P+N	5.0+1		1		1											1	BL
BR/(C)=0 → P+N	6.0+1		1	1												1	BZ
BR/(C)=1 → P+N	7.0+1		1	1	1											1	BC

Ist die Sprungbedingung nicht erfüllt wird der Befehlszähler (alt) um +2 erhöht.

Sprungbefehle, branch, über die Scratchpadseite "S"

Indirekte bedingte und unbedingte Sprungbefehle von der momentanen Seite in die gleiche, oder in eine andere Page.

Die meisten Rechenoperationen der Mikrobefehle arbeiten mit Scratchzellen.

Es steht die komplette Adresse, das Sprungziel, in der Scratchpadzelle Bit 1...16.

Wenn die Sprungbedingung erfüllt ist, wird der Inhalt des im Adressteil des Befehlswortes angegebene scratchpad Speicherwortes in den Befehlszähler übernommen.

Beim Unterprogrammsprung BS, wird vor Ausführung des Sprunges der um +2 erhöhte alte Befehlszähler P in das A-Register A2 - A16 u. (C) nach A1 geladen (P)+2 → A2- A16, (C) → A1.

	Mikrobefehlscode	> Sprungziel <	
BS → (S+N)	0.4	1	BSX
BR → (S+N)	1.4	1 1	BRX
BR/(A) = 0 → (S+N)	2.4	1 1	BEX
BR/(A) ≠ 0 → (S+N)	3.4	1 1 1	BUX
BR/(A1_A9) odd Parity → (S+N)	4.4	1 1	BPX
BR/(A1) = 1 → (S+N)	5.4	1 1 1	BLX
BR/(C) = 0 → (S+N)	6.4	1 1 1	BZX
BR/(C) = 1 → (S+N)	7.4	1 1 1 1	BCX

Ist die Sprungbedinung nicht erfüllt wird der Befehlszähler (alt) um + 2 erhöht.

Sprungbefehl, branch, über die laufende Seite "P"

Indirekte bedingte und unbedingte Sprungbefehle von der momentanen Seite in die gleiche oder in eine andere Page.
Wenn die Sprungbedingung erfüllt ist, wird der Inhalt des im Adressteil des Befehlswortes angegebenen Speicherwortes der Seite in den Befehlszähler übernommen.

Beim Unterprogrammsprung BS, wird vor Ausführung des Sprunges der um +2 erhöhte alte Befehlszähler P in das A-Register A2 → A16, u. (C) nach A1 geladen
(P) +2 → A2 – A16, (C) → A1.

	Mikrobefehlscode	> Sprungziel <	
BS → (P+N)	0.4+1	1	BSI
BR → (P+N)	1.4+1	1	BRI
BR/(A) = 0 → (P+N)	2.4+1	1	BEI
BR/(A) ≠ 0 → (P+N)	3.4+1	1	BUI
BR/(A1...A9) odd Parity → (P+N)	4.4+1	1	BPI
BR/(A1) = 1 → (P+N)	5.4+1	1	BLI
BR/(C) = 0 → (P+N)	6.4+1	1	BZI
BR/(C) = 1 → (P+N)	7.4+1	1	BCI

Ist die Sprungbedingung nicht erfüllt wird der Befehlszähler um +2 erhöht.

2. Reihe des Mikrobefehlssatzes, Gruppe 5 und 5a

Direkoperand, Literalbefehle "N" ist hier ein Festwert

Mit diesen Befehlen werden Festwerte, Konstante in den Akkumulator gespeichert bzw. der Akku-Inhalt wird logisch oder auch arithmetisch mit einer Konstanten verarbeitet. Das Ergebnis steht im Akkumulator.

Das Bit 1 des Befehlswortes ist nicht Bestandteil des Operation Codes, sondern gehört für ungerade Festwerte, mit zum Binärwert N.

Die Gruppe 5a wird für die Verarbeitung der ungeraden Konstanten benötigt.

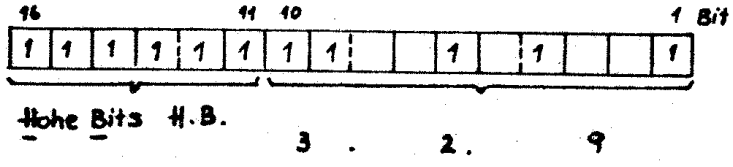
Bei dem Befehl mit dem OP-Code 13.0 (A)+N+H.B. A,C erfolgt bei der Verarbeitung ein automatische Hinzufügen der hohen Bits (H.B.) Bit 11...16.

Will man z.B. von einem Akku-Inhalt einen Wert subtrahieren läßt sich diese Rechenoperation über (A)+N+H.B.→A,C durchführen. Entsteht bei dieser Operation ein Überlauf wird C auf logisch Eins gesetzt.
(A) - 13.6 → A z.B.

Das Komplement von 13.6 = 15.15.2.9 - (A) + 3.2.9 + H.B. → A,C

Der Festwert N belegt nur die Bits 1...10 *

		Mikrobefehlscode															Festwert N	
N→A	8.0	1																LAD
(A)∧N→A	9.0	1			1													MAD
(A)⊕N→A	10.0	1		1														EAD
(A)∨N→A	11.0	1		1	1													OAD
Sonderbefehl	12.0	1	1															SY
(A)+N+H.B.→A,C	13.0	1	1		1													AFD
(A,C)+N→A,C	14.0	1	1	1														CAD
(A)+N→A,C	15.0	1	1	1	1													AAD
		16					11	10									1	Bit



2. Reihe, Gruppe 6

Befehle aus dem 1. Speicherbereich, der Zeropage "Z"

Der Inhalt des Datenwortes der Null-Seite, dessen Adresse im Adreßteil des Befehlswortes steht, wird mit dem Inhalt des Akku's verknüpft und das Ergebnis in den Akku geladen.

Der Inhalt des Operandenspeichers bleibt erhalten, der alte Inhalt des Akku's geht verloren.

	Mikrobefehlscode						Adresse									
$(N)_Z \rightarrow A$	8.4	1				1										LA
$(A) \wedge (N)_Z \rightarrow A$	9.4	1			1	1										MA
$(A) \oplus (N)_Z \rightarrow A$	10.4	1		1		1										EA
$(A) \vee (N)_Z \rightarrow A$	11.4	1		1	1	1										OA
$(N)_Z + 1 \rightarrow A$	12.4	1	1			1										IA
$(N)_Z - 1 \rightarrow A$	13.4	1	1		1	1										DA
$(A,C) + (N)_Z \rightarrow A,C$	14.4	1	1	1		1										CA
$(A) + (N)_Z \rightarrow A,C$	15.4	1	1	1	1	1										AA

2. Reihe, Gruppe 7

Befehle aus dem laufenden Speicherbereich, Currentpage "P"

Der Inhalt des Datenwortes der laufenden Seite, dessen Adresse im Adreßteil des Befehlswortes steht, wird mit dem Inhalt des Akku's verknüpft und das Ergebnis in den Akku geladen.

Der Inhalt des Operandenspeichers bleibt erhalten und der alte Inhalt des Akku's geht verloren.

	Mikrobefehlscode						Adresse									
(P+N) → A	8.4+1	1				1									1	LA
(A) ∧ (P+N) → A	9.4+1	1			1		1									1	MA
(A) ⊕ (P+N) → A	10.4+1	1		1			1									1	EA
(A) ∨ (P+N) → A	11.4+1	1		1	1		1									1	OA
(P+N) + 1 → A	12.4+1	1	1				1									1	IA
(P+N) - 1 → A	13.4+1	1	1		1		1									1	DA
(A,C) + (P+N) → A,C	14.4+1	1	1	1			1									1	CA
(A) + (P+N) → A,C	15.4+1	1	1	1	1		1									1	AA

3. Reihe, Gruppe 8

Linkes Byte aus dem Scratchpadbereich "S" mit Zurückschreiben

Direkte 1 Byte - Abspeicherbefehle (hohes Byte)

Der Inhalt des Speicherwertes des Scratchpadseite, dessen Adresse im Adreßteil des Befehlswortes angegeben ist, wird mit dem Inhalt des Akku's (Bit 1...8) verknüpft und das Ergebnis unter der gleichen Adresse im Scratchpad-Bereich abgespeichert. Im Akku steht der Ergebniswert.

Der ursprüngliche Inhalt des Operanden speichers geht verloren.

In dieser Befehlsgruppe können nur Bytes mit geradzahliger Adresse adressiert werden.

		Mikrobefehlscode						Adresse								
$(A) \rightarrow S+N_L$ *	0.8					1										SML
$(A) \wedge (S+N)_L \rightarrow A, S+N_L$	1.8				1	1										MML
$(A) \oplus (S+N)_L \rightarrow A, S+N_L$	2.8			1		1										EML
$(A) \vee (S+N)_L \rightarrow A, S+N_L$	3.8			1	1	1										OML
$(S+N)_L + 1 \rightarrow A, S+N_L$	4.8		1			1										IML
$(S+N)_L - 1 \rightarrow A, S+N_L$	5.8		1		1	1										DML
$(A, C) + (S+N)_L \rightarrow A, C, S+N_L$	6.8		1	1		1										CML
$(A) + (S+N)_L \rightarrow A, C, S+N_L$	7.8		1	1	1	1										AML

* Bei diesem Befehl bleibt der Akku-Inhalt unverändert.

3. Reihe, Gruppe 9

Einzelbyte Befehle mit indirekter Adressierung über den Scratchpadbereich "S" mit Zurückschreiben

Indirekte 1 Byte - Abspeicherbefehle (hohes oder niederes Byte)

Der Inhalt des Speicherwortes der Scratchpadseite, dessen Adresse im Adreßteil des Befehlswortes steht, ist die Adresse des Operanden (1Byte) der mit dem Inhalt des Akku's verknüpft wird. Das Ergenis wird im Operandenspeicher abgespeichert. Im Akku steht der Ergebniswert.

Der ursprüngliche Inhalt des Operandenspeichers geht verloren.

		Mikrobefehlscode						Adresse								
$(A) \rightarrow (S+N)_B$ *	0.8 + 1					1										1 SMB
$(A) \wedge ((S+N))_B \rightarrow A, (S+N)_B$	1.8 + 1				1	1										1 MMB
$(A) \oplus ((S+N))_B \rightarrow A, (S+N)_B$	2.8 + 1			1		1										1 EMB
$(A) \vee ((S+N))_B \rightarrow A, (S+N)_B$	3.8 + 1			1	1	1										1 OMB
$((S+N))_B + 1 \rightarrow A, (S+N)_B$	4.8 + 1		1			1										1 IMB
$((S+N))_B - 1 \rightarrow A, (S+N)_B$	5.8 + 1		1		1	1										1 DMB
$(A,C) + ((S+N))_B \rightarrow A, C, (S+N)_B$	6.8 + 1		1	1		1										1 CMB
$(A) + ((S+N))_B \rightarrow A, C, (S+N)_B$	7.8 + 1		1	1	1	1										1 AMB

* Bei diesem Befehl bleibt der Akku Inhalt unverändert.

3. Reihe, Gruppe 10

Befehle in den Scratchpadbereich "S" mit Zurückschreiben

Direkte 2 Byte Abspeicherbefehle

Der Inhalt des Adresswortes (2Byte) der Scratchpadseite, dessen Anfangsadresse im Adreßteil des Befehlswortes angegeben ist, wird mit dem Inhalt des Akku's verknüpft und das Ergebnis in den gleichen Speicherbereich abgespeichert.

Im Akku steht der Ergebniswert.

Der ursprüngliche Inhalt des Operandenspeichers geht verloren.

	Mikrobefehlscode						Adresse										
$(A) \rightarrow S+N$ *	0.12					1	1										SMX
$(A) \wedge (S+N) \rightarrow A, S+N$	1.12				1	1	1										MMX
$(A) \oplus (S+N) \rightarrow A, S+N$	2.12			1		1	1										EMX
$(A) \vee (S+N) \rightarrow A, S+N$	3.12			1	1	1	1										OMX
$(S+N) + 1 \rightarrow A, S+N$	4.12		1			1	1										IMX
$(S+N) - 1 \rightarrow A, S+N$	5.12		1		1	1	1										DMX
$(A,C) + (S+N) \rightarrow A,C, S+N$	6.12		1	1		1	1										CMX
$(A) + (S+N) \rightarrow A,C, S+N$	7.12		1	1	1	1	1										AMX

* Bei diesem Befehl bleibt der Akku-Inhalt unverändert.

3. Reihe, Gruppe 11

Befehle mit indirekter Adressierung über den Speicherbereich "S" mit Zurückschreiben

Indirekter 2 Byte Abspeicherbefehl

Der Inhalt des Adreßwortes der Scratchpad Seite, dessen Anfangsadresse im Adreßteil des Befehlswortes steht, ist die Adresse des Operanden (2 Byte), der mit dem Inhalt des Akku's verknüpft wird.

Das Ergebnis wird im Operandenspeicher abgespeichert. Im Akku steht der Ergebniswert.

Der ursprüngliche Inhalt des Operandenspeichers geht verloren.

		Mikrobefehlscode						Adresse								
$(A) \rightarrow (S+N)$ *	0. 12+1					1	1								1	SMI
$(A) \wedge ((S+N)) \rightarrow A, (S+N)$	1. 12+1				1	1	1								1	MMI
$(A) \oplus ((S+N)) \rightarrow A, (S+N)$	2. 12+1			1		1	1								1	EMI
$(A) \vee ((S+N)) \rightarrow A, (S+N)$	3. 12+1			1	1	1	1								1	OMI
$((S+N)) + 1 \rightarrow A, (S+N)$	4. 12+1		1			1	1								1	IMI
$((S+N)) - 1 \rightarrow A, (S+N)$	5. 12+1		1		1	1	1								1	DMI
$(A,C) + ((S+N)) \rightarrow A,C,(S+N)$	6. 12+1		1	1		1	1								1	CMI
$(A) + ((S+N)) \rightarrow A,C,(S+N)$	7. 12+1		1	1	1	1	1								1	AMI

* Bei diesem Befehl bleibt der Akku-Inhalt unverändert.

4. Reihe, Gruppe 12

Linkes Byte aus dem Scratchpadbereich "S" ohne Zurückschreiben

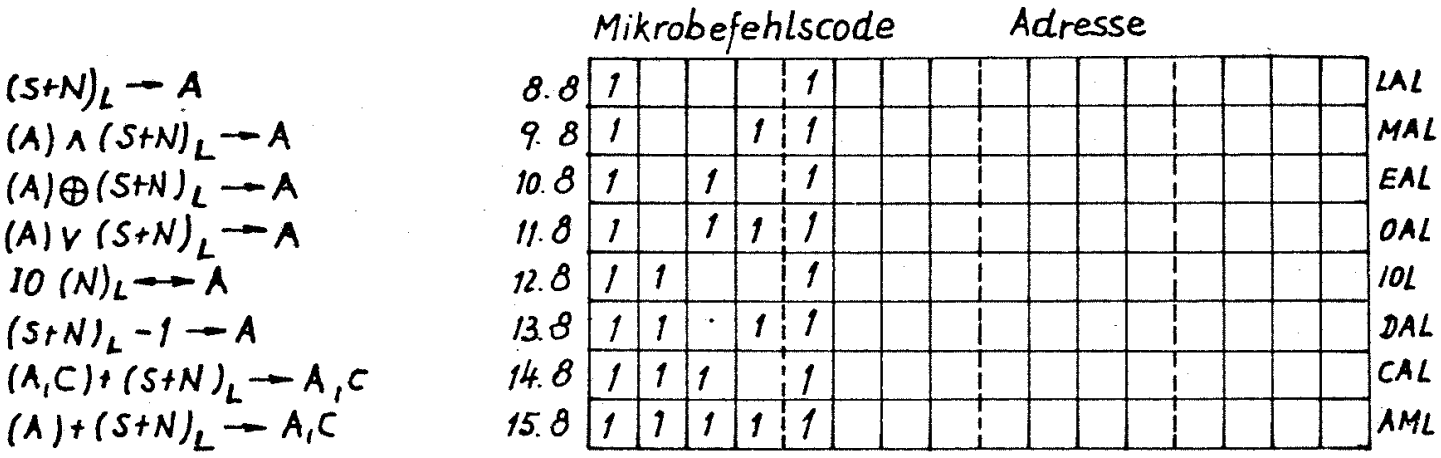
Direkte 1 Byte Ladebefehl (hohes Byte)

Der Inhalt des Speicherwortes der Scratchpad-Seite, dessen Adresse im Adreßteil des Befehlswortes angegeben ist, wird mit dem Inhalt des Akku's (Bit 1...8) verknüpft.

Im Akku steht der Ergebniswert. Der alte Inhalt des Akku's geht verloren.

In dieser Befehlsgruppe können nur Bytes mit geradzahliger Adresse adressiert werden.

Die IO Befehle werden gesondert behandelt.



4. Reihe, Gruppe 13

Einzelbyte Befehle mit indirekter Adressierung
von dem Scratchpadbereich "S" ohne Zurückschreiben

Indirekte 1 Byte Ladebefehle (hohes oder niederes Byte)

Der Inhalt des Speicherwortes der Scratchpad-Seite, dessen Adresse im Adreßteil des Befehlswortes steht, ist die Adresse des Operanden (1 Byte), der mit dem Inhalt des Akku's (B1...B8) verknüpft wird. Das Ergebnis steht im Akkumulator. Der ursprüngliche Inhalt des Operandenspeichers bleibt erhalten.

Die IO Befehle werden gesondert behandelt.

		Mikrobefehlscode						Adresse								
$((S+N))_B \rightarrow A$	8.8+1	1				1									1	LAB
$(A) \wedge ((S+N))_B \rightarrow A$	9.8+1	1			1	1									1	MAB
$(A) \oplus ((S+N))_B \rightarrow A$	10.8+1	1		1		1									1	EAB
$(A) \vee ((S+N))_B \rightarrow A$	11.8+1	1		1	1	1									1	OAB
$IO\ ((S+N))_B \leftrightarrow A$	12.8+1	1	1			1									1	IOB
$((S+N))_B - 1 \rightarrow A$	13.8+1	1	1		1	1									1	DAB
$(A,C) + ((S+N))_B \rightarrow A,C$	14.8+1	1	1	1		1									1	CAB
$(A) + ((S+N))_B \rightarrow A,C$	15.8+1	1	1	1	1	1									1	AAB

4. Reihe, Gruppe 14

Befehle aus dem Scratchpadbereich"S" ohne Zurückschreiben

Direkte 2 Byte Ladebefehle

Der Inhalt des Datenwortes (2 Byte der Scratchpad-Seite, dessen Anfangsadresse im Adreßteil des Befehlswortes angegeben ist, wird mit dem Inhalt des Akku's verknüpft. Im Akku steht der Ergebniswert.

Der Inhalt des Operandenspeichers bleibt erhalten, der alte Akku Inhalt geht verloren.

Die IO Befehle werden gesondert behandelt.

	Mikrobefehlscode						Adresse									
$(S+N) \rightarrow A$	8.12	1				1	1									LAX
$(A) \wedge (S+N) \rightarrow A$	9.12	1			1	1	1									MAX
$(A) \oplus (S+N) \rightarrow A$	10.12	1		1		1	1									EAX
$(A) \vee (S+N) \rightarrow A$	11.12	1		1	1	1	1									OAX
$IO\ (N) \rightarrow A$	12.12	1	1			1	1									IOX
$(S+N) - 1 \rightarrow A$	13.12	1	1		1	1	1									DAX
$(A_1C) + (S+N) \rightarrow A_1C_1$	14.12	1	1	1		1	1									CAX
$(A) + (S+N) \rightarrow A_1C$	15.12	1	1	1	1	1	1									AAX

4. Reihe, Gruppe 15

Befehle mit indirekter Adressierung aus dem Scratchpadbereich "S" ohne Zurückschreiben

Indirekte 2 Byte Ladebefehle

Der Inhalt des Adreßwortes der Scratchpadseite, dessen Adresse im Adreßteil des Befehlswortes steht, ist die Adresse des Operanden (2 Byte) der mit dem Inhalt des Akku's verknüpft wird. Im Akku steht der Ergebniswert. Der Inhalt des Operandenspeichers bleibt erhalten, der alte Akku Inhalt geht verloren.

Die IO Befehle werden gesondert behandelt.

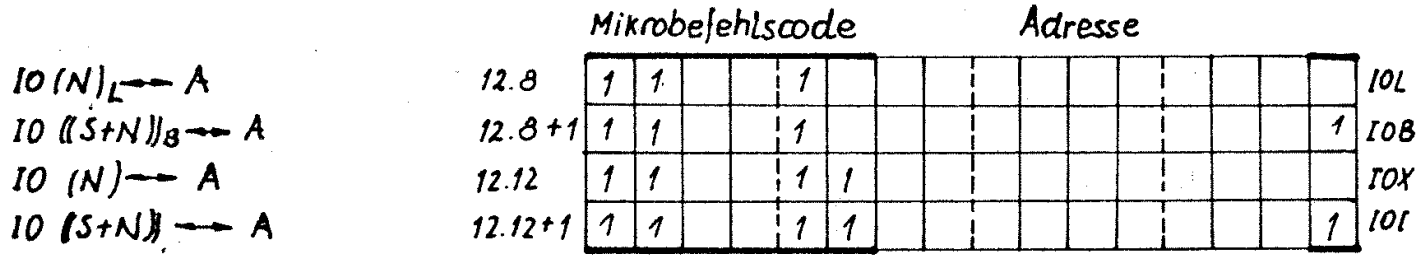
		Mikrobefehlscode						Adresse									
((S+N)) → A	8.12 +1	1				1	1									1	LAI
(A) ∧ ((S+N)) → A	9.12 +1	1			1	1	1									1	MAI
(A) ⊕ ((S+N)) → A	10.12 +1	1		1		1	1									1	EAI
(A) ∨ ((S+N)) → A	11.12 +1	1		1	1	1	1									1	OAI
IO((S+N)) → A	12.12 +1	1	1			1	1									1	IOI
((S+N)) - 1 → A	13.12 +1	1	1		1	1	1									1	DAI
(A,C) + ((S+N)) → A,C	14.12 +1	1	1	1		1	1									1	CAI
(A) + ((S+N)) → A,C	15.12 +1	1	1	1	1	1	1									1	AAI

IO Befehle (input output Befehle)
(siehe Ein-Ausgabekonzept)

Im Bit 4 des Befehlswortes wird angegeben, ob es sich bei dem Befehl um eine Eingabe oder Ausgabe handelt.

Bei der Ausgabe wird der Inhalt des Akku's in das im Adreßteil des Befehlswortes adressierte periphere Gerät über die IO-Karte ausgegeben.

Bei der Eingabe wird die Information, die über die IO-Karte anliegt, von dem im Adreßteil des im Befehlswort adressierten peripheren Gerät in den Akku übernommen.



Beschreibung einiger Sonderbefehle

LOESCHEN RECHNER (siehe Netzausgallunterbrechung)

Dieser Befehl bewirkt eine Löschung der Rechner Register I, P und R. Der Rechner gibt ein Reset-Signal ab.

PARITÄTSFEHLER
PARITYFEHLER

Mit jedem Uhrimpuls wird im Rechner (A)→X0 gerettet, über BS → 12 wird nach Zelle 12 gesprungen.

PARITY 12.2.0.4 (SY RS TC)

Durch den Sonderbefehl Uhrsperre C kann man z.B. prüfen, ob diese Unterbrechung im Haupt- oder im Geräteprogramm stattfand. Bei (C) = Ø erfolgt Parityfehler im Hauptprogramm.

IO - UNTERBRECHUNG 12.0.0.7

Über diesen Sonderbefehl kann man prüfen ob ein IO-Interrupt vorliegt.

Sonderbefehl

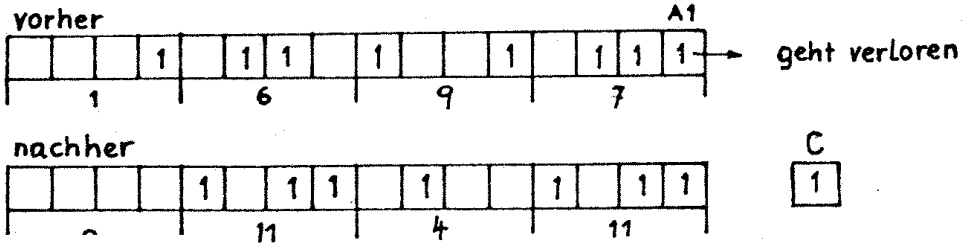
(A) Rechtsshift um 1Bit; ∅ → A16	12.2.0.0	1	1					1											RS
(A) Rechtsshift um 1Bit; (C) → A16	12.3.0.0	1	1					1	1										RC
Löschen Rechner	12.1.0.1	1	1						1									1	GR
Löschen Parityfehler (Option)	12.2.0.4	1	1					1	x							1			PR
									wahlweise										

*(C) → C	12.0.0.0	1	1																CC
1 → C	12.0.0.1	1	1															1	SC
(A1) → C	12.0.0.2	1	1														1		LC
(A16) → C	12.0.0.3	1	1														1	1	HC
Uhrunterbrechungssperre	12.0.0.4	1	1													1			TC
Netzausfall	12.0.0.5	1	1													1		1	FC
Paritätsfehler	12.0.0.6	1	1													1	1		PC
IO Unterbrechungssperre	12.0.0.7	1	1													1	1	1	DC

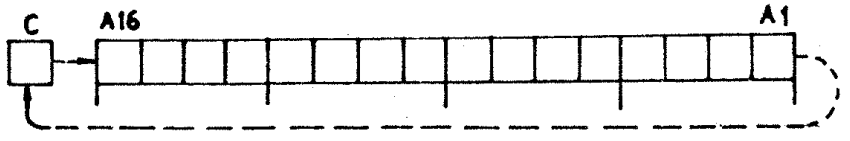
Kombinationen beider Befehlsgruppen:

Befehlscode	Assembler schreibweise			Mikrobefehlscode	
12.2.0.1	SY	RS	SC	(A) > 1 → A, 1 → C	Rechtsshift
12.2.0.2	SY	RS	LC	(A) > 1 → A, (A1) → C	Rechtsschift
12.2.0.3	SY	RS	HC	∅ → A16, A16 → C	
12.2.0.4	SY	PR	FC	LOE PAR, PWF → C	
12.2.0.5	SY	PR	TC	UHRSPERR → C, LOE PAR.	
12.3.0.2	SY	RC	LC	(A) > 1 → A, (C) → A16, (A1) → C, Rundshift	

Rechtsshift um 1 Bit, 1→C.



Rundshift



(A) > 1 → A,
(C) → A16, (A1) → C

* "C" soll erhalten bleiben.

Mikroinstruktionsverarbeitung

Verarbeitung der Mikroinstruktionen

Die Verarbeitung eines Mikrobefehls erfolgt je nach Befehlsart in unterschiedlichen Schritten. Die indirekte Befehlsverarbeitung benötigt die maximale Verarbeitungszeit, sie durchläuft folgende Phasen.

Instruktionsphase
Substitutionsphase
Ausführungsphase

Die Instruktionsphase IPH, der 1. Arbeitschritt, besteht aus folgenden Tätigkeiten

- Abspeichern des aktuellen Befehlszählerstand in das im Rechner befindliche P-Register

- Aus dem Speicher den Mikrobefehl lesen, dabei den OP-Code von der N-Adresse trennen. Der Operantionscode wird in das I-Register, die Adresse in das R-Register gespeichert.

Die Substitutionsphase SPH

Die Substitutionsphase wird nur bei indirekter Befehlsverarbeitung durchlaufen.

Lesen aus dem Speicherbereich der im Mikrobefehl angegebenen Adresse. Der gelesene Wert ist die absolute Adresse die für die Ausführungsphase benötigt wird.

Die Ausführungsphase APH

- Entsprechend der im OP-Code stehende Instruktion wird hier diese Operation ausgeführt.

- Die Adresse des Befehlszählers wird um 2 für die nächste Befehlsverarbeitung, erhöht.

Anmerkung für die Rechner Ablaufsteuerung

$f_0 \ldots f_6$ sind Zähler Flip Flops des Rechners. Sie dienen zur Steuerung des Befehlsablaufes. Weitere Kenntnisse für das Verständnis ist nicht erforderlich.

BR → Nz; BR → P+N; Direktoperant (lit)
BR → (S+N); BR → (P+N);

Rechner-Ablaufsteuerung

			f_6	f_5	f_4	f_3	f_2	f_1	f_0	Sign.	Funktionen
I	H	R	0	0	0	0	0	0	0	PI	(E) → I; (R) → P; B → BA
		W	0	(0)	0	(0)	0	1	1		
	L	R	0	0	0	0	1	1	0	BR	(E) → B, BA, BR
		W	0	1	0	(0)	1	1	1	RHL	I9, I10, Page → B, RH; (BR) → RL
S	L	R	1	1	0	0	1	1	0	BR	(E) → B, BA, BR
		W	1	1	0	0	1	0	1		
	H	R	1	1	0	0	0	0	0		
		W	1	1	0	1	0	1	1	RHL	(E) → B, RH; (BR) → RL
A	L	R	1	1	1	1	1	1	0	TRL BR	(E) $\overset{+}{\wedge v}$ (AL) (C) → B, BA, BR, fl
		W	1	1	1	1	1	0	1	AL BR	(PL) + 2 → B, BR, fz; (BA) → AL
	H	R	1	1	1	1	0	0	0		(E) $\overset{+}{\wedge v}$ (AH) (fl) → B, BA, C
		W	1	0	1	0	0	0	1	AH RHL	(PH) + (fz) → B, RH; (BA) → AH; (BR) → RL

BR	BS
	(PL) + 2 → B, BA, BR, fl
(E) → B, BR; (BA) → AL	
	(PH) + fl → B, BA
(E) → B, RH; (BR) → RL; (BA) → AH	

R = Read-Teilzyklus; W = Write-Teilzyklus
L = Byte low; H = Byte high
I = Instruktionsphase; S = Substitutionsphase; A = Ausführungsphase

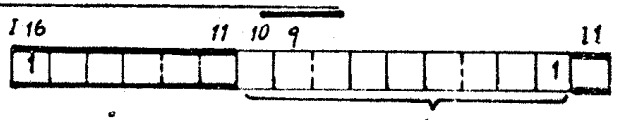
Beispiel:

Mikrobefehlsverarbeitung dargestellt in den einzelnen Phasen.

IPH = Instruktionsphase APH = Ausführungsphase BA = Hilfsspeicher für Akku
SPH = Substitutionsphase ALU = Arithmetic Logic unit BR = Hilfsspeicher für R-Register

Befehl

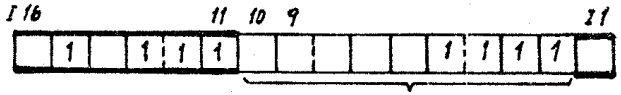
SGO N → A (2 → A) SGO = Konnektor im Mikro, 0.1.6.12 8.0.0.2 SGO LAD2 SGO 2:=A



P-Reg : >SGO< = 0.1.6.12 ⟶ Befehlszählerstand
IPH : $(SGO)_{HIGH}$ = 8.0 = OP Code (I11-I16) → E, → I-Reg. Operationscode erfassen
APH : $(SGO)_{LOW}$ = 0.0.2 N (+ I9 u. I10) = Wert → E → BA Festwertverarbeitung u. Zwischensp. im BA-Reg. N = 0.0.0.2
$(P)_{LOW}$ + 2 → BR, (BA) → A_{LOW}
$(P)_{HIGH}$ + Ü → RH, (BA) → A_{HIGH}, BR-R_L (P) = alter Befehlszählerstand + 2, (A) = 2 (Festwert)

Befehl

MP9 (S+N) - 1 → A, S+N MP9 = Konnektor im Mikro, 0.10.10.14 5.12.1.14 MP9 DMX XCI (XCT) - 1:= XCT, A

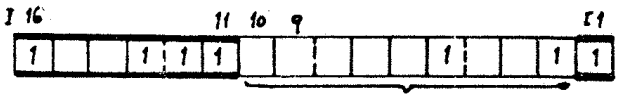


P-Reg : >MP9< = 0.10.10.14 ⟶ Befehlszählerstand
IPH : $(MP9)_{HIGH}$ = OP Code (I11-I16) → E, → I-Reg. Operationscode erfassen
$(MP9)_{LOW}$ = N (+I9 u. I10) → R-Reg. = XCT, Page=P Adr. XNR einstellen → R, Page SC (SC=6.0) · N = 0.0.1.14
APH : $(XCT)_{LOW}$ = Wert - 1 → BA (über ALU) Befehlsverarbeitung nied. Byte, +P = 6.0
$(P)_{LOW}$ + 2 → BR, (BA) → A_{LOW} Niederer Befehlsz. + 2, 1.Teilergebnis → A_L (R) = 6.0.1.14
$(XCT)_{HIGH}$ = Wert → (BA) (über ALU) Befehlsverarbeitung hohes Byte 2.Teilerg.
$(P)_{HIGH}$ +Ü → RH, (BA) → A_H, BR → R_L Hoher Befehlsz. + Übertrag, 2.Teilerg. (BA) → A_{HIGH}
(P) = alter Befehlszählerstand + 2, (A) = Gesamtergebnis

$\underline{(A) \wedge ((S+N)) \rightarrow A}$
$(ADX) = Wert$

Befehl

0.14.3.4 9.12.1.3 MAI XRC (A) & ((XRC)) :=A



P-Reg : 0.14.3.4 Befehlszählerstand

IPH : $(Befehl)_H$ = OP Code (I11-I16) → E, → I-Reg. Operationscode erfassen

$(Befehl)_{LOW}$ = N (+I9 I10) → R-Reg. = XRC, Page- P Adr. XCR einstellen → R Page = 6.0

N= 0.0.1.2

SPH : $(XRC)_{LOW}$ = ADX_{LOW} → E → BA, BR Inhalt niederes Byte XRC=ADX } → R-Register

$(XRC)_{HIGH}$ = ADX_{HIGH} → E → RH (BR) → RL Inhalt hohes Byte XRC - ADX

+P= 6.0

(R)= 6.0.1.2

APH : $(ADX)_{LOW}$ = Wert ∧ $(A)_L$ → BA (über ALU) Befehlsverarbeitung niederes Byte

$(P)_{LOW}$ +2 → BR, (BA) → A_{LOW} Niederer Befehlszähler + 2, 1. Teilerg. → A_{LOW}

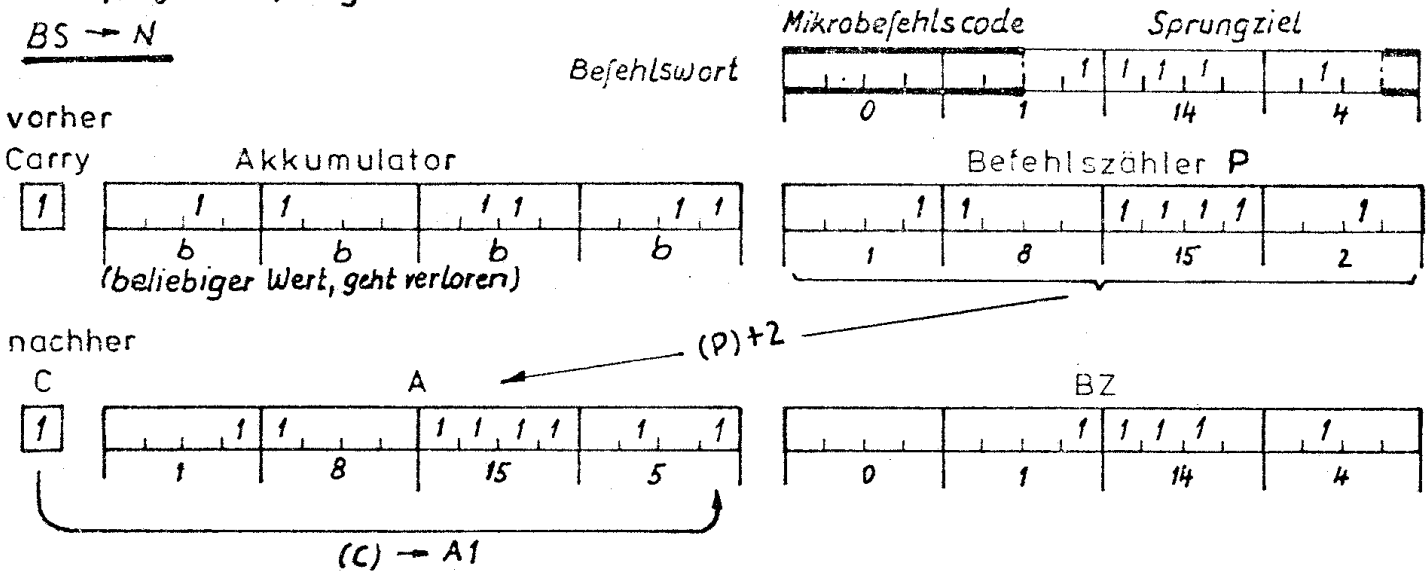
$(ADX)_{HIGH}$ = Wert ∧ $(A)_H$ → BA, BR. Befehlsverarbeitung hohes Byte, 2.Teilerg. → BA

$(P)_{HIGH}$ +Ü → RH, (BA) → A_H (BR) → RL Hoherbefehlszählerinhalt + Übertrag, 2. Teilerg. (BA) → A_H

(P) = alter Befehlszählerstand + 2, (A) = Gesamtergebnis

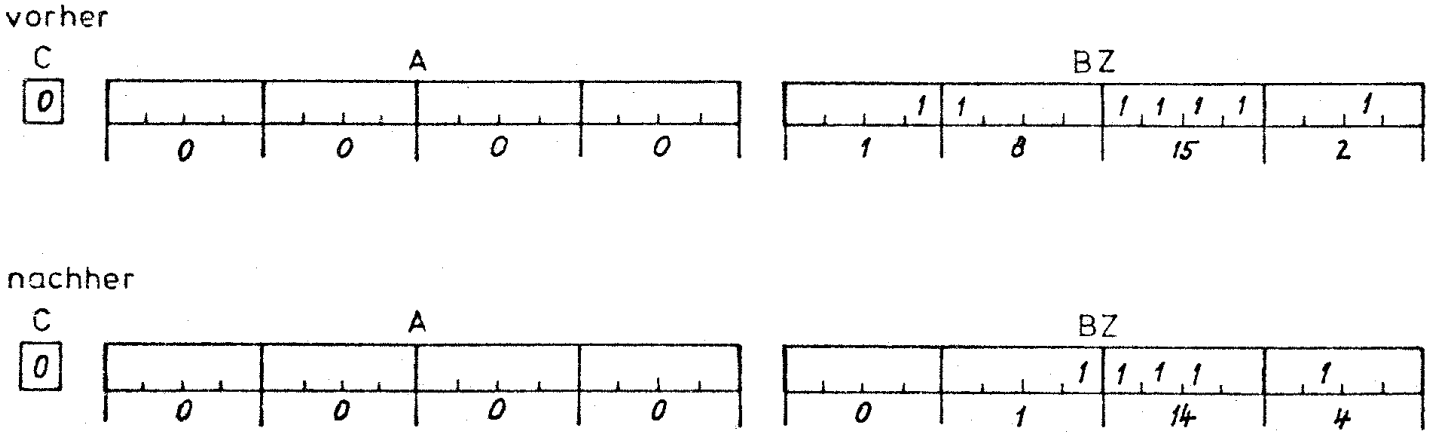
Unterprogrammsprung

BS → N

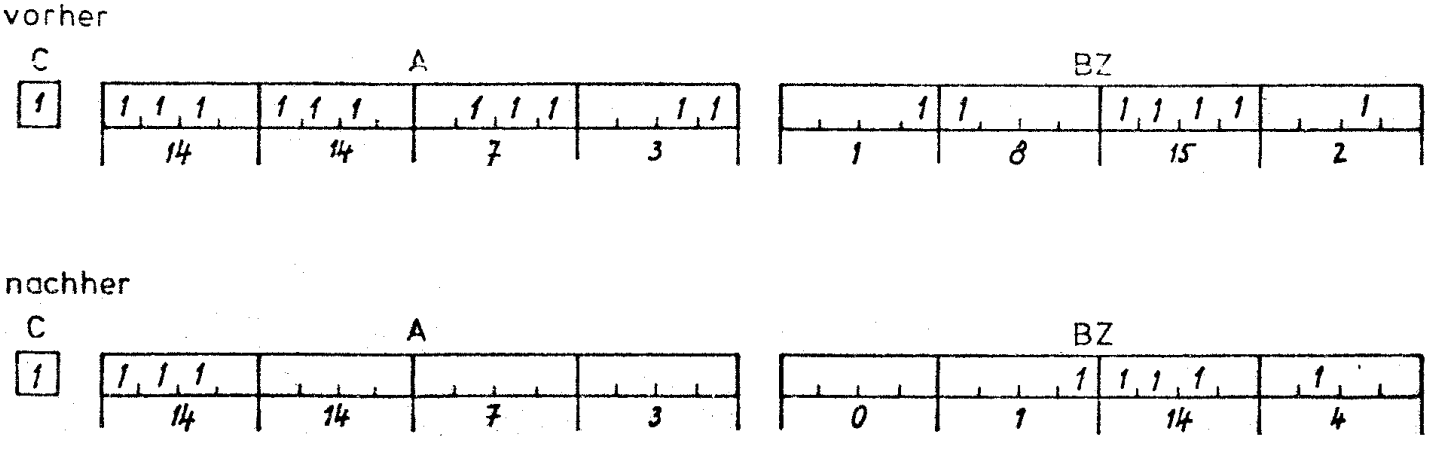


b = beliebig

BR → N
BR / (A) = ∅ → N
BR / (C) = ∅ → N

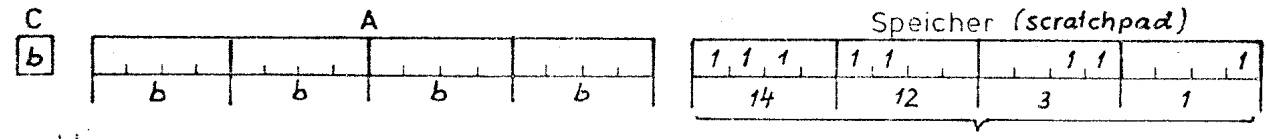


BR / (A) ≠ ∅ → N
BR / (A1) = 1 → N
BR / (C) = 1 → N
BR / (A1 ... A9) odd Parity → N ungerade Parität

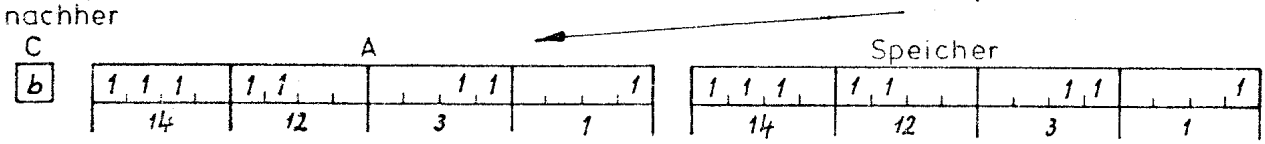


XWK → A

vorher

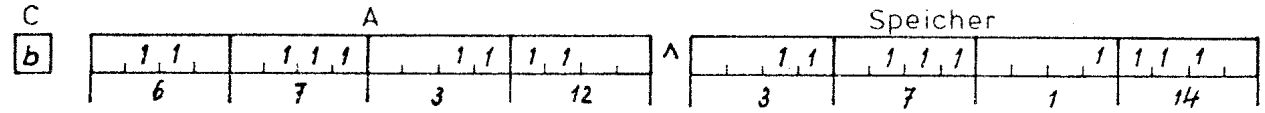


nachher

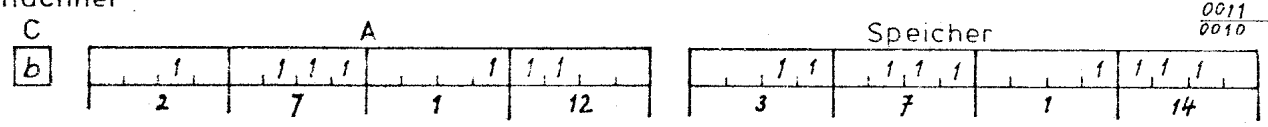


(A) ∧ (XWK) → A

vorher



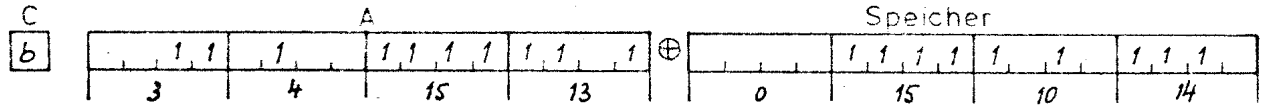
nachher



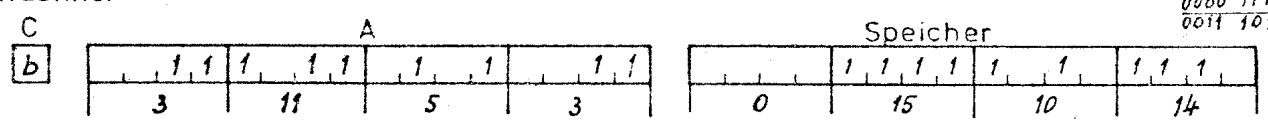
0110 0111 0011 1100
0011 0111 0001 1110 ∧
0010 0111 0001 1100

(A) ⊕ (XWK) → A

vorher



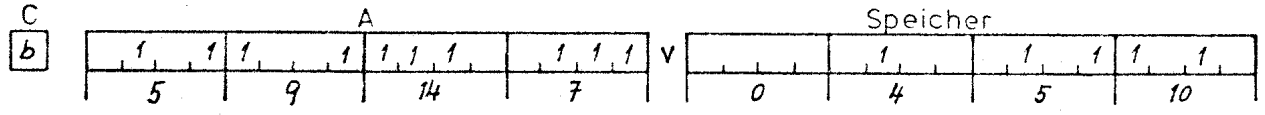
nachher



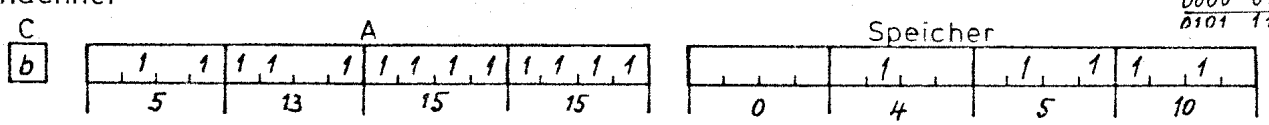
0011 0100 1111 1101
0000 1111 1010 1110 ⊕
0011 1011 0101 0011

(A) ∨ (XWK) → A

vorher



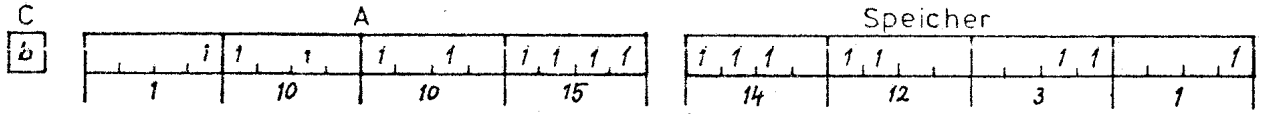
nachher



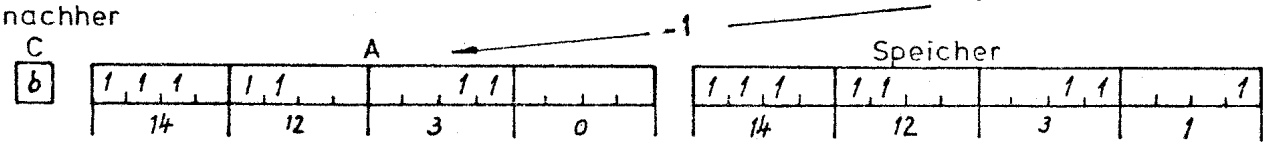
0101 1001 1110 0111
0000 0100 0101 1010 ∨
0101 1101 1111 1111

(XWK) − 1 → A

vorher



nachher



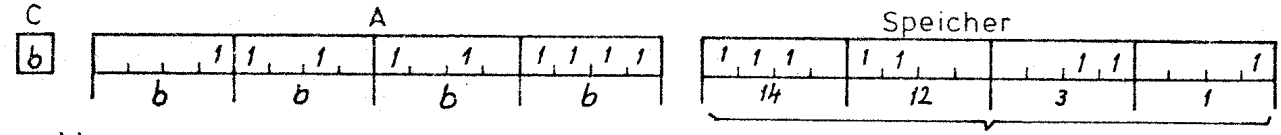
b = beliebig

XWK
↓
Scratchpadzelle

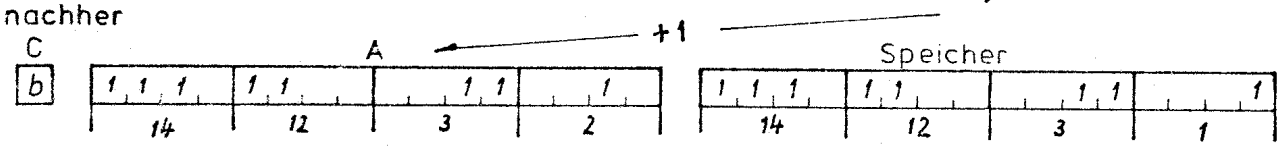
b = beliebig

(XWK) + 1 → A

vorher

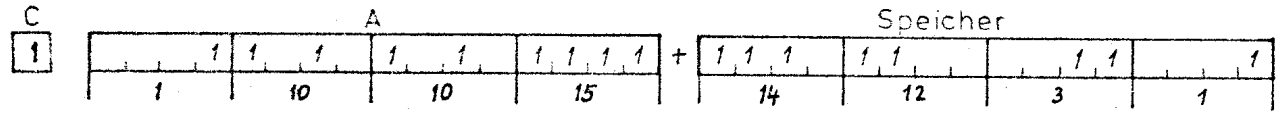


nachher



(A,C) + (XWK) → A,C

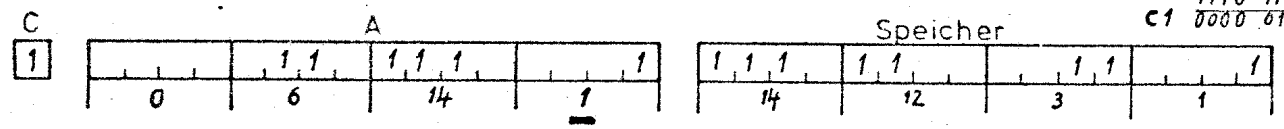
vorher



1(C) alt

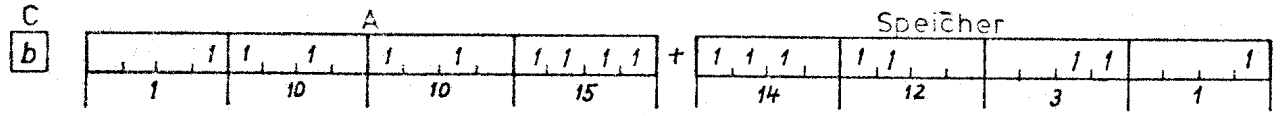
0001 1010 1010 1111
1110 1100 0011 0001 +
C1 0000 0110 1110 0001

nachher



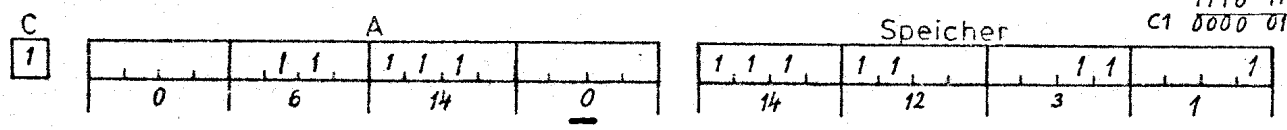
(A) + (XWK) → A,C

vorher



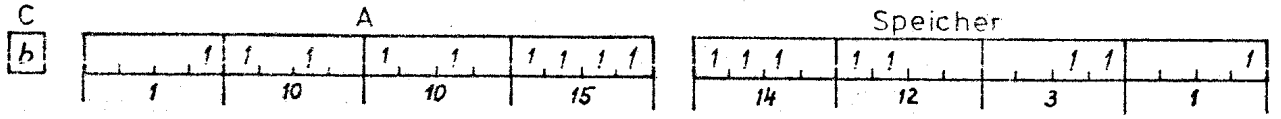
nachher

0001 1010 1010 1111
1110 1100 0011 0001 +
C1 0000 0110 1110 0000

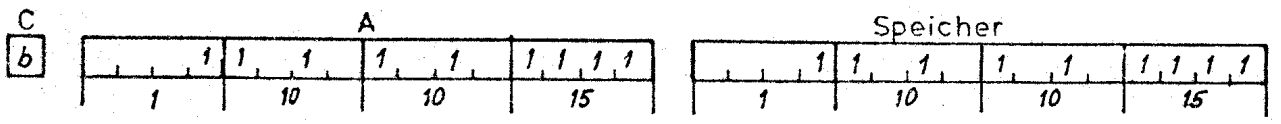


(A) → XWK

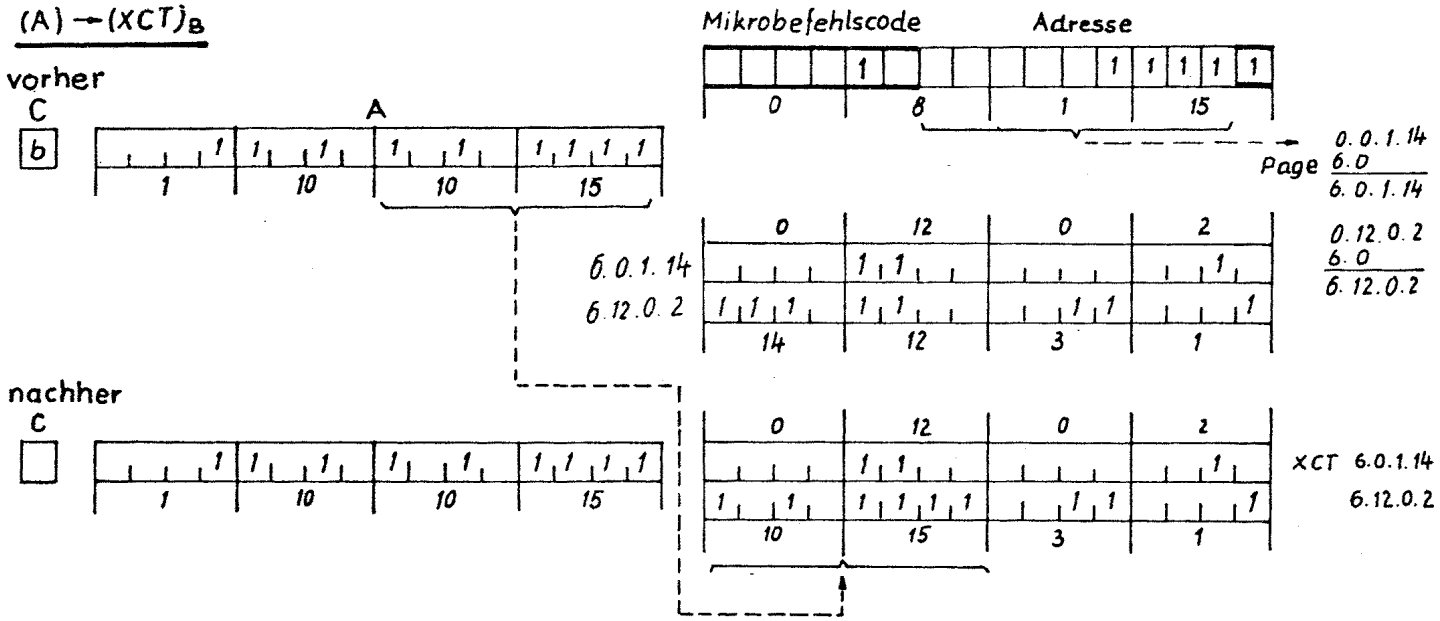
vorher

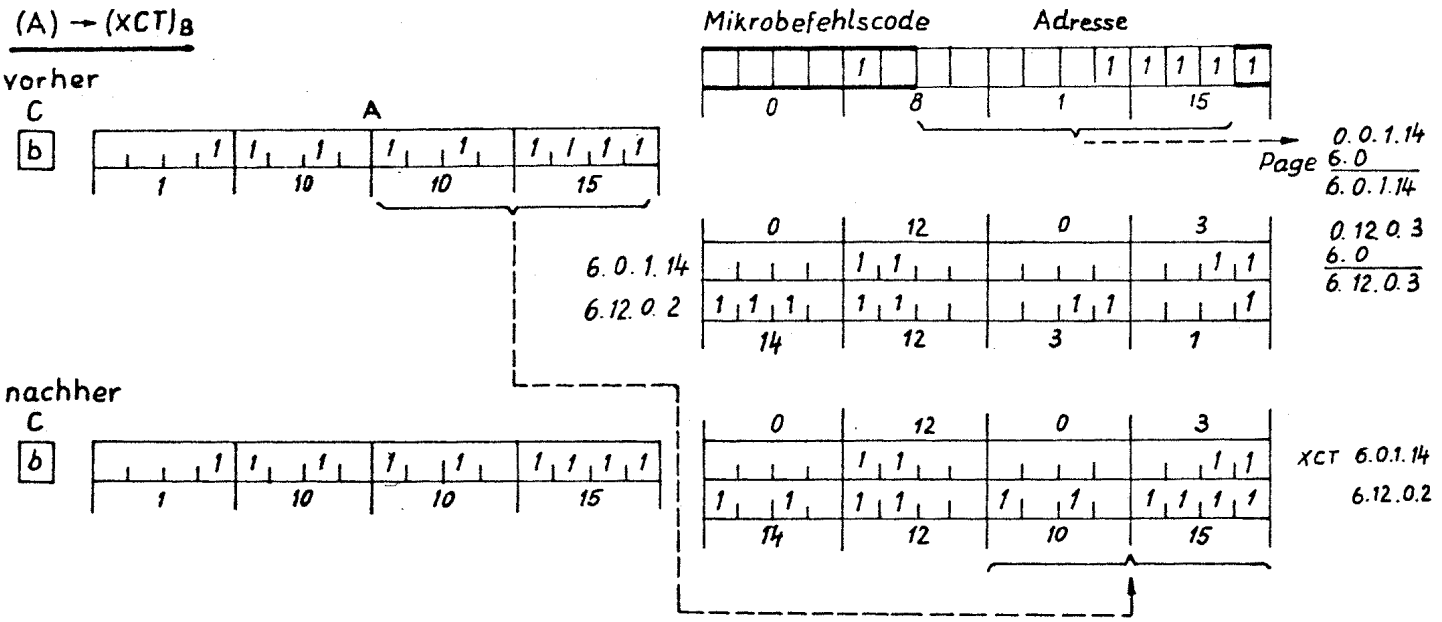


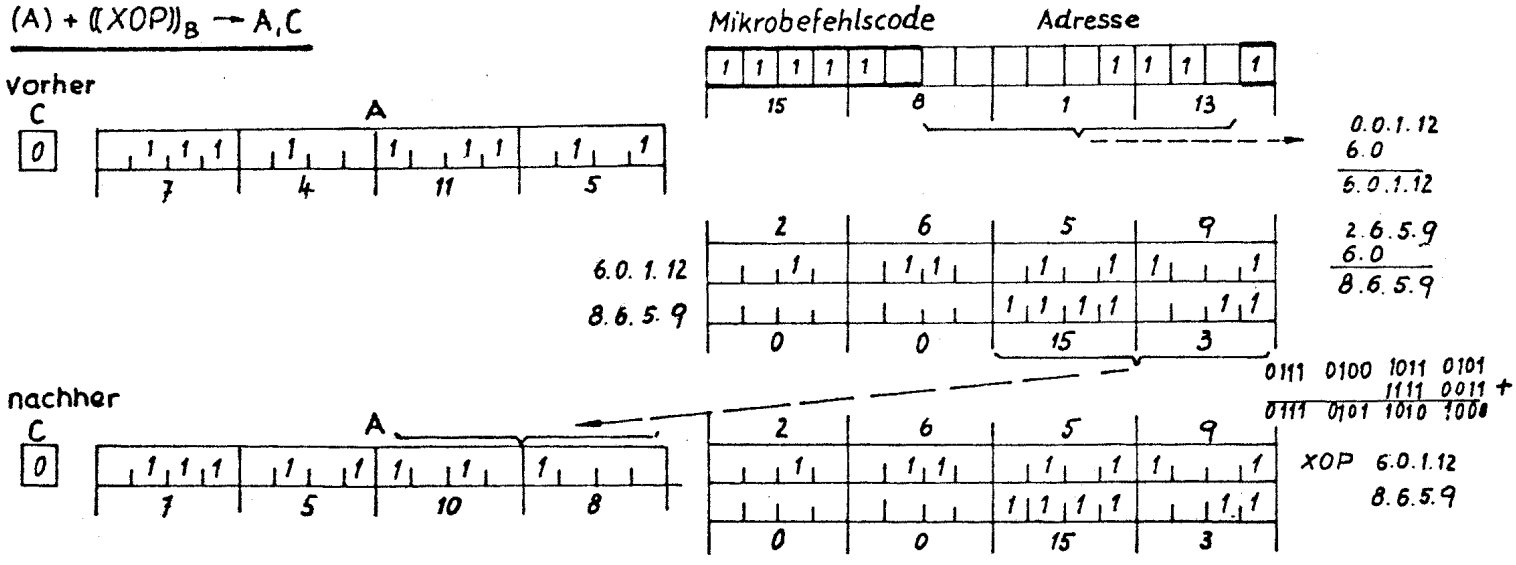
nachher



Scratchpad auf 6.0.0.0 angenommen

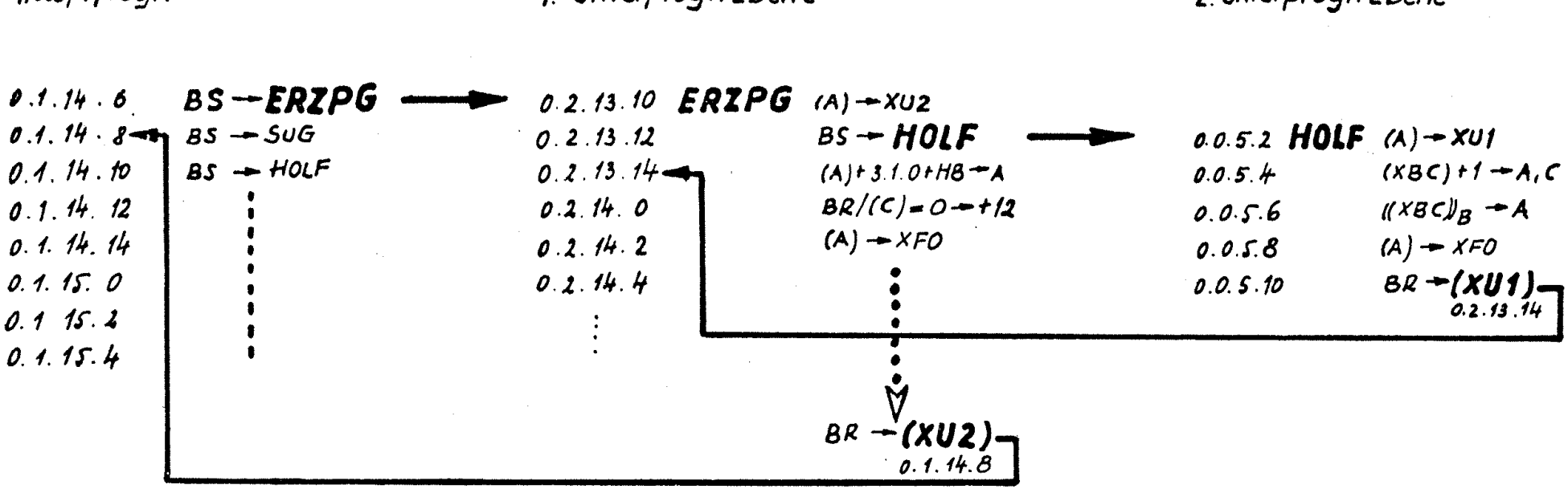






Direkte Unterprogrammsprünge

Beispiel:



Befehl

0.1.14.6 0.2.13.10 BS → ERZPG

0						2			13				10		
0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	0

Mikrobefehlscode — Sprungziel >ERZPG<

	0				1				14				6			
alt	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	0
neu	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	0
	0				2				13				10			

} P-Register Befehlszähler

Vor Ausführung des Sprungs wird der um +2 erhöhte alte Befehlszählerstand (P)+2 → A2...A16, (C) → A1 geladen

0							1				14				8		
0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0

AKKU neu

(A) alt uninteressant

Befehl

0.2.13.12 0.0.5.2 BS → HOLF

0						0			5				2		
0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0

	0				2				13				12			
alt	0	0	0	0	0	0	1	0	1	1	0	1	1	1	0	0
neu	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0
	0				0				5				2			

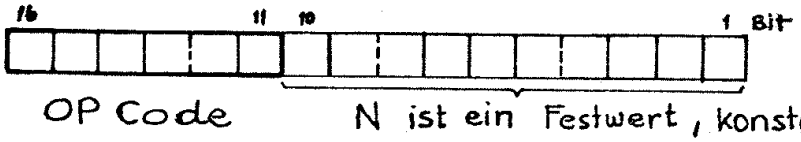
} P-Register Befehlsz.

0						2				13				14			
0	0	0	0	0	0	1	0	1	1	0	1	1	1	1	0		

AKKU neu

Adressierungsmöglichkeiten

Direktoperand, Literalbefehl (Gruppe 5 und 5a)

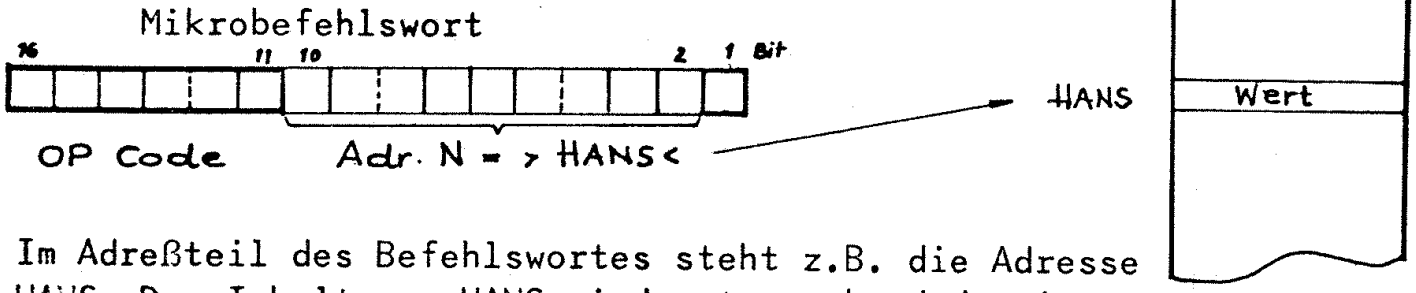


Beim Direktoperand werden Festwerte, Konstante in den Akkumulator gespeichert oder der Akku-Stand wird logisch mit einem Festwert verknüpft.

Das Bit 1 des Befehlswortes ist nicht Bestandteil des OP-Codes, sondern gehört für ungerade Festwerte mit zu N.

Die Sonderbefehle werden auf Seite separat behandelt.

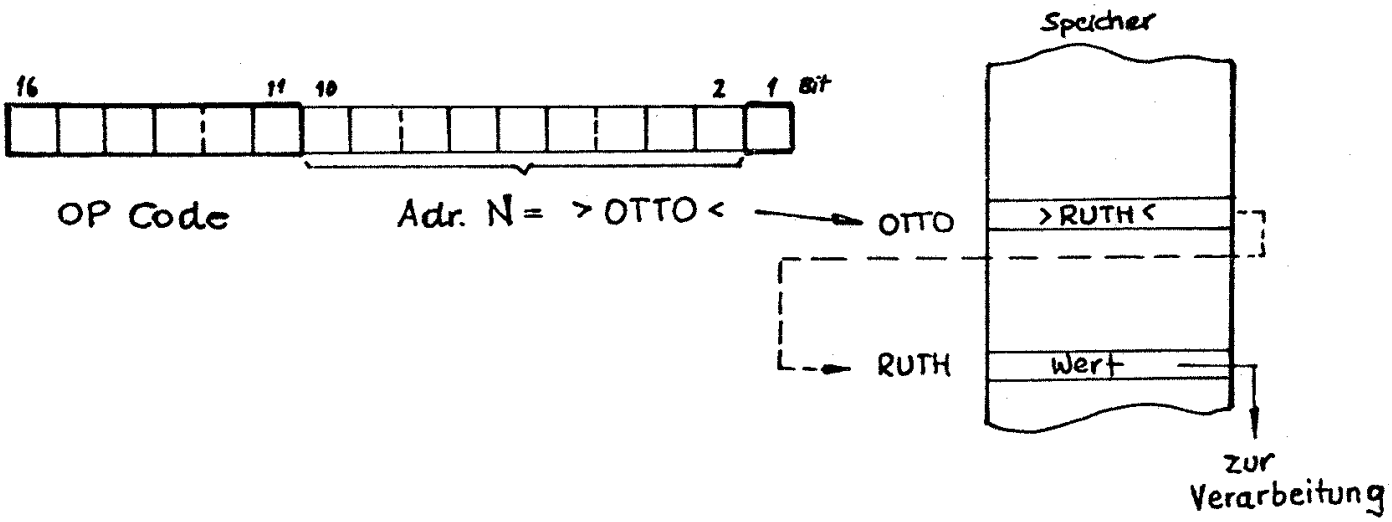
Direktadressierung (Gruppe 1,2,6,7,8,10,11,12 u.14)
(schematisch)



Im Adreßteil des Befehlswortes steht z.B. die Adresse HANS. Der Inhalt von HANS wird entsprechend der im OP-Code stehenden Instruktion verarbeitet.

Es kann nur geradezahlig adressiert werden, da Bit 1 mit zum OP-Code gehört.

Indirekte Adressierung (Gruppe 3,4,5,13 u.14)
(schematisch)



Im Adreßteil N des Befehlswortes steht die Adresse OTTO. Der Inhalt von OTTO ist wiederum eine Adresse RUTH. Der Inhalt von RUTH wird entsprechend im OP-Code stehender Instruktion verarbeitet.

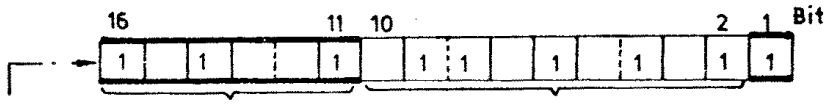
Es kann nur geradzahlig adressiert werden, da Bit 1 mit zum OP-Code gehören.

Beispiel: Direkte Befehlsverarbeitung

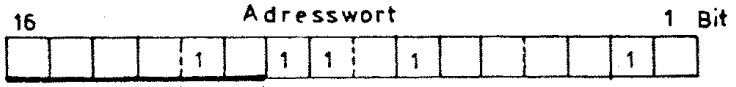
> Befehlszähler <
0.11.10.6

Mikrobefehl
10.5.9.11

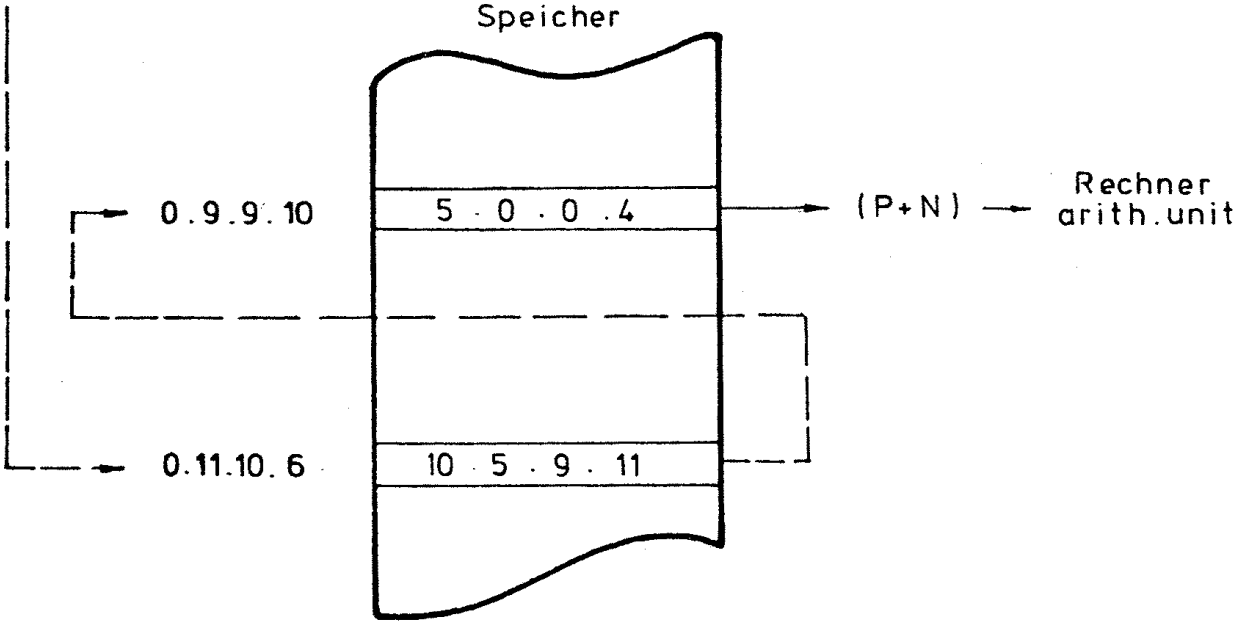
EA (A) ⊕ (P+N) → A
1.1.0.15

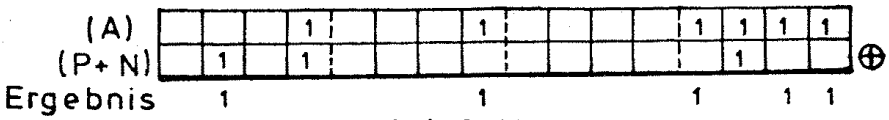


10.4.0.1
0.1.9.10
10.5.9.11



P 0.8.0.0
N 0.1.9.10
Adr. P+N 0.9.9.10
(P+N) 5.0.0.4

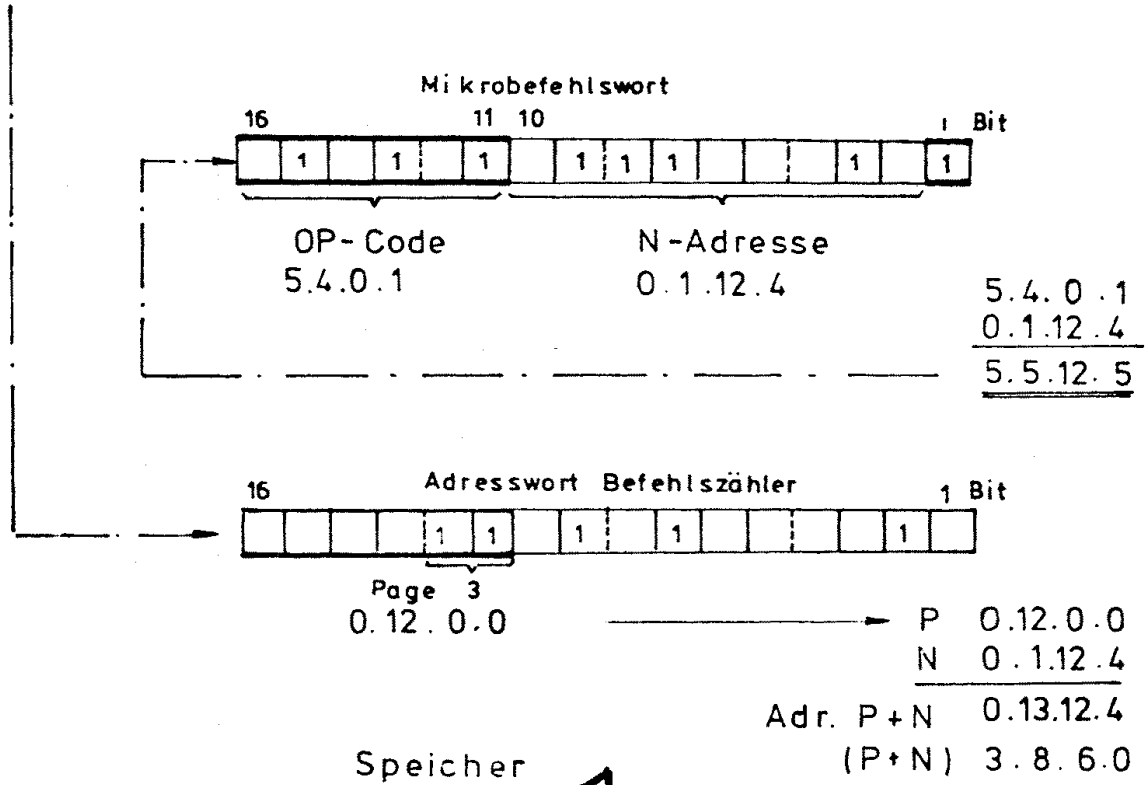


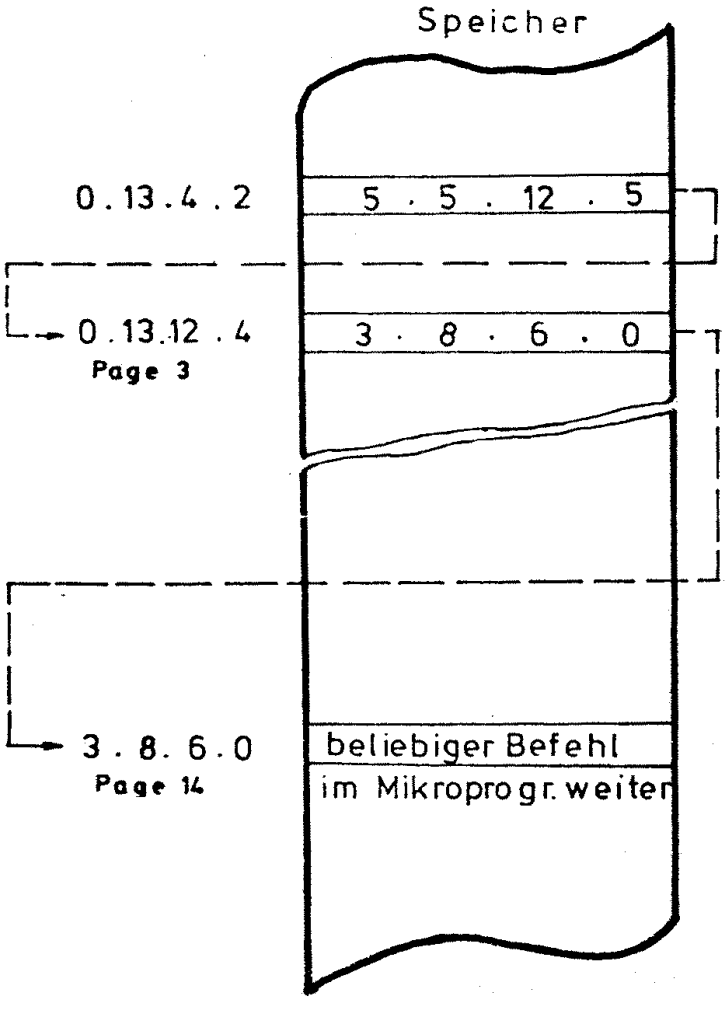


4.1.0.11

Beispiel: Indirekte Befehlsverarbeitung
Indirekter Sprung über mehrere Speicherseiten
pag 3 → page 14

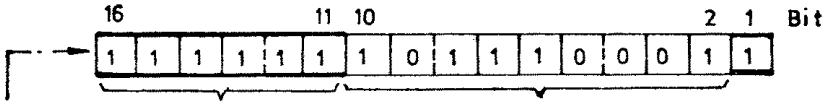
>Befehlszähler< Mikrobef.
0.13.4.2 5.5.12.5 BLI BR/(A1) = 1→(P+N)





Beispiel: Indirekte Befehlsverarbeitung

>Befehlszähler<	Befehl		
0.0.4.6	15.14.14.2	AAI	(A)+ ((S+N)) → A, C 0.0.0.1 15.15.15.15

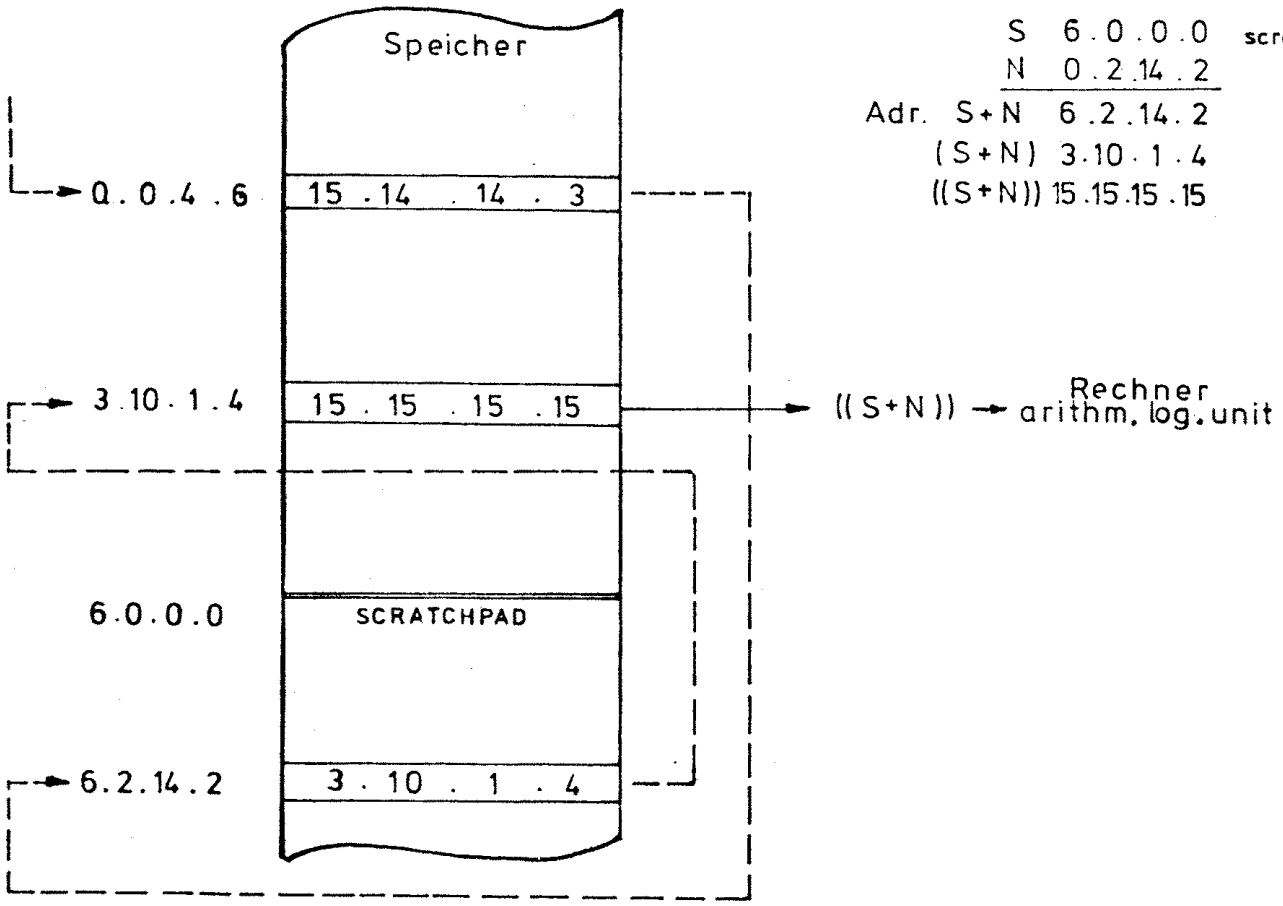


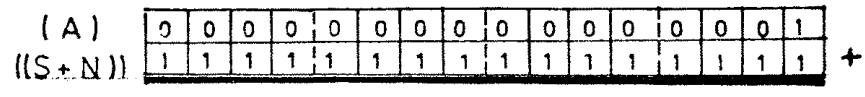
OP Code
Bit 1 u. 11...16
15 .12 .0.1

N-Adresse
0.2.14.2

15.12.0.1
0.2.14.2
15.14.14.3

S 6.0.0.0 scratchpad
N 0.2.14.2
Adr. S+N 6.2.14.2
(S+N) 3.10.1.4
((S+N)) 15.15.15.15





Ergebnis (A)=∅ , (C)=1

Beispiele der Mikroprogrammierung

Um das Verständnis für die Mikroprogrammierung anzuregen werden nachfolgend einige einfache Beispiele aufgezeigt.

Einer Komplement (1-er-Komplement)

Eine Subtraktion wird bei den meisten Computern über das Einer Komplement, als eine spezielle Form der binären Addition, durchgeführt.

Das Einer Komplement einer binären Ziffer läßt sich wie folgt ganz einfach ermitteln in dem wir jeder Ø Ziffer durch eine 1, jede 1 durch eine Ø ersetzen.

Z.B. Dezimal 18 = Binär 0001 · 0010 = hexadez. 1.2
Einer Kompl. 1110 · 1101 = hexadez. 14.13

Steht der Wert in hexadezimaler Form zur Verfügung dann ergänzt man auf 15

Z.B. 0.9.4.2 Zwei und wieviel ist 15 und 13
Vier und wieviel ist 15 und 11
Neun und wieviel ist 15 und 6
Null und wieviel ist 15 und 15
Einer Komplement = 15. 6. 11. 13

Vorgang der Subtraktion mit Hilfe des Einer Komplement

Dezimal 14 - 8 $(14_{10} - 8_{10}) = 6$

Minuend Subtrahend

Binär 14 = 1110 (1110_2) binär 8 = 1000

0111+ ← Kompl.= 0111

Extra Ziffer 1 0101

1

$0110_2 = 6_{10}$

Wir wandeln den Subtrahenden in das Einer Komplement um und addieren die beiden Zahlen.
Ergibt die Summe eine Ziffernstelle mehr als die addierten Ziffernstellen (d.h. wenn die Summenziffer eine zusätzliche Ziffernstelle nach links ergeben hat) so tragen wir diese an die erste Ziffernstelle rechts zurück und addieren sie.
Wenn die Differenz eine negative Menge ergibt, so entsteht keine Extra-Ziffer.

z.B. 6 - 8 = 2

$6_{10} = 0110_2$

$8_{10} = 1000_2$ Einer Komplement = 0111_2 = 0110 / 0111+ / 1101 (keine Extra Zif.)

Rückkomplement 0010 = -2_{10}

Beispiel 1

Vergleich zweier Zahlen

Die Zahlenwerte von X1 u. X2 sollen verglichen und eine Programmverzweigung, entsprechend den unten angegebenen Werten durchgeführt werden.

X1 < X2	HIGH	Eine Hilfszelle MIN 15.15.15.15., alle Bits auf
X1 > X2	LOW	Logisch 1 gesetzt, steht zur Verfügung. MIN ist z.B.
X1 = X2	EQUAL	eine Zelle in der Zeropage

Lösung:

(X1) - 1 → A
(A) ⊕ (MIN) → A
(A) + (X2) → A, C
} Subtraktion X2 - X1 Einer Komplement

BR/(A) = ∅ → EQUAL
BR/(C) = 1 → HIGH
BR → LOW

	X1 = 6, X2 = 4	X1 = 6, X2 = 6	X1 = 5, X2 = 6
(X1)-1 =	0101	0101	0100
	1111 ⊕	1111 ⊕	1111 ⊕ 1er Komplement
	1010	1010	1011
	0100 +	0110 +	0110 +
	1110	C1 0000	C1 0001
	LOW	EQUAL	HIGH

Falls der Inhalt von X1 auch Null sein kann, kommt nachfolgende Subtraktionsart zur Anwendung.

1 → C
(X1) → A
(A) ⊕ (MIN) → A
(A, C) + (X2) → A, C
} Subtraktion X2 - X1

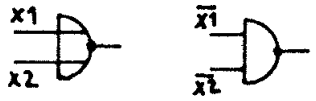
BR/(A) = ∅ → EQUAL
BR/(C) = 1 → HIGH
BR → LOW

* (X1)-1 A ≙ (S+N)-1 A
EQUAL, HIGH, LOW sind z.B. Konnektoren irgenwo in der laufenden Seite.

Beispiel 2

Ohne den Mikrobefehl (A) v (S+N) → A zu verwenden, sollen zwei Werte X1 u. X2 logisch verknüpft werden.

A



$x1 \vee X2 = \overline{\overline{x1} \wedge \overline{x2}}$

Probe: X1=3, X2=5

0011
0101 ∨
0111 Ergebnis

x1	x2	∧	∨	⊕	+
0	0	0	0	0	0
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	1	0	0+Ü

7 Befehle

(x1) → A
(A) ⊕ (MIN) → A
(A) → XWK Hilfs- oder Arbeitszelle z. Zwischenspeich.
(x2) → A
(A) ⊕ (MIN) → A
(A) ∧ (xWK) → A
(A) ⊕ (MIN) → A

$\overline{X1} \rightarrow$ xWK
$\overline{X2}$
$\overline{X2} \wedge \overline{X1}$ umkehren (negieren)

0011
1111 ⊕
1100 → xWK

0101
1111 ⊕
1010
1100 ∧ (xWK)
1000
1111 ⊕
0111 Ergebnis

B

x1	x2	∧	∨	⊕	+
0	0	0	0	0	0
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	1	0	0+Ü

(x1) → A
(A) ∧ (x2) → A
(A) → XWK
(x1) → A
(A) ⊕ (x2) → A
(A) + (XWK) → A

6 Befehle

0011
0101 ∧
0001 → xWK

0011
0101 ⊕
0110
0001 + (xWK)
0111

C

x1	x2	∧	⊕	+	∨
0	0	0	0	0	0
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	0	0+Ü	1

(x1) → A
(A) ∧ (x2) → A
(A) ⊕ (x1) → A
(A) + (x2) → A

4 Befehle

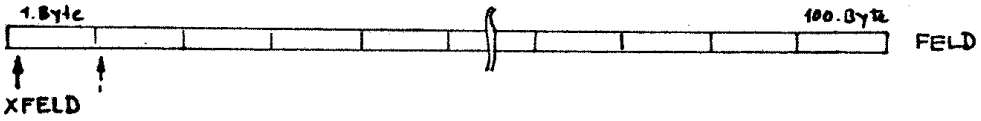
Beste Lösungsart

0011
0101 ∧
0001
0011 ⊕
0010
0101 +
0111

Beispiel 3

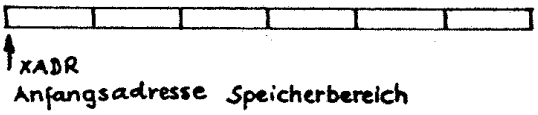
Aus Feld byteweise Werte über die IO-Karte an peripheres Gerät ausgeben.
IO-Karte 6, Zeile 0 XANZAH, Zelle mit 100 laden (Anzahl 100 Bytes).

```
        ϕ → A                       Löschen Zähler
        (A) → XZAEHL
        BS → UHR
ANF     ((XFELD))B → A              Wert 1. Byte ausgeben (2., 3., ... 100.)
        IOX (6,0) A
        (XFELD) +1 → XFELD          nächstes Byte adressieren
        (XZAEHL) +1 → XZAEHL        Zähler um 1 erhöhen.
        (A) ⊕ (XANZAH) → A          vergleichen mit der Anz. Bytes.
        BR/(A) ≠ ϕ → ANF
WEIT    :
```



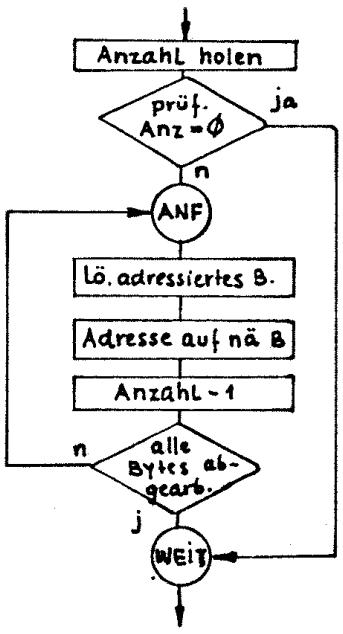
Beispiel 4

Byteweises Löschen eines Speicherbereiches.
Ist die Anzahl der zu löschenden Bytes Null, ist auf Weit (Weiter) zu verzweigen.



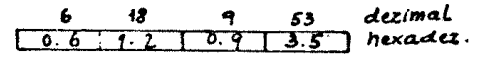
XLA Anzahl der zu löschenden Bytes

```
        (XLA) → A                   Prüfen ob die Anz. = ϕ
        BR/(A) = ϕ → WEIT
ANF     ϕ → A
        (A) → ((XADR))B             lö 1. Byte (2., 3., ... n tes)
        (XADR) +1 → A, XADR         Byte adresse erhöhen
        (XLA) -1 → A, XLA           Anzahl -1
        BR/(A) ≠ ϕ → ANF
WEIT    :
```



Beispiel 5

Suche aus einer Kette von Werten den Maximalwert und speichere diesen in die Zelle XMAX.
XANZ, Anzahl der zu prüfenden Bytes. Ist diese Zelle zu Beginn Null, wird auf Fehler verzweigt. (XANZ) = 4

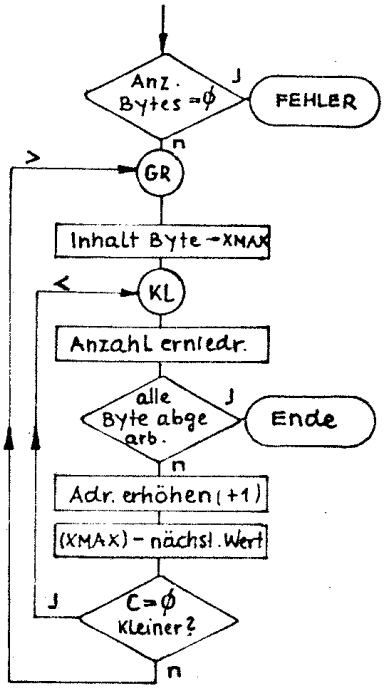


XADR

```
        (XANZ) → A
        BR/(A) = 0 → FEHL
> GR    ((XADR))B → A
        (A) → XMAX
< KL    (XANZ) - 1 → A, XANZ
        BR/(A) = ϕ → ENDE (WEIT)
        (XADR) + 1 → A, XADR
        (XMAX) → A
        (A) ⊕ (MIN) → A
        (A) + ((XADR))B → A, C
        BR/(C) = ϕ → KL
        BR → GR
```

XMAX	XANZ
	4
0.6	
1.2	3
3.5	2
	1
	ϕ
	ENDE

```
0000 0110
1111 1001 Kompl.
0001 0010 +
C1 0000 1011 GR

0001 0010
1110 1101 Kompl.
0000 1001 +
1111 0110 KL

0001 0010
1110 1101 Kompl.
0011 0101 +
C1 0010 0110 GR
```



Beispiel 6

Unterprogramm: Code Wandlung über Tabelle

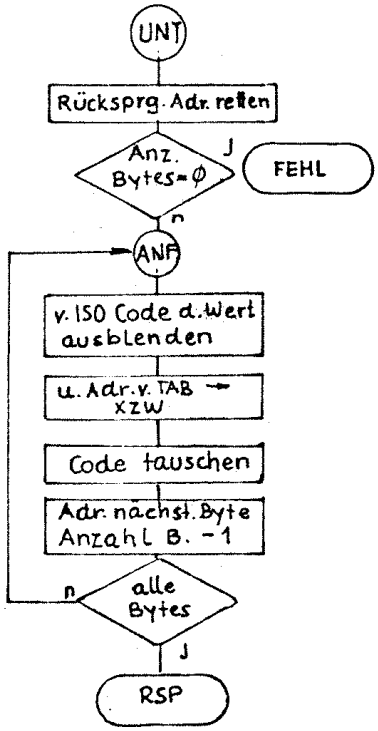
dezimal	ISO 7 Bit	Fernschr. Code
0	0.0 =	1.6
1	0.1 =	1.7
2	0.2 =	1.3
3	0.3 =	0.1
4	0.4 =	0.10
5	0.5 =	1.0
6	0.6 =	1.5
7	0.7 =	0.7
8	0.8 =	0.6
9	0.9 =	1.8

```
UNT (A) → XU
    (XANZ) → A
    BR/(A) = ϕ → FEHL
ANF ((XADR))B → A
    (A) ∧ 15 → A
    (A) + (TAB) → A
    (A) → XZW
    ((XZW))B → A
    (A) → (XADR)B        } Wandlung
    (XADR) + 1 → A, XADR
    (XANZ) - 1 → A, XANZ
    BR/(A) ≠ ϕ → ANF
    BR → (XU)
```

TAB

1000	0 0.0	1 1.7		
1002	2 1.3	3 0.1		
1004	4 0.10	5 1.0		
1006	6 1.5	7 0.7		
1008	8 0.6	9 1.8		

z.Beisp. ISO Code 3.6
6 ausblenden u.
TAB Anf.Adr. = 1006 → XZW
((XZW)) → 1.5 → XADR
6
(steht 3.6 im Byte 1.5)



Beispiel 7

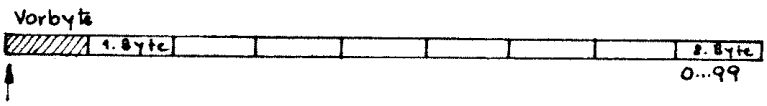
Liegt der Byte Wert im Bereich von 1 - 9 verzweige nach KON 1
10 -19 " " KON 2
20 -29 " " KON 3
Ist der Wert 0 oder größer 29 verzweige nach Fehler.

```
((XADR))B → A
BR/(A) = ϕ → FEHL
(A) ⊕ (MIN) → A
(A) + 10 → A,C
BR/(C) = 1 → KON1
(A) + 10 → A,C
BR/(C) = 1 → KON2
(A) + 10 → A,C
BR/(C) = 1 → KON3
BR → FEHL
```

1	9	10	19 (1.3)	20 (1.4)	29 (1.13)	30 (1.14)
15.14	15.6	15.5	14.12	14.11	14.2	14.1
10 +	10 +	10+	10+	10+	10 +	10 +
C1 0.8	C1 0.0	15.15	15.6	15.5	14.12	14.11
KON1	KON1	10	10+	10+	10+	10+
		C1 0.9	C1. 0.0	15.15	15.6	15.5
		KON2	KON2	10+	10+	10+
				C1 0.9	C1 0.0	15.5
				KON3	KON3	FEHL

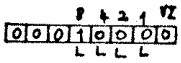
Beispiel 8

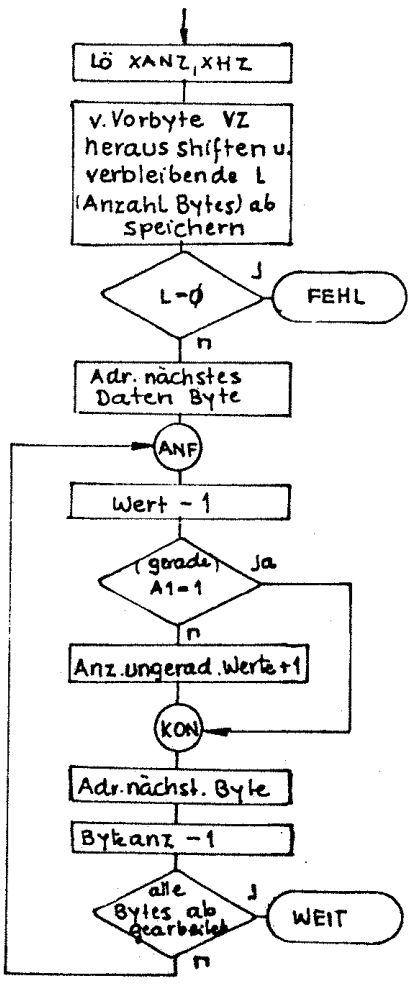
Datenfeld mit Zahlen.
Die Anzahl der ungeraden Werte in XHZ (Hilfszelle) abspeichern.




```
ϕ → A
(A) → XANZ
(A) → XHZ
((XADR))B → A
(A) > 1 → A        Vorzeich. herausshiften
BR/(A) = ϕ → FEHL
(A) → XANZ
(XADR) + 1 → A, XADR
ANF ((XADR))B - 1 → A
BR/(A1) = 1 → KON
(XHZ) + 1 → A, XHZ
KON (XADR) + 1 → A, XADR
(XANZ) - 1 → A, XANZ
BR/(A) = ϕ → WEIT
BR → ANF
```

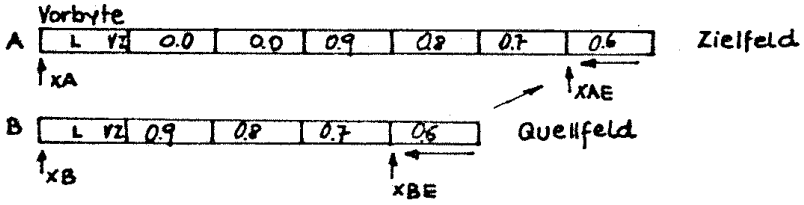
16		17	
1.0		1.1	
- 1		- 1	
0.15	A1 = 1	1.0	A1 = ϕ
	KON	+1 → XHZ	

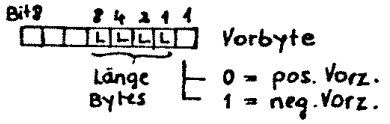




Beispiel 9

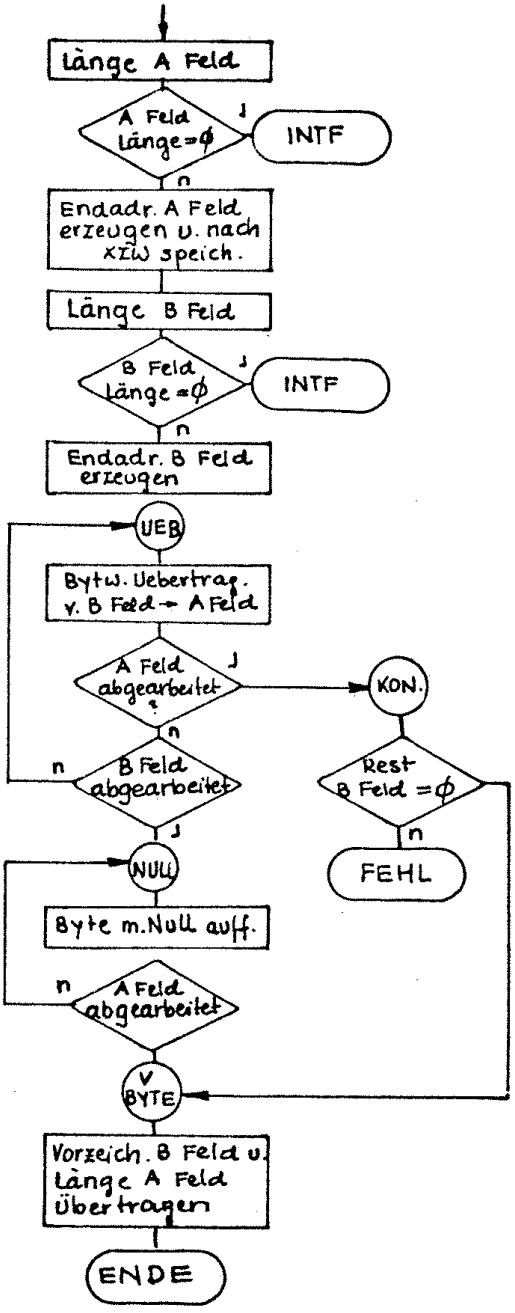
Übertrage den Inhalt des Dezimalfeldes B rechtsbündig in das Dezimalfeld A.
Ist Feld B größer als Feld A wird INTF angezeigt.
Ist Feld A größer sind die rechstlichen Bytes mit Nullen aufzufüllen.

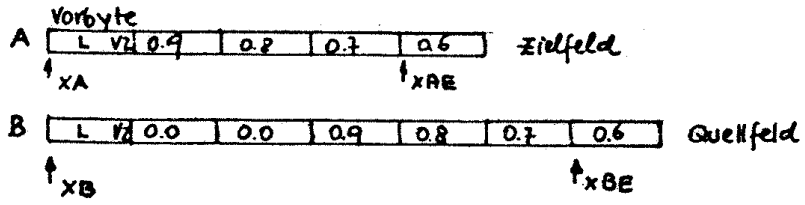





```
     ((XA))B → A
     (A) > 1 → A
     (A) → XZW        Länge A Feld
     BR/(A) = ϕ → INTF
     (A) + (XA) → A
     (A) → XAE
     ((XB))B → A
     (A) > 1 → A
     BR/(A) = ϕ → INTF
     (A) + (XB) → A
     (A) → XBE        Endadr. B Feld

KONTR (XBE) - 1 → A, XBE
      (A) ⊕ (XB) → A
      BR/(A) = ϕ → VBYTE
      ((XBE))B → A
      BR/(A) = ϕ → KONTR
      BR → FEHL

UEB  ((XBE))B → A          Übertrag
     (A) → (XAE)B
     (XAE) - 1 → A, XAE
     (A) ⊕ (XA) → A
     BR/(A) = ϕ → KONTR    } A Feld abgearbeitet
     (XBE) - 1 → A, XBE
     (A) ⊕ (XB) → A
     BR/(A) ≠ 0 → UEB      } B Feld abgearbeitet.

NULL ϕ → A                 mit ϕ auffüllen
     (A) → (XAE)B          B-Feld fertig
                           A-F. noch nicht
     (XAE) - 1 → A, XAE
     (A) ⊕ (XA) → A
     BR/(A) ≠ 0 → NULL

VBYTE ((XB))B → A
      (A) ∧ 0.1 → A
      (A) + (XZW) → A
      (A) → (XA)B
```

(Vorbyte A) = 0.12
(Vorbyte B) = 0.9

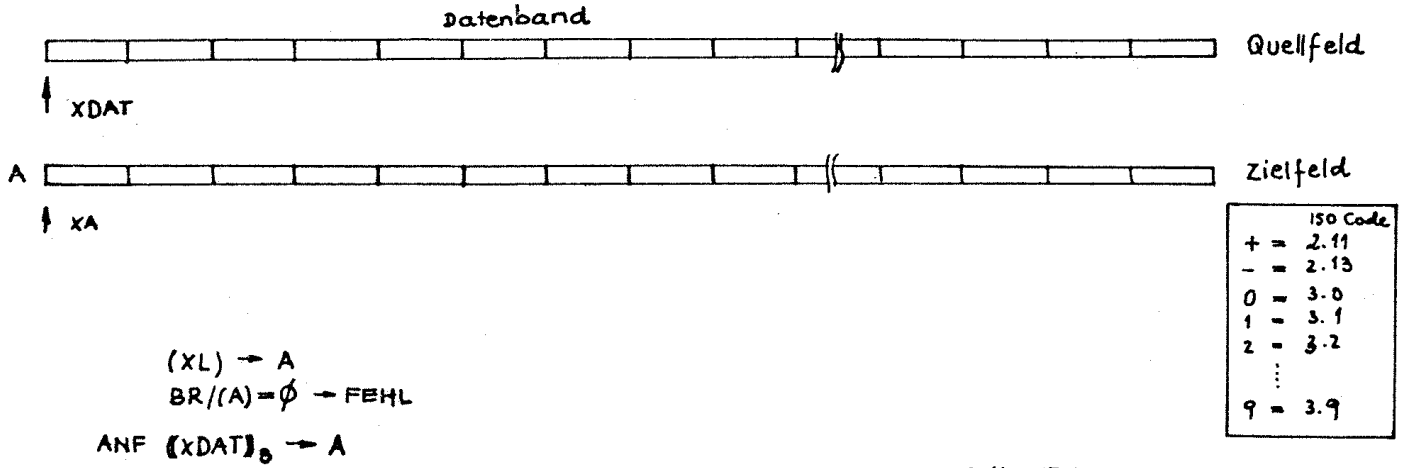




(Vorbyte A) = 0.9
(Vorbyte B) = 0.12

Beispiel 10

Aus einem ISO Code Datenband, Quellfeld,
sind in das Zielfeld A, nur die Ziffern u. Vorzeichen zu
übertragen. XL = Anzahl der Zeichen.



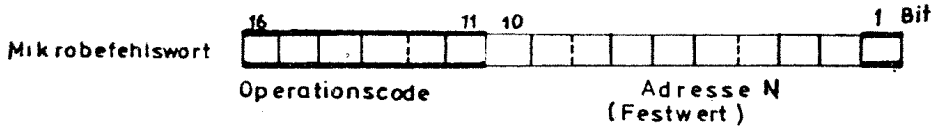

```
        (XL) → A
        BR/(A)=ϕ → FEHL
ANF     ((XDAT))B → A
        (A)+3.13.5+HB → A,C      A - 2.11  VZ+
        BR/(A)=ϕ → UEB
        (A)+3.15.14+HB → A,C     A - 0.2   VZ-
        BR/(A)=ϕ → UEB
        (A)+3.15.13+HB → A,C     A - 0.3
        BR/(C)=ϕ → NAZ           nächstes Zeich.
        (A)+3.15.6+HB → A,C      A - 0.10
        BR/(C)=1 → NAZ
UEB     ((XDAT))B → A            Zeichen übertragen
        (A) → (XA)B
NAZ     (XA)+1 → A, XA           Adr. der Felder
        (XDAT)+1 → A, XDAT       erhöhen
        (XL)-1 → A, XL
        BR/(A)≠ϕ → ANF
```

```
   2.11  VZ+
+    2
  2.13  VZ-
+    3
  3.0
+ 10     } Ziffern
  3.10
```

VZ+	VZ-	Kein VZ	Ziffern 0...9		Buchstaben etc.	
2.11	2.13	2.15	3.0	3.9	3.10	
15.15.13.5+	15.15.13.5	15.15.13.5	15.15.13.5	15.15.13.5	15.15.13.5	- 2.11
C1 0.0.0.0	C1 0.0.0.2	C1 0.0.0.4	C1 0.0.0.5	C1 0.0.0.14	C1 0.0.0.15	
	15.15.15.14	15.15.15.14	15.15.15.14	15.15.15.14	15.15.15.14	- 0,2
	C1 0.0.0.0	C1 0.0.0.2	C1 0.0.0.3	C1 0.0.0.12	C1 0.0.0.13	
		15.15.15.13	15.15.15.13	15.15.15.13	15.15.15.13	- 0,3
		15.15.15.15	C1 0.0.0.0	C1 0.0.0.9	C1 0.0.0.10	
			15.15.15.6	15.15.15.6	15.15.15.6	- 0,10
			15.15.15.6	15.15.15.15	C1 0.0.0.0	

bei Ziffen 0...9
C=ϕ

Codierung



Zerlegen der Befehlszähler Adr. z.B. 1.7.10.2
(siehe Beispiel Speicheraufbau)

1 . 7 . 10 . 2 hexa
0001.0111.1010.0010 binär
Page 5 = 1.4.0.0
Adresse 0 . 3 . 10 . 2 hexa

Page 5 1.4.0.0 – 1.7.15.14

Konnektor suchen

>Befehlsz.< Mikrobefehl
0.6.9.14 3.2.4.13 BR/(A) ≠ ∅ → UEB1
zerlegen des Mikrobefehlswort in
Operationscode u. Adresse

3 . 2 . 4 . 13 Befehl
0011.0010.0100.1101
OP code 3 . 0 . 0 . 1
Adresse 0 . 2 . 4 . 12

0 . 6 . 9 . 14 Adresse
0000.0110.1001.1110
Page 0 . 4 . 0 . 0
Adr. 0 . 2 . 4 . 12
UEB1 0 . 6 . 4 . 12

0.	Page Seite	0.0.0.0 – 0.3.15.14 zero p.
1.	–"–	0.4.0.0 – 0.7.15.14
2.	–"–	0.8.0.0 – 0.11.15.14
3.	–"–	0.12.0.0 – 0.15.15.14
4.	–"–	1.0.0.0 – 1.3.15.14
5.	–"–	1.4.0.0 – 1.7.15.14
6.	–"–	1.8.0.0 – 1.11.15.14
7.	–"–	1.12.0.0 – 1.15.15.14
8.	–"–	2.0.0.0 – 2.3.15.14
9.	–"–	2.4.0.0 – 2.7.15.14
10.	–"–	2.8.0.0 – 2.11.15.14
11.	–"–	2.12.0.0 – 2.15.15.14
12.	–"–	3.0.0.0 – 3.3.15.14
13.	–"–	3.4.0.0 – 3.7.15.14
14.	–"–	3.8.0.0 – 3.11.15.14
15.	–"–	3.12.0.0 – 3.15.15.14
16.	–"–	4.0.0.0 – 4.3.15.14
17.	–"–	4.4.0.0 – 4.7.15.14
18.	–"–	4.8.0.0 – 4.11.15.14
19.	–"–	4.12.0.0 – 4.15.15.14
20.	–"–	5.0.0.0 – 5.3.15.14
21.	–"–	5.4.0.0 – 5.7.15.14
22.	–"–	5.8.0.0 – 5.11.15.14
23.	Seite Page	5.12.0.0 – 5.15.15.14
		6.0.0.0 – 6.3.15.14 scratchpad p.

Das Mikroprogrammierbeispiel Nr. 5 wird zur Übung manuell codiert

Erste Befehlsadresse 1.7.0.0, >WEIT< = 1.15.0.6,
>FEHL< = 1.7.2.4
>XANZ< = 6.1.14.2, >XADR< = 6.1.14.4 >XMAX< = 6.1.14.6

1.7.0.0		(XANZ) → A	LAX	8.13.14.2	←	8.12.0.0 0.1.14.2	
1.7.0.2		BR/(A) = ϕ → FEHL	BE	2.7.2.5	←	2.0.0.1 1.7.2.4	
1.7.0.4	GR	((XADR))$_B$ → A	LAB	8.9.14.5	←	8.8.0.1 0.1.14.4	
1.7.0.6		(A) → XMAX	SMX	0.13.14.6	←	0.12.0.0 0.1.14.6	
1.7.0.8	KL	(XANZ) - 1 → XANZ	DMX	5.13.14.2	←	5.12.0.0 0.1.14.2	
1.7.0.10		BR/(A)=0 → (AWEIT)	BRI	2.7.1.9	←	2.4.0.1 0.3.1.8	* Seitenwechsel
1.7.0.12		(XADR)+1 → A, XADR	IMX	4.13.14.4	←	4.12.0.0 0.1.14.4	
1.7.0.14		(XMAX) → A	LAX	8.13.14.6	←	8.12.0.0 0.1.14.6	
1.7.1.0		(A)⊕(MIN) → A	EA	10.7.1.11	←	10.4.0.1 0.3.1.10	
1.7.1.2		(A)+((XADR))$_B$ → A, C	AAB	15.9.14.5	←	15.8.0.1 0.1.14.4	
1.7.1.4		BR/(C) = ϕ → KL	BZ	6.3.0.8	←	6.0.0.1 0.3.0.8	**
1.7.1.6		BR → GR	BR	1.3.0.5	←	1.0.0.1 0.3.0.4	
1.7.1.8		(AWEIT)		1.15.0.6			
1.7.1.10	MIN	15.15.15.15					

* 1.7.1.8
0001.0111.0001.1000
0.3.1.8 Adr.

** 1.7.0.8
0001.0111.0000.1000
0.3.0.8

Mikroprogr. Beisp. Nr. 5

Programm: Maximalwert suchen

für manuelle Codierung u. Eingabe

Programmierer: Huber Fritz

Datum: 3.4.74

hohe Adresse: 17

Blatt Nr. 1

Ausführungsstand: ∅

Mikro

niedrige Adresse	symbolische Adresse	Sprung-richtung	mathematische Sprache (Befehl)	hex. Codierung
∅ 0	ANF		(XANZ) → A	8 13 14 2
2			BR/(A) = ∅ → FEHL	2 1 2 5
4	GR	→	((XADR))B →	8 9 14 5
6			(A) → XMAX	∅ 13 14 6
8	KL	→	(XANZ) −1 → A, XANZ	5 13 14 2
10			BR/(A) = ∅ → (AWEIT)	2 7 1 9
12			(XADR) + 1 → A, XADR	4 13 14 4
14			(XMAX) → A	8 13 14 6
1 0			(A) ⊕ (MIN) → A	10 7 1 11
2			(A) + ((XADR))B → A, C	15 9 14 5
4			BR/(C) = ∅ → KL	6 3 ∅ 8
6			BR → GR	1 3 ∅ 5
8			(AWEIT)	1 15 ∅ 6
10	MIN		15. 15. 15. 15	
12				
14				
0				
2				
4				
6				
8				
10				
12				
14				
0				
2				
4				
6				
8				
10				
12				
14				

Mikroprammbeispiel Nr. 5 für Assemblierung

HOHE ADRESSE

PROGRAMM: Maximalwert suchen

PROGRAMMIERER: HUBER.F

AUSFÜHRUNGSSTAND: 0

BLATT NR.: 1

DATUM: 3.4.74

NIEDRIGE ADRESSE	SS	SYMB NAME	TT	OP-CODE	ADRESSTEIL ODER KOMMENTAR	LFD. NR.
	S1				MAXIMALWERT SUCHEN	
	*	XANZ			1.14.2	
	*	XADR			1.14.4	
	*	WEIT			1.15.0.6	
	*	ANF			1.7.0.0	
	*					
	*	ANF		LAX	XANZ ANF (XANZ) := A	
				BE	FEHL BR/(A)=0 := FEHL	
		GR		LAB	XADR GR ((XADR)).B := A	
				SMX	XMAX (A) := XMAX	
		KL		DMX	XANZ KL (XANZ)-1 := XANZ	
				BRI	AWEIT BR/(A)=0 := (AWEIT)	
				IMX	XADR (XADR)+1 := A, XADR	
				LAX	XMAX (XMAX) := A	
				EA	MIN (A)*(MIN) := A	
				AAB	XADR (A)+((XADR)).B := A, C	
				BZ	KL BR/(C)=0 := KL	
				BR	GR BR := GR	
		AWEIT			WEIT AWEIT WEIT	
		MIN			15.15.15.15	
		FEHL		BR	FEHL BR AUF DER STELLE	
	**					

Anhang

BITS 8 765		BITS 4 321		BITS 8 765		BITS 4 321	
HEX	DEZIMAL	HEX	DEZIMAL	HEX	DEZIMAL	HEX	DEZIMAL
00	0	00	0	00	0	00	0
01	4.096	01	256	01	16	01	1
02	8.192	02	512	02	32	02	2
03	12.288	03	768	03	48	03	3
04	16.384	04	1.024	04	64	04	4
05	20.480	05	1.280	05	80	05	5
06	24.576	06	1.536	06	96	06	6
07	28.672	07	1.792	07	112	07	7
08	32.768	08	2.048	08	128	08	8
09	36.864	09	2.304	09	144	09	9
10	40.960	10	2.560	10	160	10	10
11	45.056	11	2.816	11	176	11	11
12	49.152	12	3.072	12	192	12	12
13	53.248	13	3.328	13	208	13	13
14	57.344	14	3.584	14	224	14	14
15	61.440	15	3.840	15	240	15	15
	16^3		16^2		16^1		16^0

Umrechnung Binär → Hexadezimal → Dezimal

0011 0110 1111 = Binärzahl
03.06.15 = Hexadezimalzahl
879 = Dezimalzahl

BYTE		BYTE		
0000	0011	0110	1111	= Binär
00	03	06	15	= Hexa
0 + 768 +		96 + 15		= Dezimal = 879

Umrechnung Dezimal → Binär → Hexadezimal

879 : 2 = 439,R	1
439 : 2 = 219,R	1
219 : 2 = 109,R	1
109 : 2 = 54,R	1
54 : 2 = 27,R	0
27 : 2 = 13,R	1
13 : 2 = 6,R	1
6 : 2 = 3,R	0
3 : 2 = 1,R	1
1 : 2 = 0,R	1
0 : 2 = 0,R	0

(oder) 879 suche davon aus Tabelle den nächsten kleineren Wert.

- 768 → 03
- 111
- 96 → 06
- 15 → 15

03 06 15 Hexadezimal

0000 0011 0110 1111 Binär
00 03 06 15 Hexadezimal

Logische Operationen

Logisches Und

				1	1	0	1	13
				0	1	1	0	6 ∧
				0	1	0	0	4

Nur '1' wenn beide 1

Logisches Oder

				1	0	1	0	10
				1	1	0	0	12 ∨
				1	1	1	0	14

Nur '0' wenn beide 0

Exclusives Oder

				1	1	0	0	12
				1	0	1	0	10 ⊕
				0	1	1	0	6

Nur '1' wenn ungleich

Right shift

				1	1	1	0	14
				0	1	1	1	7 RS

Right shift um 1 Bit (= wie : 2)

Addition von Binärzahlen

				1	0	1	0	10
				1	1	0	0	12 +
			1	0	1	1	0	22

1 + 1 = 0 u. Übertrag

X	Y	Z			
		∧	∨	⊕	+
0	0	0	0	0	0
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	1	0	0+Ü

Wahrheitstabelle
(Truth table)

Begriffe u. Symbole

Logische Zustände

0 = Ø = Zustand Null zero
1 = (L) = Zustand Eins one

Potential Zustände

L = low Zustand mit negativem Potential
H = high Zustand mit positivem Potential
-3V ist positiver als -10V

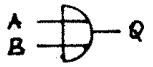
Positive u. Negative Logik

Positive Logik	Negative Logik
0 ≙ low	0 ≙ high
1 ≙ high	1 ≙ low

Gatter

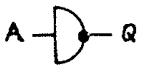
UND

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



ODER

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



NICHT

A	Q
0	1
1	0

INVERTER

NICHT

NAND

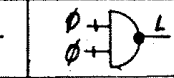
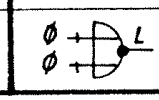
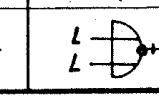
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

NOR

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

EXCLUSIV ODER

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

	UND	ODER
NAND		
NOR		

TRIUMPH | ADLER

– 107 HILTA
ALPHA – TASTATUR

Blatt 1a

Taste Nr	Symbol	Code	Tast. Zuorang.
1	1 !	3.1 2.1	x4 y0
2	Q	5.1 5.1	x5 y0
3	A	4.1 4.1	x6 y0
4	2 "	3.2 2.2	x7 y0
5	Y	5.9 5.9	x8 y0
6	W	5.7 5.7	x9 y0
7	S	5.3 5.3	x10 y0
8	3 #	3.3 2.3	x11 y0
9	X	5.8 5.8	x12 y0
10	E	4.5 4.5	x13 y0
11	D	4.4 4.4	x14 y0
12	4 ¤	3.4 2.4	x15 y0
13	C	4.3 4.3	x0 y1
14	R	5.2 5.2	x1 y1
15	F	4.6 4.6	x2 y1
16	5 %	3.5 2.5	x3 y1
17	V	5.6 5.6	x4 y1
18	T	5.4 5.4	x5 y1
19	G	4.7 4.7	x6 y1
20	6 &	3.6 2.6	x7 y1
21	B	4.2 4.2	x8 y1
22	Z	5.10 5.10	x9 y1
23	H	4.8 4.8	x10 y1
24	7 '	3.7 2.7	x12 y1
25	N	4.14 4.14	x13 y1

Taste Nr	Symbol	Code	Tast. Zuordnung
26	U	5.5 5.5	x14 y1
27	J	4.10 4.10	x15 y1
28	8 (	3.8 2.8	x0 y2
29	M	4.13 4.13	x1 y2
30	I	4.9 4.9	x2 y2
31	K	4.11 4.11	x3 y2
32	9)	3.9 2.9	x4 y2
33	, <	2.12 3.12	x5 y2
34	O	4.15 4.15	x6 y2
35	L	4.12 4.12	x7 y2
36	0 0	3.0 3.0	x8 y2
37	. >	2.14 3.14	x9 y2
38	P	5.0 5.0	x10 y2
39	; +	3.11 2.11	x11 y2
40	- =	2.13 3.13	x12 y2
41	/ ?	2.15 3.15	x13 y2
42	@	4.0 4.0	x14 y2
43	: *	3.10 2.10	x16 y2
44	^	5.14 5.14	x0 y3
45	[	5.11 5.11	x1 y3
46	]	5.13 5.13	x2 y3
47	\	5.12 5.12	x1 -y0
48	—	5.15 5.15	x2 y0
49	SHIFT		SH
50	(4)	1.3 1.7	x0 y0

Ohne Kleinschreibung

STAND: FERTIGUNGSFREIGABE

Erstausgabe 28. Sept. 1972 BM Neuherausgabe 24 Okt. 1972 BM	Ers.	Name: DORRFUSS	Abt.: EE-3	Es folgt Blatt 2

ALPHA - TASTATUR

Taste Nr.	Symbol	Code	Tast Zuordnung
52	②	1. 1 1. 5	x3 y0
56	SPACE	2. 0 2. 0	x11 y1
59	SHIFT		$\overline{SH}$
61	①	1. 0 1. 4	x4 y3
63	③	1. 2 1. 6	x3 y3
64	Ⓡ		$\overline{RP}$

TRIUMPH | ADLER

ZEHNER – TASTATUR

Taste Nr	Symbol	Code	Tast. Zuordnung
70	1	11. 1	x8 y3
71	2	11. 2	x12 y3
72	3	11. 3	x0 y4
73	4	11. 4	x9 y3
74	5	11. 5	x13 y3
75	6	11. 6	x1 y4
76	7	11. 7	x10 y3
77	8	11. 8	x14 y3
78	9	11. 9	x2 y4
80	0	11. 0	x7 y3
81	00	11. 10	x11 y3
82	,	10. 12	x15 y3
84	—	10. 13	x5 y3
85	C	15. 15	x6 y3

TRIUMPH | ADLER

IO - Kartenbelegung

Drucker 2

Blatt 2

Gerät: AF Bauprinzip IO-Karte: AAG 21 Baugruppe Mikromodul: AAQ 03

Geräte Nr.: 10 Fertigungs Nr. IO-Karte: E 11-0087 Z.Nr. Mikromodul: E 19-0007

Kartenadresse: 0

INPUT	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	INPUT
Zeile 0				TS20	TS10	(PE)	$\overline{C10}$	$\overline{C00}$				TS21	TS11	$\overline{C21}$	$\overline{C11}$	$\overline{C01}$	Zeile 1
Zeile 2																	Zeile 3
Zeile 4																	Zeile 5
Zeile 6																	Zeile 7
Zeile 0				K20	K10	$\overline{M0}$						K21	K11	$\overline{M1}$			Zeile 1
Zeile 2																	Zeile 3
Zeile 4																	Zeile 5
Zeile 6																	Zeile 7
OUTPUT	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	OUTPUT

Kartenadresse: 1

30. April 1975 EE 2

INPUT	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	INPUT
Zeile 0																	Zeile 1
Zeile 2																	Zeile 3
Zeile 4											TEST SYST				WS	P	Zeile 5
Zeile 6	0	0	0	0	0	0	$\overline{DS}$	$\overline{DP}$	$\overline{D8}$	$\overline{D7}$	$\overline{D6}$	$\overline{D5}$	$\overline{D4}$	$\overline{D3}$	$\overline{D2}$	$\overline{D1}$	Zeile 7
Zeile 0										RL6	RL5	RL4	RL3	RL2	RL1	RL0	Zeile 1
Zeile 2	RLWR	(RLSP)							RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0	Zeile 3
Zeile 4	Q		IXL5	IXL4	IXL3	IXL2	IXL1	IXL0					SEL	KLX			Zeile 5
Zeile 6									SL7 α	SL6 NMR	SL5 PER	SL4 INT	SL3 LWL	SL2 R	SL1 MOR	SL0 STP	Zeile 7
OUTPUT	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	OUTPUT

Signalbezeichnungen:

- C: Codierbits für EF
- TS: Taktscheibe
- M: EF-Motor
- K: EF-Kupplungsmagnet
- P: Datenanforderung v. Drucker
- PE: Papierendekont.
- D: Tastaturdaten
- ML: Hauptlampen
- RL: Resttestlampen
- SL: Systemlampen
- IXL: Information an Drucker
- Q: Quittung v. Drucker
- KLX: Druckerkopplung
- SEL: Select-Nadelspannung
- DS: Strobe
- DP: Paritybit

Erstausgabe	17.2.75	BM	Ers i / d	Name Ster	Abt. EE3	Es folgt Blatt -.-
Neuausgabe		BM				

```
STMN   LOCATION   OBJ. CODE     SOURCE STATEMENT

 135   00000800   00030702   INTF2   BS  INTF          INTF2   BS := INTF          0081
 136   00000802   01040800   SIMSPV  BRX XTAS          SIMSPV  BR := (XTAS)         178
 137   00000804   00120800   SIMTAS  SMX XTAS          SIMTAS, (A) := XTAS          179
 138   00000806   01040802          BRX XDFU                  BR := (XDFU)         180
 139   00000808   00120802   SIMDFU  SMX XDFU          SIMDFU  (A) := XDFU          181
 140   00000810   01040804          BRX XPT1                  BR := (XPT1)         182
 141   00000812   00120804   SIMPT1  SMX XPT1          SIMPT1  (A) := XPT1          183
 142   00000814   01040806          BRX XPT2                  BR := (XPT2)         184
 143   00000900   00120806   SIMPT2  SMX XPT2          SIMPT2  (A) := XPT2          185
 144   00000902   01040808          BRX XDIS                  BR := (XDIS)        0186
 145   00000904   00120808   SIMDIS  SMX XDIS          SIMDIS  (A) := XDIS         0187
 146   00000906   01040810          BRX XMKK                  BR := (XMKK)        0188
 147   00000908   00120810   SIMMKK  SMX XMKK          SIMMKK  (A) := XMKK         0189
 148   00000910   01040812          BRX XKLSL                 BR := (XKLSL)       0190
 149   00000912   00120812   SIMKSL  SMX XKLSL          SIMKSL  (A) := XKLSL        0191
 150   00000914   01040814          BRX XKSSS                 BR := (XKSSS)       0192
 151   00001000   00120814   SIMKSS  SMX XKSSS          SIMKSS  (A) := XKSSS        0193
 152   00001002   01040900          BRX XMB1                  BR := (XMB1)         194
 153   00001004   00120900   SIMMB1  SMX XMB1          SIMMB1  (A) := XMB1          195
 154   00001006   01040902          BRX XMB2                  BR := (XMB2)         196
 155   00001008   00120902   SIMMB2  SMX XMB2          SIMMB2  (A) := XMB2          197
 156   00001010   01040904          BRX X1ZG                  BR := (X1ZG)         198
 157   00001012   00120904   SIMG1   SMX X1ZG          SIMG1   (A) := X1ZG          199
 158   00001014   01040906          BRX X2ZG                  BR := (X2ZG)        0200
 159   00001100   00120906   SIMG2   SMX X2ZG          SIMG2   (A) := X2ZG          201
 160   00001102   08120002   SIMEND  LAX X1            SIMEND  (X1) := A            202
 161   00001104   04040008          RTX XTP                   RT := (XTP)          203
 162   00001106   15151506   DEZ         15.15.15.6    DEZ     -10                 0173
 163   00001108   15150912               15.15.9.12             -100                0174
 164   00001110   15120108               15.12.1.8              -1000               0175
 165   00001112   130B1500               13.B.15.0              -10000              0176
 166   00001114   00000000.              0                      0                   0177
```