Ausrüstungen
für Forschungs- und
Ausbildungsstätten

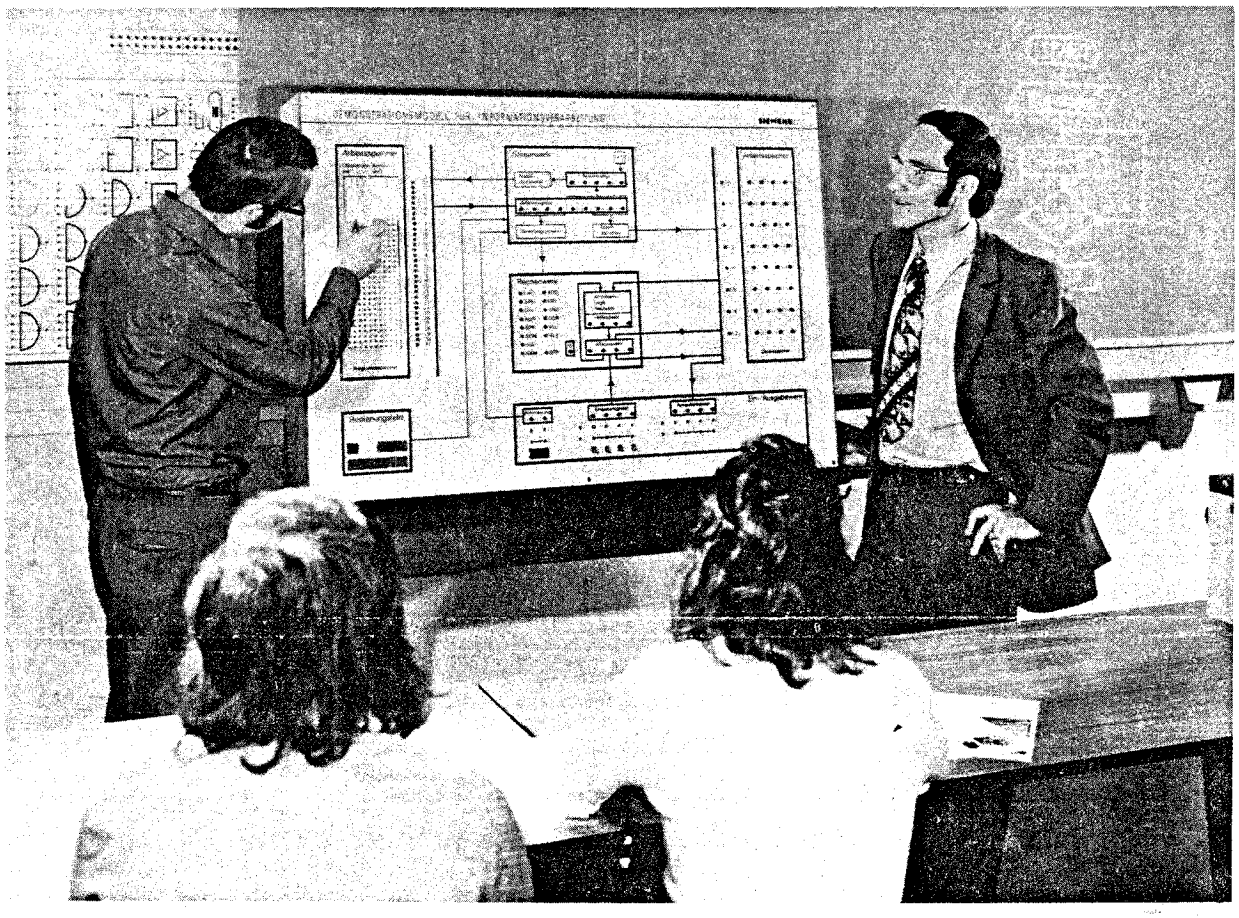
Demonstrationsmodell für Informationsverarbeitung

Betriebsanleitung
Juli 1974
E 484

Inhaltsverzeichnis

Demonstrationsmodell für Informationsverarbeitung

Der Digitalrechner ist heute das Herzstück der modernen Automatisierungstechnik in Wirtschaft, Industrie, Verwaltung und Wissenschaft. Trotzdem herrscht gerade auf diesem Gebiet ein enormer Mangel an ausgebildeten Fachkräften.
Ein neuer Lehrzweig, die Informatik, soll die mathematischen und technischen Grundlagen zum Verständnis und zur Anwendung dieser Automatisierungstechnik vermitteln. Die dabei zu erlernenden neuen Denkweisen unterscheiden sich gegenüber den bisherigen Fachdisziplinen derart grundlegend, und sind oft so abstrakt, daß man sich schon frühzeitig um Unterrichtshilfsmittel bemüht hat.
Abgesehen von den hohen Kosten erlauben die üblichen Datenverarbeitungsanlagen ebenso wie Kleinrechner jedoch keinen Einblick in die internen Abläufe und tatsächlichen Zusammenhänge.



Unterrichtsziele

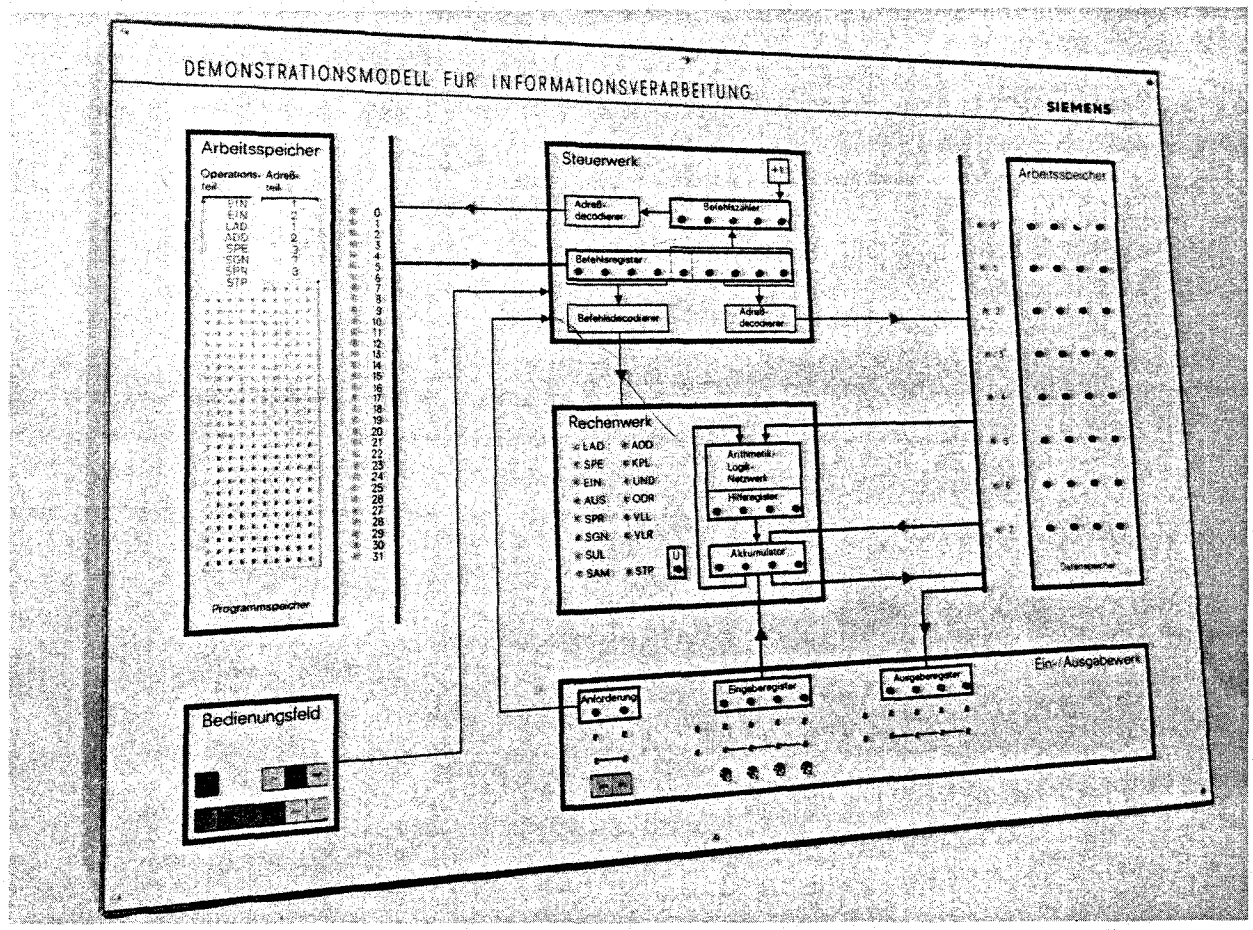
Für den Einsatz an Fachhochschulen, Technikerschulen, Berufsschulen, Fachoberschulen und Schulen für Datenverarbeitung sowie Gymnasien wurde ein neues Demonstrations- und Übungsgerät entwickelt, mit dessen Hilfe die Grundlagen der Informatik von Anfang an durch Anschauungsunterricht vermittelt werden und die Prinzipien der automatischen Informationsverarbeitung schrittweise erarbeitet werden können.
Insbesondere werden Struktur und Arbeitsweise von Datenverarbeitungsanlagen (Computer), informationstechnische Begriffe und Methoden sowie digitale Schaltungen und logische Grundfunktionen durch Vorführungen und Übungen vermittelt. Darüberhinaus dient das Gerät zur Einarbeitung in die Logik des Programmierens und zur Darstellung einfacher und übersichtlicher Anwendungsbeispiele aus dem wissenschaftlichen, kommerziellen und prozeßtechnischen Bereich.
Damit können über die grundlegenden Einsichten hinaus sowohl die Möglichkeiten als auch die Einschränkungen beim Einsatz von Digitalrechnern erfaßt werden.

Aufbau

Das "Demonstrationsmodell für Informationsverarbeitung" entspricht in seiner Grundeinheit einem programmgesteuerten Digitalrechner mit Parallelverarbeitung. Das heißt, das Modell ist ein funktionsfähiger universeller Rechenautomat nach dem Vorbild größerer DV-Anlagen. Hier kann jedoch in die wesentlichen Bausteine eines Computers jederzeit Einsicht genommen werden.

Frontplatte

Auf der Frontplatte (1190 x 825 mm) des Modells sind die wesentlichen Bausteine eines Rechners in Blockstruktur übersichtlich dargestellt und durch Signalflußlinien verbunden. Die Informationen innerhalb der einzelnen Blöcke werden durch farbige Lämpchen gekennzeichnet (hell ≙Signalzustand "1"), Informationsfluß bzw. Funktions- und Programmabläufe werden durch Leuchtpfeile signalisiert.

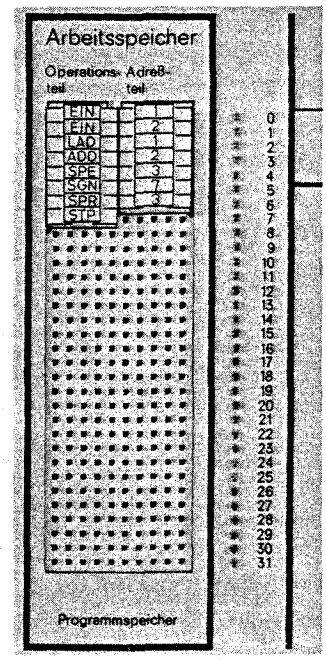


Modellsteuerung

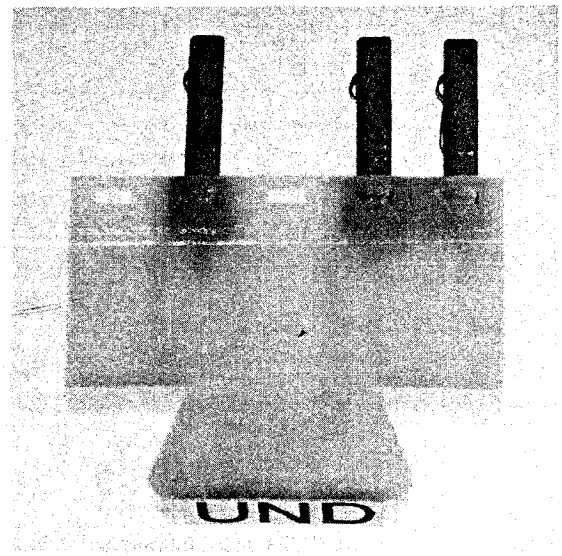
Das Bedienungsfeld des Modells ermöglicht eine überaus freizügige Steuerung des Funktionsablaufes. Insgesamt können 6 verschiedene Betriebsarten mittels gegeneinander verriegelter Tasten gewählt werden. Vier Betriebsarten ermöglichen den automatischen Programmablauf in verschiedenen Ablaufgeschwindigkeiten 0,5 Hz, 2 Hz, 8 Hz und 40 000 Hz Taktfrequenz. Damit können einzelne Taktschritte, Befehle und Programmschleifen in Zeitlupe beobachtet oder der rasche Ablauf eines gesamten Programms in Bruchteilen von Sekunden und damit die Leistungsfähigkeit von elektronischen Rechnern demonstriert werden. Mit den zwei weiteren Betriebsarten werden einzelne Takte, oder Befehle (1 Befehl = 4 Takte) von Hand ausgelöst, so daß jede einzelne Phase beliebig lang festgehalten werden kann. Die einzelnen

Betriebsarten können während des Programmablaufs umgeschaltet werden, da eine Synchronisationsschaltung diese Umschaltung nur zum Taktanfang wirksam werden läßt. Drei weitere Tasten erlauben, Programme jederzeit zu stoppen (STOP) bzw. durch Drücken der Tasten RÜCKSETZEN und START neu zu beginnen.

Programmierung

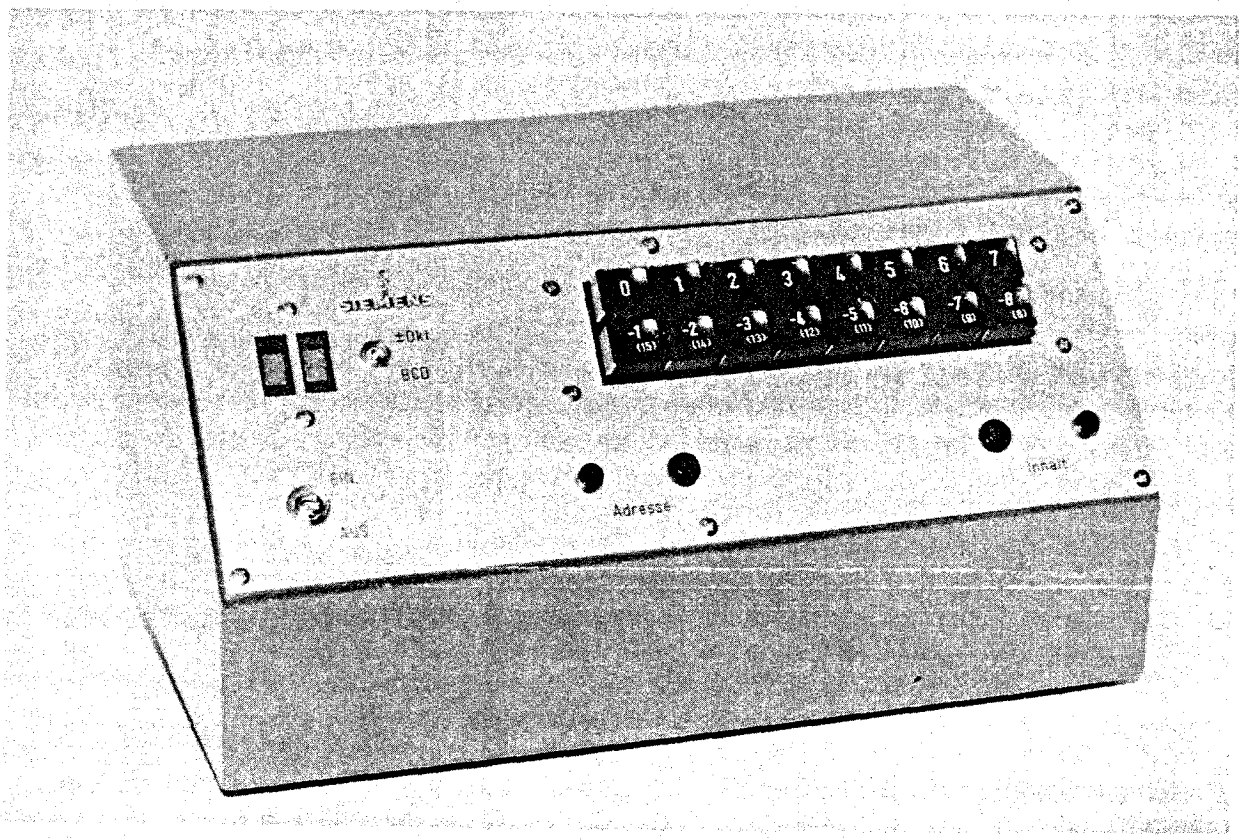


Die Programmierung erfolgt in Maschinensprache. Um nicht in binärcodierter Form (d.h. in Nullen und Einsen) programmieren zu müssen, wurde der Programmspeicher in Form eines Buchsenfeldes aufgebaut, in das die einzelnen Befehle durch vorgefertigte Steckerbausteine "eingeschrieben" werden. Diese beschrifteten Stecker lassen die Befehls- bzw. Adreßverschlüsselung in ihrer codierten Form erkennen, ohne die Übersichtlichkeit des Programms zu schmälern. Auf diese Weise ist das im Buchsenfeld gesteckte Programm (maximal 32 Befehle) wie auf einem Schnelldruckerprotokoll lesbar und kann jederzeit, z.B. beim Testen, durch Umstecken korrigiert werden. Darüber hinaus bleibt die Möglichkeit bestehen, mit Bitsteckern auch rein binär zu programmieren.



Anschlußmöglichkeiten

Das Ein-/Ausgabewerk des Modells wurde aus Sicherheitsgründen durch Eingabe- und Ausgaberelais potentialfrei gehalten. Es erlaubt neben der einfachen Kippschaltereingabe die digitale Ein- und Ausgabe statischer Informationen über Buchsenleisten (je 4 Bit). Damit können einerseits vereinfachte Peripheriegeräte angeschlossen werden (siehe Bild unten: Eingabetastatur und Dezimalanzeige), andererseits wird damit die Ankopplung des Schulungsrechners an kleinere "Prozesse" (z.B. Relaissteuerungen) möglich. Darüber hinaus wurde im Modell eine dynamische Digitaleingabe nach Art einer Alarmeingabe oder eines Interrupts verwirklicht. Diese sogenannte Anforderungssteuerung speichert Impulse und erzeugt nach Ablauf des gerade laufenden Programms externgesteuerte neue Programmstarts bei zwei verschiedenen Befehlszählerständen (0 und 1), die den beiden Anforderungen fest zugeordnet sind. Damit kann das Prinzip eines Echtzeitrechners (real-time) demonstriert werden.
(Nähere Angaben und Definitionen siehe Ein-/Ausgabewerk, Seite 11)



Eingabetastatur und Dezimalanzeige

Handhabung

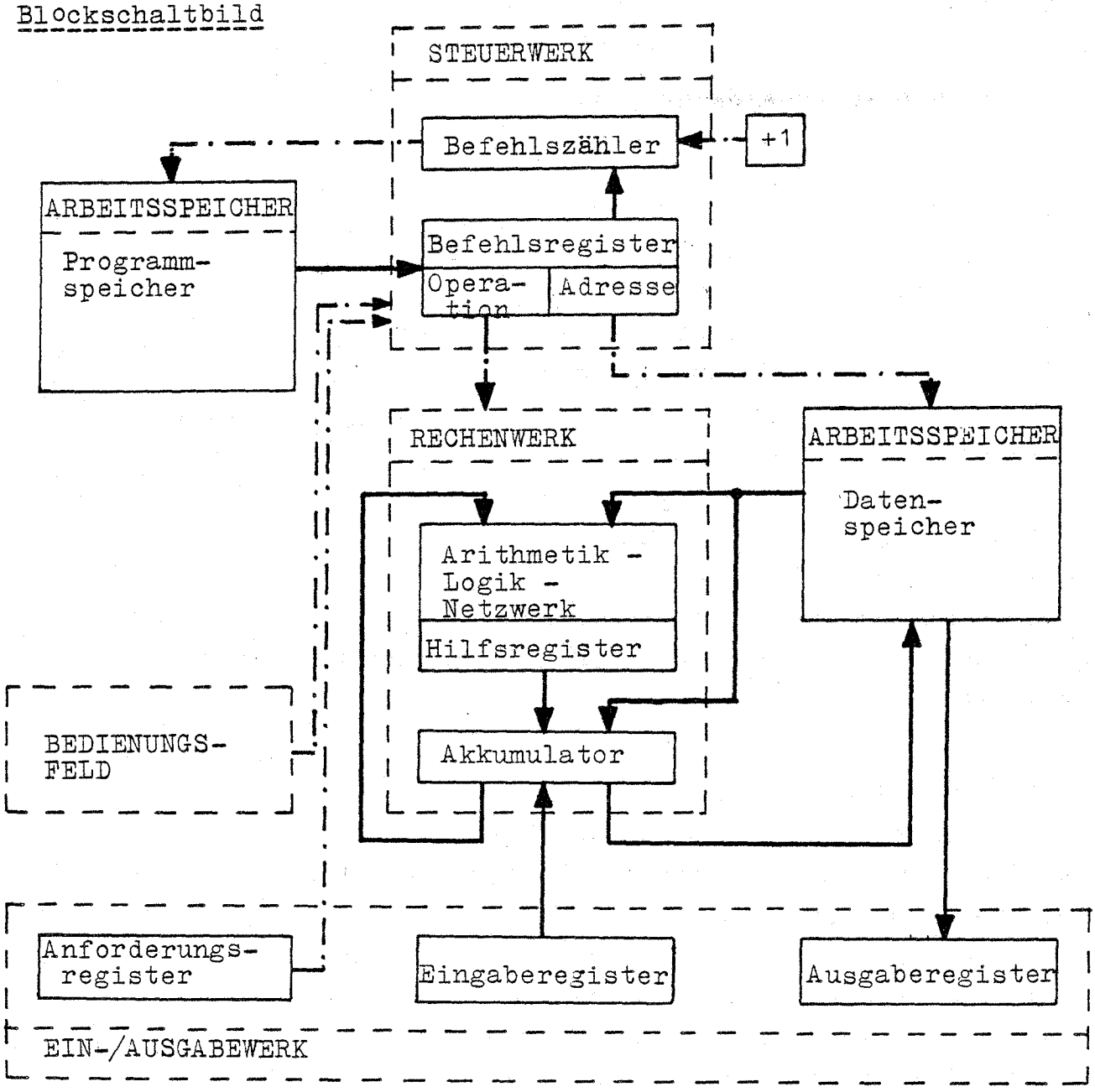
Das Demonstrationsmodell für Informationsverarbeitung wird mit kratzfester und beschreibbarer (Fettstift) Resopalfrontplatte geliefert. Bedienungen und Anschlüsse sind "Schülersicher" ausgeführt. Das Gerät kann entweder auf einem Tisch aufgestellt oder an die Wand gehängt werden und wird durch Anschluß an die Netzsteckdose betriebsbereit. Es kann mittels zweier seitlich am Gerät angebrachter versenkbarer Tragegriffe von zwei Schülern bequem getragen werden. Wir empfehlen jedoch zusätzlich, unseren fahrbaren Arbeitstisch zu verwenden, um das Modell mit allem Zubehör ohne Zeitverlust zwischen verschiedenen Unterrichtsräumen transportieren zu können.

Die Frontplattenaussage des Demonstrationsmodells kann durch Schaltbilder (15 Stück im Lieferumfang), die auf die Frontplatte des Modells aufgebracht werden, beträchtlich erweitert werden. Damit lassen sich einerseits Befehlsabläufe von der Hardwarefunktion verständlich machen; andererseits können logische Funktionen sowie Digitale Schaltungen demonstriert und erläutert werden.

Weitere Hinweise und Unterlagen, z.B. zur Verwendung des Gerätes zum schulmäßigen Üben der Fehlersuche, finden sich in der Betriebsanleitung zum Demonstrationsmodell.

Funktion

Blockschaltbild



Beschreibung der Funktionsblöcke

1. Steuerwerk

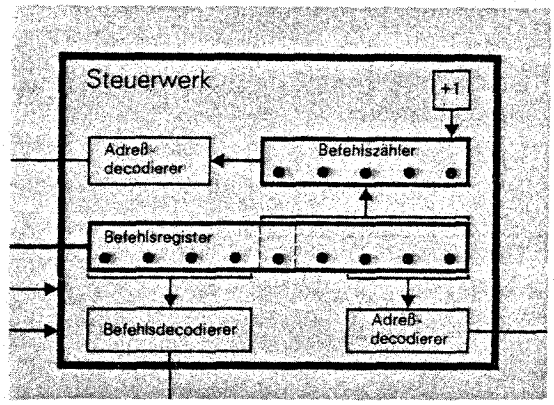
Befehlszähler:

Der Befehlszähler (5 Bit) enthält die duale Adresse des nächsten zu verarbeitenden Befehls. Er ist damit Ausgangspunkt eines jeden Funktionsablaufs im Rechner. Der Befehlszählerstand wird bei je-

dem normalen Befehlszyklus mit dem Takt 4 um 1 erhöht, so daß ein Befehl nach dem anderen ablaufen kann. Bei Sprungbefehlen wird der Befehlszähler (mit dem Takt 3) neu eingestellt (gesetzt), um Programmverzweigungen oder Schleifen zu ermöglichen.

Befehlsregister

Das Befehlsregister (9 Bit) speichert das aktuelle Befehlswort bei seiner Bearbeitung und zeigt die binäre Codierung von Operations- und Adreßteil auf (Befehlsformat).



Befehlszähler und Befehlsregister ermöglichen den automatischen Ablauf von Programmen. Sie sind der wesentlichste Teil des Steuerwerks. Rein technisch gesehen schließt das Steuerwerk die Takterzeugung und Taktverteilung mit ein, die jedoch nicht direkt angezeigt werden.

2. Arbeitsspeicher

Programmspeicher:

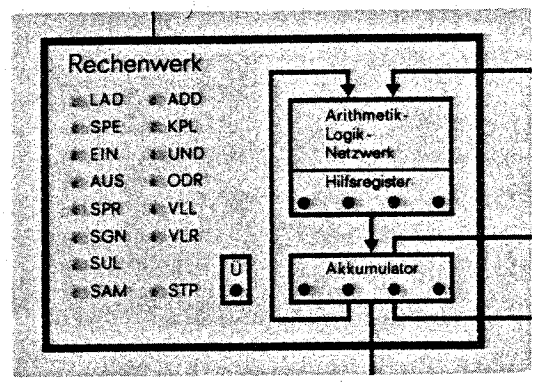
Der Programmspeicher nimmt das Programm (Befehlsfolge) auf. Er ist frei programmierbar durch Stecken von Operations- und Adreßbausteinen entsprechend der Gliederung eines Befehlswortes in die Operation und die Operandenadresse (Einadreßbefehlssystem!) Die aktuelle Befehlsadresse wird während eines Befehlszyklus in einer Signallampenleiste , welche die Adreßleitung symbolisiert, anzeigt.

Datenspeicher:

Der Datenspeicher ist als frei beschreibbarer Halbleiterspeicher ausgebildet. Er beinhaltet 8 Worte zu je 4 Bit zur Aufnahme der Daten. Über Signallämpchen werden Speicherinhalt und -adresse sichtbar gemacht.

Datenspeicher und Programmspeicher sind im allgemeinen in einem Arbeitsspeicher (Zentralspeicher) zusammengefaßt, so daß die Befehlsworte wie die Daten eingelesen (nicht gesteckt) und gespeichert werden. Aus didaktischen Gründen - direkte Lesbarkeit der Programme und klare Unterscheidung-wurde jedoch eine Trennung vorgenommen.

3. Rechenwerk:



Im Rechenwerk werden die 4 Bit breiten Daten parallel verarbeitet nach Maßgabe der Befehle. Das Befehlsspektrum wurde möglichst breit gefächert.
Es besteht aus 4 Transferbefehlen für internen und externen Datenverkehr, 4 Sprungbefehlen, einem unbedingten und drei bedingten Sprungbefehlen, 2 arithmetischen Befehlen, 2 logischen Befehlen sowie 2 Verschiebebefehlen. Ein Teil der Befehle ist substituierbar, um eine Adreßrechnung bzw. auch Indizierung zu ermöglichen.
Die Befehlsanzeige - jeder aktuelle Befehl leuchtet nach seiner Decodierung in den beiden Lampenleisten auf - wurde aus Gründen der Übersichtlichkeit im Rechenwerk zusammengefaßt, obwohl natürlich nicht alle Befehle Rechenwerkbefehle sind. So wird z.B. der unbedingte Sprungbefehl SPR nur im Steuerwerk bearbeitet. Die bedingten Sprungbefehle SAM, SGN, SUL werden dagegen schon wieder mit Bedingungen aus dem Rechenwerk verknüpft.

Akkumulator:

Der Akkumulator ist dem Rechenwerk direkt zugeordnet, und dient als Verarbeitungs- und Ergebnisregister. Er nimmt vor der Operation das zu verarbeitende Datenwort auf und enthält nach der Operation das Ergebnis. Sollen demnach zwei Operanden verknüpft werden, so muß zunächst der eine Operand in den Akkumulator

gebracht werden - durch den Befehl "LAD w". Der zweite Operand wird nun mit dem Akkumulatorinhalt verknüpft - z.B durch den Befehl "ADD w" oder "UND w". Das Ergebnis wird in den Akkumulator eingetragen und überschreibt damit den ersten Operanden. Auch andere Operationen wie Verschiebebefehle oder Komplementierungsbefehl, die sich nur auf einen Operanden beziehen, werden mit dem Akkumulatorinhalt durchgeführt, so daß auch hier der Operand zunächst in den Akkumulator geladen werden muß. An diese Operationen muß sich im allgemeinen ein Rücktransfer des Ergebnisses in den Speicher anschließen, durch den Befehl "SPE w". Moderne Datenverarbeitungsanlagen werden zunehmend mit mehr als einem (4, 8, 16) Verarbeitungsregister ausgestattet. Wenn dabei auch Register untereinander verknüpft werden können, z.B.: ADD R1, R2 (Addition der Inhalte der Register 1 und 2), so spricht man von Registermaschinen im Unterschied zu den Akkumulatormaschinen. Dabei geht natürlich das bisher einheitliche Befehlsformat, die Gliederung eines Befehlswortes in Operationsteil und Adreßteil verloren. In den Operationsteil muß dabei eine Formatangabe integriert werden. Im Wesentlichen handelt es sich dabei um die Angabe, ob im bisherigen Adreßteil 2 Registeradressen bzw. eine Registeradresse und eine Speicheradresse oder eine Sprungadresse stehen.Es gibt auch Zweiadreßmaschinen.

Hilfsregister

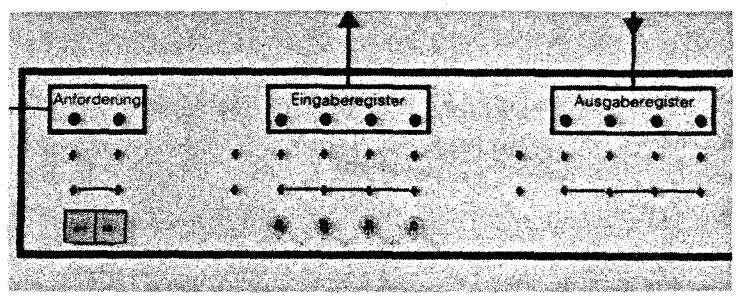
Das Hilfsregister zeigt Ergebnisse am Ausgang des Arithmetik- und Logiknetzwerks auf, bevor die Ergebnisse in den Akkumulator eingeschrieben werden und damit einen der Operanden überschreiben.

Überlauf

Der Überlauf wird dann gesetzt, sobald bei der Addition der Bereich der zulässigen positiven oder negativen Zahlen (-8, -7,... -1, 0, 1, 2,...7) nach oben oder unten überschritten wird. Der Überlauf ist der sog. Festpunktzahlendarstellung mit Vorzeichen zugeordnet und darf nicht mit dem Übertrag verwechselt

werden, welcher der Zahlendarstellung ohne Vorzeichen (absolute Zahlen) zugeordnet ist. Der Übertrag wurde im Modell nicht berücksichtigt. Er würde im Modell dann gesetzt werden, wenn bei der Addition der Bereich der darstellbaren absoluten Zahlen (0, 1, 2,...15),d.h. die 15,nach oben überschritten wird. Er hat jedoch kaum praktische Bedeutung bei der Programmierung. (Das Auftreten des Übertrags müßte dann durch einen bedingten Sprungbefehl, "Springe falls Betragsüberlauf" abgefragt werden.

4. Ein- / Ausgabewerk



Über die digitalen Ein- / Ausgaberegister können statische Informationen programmgesteuert eingegeben ("EIN w") bzw. ausgegeben ("AUS w") werden. (Statische Informationen müssen so lange anstehen, bis sie abgeholt worden sind; dynamische Informationen sind Impulse, die auch dann noch gespeichert werden, wenn sie nicht mehr anstehen.) Die je 4 Bit der Ein- /Ausgabe können über Buchsenleisten (Bananenstecker) mit Ein- /Ausgaberelais potentialfrei aufgelegt bzw. abgenommen werden, so daß beim Anlegen von gefährlichen Spannungen lediglich die Relais, nicht aber die Rechnerelektronik zerstört wird. Die Relais haben eine Schaltleistung von 60 VA und können maximale Spannungen von 100 V oder maximale Ströme von 2 A im Dauerbetrieb schalten. Zwischen den Buchsen werden jedoch lediglich Kontakte geschlossen, so daß weder Spannungssignale ausgegeben noch Signalspannungen aufgelegt werden müssen. Zwischen den Buchsen des Ausgaberegisters wird beim Ausgabesignal "1" lediglich ein Kontakt geschlossen. Ebenso muß für die Eingabe des Signals "1" zwischen den Buchsen des Eingaberegisters lediglich ein Kontakt geschlossen werden. Die Schalthäufigkeit kann 20 Schaltspiele pro Sekunde nicht überschreiten.

Die Anforderungssteuerung ANF0/ANF1 ermöglicht den externgesteuerten Start von Programmen. Impulse (dynamische Eingabe), die an den Anforderungseingängen durch einmaliges Drücken der Tasten oder Schließen von Kontakten zwischen den Buchsen einlaufen, werden gespeichert und prioritiert (ANF0 hat Vorrang vor ANF1). Nach Ablauf des gerade laufenden Programms werden unmittelbar anschließend neue Programmstarts von fest zugeordneten Befehlszählerständen ausgelöst. ANF0 erzeugt einen Programmstart ab Befehlszählerstand "0", ANF1 bei Befehlszählerstand "1". Damit können zwei beliebige Programme (Alarmprogramme oder Ein-/Ausgabeprogramme oder Zählerprogramme) von angeschlossenen Geräten "prozeßgerecht" gestartet werden (real-time-processing).

Befehlsliste

Befehlsart	Name	Bedeutung
Transfer-befehle LDA	LAD w	Laden des Akkumulators *) Der Inhalt der Arbeitsspeicherzelle w (Adresse) wird in den Akkumulator übertragen.
STA	SPE w	Speichern *) Der Inhalt des Akkumulators wird in die Arbeitsspeicherzelle w (Adresse) übertragen.
IN	EIN w	Eingabe *) Der Inhalt der Digitaleingabe wird in den Akkumulator und in die Arbeitsspeicherzelle w (Adresse) übertragen.
OUT	AUS w	Ausgabe *) Der Inhalt der Arbeitsspeicherzelle w (Adresse) wird in das Digitalausgaberegister übertragen.
Arithmetische Befehle (Festpunkt)	ADD w	Addition *) Der Inhalt der Arbeitsspeicherzelle w (Adresse) wird zum Akkumulatorinhalt addiert.
CMA	KPL	Komplementieren Der Akkumulatorinhalt wird komplementiert (2er Komplement)
Logische Befehle ANA	UND w	UND *) Der Inhalt des Akkumulators wird mit dem Inhalt der Arbeitsspeicherzelle w (Adresse) im Sinn des "logischen UND" verknüpft.
	ODR w	ODER *) Der Inhalt des Akkumulators wird mit dem Inhalt der Arbeitsspeicherzelle w (Adresse) im Sinne des "logischen ODER" verknüpft.

*) Befehl ist substituierbar

Befehlsart	Name	Bedeutung
Verschiebe-befehle RLC	VLL	Verschiebe logisch links Der Akkumulatorinhalt wird um 1 Bit nach links verschoben. Von rechts werden Nullen nachgezogen. Die Vorzeichenstelle wird wie jede andere Bitstelle behandelt.
RRC	VLR	Verschiebe logisch rechts Der Akkumulatorinhalt wird um 1 Bit nach rechts verschoben. Von links werden Nullen nachgezogen. Die Vorzeichenstelle wird wie jede andere Bitstelle behandelt.
Sprung-befehle JMP	SPR a	Springe Das Programm wird mit dem Befehlswort a (Programmspeicheradresse) fortgesetzt.
JZ	SGN a	Springe falls Akkumulator gleich Null Ist der Inhalt sämtlicher Akkumulatorbitstellen gleich Null, so wird das Programm mit dem Befehlswort a (Programmspeicheradresse) fortgesetzt. Ist der Akku-Inhalt ungleich Null, so wird das Programm mit dem auf den SGN-Befehl folgenden Befehlswort fortgesetzt.
JM	SAM a	Springe falls Akkumulator minus Ist die Stelle 1 (links) des Akkumulators mit Eins gesetzt (negatives Vorzeichen), so wird das Programm mit dem Befehlswort a (Programmspeicheradresse) fortgesetzt. Anderenfalls wird das Programm mit dem auf den SAM-Befehl folgenden Befehlswort fortgesetzt.
JC	SUL a	Springe falls Überlauf Ist das Überlaufregister mit Eins gesetzt, so wird das Programm mit dem Befehlswort a (Programmspeicheradresse) fortgesetzt und der Überlauf gelöscht. Anderenfalls wird das Programm mit dem auf den SUL-Befehl folgenden Befehlswort fortgesetzt.
HLT Organisatorische Befehle	STP	Stop Das Programm wird gestoppt.

Codierung

Operationscode

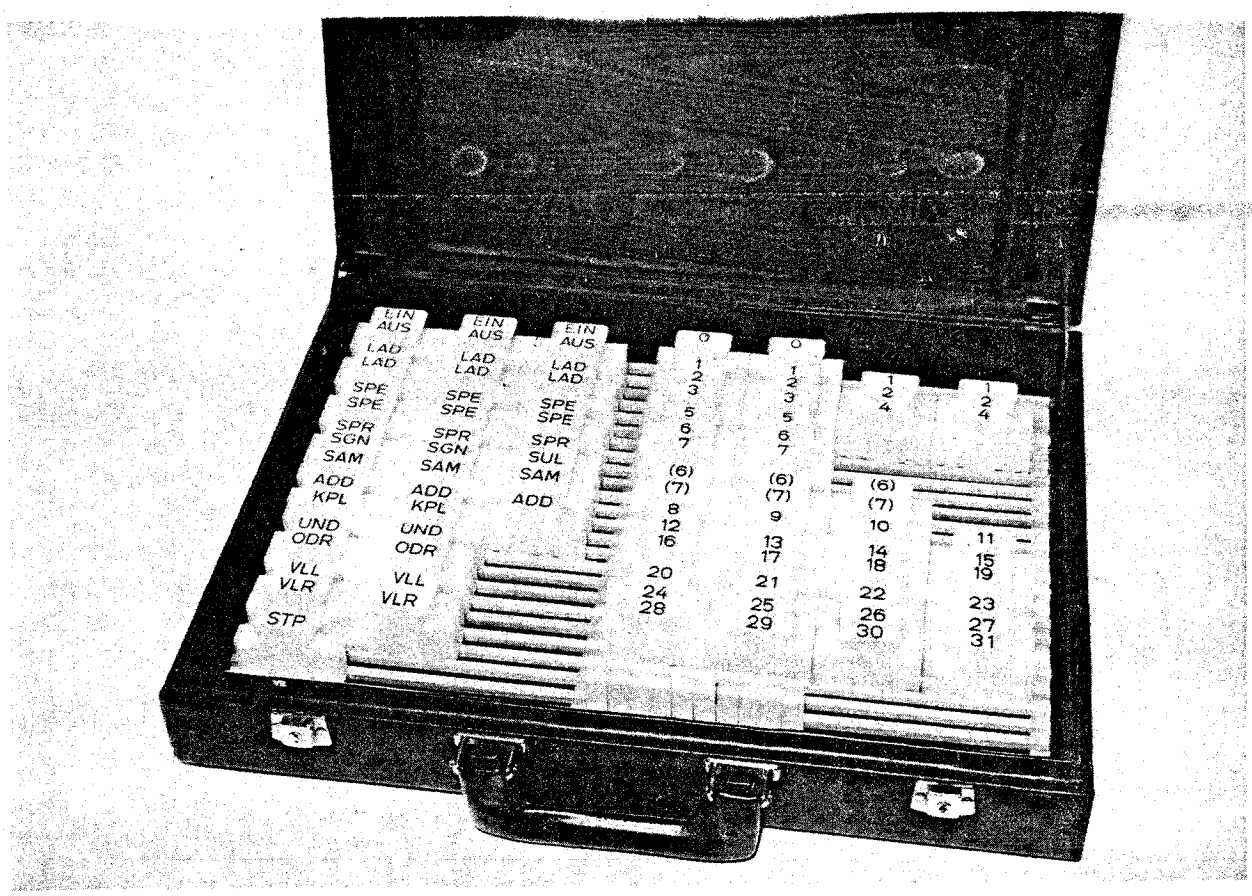
Dezimale Beschriftung	Duale Verschlüßlung	Bedeutung
LAD	000I	Transfer-befehle
SPE	00I0	
EIN	00II	
AUS	0I00	
SPR	0I0I	Sprung-befehle
SGN	0II0	
SUL	0III	
SAM	I000	
ADD	I00I	Arithmetische Befehle
KPL	I0I0	
UND	I0II	Logische Befehle
ODR	II00	
VLL	II0I	Verschiebe-Befehle
VLR	III0	
STP	0000	Stopbefehl

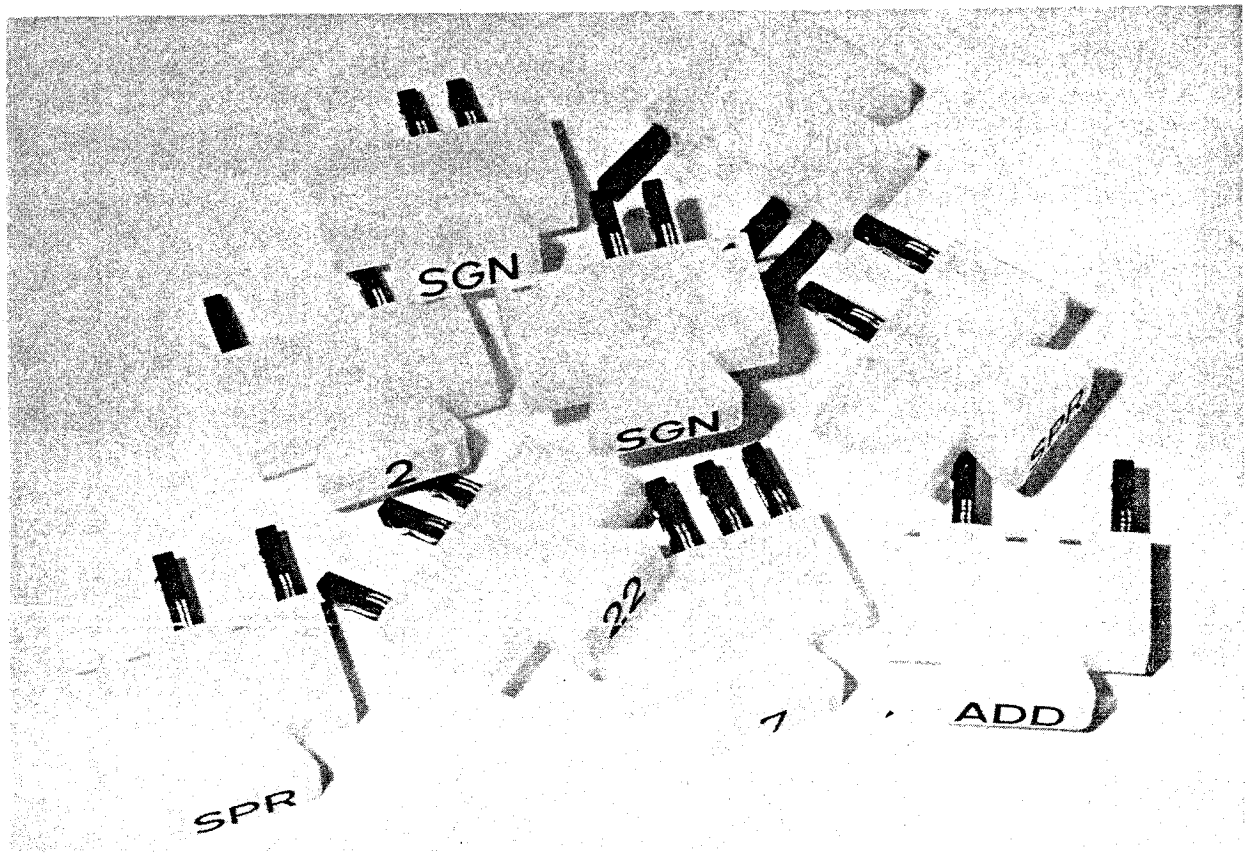
Adreßcode

Dezimale Beschriftung	Duale Verschlüßlung	Bedeutung
0	000	Adressen der Operanden
1	00I	
2	0I0	
3	0II	
4	I00	
5	I0I	
6	II0	
7	III	
(0)	I0000	Substituierte (indirekt angesprochene) Operanden-adressen
(1)	I000I	
(2)	I00I0	
(3)	I00II	
(4)	I0I00	
(5)	I0I0I	
(6)	I0II0	
(7)	I0III	

zu Adreßcode

Dezimale Beschriftung	Duale Verschlüßlung	Bedeutung
0	00000	Befehlszähler-adressen (Sprung-adressen)
1	000I0	
2	000I0	
3	000II	
4	00I00	
5	00I0I	
6	00II0	
7	00III	
8	0I000	
9	0I00I	
10	0I0I0	
11	0I0II	
12	0II00	
13	0II0I	
14	0III0	
15	0IIII	
16	I0000	
17	I000I	
18	I00I0	
19	I00II	
20	I0I00	
21	I0I0I	
22	I0II0	
23	I0III	
24	II000	
25	II00I	
26	II0I0	
27	II0II	
28	III00	
29	III0I	
30	IIII0	
31	IIIII	


EIN AUS
LAD LAD
SPE SPE
SPR SGN SAM
ADD KPL
UND ODR
VLL VLR
STP
EIN AUS
LAD LAD
SPE SPE
SPR SGN SAM
ADD KPL
UND ODR
VLL VLR
EIN AUS
LAD LAD
SPE SPE
SPR SUL SAM
ADD
0
1 2 3 5 6 7 (6) (7) 8 12 16 20 24 28
0
1 2 3 5 6 7 (6) (7) 9 13 17 21 25 29
1 2 4 (6) (7) 10 14 18 22 26 30
1 2 4 11 15 19 23 27 31


SGN
SGN
SPR
2
22
1
ADD
SPR

Einsatzmöglichkeiten

Das "Demonstrationsmodell für Informationsverarbeitung" ist zur Demonstration der Grundfunktionen und funktionalen Zusammenhänge von Digitalrechnern und von Anwendungsbeispielen der automatisierten Informationsverarbeitung geeignet. Damit können die Grundlagen und Prinzipien der Informatik von Anfang an sehr anschaulich vermittelt und schrittweise erarbeitet werden. Über die Darstellung der Struktur und Arbeitsweise von Datenverarbeitungsanlagen und deren informationstechnische Prinzipien und digitale Schaltungen hinaus dient das Modell zur Einarbeitung in die Logik des Programmierens. Das Zusammenwirken von Hardware (Geräte) und Software (Programme), Zentraleinheit und Peripherie (angeschlossene Ein-/Ausgabegeräte) soll bei der Durchführung von Anwendungsbeispielen aus dem wissenschaftlichen, kommerziellen und prozeßtechnischen Bereich erfahren werden. Nicht zuletzt werden die Vorstellungen über die Einsatzmöglichkeiten von Digitalrechnern präzisiert.

A. Demonstrationen und Übungen zur Grundlagenausbildung in Informatik

Begriffsbildung zur Informatik

Informationsdarstellung

Binärzeichen
Codes zur Zeichendarstellung, Umcodierung
Stellenwertsysteme
- Dualsystem
- Okalsystem
- Umwandlungen zwischen Stellenwertsystemen (dual ⟷ dezimal)

Sicherung von Informationen (Paritybit, ...)
Zahlendarstellung
- Festpunktzahlen ohne Vorzeichen (Beträge)
- Festpunktzahlen mit Vorzeichen
- Gleitpunktzahlen (Komma!)

Befehlsdarstellung
- Operationstypen (Befehlsarten)
- Befehlsdarstellung in Bits
- Befehlsformate

Arithmetik der Informationsverarbeitung

Dualaddition ohne Vorzeichen
Komplementbildung und Subtraktion
Multiplikations- und Divisionsalgorithmus

Grundfunktionen der Logik und Boolesche Algebra

Konjunktion, Disjunktion, Negation
Assoziatives, kommutatives, distributives Gesetz
De Morgansches Theorem
Wertetafel, Ableitung der Schaltfunktion
(z.B. Äquivalenz, Antivalenz...)

Grundbausteine der Rechnertechnik

Grundschaltungen der EDV
- Register (Speicher, Schieberegister...)
- Zähler
- Decodierer
- Paralleladdierer, Komplementierer, Logiknetzwerk

Realisierung der logischen und arithmetischen Funktionen
Prinzipien der Speicherung
Grundlegende Eigenschaften der Rechner-Funktionseinheiten
Technischer Aufbau und Organisation einer Zentraleinheit
Aufgaben der Peripheriegeräte
- Ein- und Ausgabegeräte
- Nahtstellen, Kanalwerk
- Simultanarbeit
- Datenübertragung

Beurteilung von DV-Anlagen (Leistungsparameter)

Grundlagen der Programmierausbildung

Begriffsbildung zur Software
Prinzipielles Vorgehen beim Aufbau von Programmen
- Problemanalyse
- Programmablaufplan
- Herstellung des Maschinenprogramms
Struktur von Programmen
Testen von Programmen
Prinzip eines Assemblers
Der Weg zur problemorientierten Sprache (Compiler)
Prinzipien von Betriebssystemen
- Prioritäten, Real-Time-Verarbeitung, Multiprogramming
- Hilfsprogramme

B) Demonstrationen und Übungen anhand von Anwendungsbeispielen aus der automatischen Informationsverarbeitung (siehe Aufgabensammlung)

Gruppe 1: Prozeßrechnereinsatz
Automatische Informationsverarbeitung bei technischen Prozessen.

Gruppe 2: Kommerzielle Datenverarbeitung
Informationsverarbeitung, demonstriert an Beispielen aus dem EDV-Bereich.

Gruppe 3: Technisch-wissenschaftlicher Rechnereinsatz
Informationsverarbeitung in Rechenautomaten am Beispiel der Lösung von Rechenaufgaben.

Die angegebenen Einsatzmöglichkeiten zeigen auf, daß dieses Modell im Gegensatz zu Tischrechnern in jeder Phase des Informatikunterrichts eingesetzt werden kann. Darüber hinaus werden die technischen Strukturen von DV-Anlagen richtig aufgezeigt, die weit über den Tischrechnereinsatz hinausführen und eine Fehleinschätzung verhindern.

Unterrichtsformen

Das Demonstrationsmodell für Informationsverarbeitung dient als Unterrichtshilfsmittel zur Gestaltung eines anschaulichen und effektiven Unterrichts. Dabei sind folgende Unterrichtsformen denkbar:

- Demonstrationen von Lehrschritten und Übungsbeispielen durch den Pädagogen selbst.
- Demonstration von Übungsbeispielen durch beauftragte Schüler unter Anleitung des Pädagogen.
- Erarbeitung von Lehrschritten im Unterrichtsgespräch zwischen Lehrer und Schülern, die von beauftragten Schülern am Modell verwirklicht und von der Klasse beurteilt werden.
- Schüler lösen bzw. nachvollziehen die gestellten Aufgaben im Gruppenunterricht an mehreren Geräten.
- Schülergruppen bzw. einzelne Schüler erarbeiten im Praktikumsunterricht die gestellten Aufgaben nach Anleitung durch die Aufgabensammlung oder selbständig.

Literaturhinweis

Für die Arbeit mit dem Modell empfehlen wir besonders die Buchreihe:

Datenverarbeitung - Programmierter Selbstunterricht -
Siemens AG

1.) Zahlensysteme - von Lothar Moos
2.) Logische Schaltungen Teil 1: Verknüpfungsglieder - von Fleischer
3.) Logische Schaltungen Teil 2: Speicherglieder - von Fleischer
4.) Struktur und Arbeitsweise von Datenverarbeitungsanlagen - von Lothar Moos

Darüberhinaus empfehlen wir:

Der Schlüssel zum Computer
von Martin F. Wolters, Econ-Verlag, Düsseldorf-Wien

Lexikon der DV
Löbel-Müller-Schmid
Verlag Moderne Industrie, Anleitung und Beispiele

PROSA 300 für Prozeßautomatisierung
Band 1: Einführung-Heinz Höppl, Siemens Aktiengesellschaft

Befehlsabläufe

Die Arbeitsweise einer Datenverarbeitungsanlage (= Zentraleinheit + Ein-/Ausgabegeräte) ist durch das Zusammenwirken von Geräten (hardware) und Programmen (software) gekennzeichnet. Die Zentraleinheit als wesentlichster Bestandteil der "hardware" stellt ein System von Funktionseinheiten zur Verfügung, das durch ein vom Menschen erstelltes Programm und nur durch ein Programm aktiviert wird. Auf Grund der unveränderlichen technischen Struktur der Zentraleinheit muß sich ein solches Programm - es stellt den veränderlichen Anteil dar - an strenge Gesetzmäßigkeiten halten:

Programme bestehen aus einer Folge von Befehlen. Die Befehle selbst sind Arbeitsanweisungen an den Rechner.

Welche Aussagen müssen Befehle zumindest beinhalten, damit sie verstanden und ausgeführt werden können?

1. Was soll getan werden?
2. Womit soll etwas getan werden?

Entsprechend diesen beiden Befehlselementen wurde ein Befehl in

1. Operationsteil und
2. Adreßteil gegliedert.

Mit dem Adreßteil wird ein Platz im Speicher gekennzeichnet, dessen Inhalt bei der Operation gemeint ist. Dieser Inhalt, der im Befehl adressierten Zelle, heißt Operand. Man bezeichnet die im Adreßteil eines Befehls stehende Adresse auch als Operanden-Adresse.

Solche Befehle heißen z.B. ADD 5 (Addiere Inhalt von 5)
SPE 3 (Speichere nach 3)

Welcher Anteil ist in diesen Beispielen Operationsteil?

- ADD, SPE

Was ist mit den Zahlen angesprochen?

- Adressen

Wo stehen diese Adressen?

- im Arbeitsspeicher

Was beinhalten diese Adressen?

- Operanden (Daten).

Transferbefehle

Ein Digitalrechner (Zentraleinheit) besteht aus Speicher, Rechanwerk, Steuerwerk und Ein-/Ausgabewerk. Über Eingabegeräte, die an das Ein-/Ausgabewerk angeschlossen sind, wird die Zentraleinheit mit Programmen und Daten versorgt, die gespeichert werden müssen. Zur Verarbeitung werden die Daten vom Speicher ins Rechenwerk gebracht. Die hier ermittelten Ergebnisse müssen ebenfalls gespeichert und später ausgegeben werden. Damit sind vier Transferbefehle angesprochen, ohne die keine Verarbeitung im Rechner möglich ist, 1.EINgabe in den Speicher, 2.AUSgabe, 3.LADen, 4.SPEichern.
Führen Sie nun mit dem Modell folgende Übung durch:

Stecken Sie die folgenden Befehle wie angegeben im Arbeitsspeicher (Programmspeicher) und führen Sie die darauf folgenden Bedienungen mit dem eingeschalteten Modell durch:

Programmspeicher

EIN	5	0
EIN	7	1
AUS	5	2
AUS	7	3
LAD	5	4
LAD	7	5
SPE	6	6
SPE	4	7

A) 1) Betriebsart BEFEHL einschalten,

2) Taste RÜCKSETZEN drücken,

3) Kippschalter auf IIII (das ist eine duale Zahl) einstellen, (alle Lampen im Eingaberegister leuchten),

4) Beobachten Sie den Datenspeicher während Sie auf Taste START drücken!

Vergleichen Sie nun den ersten gesteckten Befehl, "EIN 5", mit den Signalleuchten, die aufgeleuchtet sind! Was hat dieser Befehl bewirkt?

- Eingabe des Inhalts des Eingaberegisters in Adresse 5 des Datenspeichers.
  (Die gleichzeitige Eingabe in den Akkumulator im Rechenwerk hat praktische Vorteile bei der späteren Programmierung, ist hier aber nicht von grundsätzlicher Bedeutung).

Welche Signalleuchte weist darauf hin, daß nun der Befehl "EIN 7" folgt?

- Die Lampe 1 im Programmspeicher.

B) 1) Kippschalter auf 000I einstellen,

2) Taste START drücken und Datenspeicher beobachten!

Was konnten Sie feststellen?

- Eingabe der 000I, die im Eingaberegister steht, in den Datenspeicher an den Platz, der durch die Adreßangabe im Befehl gekennzeichnet war.

C) 1) Taste START drücken und Ausgaberegister beobachten,

2) Taste START erneut drücken!

Was haben die beiden Befehle bewirkt, die ausgelöst durch START abliefen?

- Ausgabe des Inhalts der im Befehl angegebenen Adressen ins Ausgaberegister.

D) 1) Taste START drücken und Akkumulator (Verarbeitungsregister) beobachten,

2) Taste START erneut drücken!

Was konnten Sie feststellen?
- Die Befehle LAD 5 und LAD 7 haben die Inhalte der Adresse 5 und dann der Adresse 7 in den Akkumulator gebracht (geladen).

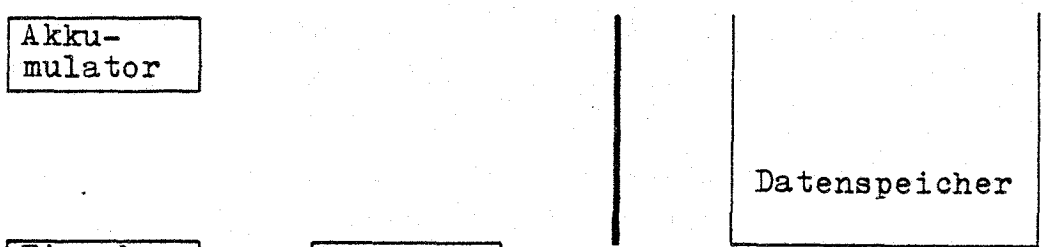
E) 1) Taste START drücken und Datenspeicher beobachten,

2) Taste START erneut drücken!

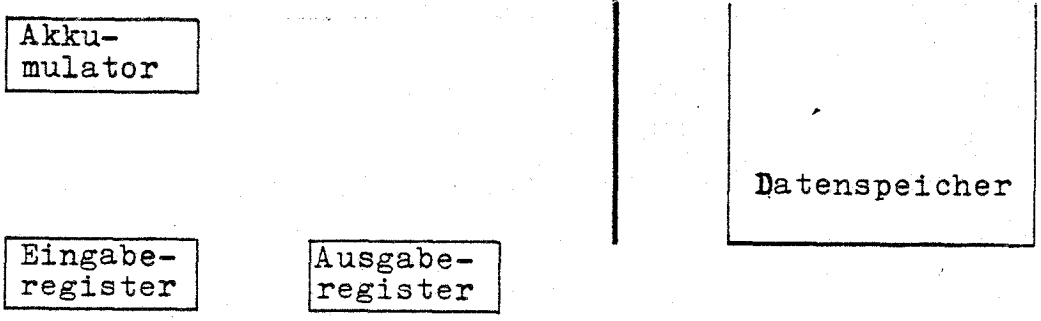
Was haben Sie beobachtet?
- Der Inhalt des Akkumulators wurde in den Datenspeicher in die Adressen 6 und 4 transferiert (gespeichert).

Tragen Sie nun die 4 Transferwege in die beiden Zeichnungen ein! EIN, AUS in Zeichnung 1) /LAD, SPE in Zeichnung 2)

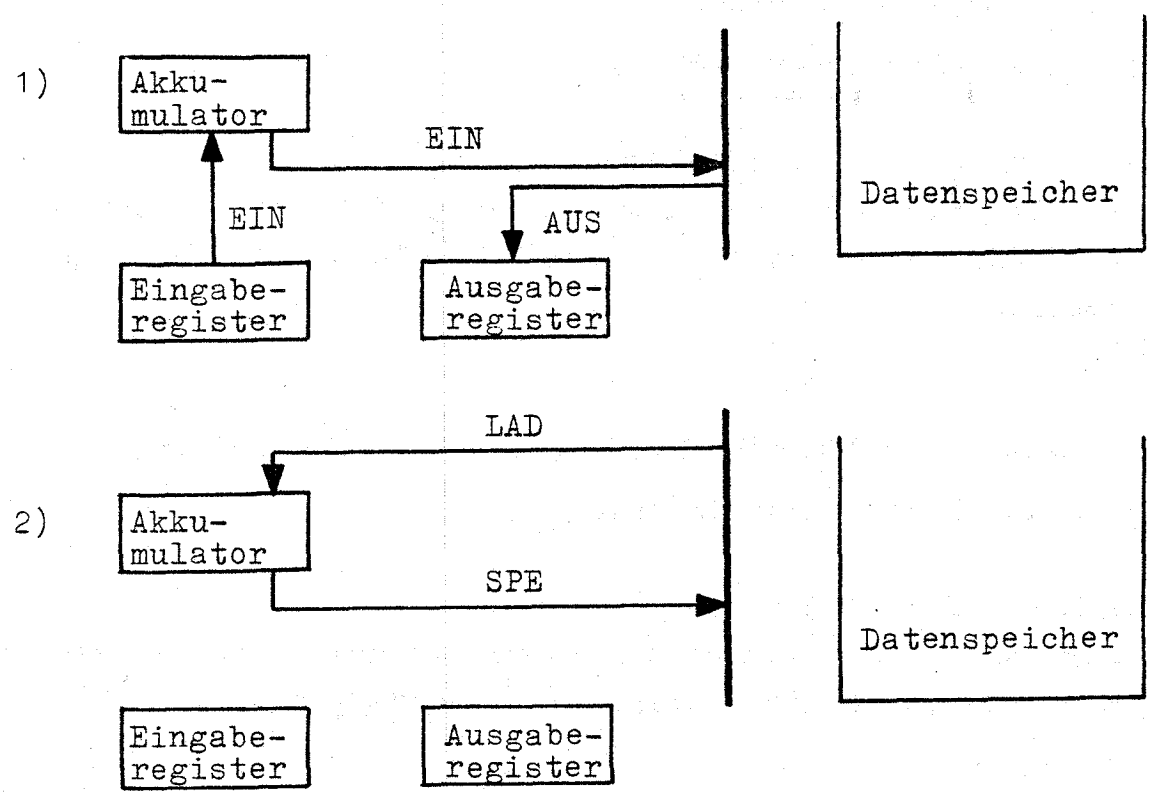
1)



2)



Lösung:



Nun stellen wir die Betriebsart SCHRITT ein, drücken Taste RÜCKSETZEN und führen alle Bedienungen nochmals durch. Wo jedoch bisher die Taste START einmal zu betätigen war, muß nun 4mal gedrückt werden, da nun ein Befehl in 4 Taktschritte unterteilt wird. Wir zählen die Taktschritte jeweils bis 4 und beobachten im Schritt 1 und 2 besonders das Steuerwerk. Die Schritte 3 und 4 vergleichen wir mit den vorherigen Ergebnissen.

Wie würden Sie die beiden Begriffe Befehlsbereitstellung und Befehlsausführung zu den Schritten 1 bis 4 zuordnen?

Befehlsbereitstellung = Schritt 1 und 2
Befehlsausführung = Schritt 3 und 4

Die Befehle stehen im Speicher. Es wird aber immer nur ein Befehl nach dem anderen ausgeführt.
Welche Funktionseinheit des Rechners veranlaßt diese Befehlsausführung?
- das Steuerwerk.

Allen Befehlen ist die nun folgende Befehlsbereitstellung gemeinsam:
Das Steuerwerk muß die Befehle lesen und interpretieren.

Welche Aufgabe hat dabei der Befehlszähler mit dem angeschlossenen Adreßdecodierer?
- Der Befehlszähler stößt den Lesevorgang an, indem er den nächsten zu bearbeitenden Befehl adressiert, auswählt, ansteuert.

Wohin wird die gelesene Information gebracht:
- in das Befehlsregister
  (Anmerkung: Ziehen Sie nach dem Taktschritt "1" Operations- und Adreßteil der angesprochenen Befehle und betrachten Sie das Befehlsregister! Sie sehen, daß die Operationsteile binär verschlüsselt sind,nach einem sog. Operationscode,und die Adreßteile als duale Zahlen vorliegen.)

Vom Befehlsregister aus kann der Befehl auf das Rechenwerk und die anderen Funktionseinheiten wirken. Dazu muß das binäre Befehlswort decodiert werden. Was bedeutet das?
- Der Rechner erkennt (entschlüsselt, decodiert) aus dem Operationscode die betroffene Operation und aus dem dualen Adreßcode, welche Adresse gemeint ist, und schaltet die zugehörigen Steuerleitungen für die folgende Befehlsausführung.

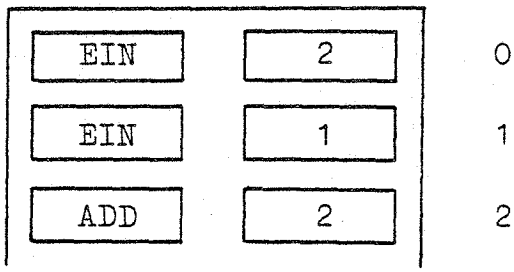
Die Erhöhung des Befehlszählers im Schritt 4 um 1, damit der nächste Befehl angesteuert werden kann, wird häufig auch noch zur Befehlsbereitstellung gezählt.

Während der Befehlsausführung werden die schon vorher gezeigten Transferoperationen durchgeführt.

Arithmetische Befehle

Die bekanntesten Verarbeitungsbefehle sind die arithmetischen Befehle (siehe auch unter Dualarithmetik). Im Modell vorhanden sind der Additionsbefehl "ADD w" und der Komplementierungsbefehl "KPL", der für die Subtraktion verwendet wird.

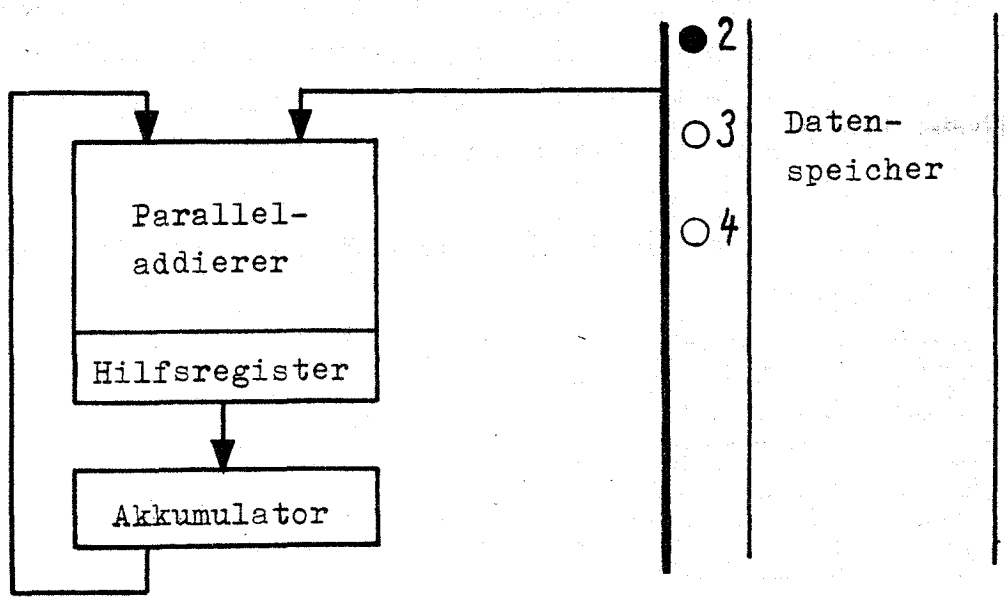
Stecken Sie die folgenden Befehle und führen Sie die angegebenen Bedienungen durch!



1) BEFEHL, RÜCKSETZEN,
   Kippschalter: OOII (duale 3),
   START

2) Kippschalter: OOIO (duale 2),
   START

3) SCHRITT, START, START,
   Beobachten Sie dabei die Befehlsbereitstellung für "ADD 2";
   START,
   Beobachten Sie das Hilfsregister;
   START,
   Beobachten Sie den Akkumulator!
   Vergleichen Sie die Abläufe der Befehlsausführung mit dem Schaltbild des Funktionsablaufs:



Was hat der Befehl "ADD 2" bewirkt?

- Der Inhalt der Adresse 2 wurde zum Akkumulatorinhalt addiert (im Hilfsregister wurde die Summe angezeigt und dann in den Akkumulator eingeschrieben).

Wiederholen Sie die Addition mit anderen Dualzahlen!

Anmerkung:

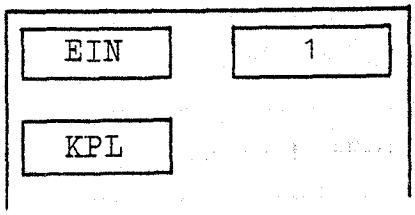
Es empfiehlt sich, die Addition als Beispiel eines grundsätzlichen Befehlsablaufs im Unterricht zu demonstrieren. Um dabei alle Vorgänge genau erläutern zu können, wurde eine Beschreibung des Befehlsablaufs, Addition, in der umseitigen Tabelle angegeben. Im Taktschritt 3 sollte dabei das Schaltbild "4-stelliges Paralleladdierwerk mit Überlaufbildung"auf die Frontplatte des Modells aufgebracht werden, um einerseits die Beschaltung (wie Bild oben) aufzuzeigen, und um andererseits Einblick in die zugehörige Hardware (Elektronik) nehmen zu können.

Erläuterung des Befehlsablaufs bei der Addition (ADD)

Betriebsart: SCHRITT
Hilfsmittel: Schaltbild "4-stelliges Paralleladdierwerk ..."

Bedienung	Takt	Vorgang
START	1	Befehl lesen. Der dem Befehlszählerstand entsprechende Befehl wird angesteuert und in das Befehlsregister eingeschrieben. Dabei signalisiert das Aufleuchten der Pfeile zum und vom Programmspeicher den Signalfluß. Das Befehlsregister enthält die binäre Darstellung des gelesenen Befehls.
START	2	Befehl decodieren. Der Befehlsregisterinhalt wird getrennt nach Operations- und Adreßteil decodiert. Das Durchschalten der decodierten Information wird durch die Leuchtpfeile vom Steuerwerk zum Rechenwerk und Datenspeicher signalisiert. Die angesprochene Operation leuchtet im Lampenfeld des Rechenwerks auf (in diesem Fall ADD). An der Adreßleiste des Datenspeichers leuchtet die angewählte Operandenadresse auf.
START	3	Befehl ausführen. Der Inhalt der angewählten Arbeitsspeicheradresse (der Operand) und der Inhalt des Akkumulators werden im Rechenwerk additiv verknüpft. Das Ergebnis erscheint im Hilfsregister. In diesem Stadium sind beide Summanden und das Ergebnis (Dualzahlen) sichtbar. (Zur Erläuterung kann das Schaltbild "4-stelliges Paralleladdierwerk" aufgelegt werden.)
START	4	Das Ergebnis wird vom Hilfsregister in den Akkumulator eingeschrieben (einer der Summanden wird überschrieben). Der Befehlszählerstand wird um 1 erhöht (entspr. der Adresse des folgenden Befehls).

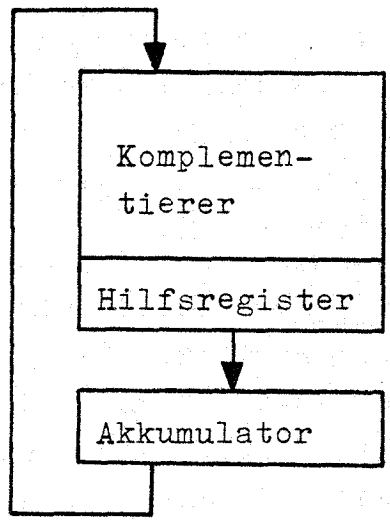
Stecken Sie nun folgende Befehle und führen Sie die angegebenen Bedienungen durch!



1) BEFEHL, RÜCKSETZEN
Kippschalter beliebig einstellen,
START

2) SCHRITT, START, START,
Befehlsbereitstellung,
START, START,

Beobachten Sie die Befehlsausführung für "KPL" im Hilfsregister und Akkumulator und vergleichen Sie diese mit dem Schaltbild des Funktionsablaufs:



Was hat der Befehl "KPL" bewirkt?
- Der Inhalt des Akkumulators wurde komplementiert (im Hilfsregister angezeigt und in den Akkumulator eingeschrieben).

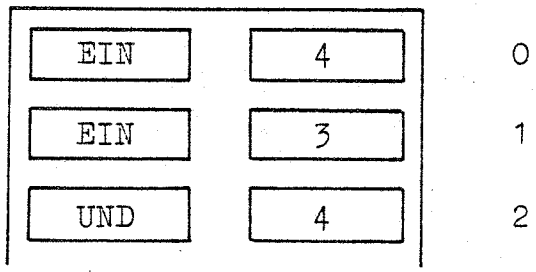
Wiederholen Sie die Komplementierung mit anderen Dualzahlen!

Anmerkung: Verwenden Sie das Schaltbild "Komplementierwerk", das Sie auf die Modellfrontplatte aufbringen können! Weitere Erläuterungen zur Komplementbildung finden Sie im Beispiel "Duale Subtraktion" auf Seite 110, das Sie auch vorab behandeln können.

Logische Befehle

Bei den arithmetischen Befehlen wurden Dualzahlen verarbeitet. Die logischen Befehle für die UND-, ODER-Verknüpfung beziehen sich auf Bitstellen. Es werten Bit 1 und Bit 1 oder Bit 3 und Bit 3 der angesprochenen Operanden für sich verknüpft.

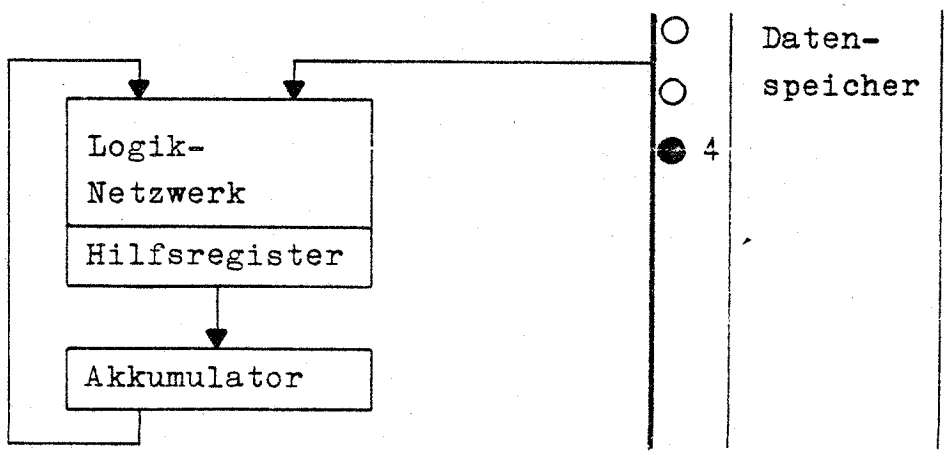
Stecken Sie die Befehle und führen Sie die Bedienungen wie angegeben durch:



1) BEFEHL, RÜCKSETZEN,
   Kippschalter: z.B. OOII,
   START;

2) Kippschalter: z.B. OIOI,
   START;

3) SCHRITT, START, START,
   Befehlsbereitstellung;
   START, START,
   Beobachten Sie die Befehlsausführung für "UND 4" und beobachten Sie dabei Hilfsregister und Akkumulator sowie den Inhalt der Adresse 4. Vergleichen Sie den Ablauf mit dem Schaltbild des Funktionsablaufs:



Was hat der Befehl "UND 4" bewirkt?

- Der Inhalt der Adresse 4 wurde mit dem Akkumulatorinhalt bitweise nach der logischen Funktion UND verknüpft. (Anzeige im Hilfsregister, Einschreibung in Akkumulator.)

Wiederholen Sie diesen Befehl mit anderen Kippschaltereinstellungen!

Ersetzen Sie nun die Operation UND im Programmspeicher durch ODR und führen Sie die gleichen Bedienungen durch!

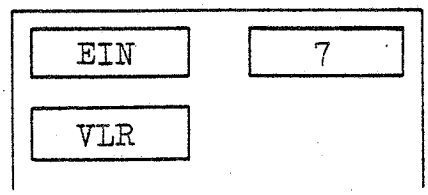
Anmerkung:

Verwenden Sie das Schaltbild "ODER-Verknüpfung", das Sie zur Erläuterung der binären Verknüpfung auf die Frontplatte des Modells aufbringen können. (Das Schaltbild "UND-Verknüpfung" kann aus dem vorhandenen Schaltbild "ODER-Verknüpfung" abgeleitet werden.)
Ein noch tieferes Eindringen in die logischen und elektronischen Vorgänge im Rechner bei den Befehlsabläufen erlauben die Schaltbilder "ODER-Verknüpfung (Logische Funktion im Rechner)" und "ODER-Verknüpfung (im Rechner)".
Nähere Erläuterungen finden Sie im Kapitel Grundfunktionen der Logik auf Seite 68 , das Sie unmittelbar anschließend erläutern könnten.

Verschiebebefehle

Die Verschiebebefehle (vergleiche Beispielgruppe "Schieberegister") beziehen sich auf Akkumulatorinhalte und sind Grundvoraussetzung für Multiplikation und Division.

Stecken Sie folgende Befehle und bedienen Sie wie angegeben:



1) BEFEHL, RÜCKSETZEN,
   Kippschalter beliebig einstellen,
   START,

2) SCHRITT, START, START,
   Befehlsbereitstellung;
   START, START,
   Befehlsausführung, beobachten Sie den Akkumulatorinhalt.

[oder: 2) BEFEHL, ANF 1, ANF 1, ANF 1]

Der Befehl VLR verschiebt den Inhalt des Akkumulators um 1 Bit nach rechts; von links werden Nullen nachgezogen.

Wiederholen Sie die Bedienungen mit anderen Zahlen (Kippschalter) und versichern Sie sich dessen, daß die Rechtsverschiebung einer Division durch 2 entspricht (Rundungsfehler!)

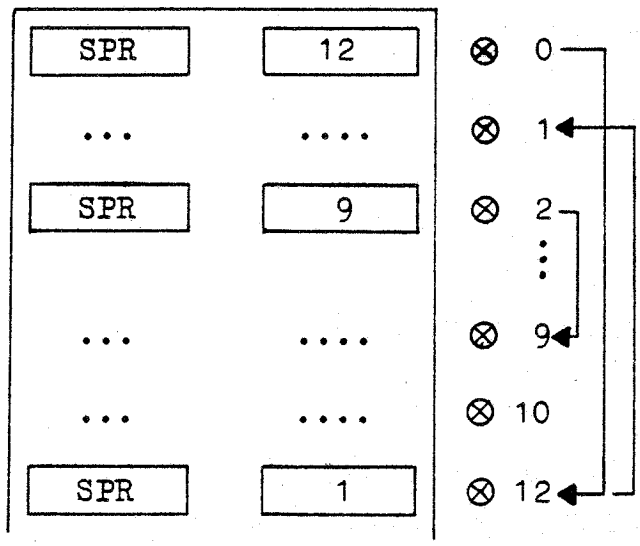
Ersetzen Sie den BEFEHL VLR durch VLL und wiederholen Sie alle Bedienungen für den Befehlsablauf "Linksverschiebung".

Anmerkung: Verwenden Sie die Frontplattenschaltbilder (Masken) "Rechtsschieberegister" und "Linksschieberegister"!

Sprungbefehle

Ein Programm wird normalerweise Befehl für Befehl in ansteigender Reihenfolge linear durchlaufen. Dieser Programmablauf kann durch die Sprungbefehle verändert werden. Neben dem unbedingten Sprungbefehl "SPR a" gibt es im Modell die bedingten Sprungbefehle: "SAM a", "SGN a" und "SUL a". ("a" ist die Fortsetzadresse des Programms, vergleiche Befehlsliste Seite 14)

Stecken Sie die folgenden Befehle und bedienen Sie wie angegeben:

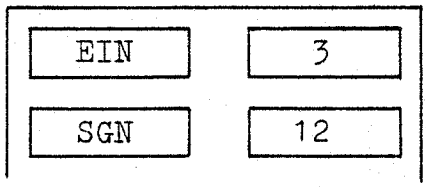


1) SCHRITT, RÜCKSETZEN,
   START, START,
   Befehlsbereitstellung;
   START, START,
   Beobachten Sie dabei die Befehlsausführung und achten Sie besonders auf Befehlszähler und Befehlsregister.

2) BEFEHL, START, START, START, START, ...
   Beobachten Sie dabei die Lampen des Programmspeichers.

Was hat der Befehl "SPR 12" bewirkt?
- Der Befehlszählerstand wurde (statt der üblichen Erhöhung um 1) auf den Wert der angegebenen Sprungadresse gesetzt. Das Programm wurde mit dem Befehl fortgesetzt, der durch die Sprungadresse (z.B. 12) vorgegeben worden war.

Stecken Sie nun folgende Befehle und bedienen Sie wieder wie angegeben:



1) BEFEHL,RÜCKSETZEN,
   Kippschalter: 00I0,
   START,
   Notieren Sie den Akkumulatorinhalt,
   START,

Warum wurde hier kein Sprung ausgeführt?
- Weil die Sprungbedingung: Springe falls Akkumulator gleich Null (SGN) nicht erfüllt war.

Wiederholen Sie diesen Befehl mit der Kippschaltereinstellung 0000!

Ersetzen Sie nun den Befehl SGN durch SAM und wiederholen Sie die Bedienungen mit den Kippschaltereinstellungen 0000, 0II0, und I0I0, II00!

Wann wird dieser Sprung ausgeführt?
- Wenn die Bedingung, Springe falls Akkumulator minus, erfüllt ist.

Stecken Sie nun folgende geänderte Befehle und bedienen Sie wie folgt:

EIN	0	0
ADD	0	1
SUL	12	2

1) BEFEHL, RÜCKSETZEN,
   Kippschalter: OOII,
   START, START,
   Notieren Sie den Inhalt des Überlaufregisters, "Ü", links vom Akkumulator

2) START
   Warum wurde kein Sprung ausgeführt?
   - Weil die Sprungbedingung, Springe falls Überlauf, nicht erfüllt war.

Wiederholen Sie nun die Bedienungen mit den Kippschaltereinstellungen OIII!
Was passiert mit dem Überlauf nach Ausführung des Sprungs?
- Er wird gelöscht (da er "bewußt" geworden ist).

Einführung in die Programmierung

1.) Umladen von Arbeitsspeicherzellen (Ergibt - Anweisung)

Aufgabenstellung: Eine Zahl, die in der Arbeitsspeicherzelle 1 steht, soll in die Zelle 3 umgespeichert werden.

Lösungsweg: Können die Adressen 1 und 3 gleichzeitig angesprochen werden?

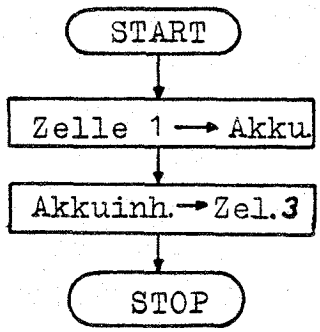
- nein

Welche Zwischenstation käme in Frage?

- der Akkumulator (Verarbeitungsregister)

Programm:

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse
0	LAD	1
1	SPE	3
2	STP	
(3	EIN	1)

Bedienung: Programm wie angegeben stecken
Betriebsart: BEFEHL;
RÜCKSETZEN, START

Der anschließende EIN-Befehl kann zur Eingabe neuer Binarzahlen dienen (Kippschalter einstellen!)

Lehrerdemonstration:

1. Die Adresse hat bei Transferbefehlen die Funktion einer Weiche.
2. Die Information in der Adresse 1 bleibt erhalten.
3. Ganz gleich, was in Adresse 3 stand, wird der Inhalt von Adresse 1 eingetragen. (Es braucht nicht gelöscht zu werden.)

Kommentar:

Die in einem Speicher (Arbeitsspeicher bzw. Zentral- oder Rechenspeicher) enthaltenen alten Programme und Daten brauchen im allgemeinen nicht gelöscht werden. Neue Programme werden nämlich wie im gegebenen Beispiel im Zuge des Ladens von Programmen im Speicher eingetragen. Ebenso werden die Ergebnisdaten in schon vorher beschriebene Zellen eingetragen. Dazu muß man sich erstens klarstellen, daß Programme bei üblichen Rechnern nicht mit Steckern eingespeichert werden, sondern daß Programme und Daten in der gleichen Form wie im Datenspeicher nebeneinander gespeichert werden. Zweitens muß man sich vor Augen halten, daß ein ordnungsgemäß ablaufendes Programm sich nur im Bereich des geladenen Programms bewegt und nur die definierten (eingegebenen) Daten verwendet. (Zu den definierten Zellen gehören auch die Ergebniszellen, die vom Assembler lediglich freigehalten werden, nicht aber gelöscht werden müssen.)

Anmerkung:

Die aus FORTRAN- und ALGOL-Programmen bekannte Ergibt-Anweisung (z.B. A = C + 1 bzw. A:= C + 1) entspricht einer Umspeicherung (C + 1 wird nach A gespeichert).

2.) Programmverzweigungen

Aufgabenstellung: Es wird eine Zahl eingegeben. Wenn diese Zahl positiv ist, soll das Programm mit dem folgenden Befehl fortgesetzt werden. Ist die Zahl aber negativ, so soll das Programm mit Befehlsadresse 9 fortgesetzt werden.

Lösungsweg: Wie kann ein Programm mit einem anderen als dem folgenden (n + 1) Befehl fortgesetzt werden?

- durch einen Sprungbefehl

Soll dieser Sprung in allen Fällen durchgeführt werden? Wenn nein, von welcher Bedingung hängt der Sprungbefehl ab?

- nein, nicht in allen Fällen
- die Bedingung heißt: die eingegebene Zahl ist negativ.

Welcher Befehl aus der Befehlsliste erfüllt diese Voraussetzung?

- der SAM-Befehl

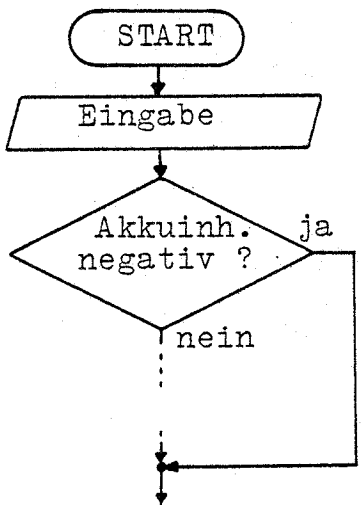
Wohin muß deshalb eine Zahl, von der die Bedingung abhängig ist, gebracht werden?

- in den Akkumulator

Führen Sie das folgende Programm zur Illustration einer Verzweigung durch und verwenden Sie bei der Eingabe positive und negative Zahlen!

Programm:

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse
0	EIN	A (z.B. 0)
1	SAM	9
9		

Bedienung: Programm wie angegeben stecken
Betriebsart: BEFEHL, PROGR. 8 Hz,
SCHRITT;
RÜCKSETZEN, mittels Kippschalter
neue Zahl einstellen, START

Lehrerdemonstration:

1.) In der Betriebsart BEFEHL kann verfolgt werden, daß die Sprungbedingung vom Akkumulatorinhalt abhängig ist.

2.) In der Betriebsart 8 Hz kann man alle möglichen Zahlen einstellen, jeweils tabellarisch aufzeichnen, wann ein Sprung erfolgte und so die Bedingung, Akkumulator minus, genau definieren:

- allen negativen Zahlen ist die Eins in der ersten Bitstelle (links) gemeinsam. (Achtung: negative Zahlen sind komplementierte Zahlen!)
- der SAM-Befehl ist damit ein Bit-Testbefehl.

Anmerkung: Bei 8 Hz kommt das Programm entweder bei Befehlsadresse 3 oder 10 zum Stehen. Das liegt daran, daß ein nicht gesteckter Befehl einem STP-Befehl entspricht, der noch bearbeitet wird, und mit einer letzten Erhöhung des Befehlszählers um 1 abschließt.

3.) Mit Hilfe der Betriebsart SCHRITT kann demonstriert werden, wie der Programmsprung technisch realisiert wird; nämlich durch Einschreiben der vorgegebenen Sprungadresse (im Adreßteil des Befehlswortes) in den Befehlszähler

4.) In allen Fällen muß darauf hingewiesen werden, daß der bedingte Sprungbefehl eine Programmverzweigung in zwei verschiedene Richtungen ermöglicht und eine Entscheidung über den einzuschlagenden Programmablauf fällt. Diese Entscheidung ist abhängig von einer Bedingung, die im Gegensatz zu unserem einfachen Beispiel im allgemeinen erst in einem vorangehenden Programmteil entsteht.

Kommentar: In unserem einfachen Beispiel wurde nur die Verzweigung selbst gezeigt. Für den Programmierer entsteht bei einem bedingten Sprungbefehl in jedem Fall die Aufgabe, beide möglichen Programmzweige zu bedenken und zu programmieren. Derartige Erweiterungen des Beispiels könnten sein:

1.) Abspeicherung der positiven Zahlen nach B (Adr. 2) und der negativen nach C (Adr. 3)

Lösung:

```
    0   EIN   A    (z.B.: 0)
    1   SAM   9
    2   SPE   2    (B)
    3   STP
    9   SPE   3    (C)
   10   STP
```

2a) Bildung und Ausgabe des Absolutbetrags der eingegebenen Zahl.

Lösung:

```
    0   EIN   A    (z.B. 0)
    1   SAM   9
    2   AUS   A
    3   STP
    9   KPL
   10   SPE   A
   11   AUS   A
        STP
```

2b) Zwischenfrage: Wie könnte diese Lösung vereinfacht - um einen Befehl verkürzt - werden?

```
    0   EIN   A
    1   SAM   9
    2   AUS   A
    3   STP
    9   KPL
   10   SPE   A
   11   SPR   2
```

3a) Erweiterung der Verzweigungen auf drei unter Einschluß der zusätzlichen Bedingung: Falls die eingegebene Zahl gleich Null ist, soll nach 16 gesprungen werden

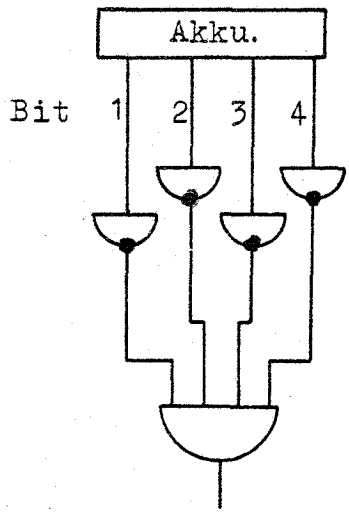
Lösung:

```
0    EIN   A
1    SAM   9
2    SGN   16
3          ←— Zahl ist positiv
:
9          ←— Zahl ist negativ
:
16         ←— Zahl ist gleich Null
```

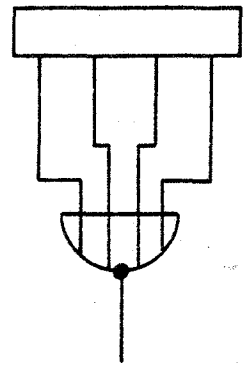
3b) Wie könnte eine Schaltung aus Verknüpfungsgliedern aufgebaut sein, welche die Bedingung, Akkumulator gleich Null, feststellt? (Diese Frage kann nur dann gestellt werden, wenn gute Kenntnisse der Booleschen Algebra vorhanden sind)

Lösung:

1. Schaltung



2. Schaltung



Erläuterung der Umformung zur 2. Schaltung:

$\overline{A} \wedge \overline{B} \wedge \overline{C} \wedge \overline{D}$ = Bedingung = $\overline{(\overline{A}\ \overline{B}\ \overline{C}\ \overline{D})}$

$= \overline{A \vee B \vee C \vee D}$

Umformungsregeln:

zwei Negation ändern eine Funktion nicht,

$\overline{\overline{A}} = A$; $\overline{A \wedge B} = \overline{A} \vee \overline{B}$; $\overline{A \vee B} = \overline{A} \wedge \overline{B}$

3.) Größenvergleich zweier Zahlen

Aufgabenstellung: In den Arbeitsspeicherzellen 4 und 5 stehen zwei positive Zahlen A und B. Speichere die größere der beiden Zahlen in Zelle C (Adresse 6) ab.

Lösungsweg: Auf welche arithmetische Operation führt ein Größenvergleich zweier Zahlen?

- auf die Subtraktion.

Geben Sie die Zuordnungen durch Unterstreichen an:

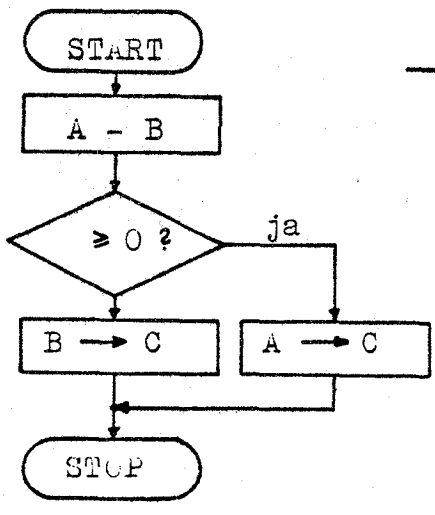
A - B = positiv, also ist A größer/kleiner B
A - B = negativ, also ist A größer/kleiner B

Wie kann der Rechner die unterschiedlichen Ergebnisse der Subtraktion verwerten?

- durch die bedingten Sprungbefehle.

Programm:

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse	
		dez.	symbol.
0	LAD	5	B
1	KPL		
2	ADD	4	A
3	SAM	7	
4	LAD	4	A
5	SPE	6	C
6	STP		
7	LAD	5	B
8	SPR	5	

Bedienung: Wie in Beispiel 2

Lehrerdemonstration: Diese Aufgabe kann nur dann vorgeführt werden, wenn die beiden Zahlen A und B vor jedem Programmlauf eingegeben werden:

```
 0   EIN   4   (A)
 1   EIN   5   (B)
 2   LAD   5
 3   KPL
 4   ADD   4
 5   SAM   9
 6   LAD   4
 7   SPE   6
 8   STP
 9   LAD   5
10   SPR   7
```

Soll von Anfang an mit den Betriebsarten PROGR. gearbeitet werden, die automatischen Ablauf bis zu einem STP-Befehl gewährleisten, so müssen solche STP-Befehle eingeschoben bzw. Steckplätze ausgelassen werden

z.B.

```
 0   EIN   4
 1   EIN   5
 2   ...
 3   LAD   5
```

In diesem Fall würde man zunächst die Zahl A mittels Kippschalter einstellen und ANF0 drücken; dann B einstellen und ANF1 drücken

oder z.B.

```
0   EIN   4
1   ...
2   EIN   5
3   ...
4   LAD   5
5   KPL
6
```

Jetzt kann wie üblich mit den Tasten RÜCKSETZEN und START gearbeitet werden und vor jedem START die gewünschte Zahl eingestellt werden.

Kommentar: Die vorliegende Aufgabe ist Voraussetzung für das Verständnis der ab Seite folgenden Sortieraufgaben.

Anmerkung: Das Beispiel setzt die Kenntnis der dualen Subtraktion voraus, die aus formalen Gründen jedoch erst später, auf Seite 110, folgt.

4.) Ausblenden von Bitstellen durch Masken (Intersektion)

Aufgabenstellung: Untersuchung einer eingegebenen Zahl A (0) auf Geradzahligkeit. Geradzahliges A soll nach B (2), ungeradzahliges A nach C (3) abgespeichert werden.

Lösungsweg: Schreiben Sie die dezimalen Zahlen 0 bis 6 mit den zugehörigen Dualzahlen untereinander und versuchen Sie eine binäre Gemeinsamkeit aller gerader Zahlen herauszufinden!

0	0	gerade
1	I	ungerade
2	I0	gerade
3	II	ungerade
4	I00	gerade
5	I0I	ungerade
6	II0	gerade

- alle geraden Zahlen enthalten in der letzten Bitstelle eine 0

Daraus folgt, es genügt die letzte Bitstelle zu betrachten.

Was muß mit den Bitstellen 1 bis 3 der geraden Zahl 0II0 geschehen, damit die Geradzahligkeit mit Hilfe des SGN-Befehls erkannt werden kann?

- die Bitstellen 1 bis 3 müssen gelöscht werden.

Wie kann mit Hilfe des UND-Befehls Bitstelle 1 bis 3 gelöscht, Bitstelle 4 aber unverändert (Maske für Biststelle 4) erhalten werden?

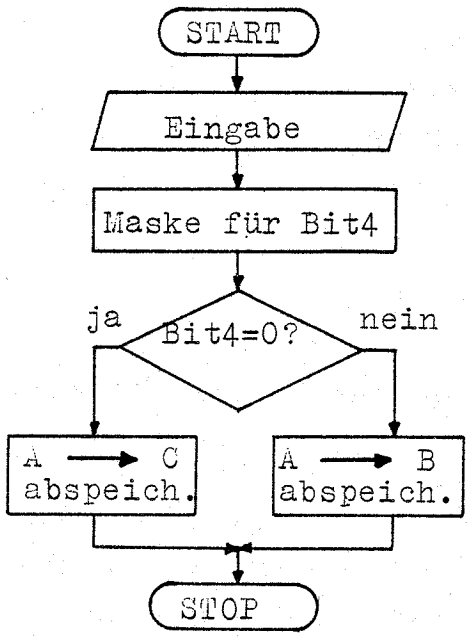
- der Operand des UND-Befehls ist an der Bitstelle 4 (an jeder auszublendenden Bitstelle) mit I zu besetzen. Da alle anderen Bitstellen mit 0 besetzt sind, muß die UND-Verknüpfung dort zum Ergebnis 0 führen.

z.B.:

	0II0	0III
∧	000I	∧ 000I
	0000	000I

Lösung:

Programmablaufplan



Befehlsfolge

Befehls-Adr.	Operation	Adresse	
Kippschalter für A bzw. auch für 000I einstellen			
0	EIN	0	(A)
		(1)	(000I)
1	UND	1	(000I)
2	SGN	6	
3	LAD	0	
4	SPE	2	
5	SPR	8	
6	LAD	0	
7	SPE	3	
8	STP		

Bedienung:

1.) Programm wie angegeben stecken, jedoch statt EIN 0, EIN 1 stecken
Betriebsart: BEFEHL;
RÜCKSETZEN,

Kippschalter auf 000I stellen, START drücken. Damit wird die Maske für die UND-Verknüpfung eingegeben.

2.) EIN 1 wieder korrigieren: EIN 0, RÜCKSETZEN,
beliebige Zahl mittels Kippschalter einstellen,
Befehl für Befehl mit START durchtasten und Programm verfolgen bzw. vorführen.

Lehrerdemonstration:

Darstellung des UND-Befehls in der Betriebsart SCHRITT. Erläuterung seiner Anwendungsmöglichkeit: Soll eine Reihe beliebiger Bitstellen isoliert betrachtet werden, so kann dies geschehen, indem man sie mit einer Maske (UND-Befehl, dessen Operand an den betreffenden Bitstellen mit I besetzt ist) ausblendet.

Kommentar:

Die Betrachtung isolierter Bitstellen ist nur in Assembler (Maschinensprache) möglich. Sie wird vor allem in der Programmierung von Prozeßaufgaben angewendet. Zum Beispiel kann jeder Bitstelle eines Eingabeworts (16, 24, 32 Bitstellen) ein Schaltzustand im Prozeß (Produktionseinrichtung oder ähnliches) zugeordnet sein. In einem Gefahrenzustand kann dann durch Ausblenden einer oder mehrerer Bitstellen ein besonders kritischer Schaltzustand vorab rasch abgefragt werden.
Da bei Prozeßaufgaben eine Rückwirkung auf den Prozeß durch Stellbefehle erforderlich ist, muß die geschilderte erste Aufgabe durch das definierte Setzen von Bitstellen ergänzt werden. Dies kann durch den ODER-Befehl erfolgen:

Es sei z.B. folgende Aufgabenstellung zu bearbeiten:
Das Eingaberegister ist an Signalgeber angeschlossen, die folgende Reaktion des Rechners durch Angabe von Stellbefehlen über das Ausgaberegister erfordern.

Eingaberegister	Ausgaberegister	
0000	0000	unabhängig vom vorhergehenden Zustand
I000	Ixxx	x ≙ vorheriger Zustand
0I00	xI00	in unveränderter
00I0	xxIx	Form
000I	xxxI	

Lösung

0	EIN	0
1	SGN	6
2	ODR	1
3	SPE	1
4	AUS	1
5	STP	
6	SPE	1
7	SPR	4

Durch die Funktion ODER werden Bitstellen, die im Operanden der ODER-Funktion mit 0 besetzt sind, unverändert gelassen. Alle mit I besetzten Bitstellen werden in den anderen Operanden eingetragen.

z.B.

	0II0	II0I
v	000I	v 000I
	0III	II0I

Anwendungen der hier beschriebenen Prinzipien finden Sie in den Prozeßrechnerbeispielen ab Seite 149 .

5.) Schleifenprogrammierung

Aufgabenstellung: Ein gegebener Verarbeitungsalgorithmus soll mehrmals durchlaufen werden.

Lösungsweg:

1.) Stecken Sie den Befehl [EIN] [0] 0, stellen Sie die Kippschalter auf φφφφ und geben Sie diese Zahl (1) ein (Betriebsart: BEFEHL; Tasten: RÜCK-SETZEN, START)

2.) Ersetzen Sie diesen Befehl durch die Befehle:

[ADD]	[0]	0
[SPE]	[1]	1
[STP]		2

Stellen Sie jetzt die Betriebsart PROGR. 8 Hz ein und führen Sie dieses Programm durch (RÜCKSETZEN, START)

Was müssen Sie tun, um dieses Programm zu wiederholen und um damit im Akkumulator dual zu zählen?

- Taste RÜCKSETZEN drücken und START

Welchen Befehl müßte man nach dem STP-Befehl stecken, um sich das RÜCKSETZEN zu ersparen?

- den Sprungbefehl [SPR] [0]
  (Springe nach Befehlsadresse 0)

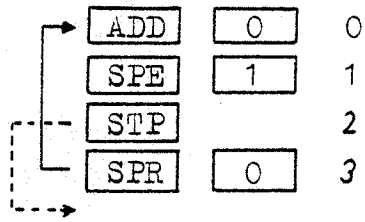
Führen Sie auch das folgende Beispiel durch:

[ADD]	[0]	0
[SPE]	[1]	1
[SPR]	[0]	2

Welchen Unterschied stellen Sie fest und warum muß diese Schleife bei praktischen Programmen unbedingt vermieden werden?

- das Programm läuft endlos und kann nur durch äußeren Eingriff (STOP-Taste) angehalten werden.
- Solche Schleifen führen zu einer Blockierung des Rechners und können von außen schwer erkannt werden. Es ist deshalb üblich, Programme, die nach einem Vielfachen der geschätzten Rechenzeit nicht beendet sind, abzubrechen, da eine endlose Schleife vermutet werden muß.

Stecken Sie Ihr Programm wieder um:



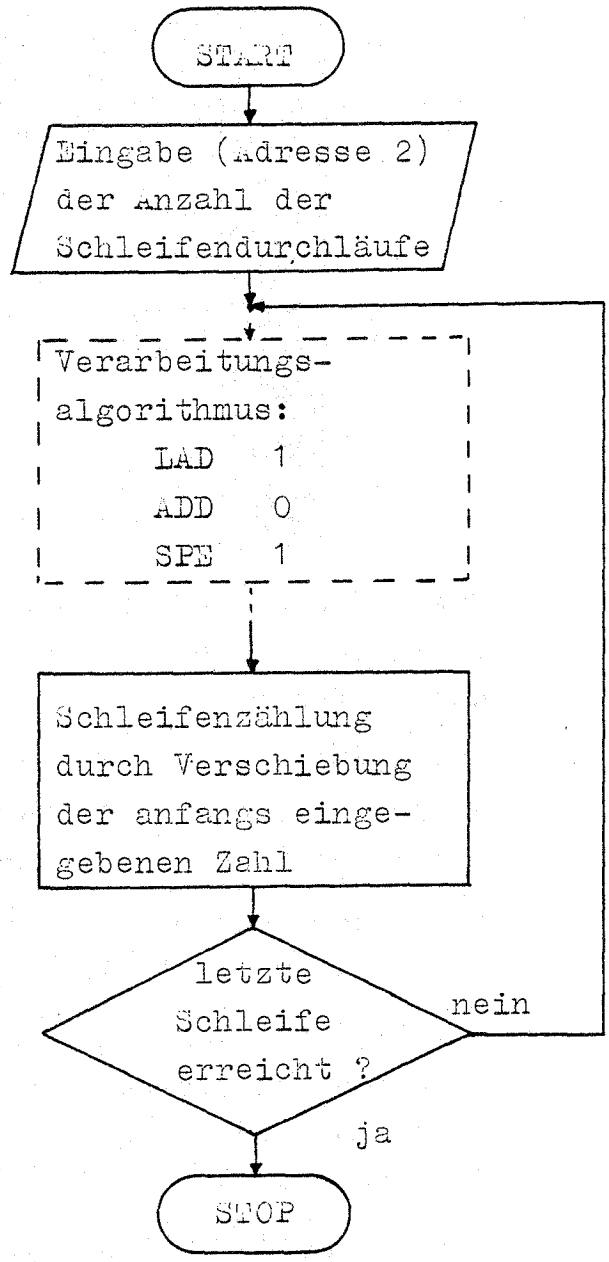
Durch welche Befehle könnte der STP-Befehl ersetzt werden, um zu erreichen, daß das Programm nach einigen Durchläufen aus der Schleife herausspringt?

- durch die bedingten Sprungbefehle:

  SUL, SGN, SAM

Führen Sie das Beispiel mit diesen Befehlen jeweils durch.

3.) Das folgende Beispiel, das Sie an Hand eines Blockdiagramms (Programmablaufplan) programmieren sollen, hat folgenden Zweck:
Eine Schleife soll 1 bis 4 mal, je nach Eingabe wählbar, durchlaufen werden.



Hilfestellung: Stellen Sie sich die Links- oder Rechtsverschiebung der Binärzahl IIII vor:

IIII → IIIO → IIOO → IOOO → 0000

bzw.: IIII → OIII → OOII → OOOI → 0000

Lösung: Aus der Vielzahl von Lösungsmöglichkeiten stellen wir folgende zwei vor:

1.)

0	EIN	2
1	LAD	1
2	ADD	0
3	SPE	1
4	LAD	2
5	VLL	
6	SPE	2
7	SAM	2
8	STP	
9	EIN	0
10	EIN	1

Die einzugebende Verschiebezahl für die wie folgt aufgelistete Anzahl von Schleifendurchläufen lautet:

4 IIII
3 .... III0
2 .... II00
1 .... I000

Eingabe der Zahlen A und B für C = A + B + B + B + ...

Wie muß diese Lösung erweitert werden, damit im Fall der Eingabe von 0000 kein Schleifenlauf durchgeführt wird?

- Einschiebung des Befehls SGN 9 zwischen Befehlsadresse 0 und 1 (Achtung: Übriges Programm verschiebt sich um 1, ebenso Sprungadressen).

Warum darf der Befehl SPE 2 nach der Verschiebung nicht fehlen?

- Weil der Akkumulator nicht zur Speicherung des Zwischenergebnisses der Verschiebung geeignet ist. Das Zwischenergebnis geht bei Folgeoperationen verloren.

2.)

0	EIN	2
1	LAD	1
2	ADD	0
3	SPE	1
4	LAD	2
5	VLR	
6	SPE	2
7	SGN	9
8	SPR	1
9	STP	
10	EIN	0
11	EIN	1

Die einzugebende Verschiebezahl für verschiedene Schleifendurchläufe lautet hier:

4 .... I000
3 .... 0I00
2 .... 00I0
1 .... 000I

Eingabe von A und B für C = A + B + B + ...

Auch hier kann die Aufgabe mit dem Befehl

1 [SGN] [10] erweitert werden.

Lehrerdemonstration:

Wie üblich kann der Programmablauf am besten in der Betriebsart BEFEHL erarbeitet werden. Die Wirkung der im Eingaberegister (Kippschalter) neugewählten Verschiebezahlen (Zyklenanzahl) kann am besten in Betriebsart PROGR. 8 Hz verfolgt werden. Es ist besonders wesentlich, darauf hinzuweisen, daß diese Zyklenanzahl im allgemeinen durch einen vorangehenden Programmabschnitt nach den im Programm sich ergebenden Notwendigkeiten selbst eingestellt wird. Oder aber ein vorhandenes Programm wird durch Änderung der Eingabedaten (auf Lochkarten) variiert.

Erweiterung:

Die Schleifenzählung durch Verschiebung hat natürlich ihre Grenzen in der Anzahl der Bits je Wort. Im allgemeinen wird für Schleifenprogramme deshalb ein echter Zähler programmiert und auf einen Endstand abgefragt.

Annahme: Ihr Zähler: { LAD A / ADD EINS / SPE A } habe zum Beispiel bis 5 gezählt, was Ihrem letzten Schleifenlauf entsprechen möge. Wie könnten Sie diesen Zustand , die Zahl 5 steht im Akku, mit dem Befehl SGN abfragen?

- Subtraktion einer 5
  und Abfrage auf Null.

Diese Subtraktion muß in jedem Schleifenlauf durchgeführt werden, ohne daß die eigentliche Zählzelle beeinflußt wird.

Statt einer Subtraktion über die Komplementbildung kann auch eine Addition von -5 durchgeführt werden. Geben Sie die Lösung an!

- <u>1. Möglichkeit</u>

```
ANFANG
        LAD   A
        ADD   EINS
        SPE   A
        ADD   MINUS 5
        SGN   ENDE
        SPR   ANFANG
ENDE
```

- <u>2. Möglichkeit</u>

```
        LAD   A
        ADD   EINS
        SPE   A
        KPL
        ADD   FÜNF
        SGN
        SPR
```

Die beiden Befehle { SGN / SPR } könnten durch den Befehl SUN (Springe falls ungleich Null) ersetzt werden, der im Modell jedoch nicht enthalten ist.

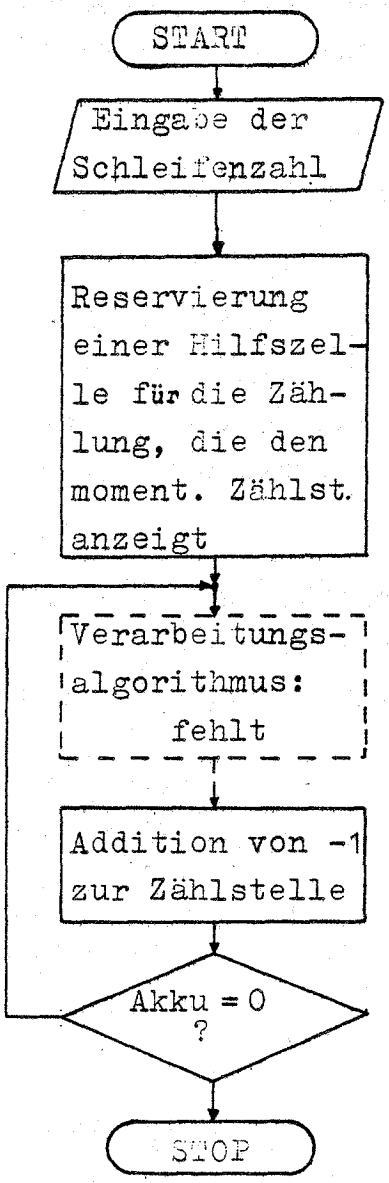
Können Sie sich eine Umkehrung dieses Zählvorgangs vorstellen, welche die Subtraktion von 5 vermeidet und die Programmierung vereinfacht? Geben Sie diese Möglichkeit an!

- Die Anzahl der Schleifenläufe wird als Zahl eingegeben. Von dieser Zahl wird bei jedem Schleifenlauf 1 abgezogen und auf Null abgefragt.

Stellen Sie dieses Programm auf!

Lösung:

Programmablaufplan:



Befehlsfolge

Befehlsadresse	Operation	Adresse dez.	Adresse Inhalt
0	EIN	0	z.B. 5 (QIOI) *
1	SPE	1	HZ
2	** LAD	1	HZ
3	ADD	2	-1 (IIII) ***
4	SPE	1	
5	SGN	7	
6	SPR	2	
7	STP		

* Die Schleifenzahl muß positiv sein (d.h. 1. Bit = 0)

** dieser Befehl ist im vorliegenden Programm unnötig. Wäre jedoch ein Verarbeitungsteil zwischengelagert, so könnte nicht mehr darauf verzichtet werden.

*** Achtung. Die Zahl -1 (IIII) muß vor dem Programmlauf eingegeben werden. Eventuell mittels Umstecken: EIN 2 statt EIN 0 oder Anfügen von EIN 2 am Programmende.

Kommentar:

Schleifen dieser Art können durchaus ineinander geschachtelt sein.
In der problemorientierten Sprache FORTRAN kann eine solche Schleife mit einer einzigen Anweisung programmiert werden. Diese Anweisung, die Schleifenanweisung lautet:

DO n i = m_1, m_2, m_3

z.B. DO 7 I = 1, 5, 1

Aus den Definitionen der Parameter n, i, m_1, m_2, m_3 kann man sofort ersehen, daß diese Anweisung in ein Schleifenprogramm ähnlich wie beim gegebenen Beispiel vom Compiler (Übersetzungsprogramm) umgesetzt werden muß.

n = Nummer der nach Abschluß der gesamten Schleife folgenden Anweisung
I = Index (z.B. ist beim 3 Schleifenlauf der Index gleich 3)
m_1 = Indexanfangswert (1. Schleifenlauf)
m_2 = Indexendwert (letzter Schleifenlauf)
m_3 = Schrittweite

Daraus folgt: Die Anzahl der Schleifenläufe (Iterationen)

$$= \frac{m_2 - m_1}{m_3} + 1$$

In der problemorientierten Sprache ALGOL lautet die Schleifenanweisung

'FOR' i = n_1 'STEP' m 'UNTIL' n_2 'DO'
z.B. 'FOR' I = 1 'STFP' 1 'UNTIL' 5 'DO'

6.) Indirekter Zugriff auf eine Adresse

Aufgabenstellung: Zum Inhalt der Zelle 1 soll diejenige Zahl addiert werden, deren Adresse in der Hilfszelle 6 steht.

Lösungsweg: Von welchem Platz im Rechner werden die Arbeitsspeicheradressen angesteuert?

- vom Adreßteil des Befehlsregisters.

Wie kann der Inhalt einer Speicherzelle zur Adresse werden?

- indem man ihn in den Adreßteil des Befehlsregisters bringt.

Wann muß dieser Transfer auf jeden Fall abgeschlossen sein?

- vor dem Beginn der eigentlichen Operation (Befehl).

Bei der hier angesprochenen Substitution (indirekter Speicherzugriff) - wenn ein Substitutionsbit gesetzt ist - wird der Inhalt der direkt angesprochenen Adresse in den Adreßteil des Befehlsregisters gebracht, so daß der betreffende Befehl nicht mit der direkt angesprochenen Adresse abläuft, sondern mit derjenigen Adresse, die in der direkt angesprochenen Adresse enthalten ist.

Programm:

Befehls-Adr.	Operation	Adresse
0	EIN	1 bzw. LAD 1
1	ADD	(6)

Kennzeichen der Substitution

Lehrerdemonstration: Vorführung substituierter Befehle und Erläuterung des Wirkungsablaufs. Besonders gut dafür geeignet ist die Betriebsart "SCHRITT". Der Befehl Nr. 1 kann natürlich auch variiert und durch andere Befehle ersetzt werden:

1	ADD	(6)	
"	"	(2)	
"	"	(7)	
1	SPE	(0)	
1	AUS	(0)	
1	UND	(0)	usw.

Kommentar: Sinnvoll wird dieses Programm durch Anfügen einer Schleife in welcher der Inhalt der Adresse 6 selbst verändert wird, so daß in jedem Schleifenlauf eine andere Adresse bearbeitet (hier: addiert) wird. Z.B. wäre folgender Anhang denkbar

```
:
LAD     6
ADD     EINS
SPE     6
bedingter Sprung
```

(Vergleiche z.B. "Mathematisches Beispiel 2")
Damit wurde das Thema Adreßrechnung bzw. Indizierung angeschnitten.

7.) Allgemeine Hinweise zur Aufstellung und zum Ablauf von Programmen

Symbolische Programmierung:

Es empfiehlt sich, zunächst mit symbolischen Adressen zu programmieren, z.B.:

```
        LAD   A
        ADD   K3      (Konstante 3)
        SAM   VERZW   (Verzweigung)
        SPE   SUMME
VERZW   KPL
        SPE   ABS     (Absolutwert)
```

Nun verfahren wir so, wie ein Übersetzer (Assembler) für ein Maschinenprogramm, das mit symbolischen Adressen geschrieben ist, verfahren würde. Wir weisen den symbolischen Adressen absolute, dezimale Adressen zu. Das Umsetzen in die duale (bzw. binärcodierte) Form braucht hier nicht mehr durchgeführt werden, da die beschrifteten Stecker (Programmierbausteine) diese Aufgabe automatisch vornehmen.

Die symbolisch angesprochenen Daten: A, K3, SUMME und ABS werden demnach auf den Arbeitsspeicher verteilt, was am besten nach der folgenden Tabelle erfolgt.

Arbeitsspeicherbelegung:

Adresse		Inahlt	
dezimal	symbolisch	dezimal	dual
0	A	bel.	bel.
1	K 3	3	00II
2	SUMME	bel.	bel.
3	ABS	bel.	bel.
4			
7			

Die so gewonnenen dezimalen Adressen z.B. 2 für SUMME werden in das Programm eingetragen.
Außerdem werden die fortlaufenden Befehlsspeicheradressen 0 ... 31 eingetragen und die symbolischen Sprungadressen - in diesem Fall VERZW - durch diese dezimalen Adressen ersetzt. Gesteckt wird nur der umrahmte Teil.

0		LAD	A	0
1		ADD	K 3	1
2		SAM	VERZW	12
3		SPE	SUMME	2
12	VERZW	KPL		
13		SPE	ABS	3

Dateneingabe:

Bei dem angegebenen Beispiel muß vor dem eigentlichen Programmlauf die Konstante 3 (K 3) eingegeben werden. Die Eingabe kann durch normale EIN-Befehle mit dezimaler

Adresse erfolgen. Besser ist die Eingabe mit dem Eingabeprogramm der Eingabetastatur, daß auch ohne Tastatur leicht vollzogen werden kann.

Eingabeprogramm

Die ersten 3 Befehle des Befehlsspeichers bleiben immer reserviert für:

0	EIN	7	Adresse
1	EIN	(7)	Inhalt
2	STP		

1.) Zunächst wird die Betriebsart PROGR 40 kHz eingestellt und die gewünschte Adresse (0 ... 7) mit der Kippschaltereingabe eingestellt. Durch Drücken der Taste "ANFO" wird die Adresse eingeschrieben.

2.) Dann wird der gewünschte Inhalt an den Kippschaltern eingestellt und die Taste "ANF1" gedrückt. Damit wird der Inhalt in die vorgewählte Arbeitsspeicheradresse eingeschrieben.

Es gilt lediglich die Einschränkung: Soll eine Eingabe in die Zelle 7 erfolgen, so muß diese Eingabe immer erst als letzte erfolgen.

Grundfunktionen der Logik

Aufgabenstellung:

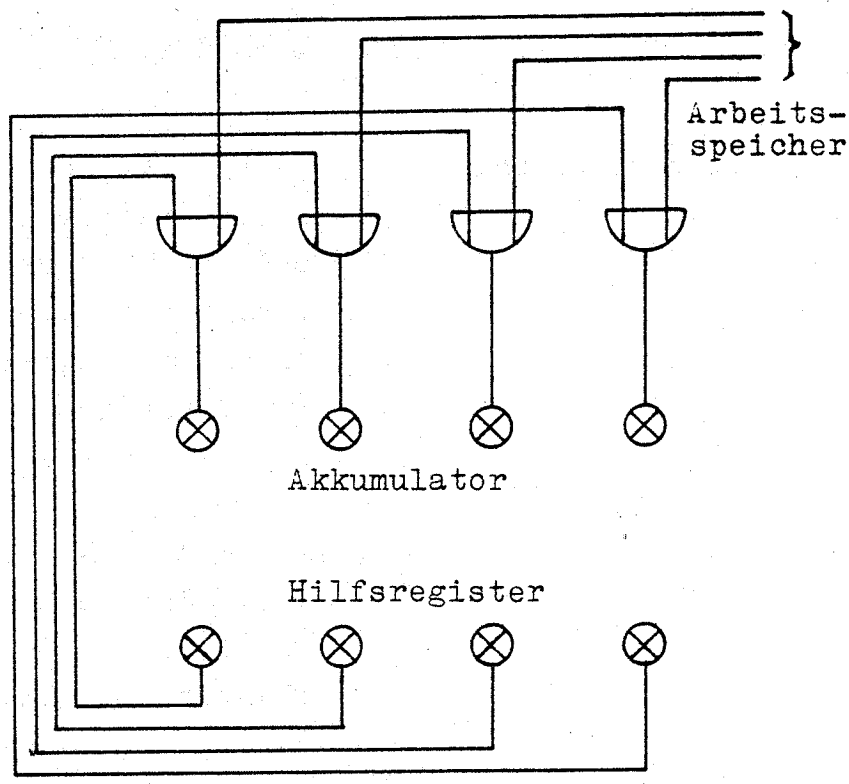
Die Wirkung der logischen Grundfunktionen Negation, UND-Verknüpfung, ODER- Verknüpfung auf binäre Eingangssignale soll demonstriert werden, um daraus die vollständige Wertetabelle (Zuordnung von Eingangs- zu Ausgangssignalen) dieser Funktionen abzuleiten.

Die technische Realisierung dieser Funktionen soll durch die symbolischen Darstellungen der Verknüpfungsglieder erläutert werden.

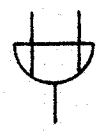
Lösungsweg:

Die logischen Verknüpfungen UND und ODER werden durch die Befehle UND und ODR des Rechners ausgelöst. Die Negation wird durch Kombination der Befehle KPL und ADD "-1" nachgebildet (Die Komplementbildung entspricht einer Negation mit nachgeschalteter Addition von eins, die mit der Addition von -1 rückgängig gemacht werden muß)
Zur Erläuterung der Symbole der logischen Verknüpfungsglieder werden vorgefertigte Schaubilder mit Ausschnitten (Masken) für die Lämpchen auf die Rechnerfrontseite aufgebracht. Damit werden Eingangs- und Ausgangssignale durch Lämpchen angezeigt und die technische Schaltung (als Logikplan) dargestellt.

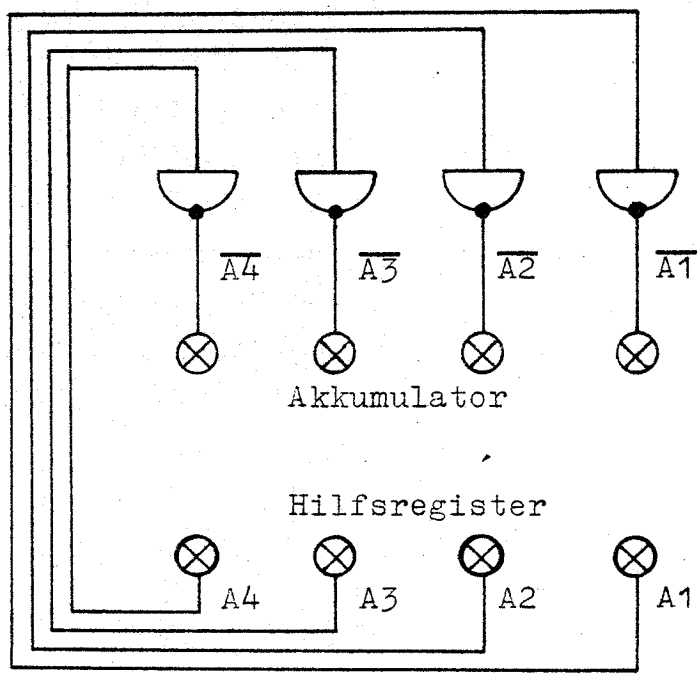
ODER:



Für die UND-Verknüpfung werden lediglich die Symbole geändert:

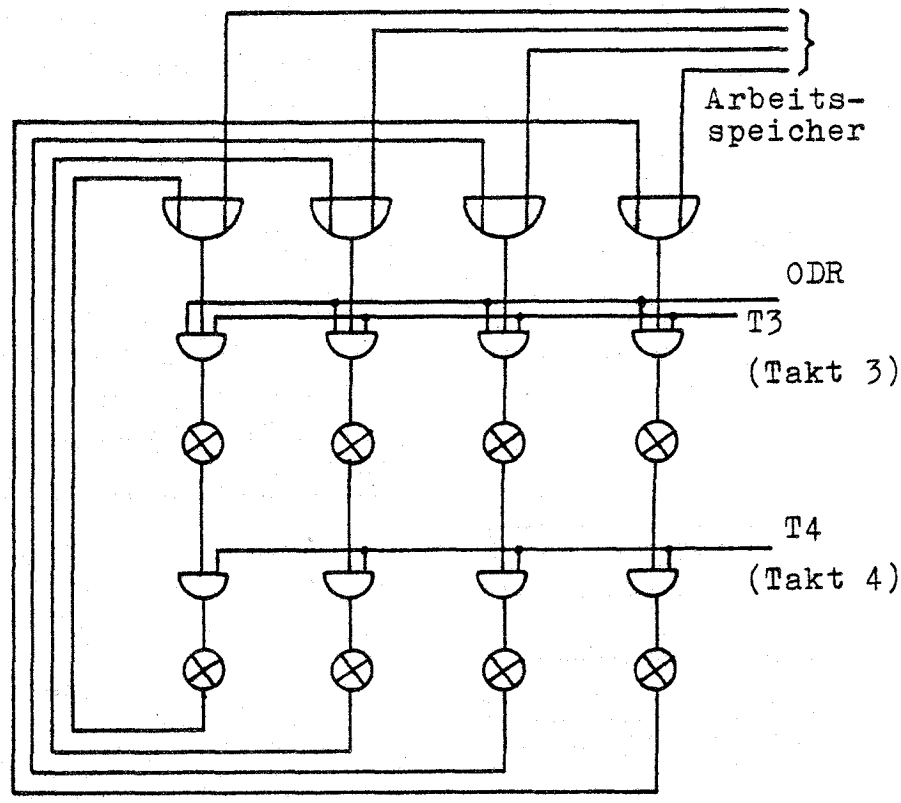
statt  setzt man 

NEGATION:



Zur Darstellung des befehls- und taktgesteuerten Ablaufs im Rechner, kann diese Darstellung folgendermaßen erweitert werden:

z.B. ODER



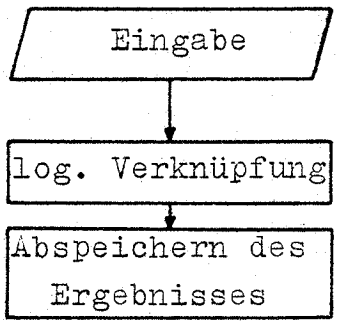
Bedienung:

Programm wie im folgenden angegeben stecken,
RÜCKSETZEN,
Kippschalter zur Eingabe der ersten zu verknüpfenden Binärkombination einstellen,
Betriebsart: BEFEHL einstellen,
Funktionstaste: START drücken,
zweite Binärkombination einstellen,
START auslösen,
Betriebsart SCHRITT zur Durchführung der logischen Verknüpfung einschalten,
Schaltbild (Maske) aufbringen,

Funktionstaste START drei mal drücken (2 mal für die übliche Befehlsbereitstellung und 1 mal für den 1. Schritt der Befehlsausführung, so daß das Ergebnis der logischen Verknüpfung im Hilfsregister steht),
Abspeicherung wieder in Betriebsart BEFEHL

Programm: 1. UND C 1 = A ∧ B

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse dez.	Adresse symb.
0	EIN	1	B
1	EIN	0	A
2	UND	1	B
3	SPE	2	C 1

2. ODER C 2 = A V B

Befehlsfolge wie vorher, jedoch UND-Befehl durch ODR ersetzt:

ODR	1	B
SPE	3	C 2

Bedienung wie vorher

3. NEGATION C 3 = $\overline{A}$

Die Befehlsfolge ist wie folgt zu ändern:

```
0   EIN   6   Minus Eins (IIII)
1   EIN   0   A
2   KPL
3   ADD   6   Minus Eins
4   SPE   4   C 3
```

Bedienung wie bei "1. UND" jedoch muß nach dem RÜCKSETZEN die —1 (IIII) an den Kippschaltern eingestellt werden.

Didaktischer Hinweis:

Sollen die logischen Verknüpfungsoperationen mit den gleichen Zahlen wiederholt durchgeführt werden, so empfiehlt sich eine andere Anordnung der Befehlsfolge: Die Eingaben werden an den Schluß des Programms gestellt.

RÜCK-SETZEN

```
0   UND/ODR   B
1   SPE       C 1/ C 2/ C 3
2   ...
3   EIN       A
4   EIN       B
```

Beim ersten Mal muß lediglich die Taste START mehrmals gedrückt werden, bis der Befehl Nr 3 und 4 erreicht ist.

Genausogut, oder besser noch, kann eine Wiederholung durchgeführt werden, indem man Sprungbefehle anfügt, z.B.:

```
0   EIN   A
1   EIN   B
2   LAD   A
3   UND   B   ← ODR   B
4   SPE   C 1 ← SPE   C 2
5   SPR   2
```

Lehrerdemonstration:

Zunächst werden beliebige Zahlen eingegeben und die Ergebnisse mit den Eingangswerten verglichen. Daraus werden die vollständigen Wertetabellen abgeleitet:

A	B	C 1 UND	C 2 ODER	C 3 = $\overline{A}$ NEGATION
0	0	0	0	I
0	I	0	I	I
I	0	0	I	0
I	I	I	I	0

Um die vollständige Wertetabelle mit einem mal zu erhalten,
gibt man für A ... IIOO
und für B ... IOIO ein.
Wenn die Ergebnisse der verschiedenen Verknüpfungsoperationen dann noch ohne Umstecken erhalten werden sollen
- zum Beispiel bei einer reinen Vorführung durch den Lehrer- empfiehlt sich das folgende vollständige Programm:

```
 0   EIN   1   B (IOIO)
 1   ...
 2   EIN   0   A (IIOO)
 3   ...
 4   UND   1   B
 5   SPE   2   C 1
 6   ...
 7   LAD   0   A
 8   ODR   1   B
 9   SPE   3   C 2
10   ...
11   LAD   0   A
12   KPL
13   ADD   6   (IIII) - 1
14   SPE   4   C 3
```

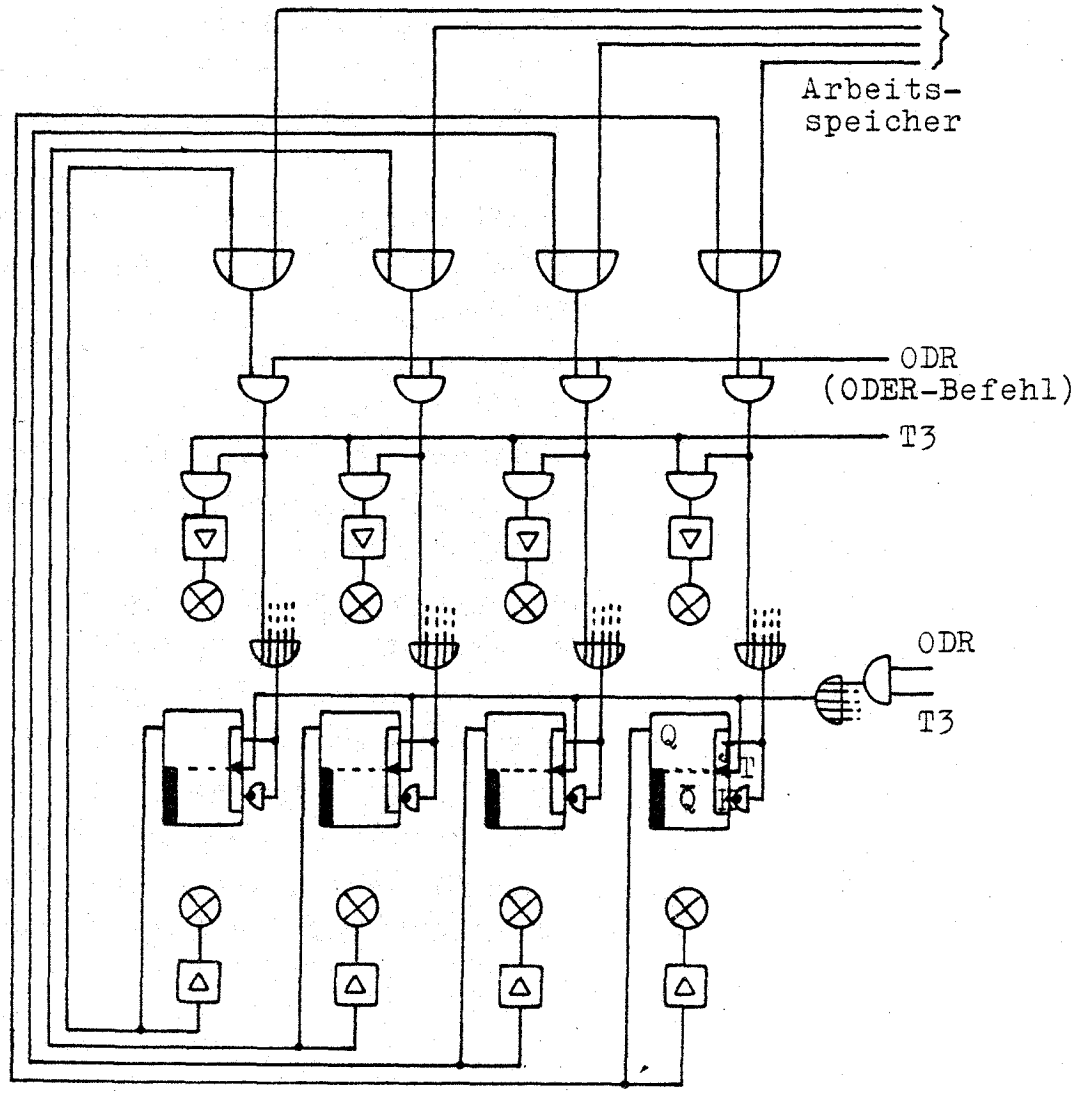
Für die Bedienung gilt:
Betriebsart PROGR. 40 kHz.
Nun braucht für jeden Abschnitt nur einmal die Taste START gedrückt werden.
(Vor dem ersten Programmlauf muß jedoch die -1 eingegeben werden.)

Die logischen Verknüpfungsglieder für die Funktionen UND, ODER sind in der Rechnerelektronik enthalten. Sie werden von den Befehlen UND, ODR durchgeschaltet. Diese Durchschaltung erfolgt, wie aus dem Schaltbild auf Seite 70 ersehen werden kann, mit Hilfe von UND-Gattern (Torschaltungen), die eine takt- und befehlsgesteuerte Durchschaltung ermöglichen: Das Ergebnis der ODER-Verknüpfung erscheint nur dann im Hilfsregister, wenn auch der Befehl ODR selbst und der 3. Takt (T 3) anstehen. Damit ist schon eine Anwendung der UND-Verknüpfung in der Technik aufgezeigt. Die Demonstration dient damit einerseits dem Verständnis der logischen Grundfunktionen, andererseits dem Verständnis der Realisierung im Rechner und der zugehörigen Programmierung. Die Anwendung dieser Grundfunktionen in der Intersektion (Maskenbildung mit Hilfe der UND-Verknüpfung) sowie die Möglichkeit der Einbringung von binären Einsen an beliebige Bitstellen mit Hilfe der ODER-Verknüpfung kann noch angesprochen und gezeigt werden.

Kommentar:

Wie die Schaltbilder zeigen, werden jeweils Eingangs- und Ausgangssignale aufgezeigt, sowie die logischen Schaltungen dargestellt, so daß im Grunde kein Unterschied gegenüber der Darstellung solcher Funktionen mit üblichen elektronischen Baukästen besteht.

Aus Gründen der Vollständigkeit wird für Interessierte der vollständige Schaltplan für die ODER-Verknüpfung im Rechner dargestellt:



Der vollständige Schaltplan für die ODER-Verknüpfung enthält über die logischen Bedingungen hinaus zusätzlich folgende Punkte:

- Berücksichtigung der übrigen Befehle durch ODER-Gatter
- Ansteuerung der Lampen über Transistorverstärker
- Übernahme und Speicherung im Akkumulator durch Registerglieder

Boolesche Algebra

Aufgabenstellung: Die Grundgesetze der Booleschen Algebra (Schaltalgebra) nämlich:

- assoziatives Gesetz
- distributives Gesetz
- de Morgansches Theorem

sollen durch unmittelbares Erleben vermittelt werden

Lösungsweg: Mit Hilfe des Rechners und der einfachen Grundfunktionen UND und ODER werden die genannten Beziehungen programmiert und die Ergebnisse verglichen

1. **Kommutatives Gesetz:** $X1 \wedge X2 = X2 \wedge X1$ (bzw. $X1 \vee X2 = X2 \vee X1$)

Programm:

	Befehls-Adr.	Operation	Adresse dezimal	Adresse symbol.
Eingabe	0	EIN	1	X 1
	1	STP		
	2	EIN	2	X 2
	3	STP		
X 1 ∧ X 2	4	LAD	1	X 1
	5	UND	2	X 2
	6	SPE	3	C 1
	7	STP		
X 2 ∧ X 1	8	LAD	2	X 2
	9	UND	1	X 1
	10	SPE	4	C 2
	11	STP		

2. Assoziatives Gesetz:

$(X1 \wedge X2) \wedge X3 = X1 \wedge (X2 \wedge X3)$

(bzw. $(X1 \vee X2) \vee X3 = X1 \vee (X2 \vee X3)$)

Programm:

```
0   EIN   1   X 1
1   STP
2   EIN   2   X 2
3   STP
4   EIN   3   X 3
5   STP
6   LAD   1   X 1
7   UND   2   X 2
8   UND   3   X 3
9   SPE   4   C 1
10  STP
11  SPR   6   <-- vor diesem Sprung werden die Adressen für
                  X 1 bis X 3 wie folgt umgesteckt:
6   LAD   2   X 2
7   UND   3   X 3
8   UND   1   X 1
9   SPE   5   C 2
```

Ergebnis: Der Inhalt von C 1 muß mit dem Inhalt von C 2 übereinstimmen.

Anmerkung: Das Programm realisiert genau genommen die Beziehung:

$(X1 \wedge X2) \wedge X3 = (X2 \wedge X3) \wedge X1$

statt = $X1 \wedge (X2 \wedge X3)$

und berücksichtigt hier aus Gründen einfacher Programmierung schon das vorher behandelte Kommutative Gesetz.

Ergebnis: Der Inhalt der Adressen 3 und 4 (C 1 und C 2) muß übereinstimmen. Damit ist das assoziative Gesetz bewiesen.

Anmerkung: Der Teil X 2 $\wedge$ X 1 braucht nicht unbedingt gesteckt werden, er kann auch durch bloßes Umstecken der Adressen im Teil X 1 $\wedge$ X 2 erreicht werden. Ebenso kann durch Ersatz der beiden UND-Befehle und durch zwei ODR-Befehle die Bedingung
X 1 $\vee$ X 2 = X 2 $\vee$ X 1
verifiziert werden.

Bedienung: Programm, wie folgt angegeben, stecken, RÜCKSETZEN, PROGR. 40 kHz,
Kippschalter für X 1 einstellen, START.
Kippschalter für X 2 einstellen, START.
Nach dem nochmaligen START steht das Ergebnis von X 1 $\wedge$ X 2 in Zelle 3,
START, das Ergebnis von X 2 $\wedge$ X 1 steht in Zelle 4

3. Distributives Gesetz:

$$(X1 \wedge X2) \vee (X1 \wedge X3) = X1 \wedge (X2 \vee X3)$$

(bzw. $(X1 \vee X2) \wedge (X1 \vee X3) = X1 \vee (X2 \wedge X3)$)

Programm:

```
0   EIN   1   X 1   }  Betriebsart
1   EIN   2   X 2   }  BEFEHL
2   EIN   3   X 3   }
3   LAD   1   }
4   UND   3   }  (X 1 ∧ X 3)
5   SPE   7   }
6   LAD   1   }
7   UND   2   }  (X 1 ∧ X 2)
8   ODR   7         =    ∨ (X 1 ∧ X 3)
9   SPE   4     C 1
10  STP
11  SPR   3   ← vor diesem Sprung werden
                die Adressen für X 1 bis X 3
                wie folgt umgesteckt:

3   LAD   2   }
4   ODR   3   }  (X 2 ∨ X 3)
5   SPE   7   }
6   LAD   1   }
7   UND   7   }  X 1 ∧ ( X 2 ∨ X 3)
8   SPE   5     C 2
9   STP
```

Ergebnis: Der Inhalt von C 1 (Adresse 4) muß mit dem Inhalt von C 2 (Adresse 5) übereinstimmen.

4. De Morgansches Theorem:

Allgemein gilt:

$\bar{f}\ (X_i, \bar{X}_k, \wedge, V) \quad f\ (\bar{X}_i, X_k, V, \wedge)$

In Worten: Die negierte Schaltfunktion ist gleich der nicht negierten Schaltfunktion mit negierten Einzelelementen
(d.h. X 1 → $\overline{X\ 1}$, ∧ → V, V → ∧)

z. B.: $\overline{\overline{X\ 1} \wedge X\ 2} = X\ 1 \vee \overline{X\ 2}$

Programm:

0	EIN	1	X 1	*
1	...			
2	EIN	2	X 2	
3	...			
4	LAD	1		
5	KPL			
6	ADD	6	- 1	
7	UND	2		
8	KPL			
9	ADD	6		
10	SPE	3		
11	...			
12	LAD	2		**
13	KPL			
14	ADD	6	- 1	
15	ODR	1		
16	SPE	4		

* Vor dem Programmlauf wird hier der Befehl "EIN 6" gesteckt und die - 1 (IIII) für die spätere Negation eingegeben.

** Aus Gründen einfacherer Programmierung wird hier das kommutative Gesetz angewendet und $\overline{X\ 2}$ v X 1 statt X 1 v $\overline{X\ 2}$ gebildet.

Bedienung:

wie unter 1.) Assoziatives Gesetz.
Die einzelnen Programmabschnitte werden durch START ausgelöst (Betriebsart PROGR. 40 kHz).
Die Ergebnisse in Zelle 3 und 4 müssen übereinstimmen.

Schaltfunktionen Äquivalenz und Antivalenz

Aufgabenstellung: Aus der vollständigen Wertetabelle sollen die Schaltfunktionen für Äquivalenz und Antivalenz aufgestellt und dann durch ein Programm realisiert werden.

Lösungsweg: Wertetabelle:

X 1	X 2	Äquivalenz	Antivalenz
0	0	I	0
0	I	0	I
I	0	0	I
I	I	I	0

Die Äquivalenz (Übereinstimmung) ist dann "wahr" (bzw. Eins) wenn X 1 mit X 2 übereinstimmt.

Die Antivalenz (Unterscheidung) ist dann "wahr" (bzw. Eins) wenn X 1 nicht mit X 2 übereinstimmt. Die Antivalenz ist die Negation der Äquivalenz und umgekehrt. Die sog. konjunktive Normalform der Schaltfunktion entsteht dann, wenn man diejenigen Zeilen der vollständigen Wertetabelle, an denen die Ausgangsgröße mit Eins besetzt ist disjunktiv (ODER) mit einander verknüpft. Innerhalb der mit Eins besetzten Zeilen werden die Eingangsgrößen konjunktiv (UND) verknüpft.

z.B.: Antivalenz
Die 2. Zeile ist mit I besetzt, die zugehörigen Eingangsgrößen sind wie folgt zu verknüpfen:
$\overline{X\,1} \wedge X\,2$

Die 3. Zeile ist wieder mit I
besetzt, die Eingangsgrößenver-
knüpfung lautet:
$X\,1 \wedge \overline{X\,2}$
Die gesamte Funktion lautet:
Antivalenz, $A = (\overline{X\,1} \wedge X\,2) \vee (X\,1 \wedge \overline{X\,2})$

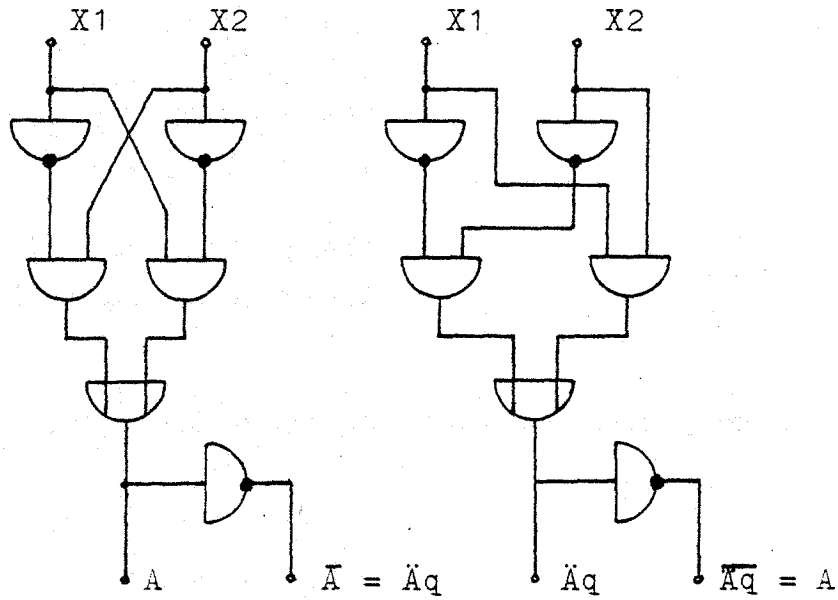
Ebenso folgt für die Äquivalenz:
Äquivalenz, $Äq = (X1 \wedge X\,2) \vee (\overline{X}1 \wedge \overline{X\,2}) = \overline{A}$

Lösung bzw. Darstellung am Rechner durch ein Programm:

Eingabe	0	EIN	1	X 1
	1	STP		
	2	EIN	2	X 2
	3	STP		
$\overline{X\,1} \wedge X\,2$	4	LAD	1	X 1
	5	KPL		
	6	ADD	6	- 1
	7	UND	2	X 2
	8	SPE	3	A
$X\,1 \wedge \overline{X\,2}$	9	LAD	2	X 2
	10	KPL		
	11	ADD	6	- 1
	12	UND	1	X 1
	13	ODR	3	A
	14	SPE	3	A
	15	STP		
	16	KPL		
	17	ADD	6	- 1
	18	SPE	4	Äq

Ergebnis:
Die Antivalenz zu den Eingangssignalen
X 1 und X 2 steht in Zelle 3 (A)
In Zelle 4 (Äq) steht die Äquivalenz
von X 1 und X 2

Logikplan der technischen Schaltung:



Lehrerdemonstration: Das Beispiel dient zur Darstellung von Abläufen in Schaltwerken bzw. Steuerwerken und zeigt den linearen Ablauf in einzelnen Folgeschritten wie in üblichen Steuerwerken. Das Beispiel stellt eine Anwendung der Gesetze der Schaltalgebra dar. Darüber hinaus können die technische Realisierung in Form von Schaltungsbildern dargestellt und die Programmierung aufgezeigt werden.

Kommentar: Das Beispiel zeigt die Ansatzpunkte auf, wie es möglich ist mit Hilfe eines Rechners neu zu konzipierende Schaltwerke bis hin zur Elektronik eines ganzen Rechners selbst, programmiert nachzubilden und ihr Verhalten schon vor der technischen Realisierung (natürlich mit enorm aufwendigen Programmen) weitgehend auszutesten.

Der Halbaddierer

Aufgabenstellung: Als Einführung in die duale Addition sollen Aufbau und Funktion eines Halbaddierers erläutert werden.

Lösungsweg: Der Halbaddierer verknüpft lediglich zwei Eingangsbits A1 und B1 und bildet daraus Summe $\Sigma 1$ und Übertrag Ü1.

Aus der vollständigen Wertetabelle kann die logische Schaltung leicht ermittelt werden:

A1	B1	Σ 1	Ü1
0	0	0	0
0	I	I	0
I	0	I	0
I	I	0	I

daraus folgt unmittelbar:

$\Sigma 1 = (\overline{A1} \wedge B1) \vee (A1 \wedge \overline{B1})$

$\underline{\ddot{U} 1 = A1 \wedge B1}$

Wenn man von der negierten Schaltfunktion ausgeht, so kann man Ü1 auch für die Bildung von Σ 1 verwenden und erhält somit eine noch einfachere Schaltfunktion für $\Sigma 1$:

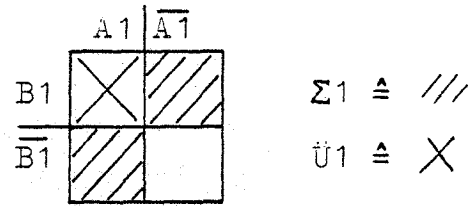
$\Sigma 1 = (\overline{A1} \wedge \overline{B1}) \vee (A1 \wedge B1)$

mit $\overline{\ddot{U}1 = A1 \wedge B1}$ folgt:

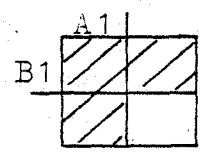
$\Sigma 1 = (\overline{A1} \wedge \overline{B1}) \vee \ddot{U}1$

$\underline{\Sigma 1 = (A1 \vee B1) \wedge \overline{\ddot{U}1}}$

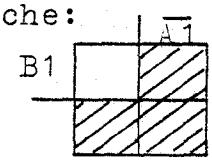
Zur Erläuterung dieser Funktionen und Umformungen kann eine graphische Darstellung, das sog. "Karnaugh-Diagramm", herangezogen werden:



$\Sigma 1$ besteht aus einem ersten Anteil A1 v B1 welcher der folgenden gestrichelten Fläche entspricht:



Da diese Fläche größer ist als $\Sigma 1$ entspricht, wird diese noch durch die Bedingungen $\wedge \overline{\ddot{U}1}$ eingeschränkt.
$\overline{\ddot{U}1}$ entspricht der Fläche:

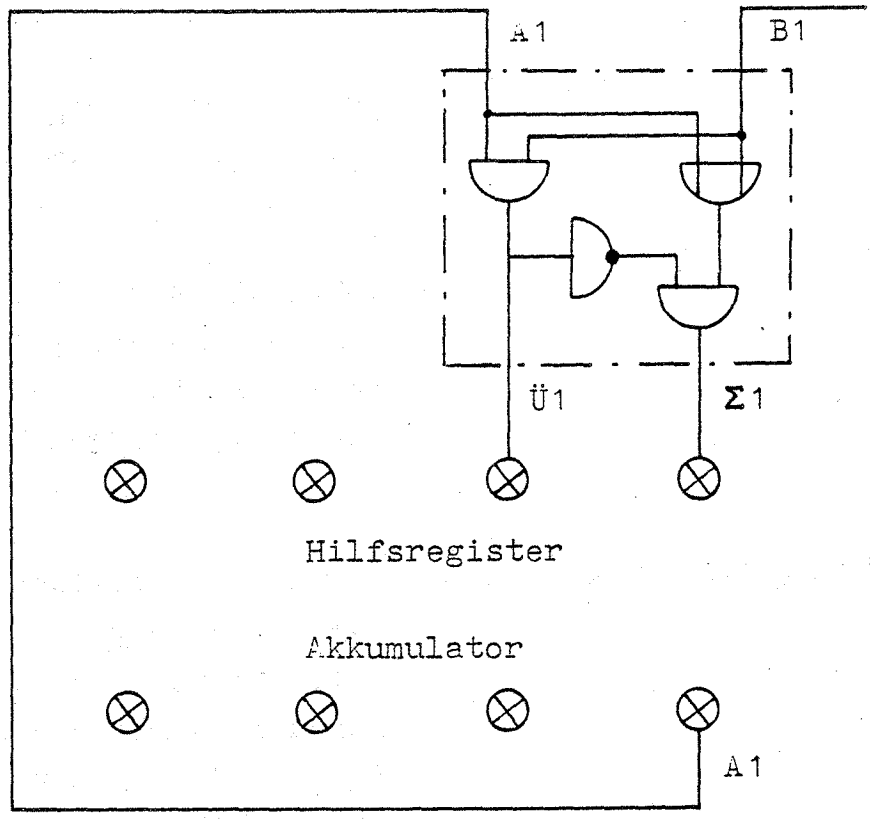


Die Schnittmenge (UND-Verknüpfung) beider Flächen entspricht der Bedingung für $\Sigma 1$.

Aus der Schaltfunktion $\Sigma 1 = (A1 \vee B1) \wedge \overline{\ddot{U}1}$ kann folgendes Schaltbild aufgestellt und - falls mit Ausschnitten für die Lampen von Akkumulator und Hilfsregister versehen - auf die Rechnerfronttafel aufgebracht werden:

Anmerkung: Erläuterungen zur Ableitung und Minimierung bzw. Umformung von Schaltfunktionen sind in allen gebräuchlichen Informatik-Lehrbüchern unter Boolescher Algebra bzw. Schaltalgebra zu finden.

Schaltplan Halbaddierer:



Die Funktion des Halbaddierers kann nun im Zusammenwirken von Schaltplan und dem folgenden Programm demonstriert werden:

Programm:

Befehls-adr.	Operation	Adresse		Betriebsart
		dez.	symb.	
0	EIN	0	B1	BEFEHL
1	EIN	1	A1	BEFEHL
2	ADD	0	B1	SHRITT

Bedienung:

Programm, wie angegeben, stecken.
Bei den beiden Eingaben am Anfang des Programms darf lediglich das letzte Bit gesetzt sein, d.h. die ersten 3 Kippschalter der Eingabe müssen immer auf Null stehen: 000X

RÜCKSETZEN, BEFEHL, Kippschalter für B1 einstellen, START → B1 wurde eingegeben

Kippschalter für A1 einstellen, START

SCHRITT, Taste START 3mal drücken (2mal für Befehlsbereitstellung, 1mal für erste Phase der Befehlsausführung → Ergebnis steht im Hilfsregister).

Lehrerdemonstration: Es werden sämtliche Möglichkeiten der Wertetabelle durchgespielt. Als Überleitung zum Volladdierer muß herausgestellt werden, daß auf der nächsten Binärstelle zusätzlich zu A2 und B2 auch der Übertrag der vorhergehenden Stelle addiert werden muß.

Kommentar: So einfache, bzw. vielfach zur Anwendung kommende Schaltungen wie der Halbaddierer werden heute nicht mehr aus einzelnen UND/ODER-Gattern aufgebaut, sondern in integrierter Form hergestellt. D.h. diese Schaltungen (ganze Addierwerke, Schieberegister, Zähler usw.) werden in Siliziumhalbleiterkristalle eingearbeitet (Abätzungen fotographisch erzeugter Muster für nachfolgende Dotierungen erzeugen die Elemente der Schaltungen und deren Verbindungen). Vielfach wird dann ein so entstandenes Gebilde nur noch als "schwarzer Kasten" angesehen. Man kennt nur noch die Definitionen und Bedingungen für Eingangs- und Ausgangssignale und macht sich keine Gedanken mehr über die inneren Funktionen der Schaltung. Dieser an sich bequeme Weg führt aber nur dann auf die Dauer nicht zu Schwierigkeiten, wenn man diese

Schaltungen im Notfall doch verstehen kann (z.B. bei mißverständlichen Definitionen, fremdsprachlich beschriebenen Schaltungen, beim Prüfen und Auftreten von Fehlern, bei vom üblichen Einsatzfall abweichenden Anwendungen, bei Anpassungsschaltungen usw.) Deshalb ist es auch heute noch sinnvoll, sich anhand solcher Schaltungsbeispiele, wie dem Halbaddierer usw. in die Technik der digitalen Schaltungen einzuarbeiten. Außerdem werden hierbei die prinzipiellen Überlegungen zur Dualarithmetik aufgezeigt.

Volladdierer, Paralleladdierwerk, Übertrag- und Überlaufbildung, sowie Komplementierwerk

Aufgabenstellung: Aufbau und Funktion des Volladdierers sollen erläutert werden.

Lösungsweg: Ausgehend von der vollständigen Wertetabelle wird die Schaltfunktion des Volladdierers ermittelt:

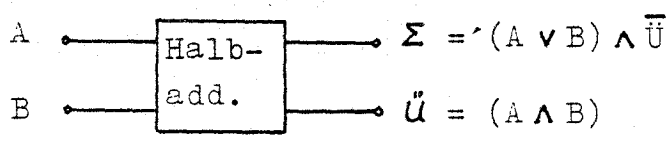
A2	B2	Ü1	Σ2	Ü2
0	0	0	0	0
0	0	I	I	0
0	I	0	I	0
0	I	I	0	I
I	0	0	I	0
I	0	I	0	I
I	I	0	0	I
I	I	I	I	I

Schaltfunktionen:

$$\Sigma 2 = (A2 \wedge \overline{B2} \wedge Ü1) \vee (\overline{A2} \wedge B2 \wedge \overline{Ü1}) \vee (\overline{A2} \wedge \overline{B2} \wedge Ü1) \vee (A2 \wedge B2 \wedge Ü1)$$

$$Ü2 = (A2 \wedge B2 \wedge Ü1) \vee (A2 \wedge \overline{B2} \wedge \overline{Ü1}) \vee (\overline{A2} \wedge B2 \wedge Ü1) \vee (A2 \wedge B2 \wedge Ü1)$$

Eine einfachere Lösung ergibt sich durch Umformung unter Berücksichtigung der Schaltfunktion des Halbaddierers:

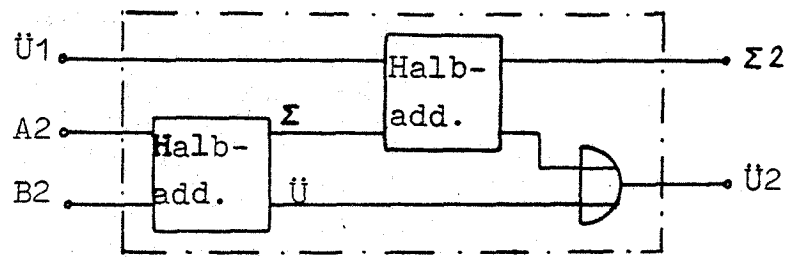


Diese Umformung kann rein algebraisch erfolgen, und zwar auf dem Weg über die negierte Schaltfunktion durch Schaffung von Ausdrücken, welche sich durch Σ und $\ddot{U}$ substituieren lassen. Streng genommen heisst das, die Vereinfachung erfolgt auf dem Weg, dass Teillösungen und Lösungen selbst nur einmal ermittelt und dann noch für andere Lösungen verwendet werden. Die ursprüngliche Schaltfunktion geht von dem Lösungsprinzip aus, dass für jede Ausgangsgrösse alle vorhergehenden Umformungen getrennt für sich, also parallel und mehrfach vorgenommen werden.
Die vereinfachte Lösung kann aber auch wie beim vorhergehenden Beispiel aus dem "Karnaugh-Diagramm" ermittelt werden.

$$\left\| \begin{array}{l} \Sigma 2 = (\ddot{U}1 \vee \Sigma) \wedge \overline{(\Sigma \wedge \ddot{U}1)} \\ \ddot{U}2 = (\Sigma \wedge \ddot{U}1) \vee \ddot{U} \end{array} \right\|$$

$$\text{mit} \quad \Sigma = (A2 \vee B2) \wedge \overline{\ddot{U}}$$
$$\ddot{U} = A2 \wedge B2$$

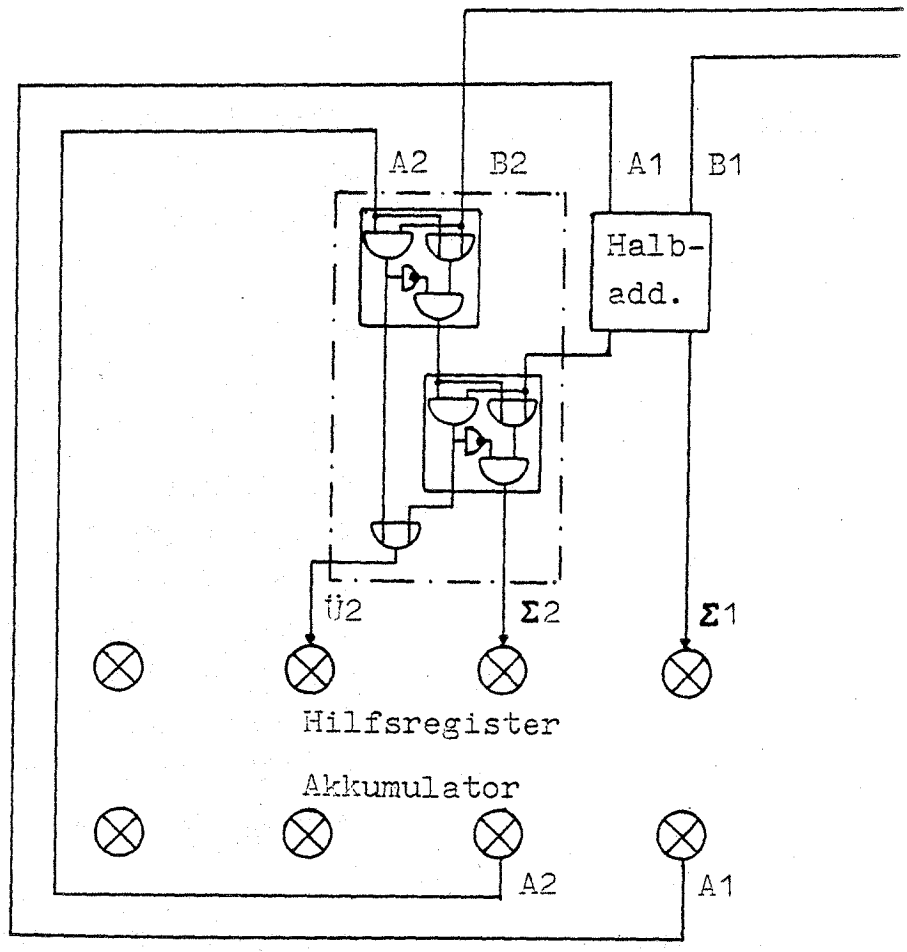
Damit sieht die Schaltung des Volladdierers folgendermassen aus:



Ü1 ist der Übertrag einer vorhergehenden Binärstelle, z.B. der Übertrag aus der Übung: Halbaddierer

Auch hier kann die Funktion der Schaltung durch eine sogenannte Maske (Schaltplan mit Ausschnitten für die Lämpchen des Demonstrationsmodells) im Zusammenwirken mit einem Programm (wie im vorhergehenden Beispiel) demonstriert werden.

Schaltplan:
Volladdierer



Programm:

Befehls-Adr.	Operation	Adresse		Betriebs-art
		dez.	symb.	
0	EIN	0	B1, B2	BEFEHL
1	EIN	1	A1, A2	BEFEHL
2	ADD	0	B1, B2	SCHRITT

Bedienung:

Wie im Beispiel "Halbaddierer".
Es ist jedoch ein zusätzliches Bit (A2, B2) mehr zu besetzen:

A im Akkumulator ... 00XX
B in Arbeitsspeicherzelle 0... 00XX

Lehrerdemonstration:

Es wird gezeigt, dass bei der Addition der zweiten Bitstelle ein möglicher Übertrag aus der vorhergehenden Stelle berücksichtigt werden muss.

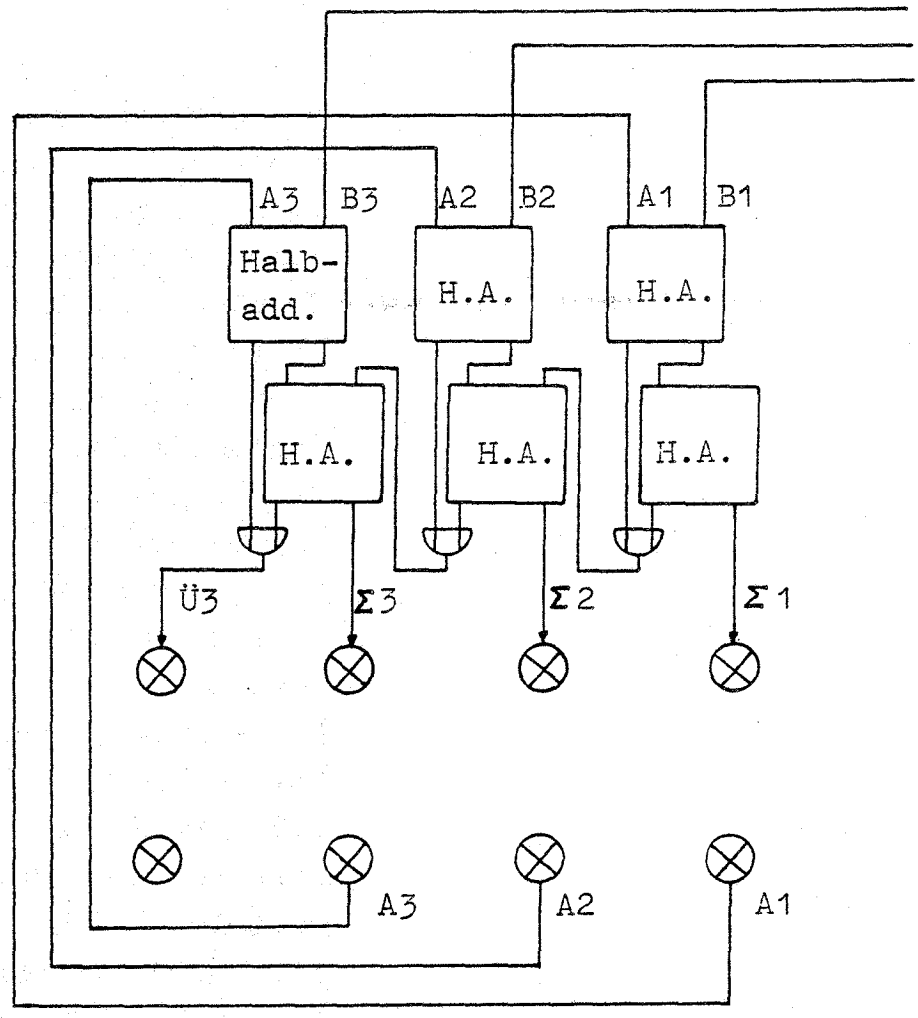
Der Übertrag Ü1 geht jetzt nicht mehr auf die nächste Bitstelle, sondern in den folgenden Volladdierer.

Erst damit erhalten wir einen vollgültigen Bitaddierer, den "Volladdierer". Damit können die Möglichkeiten der Wertetabelle durchgespielt werden. (Ü1 muss dabei jedoch gesondert ermittelt werden, indem man zunächst nur A1 und B1 eingibt, oder aber man überlegt sich selbst, ob aus A1 und B1 ein Ü1 entsteht).

Kommentar:

Das Beispiel kann noch um ein Bit erweitert werden zum 3-stelligen Paralleladdierwerk, dessen Schaltung folgendermassen aussieht:

Schaltplan:
3-stelliges Paralleladdierwerk

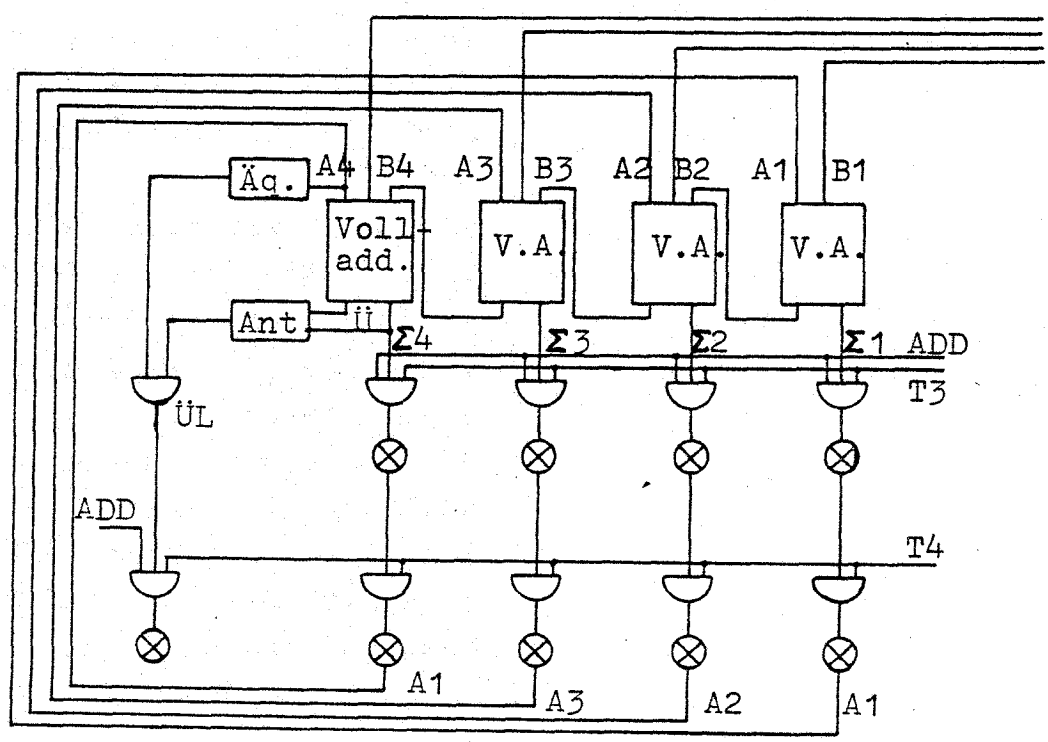


Damit können Zahlen bis zu einem maximalen Ergebnis gleich 7 addiert werden. Wird dieses Ergebnis überschritten, so tritt ein Übertrag Ü3 auf.
Wenn Ü3 zu den Bitstellen gerechnet wird, so können damit Dualzahlen bis zu einem maximalen Ergebnis von 15 addiert werden.

Wenn aber wie hier und auch allgemein üblich dieses Bit als Vorzeichenbit reserviert ist, so kehrt der Übertrag Ü3 das Vorzeichen des Ergebnisses um und erzeugt einen Fehler.
Die Gesetzmässigkeiten dieser Zahlendarstellung mit Vorzeichen werden durch Bildung des sogenannten "Überlaufs" berücksichtigt. Der Überlauf wertet die Vorzeichen der Zahlen A und B bei der Addition

aus (Äquivalenz- und Antivalenzschaltung) sowie das bisherige Ü3, das wir jetzt VZ (Vorzeichen) nennen müssen, und bildet bei einer Verfälschung des Ergebnisses den Überlauf ÜL. Dieses Fehlersignal kann mit Hilfe der Abfrage SUL (Sprünge falls Überlauf) abgefragt werden. Beim Auftreten des Überlaufs ist das Ergebnis aber nicht einfach falsch, es ist nur für den Rechner missverständlich geworden. Wenn man aber in diesem Fall das Vorzeichen zu den Binärstellen rechnet, so ist das Ergebnis noch erkennbar.
Diese Zusammenhänge und die Bedeutung des Überlaufregisters werden in der Übung "Duale Subtraktion" noch näher erläutert.
Hier sei nur noch gesagt: Der Überlauf wird gebildet, wenn bei der Addition der Bereich der darstellbaren Zahlen (-8...0...7) nach oben oder unten überschritten wird.
Das zugehörige Schaltbild zur Demonstration dieser Zusammenhänge hat das folgende Aussehen. Auch hier kann das bisherige Programm verwendet werden.

Schaltplan:
Paralleladdierwerk mit Überlaufbildung

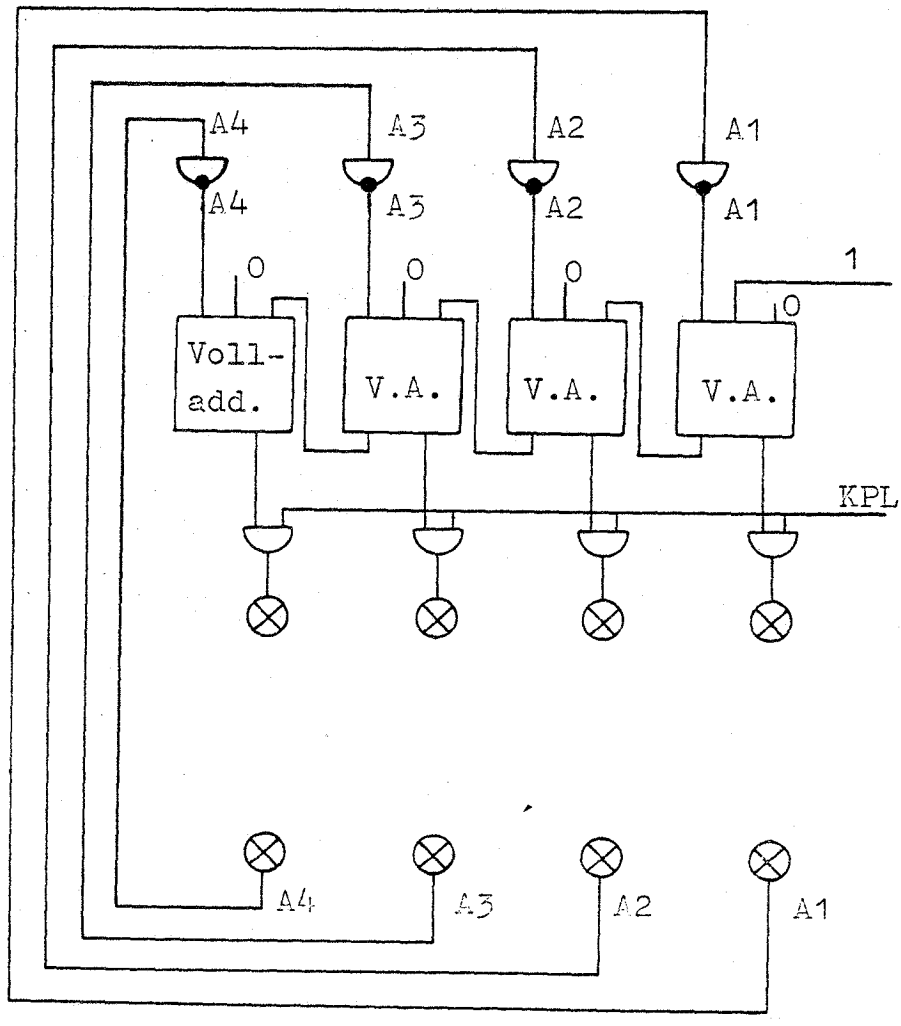


Mit Hilfe eines Paralleladdierwerks kann auch die Komplementbildung (2-Komplement) einfach durchgeführt werden. Wie auch im Beispiel "Duale Subtraktion" erläutert, muss beim 2-Komplement zu einer Umkehrung aller Bitstellen eine Addition von Eins auf die niederwertigste Stelle hinzutreten. Diese Zusammenhänge zeigt der folgende Schaltplan auf, der auf die Frontplatte des Modells zu bringen ist und zusammen mit dem Programm ...

```
0    EIN    0
1    KPL
```

... in der Betriebsart BEFEHL die Komplementbildung anschaulich demonstrieren lässt.

Schaltplan:
Komplementierwerk

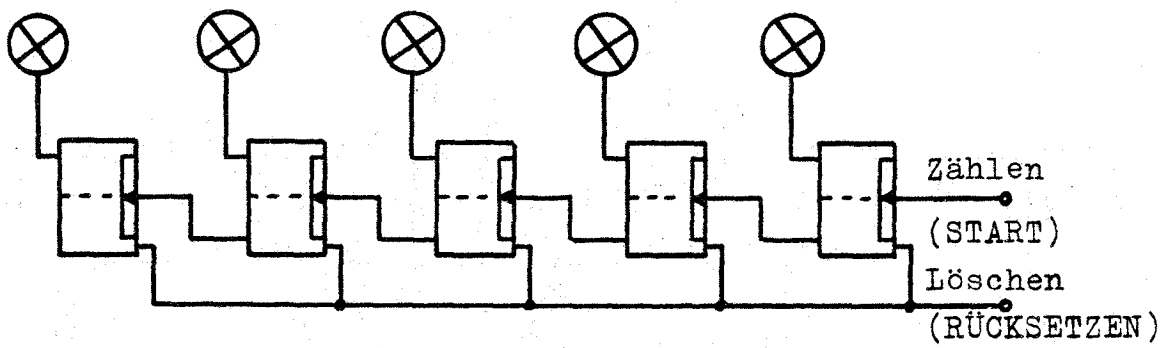


Dualzähler

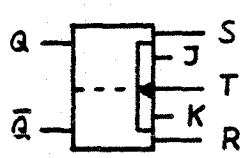
Aufgabenstellung: Aufbau und Funktion eines Dualzählers sollen erläutert und demonstriert werden.

Lösungsweg: Der technische Aufbau eines Dualzählers wird anhand des Befehlszählers im Rechner und mit Hilfe einer Schaltzeichnung erläutert. (Die Schaltzeichnung ist im Zubehör enthalten und kann auf den Befehlszähler aufgelegt werden, so daß nur die Signalleuchten durchblicken.) Zusätzlich kann die Funktion des Zählers durch ein Zählerprogramm nachgebildet werden.
In beiden Fällen wird das duale Zahlensystem anschaulich dargelegt.

Schaltzeichnung des Dualzählers (Befehlszähler)



Anmerkung: Zählstufen bzw. Registerstufen werden durch sog. Master-Slave-Flipflops realisiert.



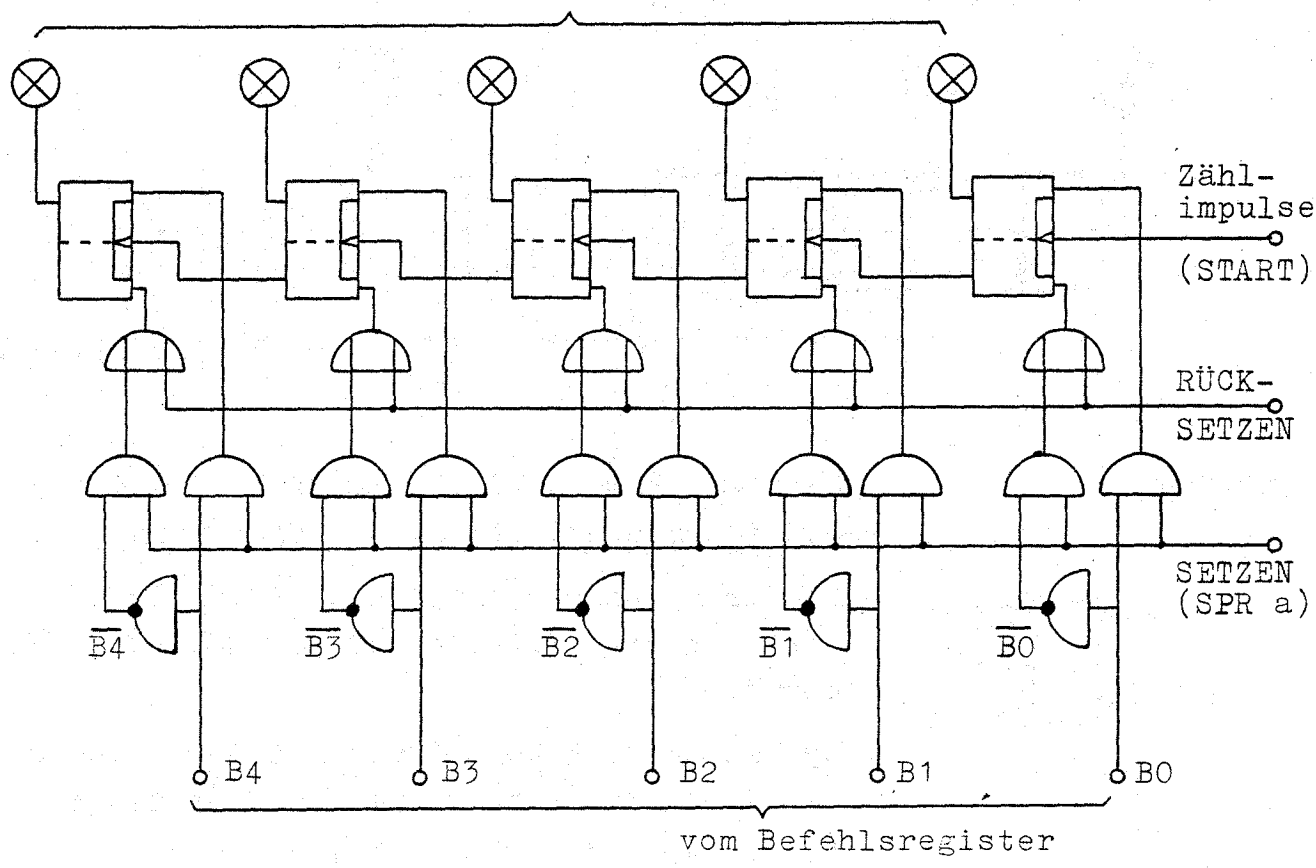
- Statische Eingänge (Setzen, Rücksetzen) ...S,R schalten sich unmittelbar auf den Ausgang... Q durch.
- Dynamische Eingänge (Vorbereitungseingänge) ... J, K,wirken erst mit einem Taktsignal ...T auf Q, bleiben dann aber gespeichert.

Diese Zeichnung kann mit Ausschnitten für die Befehlszählerlämpchen versehen und auf den Befehlszähler in der Frontplatte aufgesetzt werden.

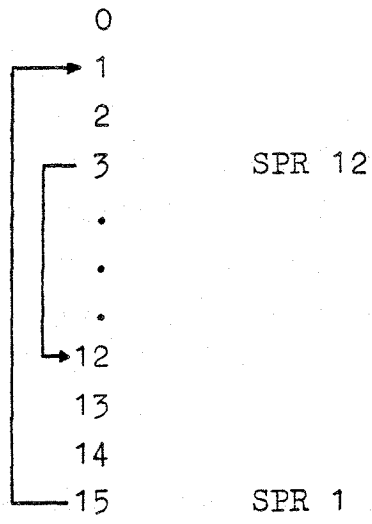
Die Taste RÜCKSETZEN wirkt auf den Löscheingang und setzt den Zähler zurück (auf Null). Mit der Taste START kann der Zähler hochgezählt werden (Betriebsart: PROGR. Es ist kein Befehl gesteckt).

Der Befehlszähler kann außerdem auch auf einen bestimmten Zählerstand (B0...B4) von außen gesetzt werden, und zwar bei Sprungbefehlen. Damit erweitert sich die Schaltzeichnung wie folgt:

Einstellbarer Dualzähler (Befehlszähler):



Zur Demonstration des Zählersetzens (neue Einstellung des Zählers) sind nun jedoch auch Sprungbefehle im Programmspeicher zu stecken, die an beliebiger Stelle stehen können, z.B.:



Solche technisch aufgebauten (in der Elektronik des Modells enthaltenen) Zähler können auch durch Zählerprogramme simuliert werden:

Eine Zählung im Akkumulator läßt sich am einfachsten mit dem folgenden "Programm" durchführen und demonstrieren:

0	EIN	0	(000I)
1	ADD	0	

Zur Bedienung müssen hier zunächst die Kippschalter auf 000I eingestellt und Betriebsart BEFEHL gewählt werden. Dann wird einmal ANF0 zur Eingabe der Zahl 1 in Adresse 0 gedrückt. Jeder Tastendruck auf ANF1 zählt nun den Inhalt des Akkumulators um 1 hoch.

Man kommt ohne die Tasten ANF0 und ANF1 mit START aus, wenn man das Programm wie folgt aufbaut:

```
0       EIN     0       (000I)
1       ADD     0
2       STP
3       SPR     1
```

Betriebsart, PROGR. 40 kHz

Lehrerdemonstration: In beiden Fällen, Befehlszähler, als auch programmierter Zähler, wird das duale Zahlensystem anschaulich dargelegt. Besonders herauszustellen ist dabei der Begriff der Stellenwertigkeit.

Kommentar: Ein ordentlicher programmierter Zähler hat natürlich eine andere umfangreichere Form als das angegebene reine Demonstrationsprogramm, das nur durch die Möglichkeiten, die in ANF0 und ANF1 enthalten sind, wirksam werden kann. Und zwar muß ein solches Programm einen Zähleranfangs- und Zählerendwert, sowie eine Schrittweite (im allgemeinen 1) einstellen lassen können:

```
0       EIN     0       (Zähleranfangswert)
1       EIN     1       (Zähler-endwert)
2       EIN     2       (Schrittweite)
3       LAD     0
4       SPE     3       (Zähler-Hilfszelle)
5       LAD     3
6       ADD     2
7       SPE     3
8       KPL
9       ADD     1
10      SGN     12
11      SPR     5
12      STP
```

Die ersten 3 Eingaben müssen in Betriebsart BEFEHL durchgeführt werden. Ab Befehlszählerstand 3 kann das Programm auch in den anderen Betriebsarten für Programmabläufe (PROGR. 40 kHz, 8 Hz, 2 Hz, 0,5 Hz) ablaufen. Das Programm sorgt für die Abspeicherung des Zähleranfangswertes in die Zähl z elle (Zähler-Hilfszelle), dann folgt die Zählung selbst mittels Addition der Schrittweite; das Ergebnis wird durch Subtraktion mit dem Zählerendwert verglichen. Mittels Abfrage auf Null kann schließlich die Zählerschleife verlassen werden.

Stellen Sie den zugehörigen Programmablaufplan auf!

Das Zählerprogramm kann zu einem mehrstelligen Oktalzähler ausgebaut werden, wenn man den Zählerinhalt zunächst in einer ersten Zelle abspeichert und jeden Übertrag für sich addiert und in einer folgenden Zelle abspeichert. Damit wäre das Oktalsystem (streng genommen das binär codierte Oktalsystem) darstellbar.

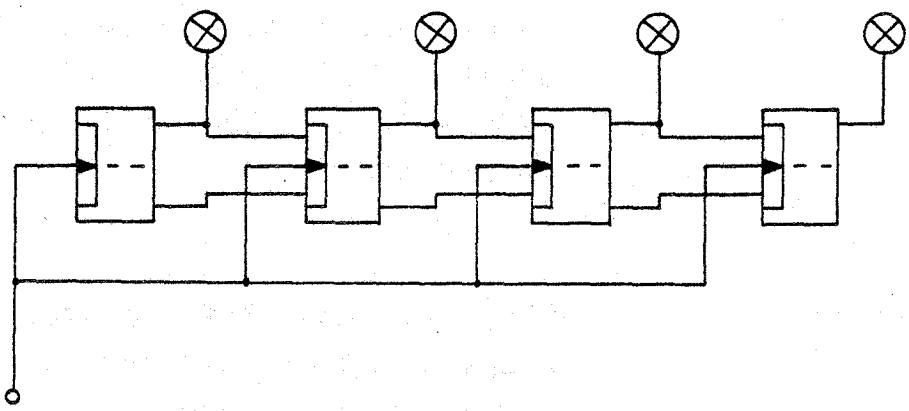
Durch Abfrage auf IOIO (10) kann ebenso gut ein mehrstelliger BCD-Zähler aufgebaut werden.

Schieberegister

Aufgabenstellung: Aufbau und Funktion eines Schieberegisters soll dargestellt werden.

Lösungsweg: Der Akkumulator ist neben anderen Funktionen auch als Schieberegister aufgebaut. Es genügt deshalb ein Schaltbild, das die Akkumulatorlämpchen frei läßt, auf die Frontplatte aufzusetzen und die Verschiebebefehle ablaufen zu lassen.

Schaltbild des Rechtsschieberegisters



Schiebetakt

Programm:

Befehlsfolge

Befehls-Adr.	Operation	Adresse
0	EIN	0
1	VLR	

Bedienung:

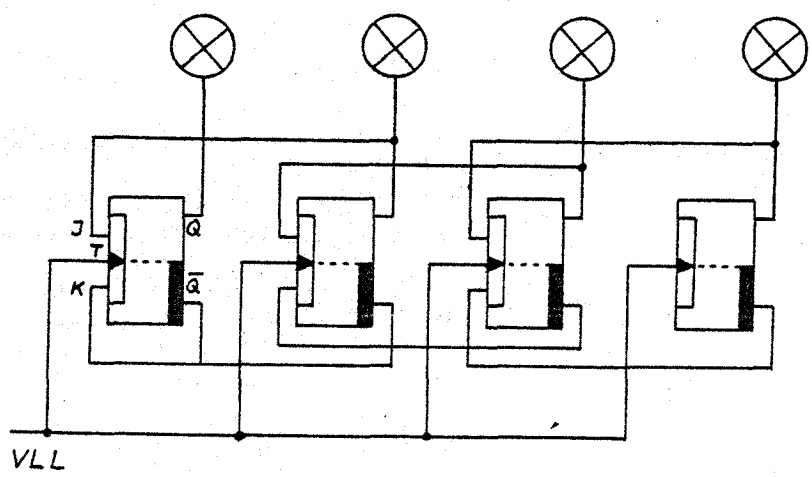
Betriebsart BEFEHL einstellen,
Programm stecken,
Verschiebezahl im Eingaberegister einstellen,
ANF0 drücken ⟶ Eingabe der Verschiebezahl
ANF1 löst jeweils eine Verschiebung aus.

Lehrerdemonstration:

Erläuterung der Schaltung des Rechtsschieberegisters: Die Ausgänge (Inhalt) der Registerstufen sind auf die Vorbereitungseingänge der rechten Nachbarregister geschaltet. Die an den Vorbereitungseingängen anstehenden Signale werden nur dann übernommen, wenn auch auf die Übernahmeeingänge ein Signal (Takt) geschaltet wird. Diese Übernahmeeingänge sind zusammengeschaltet und erhalten bei der Auswertung des Verschiebebefehls vom Rechner einen Impuls, den Schiebetakt.

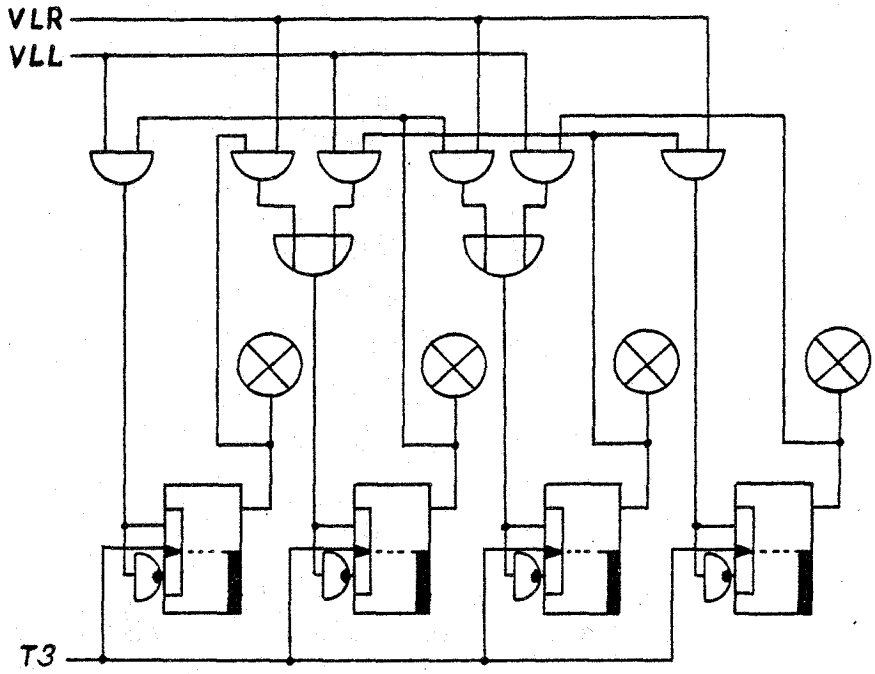
Kommentar:

Ebensogut kann das Linksschieberegister dargestellt und mit dem VLL-Befehl in Betrieb gesetzt werden.



Wenn mehrere Funktionen, wie z.B. Rechts- und Linksverschiebung,an einem Register gleichzeitig möglich sein sollen, so wird die Schaltung wesentlich komplizierter, da eine Reihe von Verriegelungen (UND/ODER-Gatter) nötig sind.

Rechts/Links-Schieberegister:



Die im Modell zwar nicht unmittelbar vorgesehene zyklische Verschiebung kann durch ein Programm nachgebildet werden:

Zyklisches Linksschieberegister

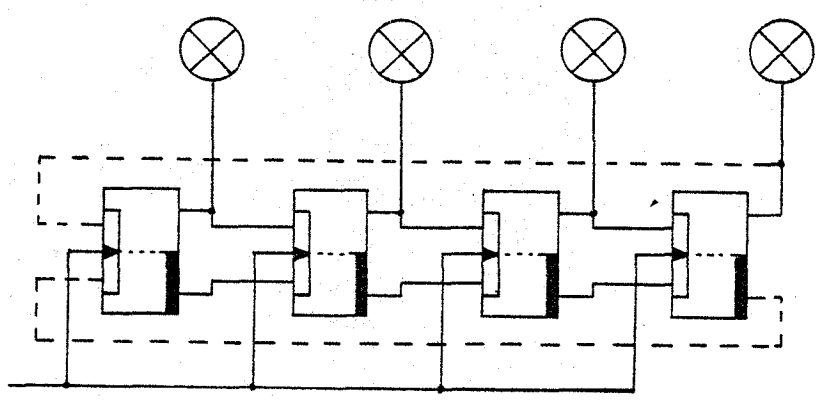
0	EIN	0	(Verschiebezahl)
1	...		
2	SAM	5	
3	VLL		
4	SPR	1	
5	VLL		
6	ODR	1	(000I)
7	SPR	1	

Das Programm läuft in Betriebsart PROGR. 40 kHz. Vor dem ersten Programmlauf muß jedoch (mit EIN 1) die Binärkombination 000I eingegeben werden.

Zyklisches Rechtsschieberegister

0	EIN	0	(Verschiebezahl)
1	...		
2	UND	1	(000I)
3	SGN	9	
4	LAD	0	
5	VLR		
6	ODR	2	(I000)
7	SPE	0	
8	SPR	1	
9	LAD	0	
10	VLR		
11	SPE	0	
12	SPR	1	

Betriebsart, PROGR. 40 kHz. Vor dem ersten Programmlauf müssen die Binärkombination 000I in Adresse 1 und I000 in Adresse 2 eingegeben werden.
Die vorliegenden, nicht zyklischen Schieberegisterversionen können durch Einzeichnen der zusätzlichen Verbindungen ergänzt werden. Z.B.: Zyklisches Rechtsschieberegister:



Die duale Addition im Rechenwerk

Aufgabenstellung: Zwei positive Zahlen, die im Arbeitsspeicher in den Zellen 1 und 2 (Adressen) stehen, sind zu addieren. Das Ergebnis ist in der Zelle 3 abzuspeichern.

Lösungsweg: Das Rechenwerk (im Falle der Addition wird der Paralleladdierer durchgeschaltet) kann den Inhalt des Akkumulators und den Inhalt einer Arbeitsspeicherzelle, die im ADD-Befehl durch ihre Adresse angesprochen wird, additiv verknüpfen. Wohin muß demnach einer der im Arbeitsspeicher stehenden Summanden gebracht werden?

- In den Akkumulator.

Daraus folgt der erste Programmschritt:
LAD w
(vergl. Befehlsliste S.13)
Das Ergebnis der Addition wird in den Akkumulator eingeschrieben. Was passiert bei weiteren Operationen mit dem Ergebnis?

- Es geht verloren.

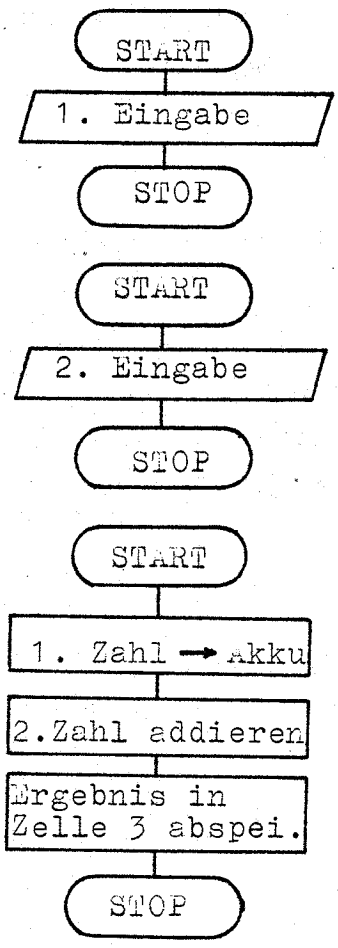
Daraus folgt der sich an die eigentliche Addition anschließende Programmschritt:
SPE w
(vergl. Befehlsliste S.13)

Bedienung:

NETZ EIN, RÜCKSETZEN
Programm, wie angegeben, stecken.
Vor dem eigentlichen Programm müssen natürlich erst die gewünschten positiven Zahlen eingegeben werden, und zwar durch die Kippschalter.
Positive Zahlen haben diese Form:
0XXX

Programm:

Programmablaufplan



Befehlsfolge

Befehls-Adr.	Operation	Adresse	Betriebsart
0	EIN	1	PROGR. 40kHz
1	STP		
2	EIN	2	
3	STP		
4	LAD	1	BEFEHL oder SCHRITT
5	ADD	2	
6	SPE	3	
7	STP		

Stoppbefehle brauchen nicht gesteckt werden. Ein freigelassener Befehlsspeicherplatz entspricht automatisch einem STP-Befehl. Die STP-Befehle im Anschluß an die EIN-Befehle sind des-

halb vorgesehen worden, damit das gesamte Programm in der Betriebsart PROGR.40kHz (schnell) ablaufen kann. Ebenso gut könnte das universellere Eingabeprogramm von S.67 gesteckt werden.

Lehrerdemonstration:

Das Beispiel dient zur Einübung der Befehle LAD, SPE und ADD, die in der Betriebsart SCHRITT vorgeführt werden. Ausgangspunkt eines jeden Befehlsablaufs ist der Befehlszähler. Mit jedem Drücken der Taste "START" wird einer von 4 Takten ausgelöst. Dabei leuchten Pfeile auf, welche den Informationsfluß illustrieren. Dadurch erläutert sich der Ablauf nahezu von selbst.

Für das Beispiel der Addition wurde eine besonders ausführliche Beschreibung dieser Vorgänge notiert. Im allgemeinen genügt jedoch die Beschreibung der Befehle in der Befehlsliste auf Seite 13 bis Seite 14

Kommentar:

Zur Darstellung soll kommen, das Zusammenwirken der Funktionseinheiten des Rechners, die Ansteuerung von Speicherplätzen durch Adressen, die binäre Befehlsdarstellung und deren Decodierung. Übungsziel ist das Verständnis der Befehlsabläufe als Voraussetzung der Programmierung.
Der Inhalt des Überlaufregisters wird in dieser Übung nicht betrachtet.
Siehe Übung "Duale Subtraktion".

Beispiel: Befehlsablauf bei der Addition (ADD)

Betriebsart: SCHRITT

Bedienung	Takt	Vorgang
START	1	Befehl lesen. Der dem Befehlszählerstand entsprechende Befehl wird angesteuert und in das Befehlsregister eingeschrieben. Dabei signalisiert das Aufleuchten der Pfeile zum und vom Programmspeicher den Signalfluss. Das Befehlsregister enthält die binäre Darstellung des gelesenen Befehls.
START	2	Befehl decodieren. Der Befehlsregisterinhalt wird getrennt nach Operations- und Adressteil decodiert. Das Durchschalten der decodierten Information wird durch die Leuchtpfeile vom Steuerwerk zum Rechenwerk und Datenspeicher signalisiert. Die angesprochene Operation leuchtet im Lampenfeld des Rechenwerks auf (in diesem Fall ADD). An der Adressleiste des Datenspeichers leuchtet die angewählte Speicheradresse auf.
START	3	Befehl ausführen. Der Inhalt der angewählten Arbeitsspeicheradresse und der Inhalt des Akkumulators werden im Rechenwerk additiv verknüpft. Das Ergebnis erscheint im Hilfsregister. In diesem Stadium sind beide Summanden und das Ergebnis (Dualzahlen) sichtbar.
START	4	Das Ergebnis wird vom Hilfsregister in den Akkumulator eingeschrieben (einer der Summanden wird überschrieben). Der Befehlszählerstand wird um 1 erhöht (entspricht der Adresse des folgenden Befehls).

Duale Subtraktion

Aufgabenstellung: Es ist die Differenz zweier Zahlen zu bilden, die im Arbeitsspeicher in den Zellen 1 und 2 stehen. Das Ergebnis soll abgespeichert werden.

Lösungsweg: In dualen Rechenwerken wird die Subtraktion mit Hilfe der Komplementbildung auf die Addition zurückgeführt. D.h. die Subtraktion besteht in einer Addition des Komplements. So werden die Gesetzmäßigkeiten einer Zahlendarstellung mit fester Stellenzahl (n Binärstellen) in idealer Weise verwertet. Diese Zusammenhänge können z.B. an Hand des sog. Zahlenrings besonders anschaulich dargestellt werden. Die Komplementbildung (2er Komplement) wird demnach - ohne dies hier erklären zu wollen - mittels einer Inversion (Negation) aller Bitstellen und einer anschließenden Addition von "Eins" auf die niederwertigste Stelle durchgeführt.

Z.B. 000I (1) ⟷ IIII (-1)
00II (3) ⟷ II0I (-3)
0II0 (6) ⟷ I0I0 (-6)

Das 1. Bit (Definition: X X X X
1. 2. 3. 4. Bit)
der negativen Dualzahlen ist in allen

Fällen mit einer "Eins" besetzt, während die positiven Dualzahlen an dieser Stelle eine "Null" haben. Man kann diese Bitstelle zur Vorzeichenstelle erklären. Eine "Eins" in der Vorzeichenstelle bedeutet dann:

1. Die Zahl ist negativ
2. Sie liegt in komplementierter Form vor.

Werden nun beispielsweise die positiven Zahlen OIOI (5) und OIIO (6) addiert, so entsteht ein mißverständliches Ergebnis, IOII. Ohne Vorzeichen kann es zwar noch richtig als 11 gelesen werden, bei jeder weiteren Addition wird diese Zahl jedoch als -5 gewertet.
Die Einführung einer Vorzeichenstelle hat den Bereich der 16 mit 4 Bitstellen darstellbaren Zahlen von 0...15 auf -8...0...+7 verschoben. Wird der Bereich der positiven Zahlen 0...7 nach oben und der Bereich der negativen Zahlen -1...-8 nach unten überschritten, so wird zur Kennzeichnung dieses Fehlers das Überlaufregister gesetzt. Der Überlauf unterscheidet sich demnach grundsätzlich vom Übertrag der Übungen "Halbaddierer" und "Volladdierer".

Diese Erläuterungen zur dualen Zahlendarstellung sind nötig, um die Ergebnisse der folgenden Übungsprogramme zur Komplementbildung und zur Subtraktion zu verstehen.

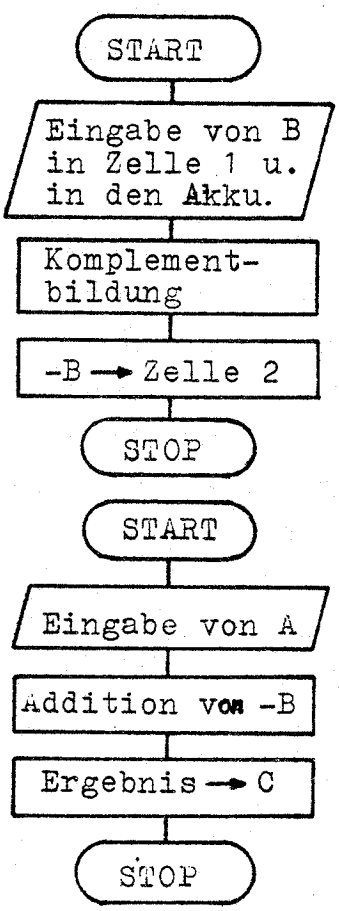
Bedienung:

NETZ EIN, RÜCKSETZEN
Programm, wie angegeben, stecken.
Das Programm ist in allen Betriebsarten ablauffähig.
Die Betriebsart SCHRITT dient zum Verständnis der Befehlsabläufe, die Betriebsart BEFEHL zur Erläuterung des Programmablaufs, die Betriebsart PROGR.40kHz zur schnellen Ermittlung von Ergebnissen.

Programm:

A - B = C

Programmablaufplan



Befehlsfolge

Befehls-Adr.	Operation	Adresse dez.	Adresse symbol.
0	EIN	1	B
1	KPL		
2	SPE	2	-B
3	STP		
4	EIN	0	A
5	ADD	2	-B
6	SPE	3	C
7	STP		

Die Adresse "0" braucht nicht gesteckt zu werden, da ein nicht belegter Adreßplatz keine "Einsen" enthalten kann und automatisch einer "Null" entspricht.

Lehrerdemonstration: Zunächst werden beliebige Zahlen durch die Kippschaltereingabe eingegeben und anschließend nur das 1. Programm zur Komplementbildung durchlaufen.
(Mit Hilfe der Taste RÜCKSETZEN kann das Programm immer wieder am Anfang begonnen werden.)

Im 2. Programm wird die komplementierte Zahl zu einer anderen einzugebenden Zahl addiert, was der Subtraktion entspricht.

Die beiden Programme dienen zum Verständnis der dualen Zahlendarstellung mit Vorzeichen (Festkommazahlen). Außerdem wird die Programmierung einer Subtraktion erlernt.

Kommentar: Üblicherweise ist in normalen Rechnern ein Subtraktionsbefehl (SUB) vorgesehen, der einen Komplementierer vor dem Addierwerk zwischenschaltet. Dadurch würde der Ablauf einer Subtraktion nach außen verschleiert werden. Hier wurde bewußt auf einen SUB-Befehl verzichtet, um die Funktion klar herauszustellen. Mit symbolischen Adressen programmiert, lautet ein übliches Programm für A - B = C so:

```
LAD   A
SUB   B
SPE   C
```

In unserem Fall würde man geschickterweise mit B anfangen, komplementieren und addieren, so daß sich der Aufwand nur um einen Befehl erhöht.

```
LAD   B
KPL
ADD   A
SPE   C
```

$\triangleq \quad C = -B + A$

Das angegebene Übungsbeispiel könnte in diesem Sinn umgeformt werden. Es müßte jedoch ein geeignetes Eingabeprogramm vorgeschaltet werden (vergl. S. 67).

Duale Addition mit gekoppelten Arbeitsspeicherzellen

Aufgabenstellung:

Gegeben sind zwei positive Zahlen, von denen jede zwei Datenworte umfaßt. Sie sind in den ASP-Zellen A1 (Adresse 0) und A2 (Adresse 1) sowie B1 (Adresse 2) und B2 (Adresse 3) enthalten.
Jede Zahl besteht aus 6 Betragsstellen. Die Bitstellen 1 enthalten das Vorzeichen.
Zum Beispiel:

A1	0	X	X	X
		2^5	2^4	2^3
A2	0	X	X	X
		2^2	2^1	2^0

Beide Zahlen sollen addiert werden.
Das Ergebnis ist in Zelle C1 (Adresse 6) und C2 (Adresse 7) abzuspeichern.

Lösungsweg:

Welche Zahlen von A und B sind zunächst zu addieren?

- A2 und B2 (niederwertige Stellen).

Welche Bedeutung hat ein möglicher Überlauf?

- Ein Überlauf entspricht der 2^3 und muß in C1 durch Addition von 1 eingetragen werden.

Was muß getan werden, damit die Inhalte von C1 und C2 weiter als positive Zahlen erkannt werden können?

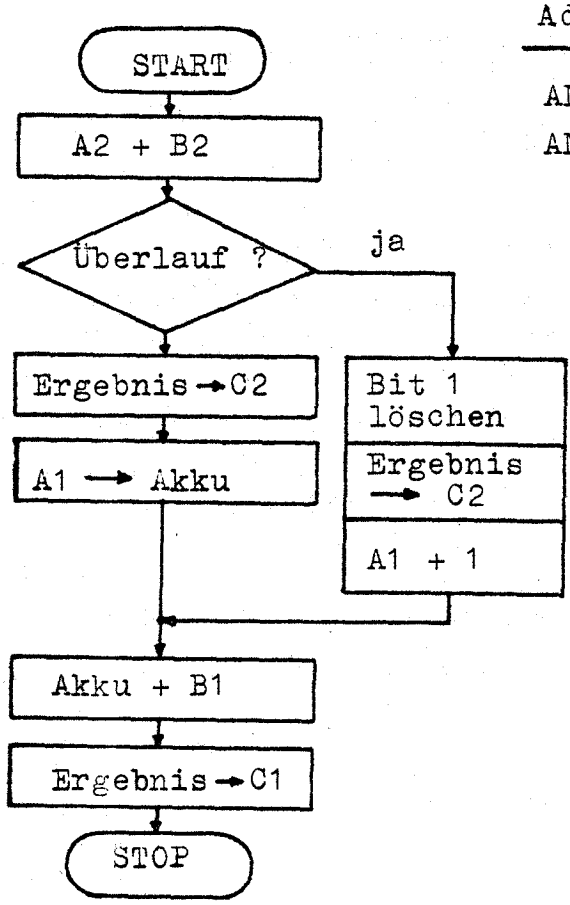
- Die 1. Bitstelle muß auf Null gesetzt werden.

Bedienung:

NETZ EIN, RÜCKSETZEN
Programm, wie angegeben, stecken.
Dateneingabe durchführen.
Betriebsart wählen, Taste START drücken.

Programm:

Programmablaufplan



Befehlsfolge

Befehls-Adr.	Operation	Adresse dez.	Adresse symb.
ANF0 → 0	EIN	7	Adresse
ANF1 → 1	EIN	(7)	Inhalt
2	STP		
3	LAD	1	A2
4	ADD	3	B2
5	SUL	11	
6	SPE	7	C2
7	LAD	0	A1
8	ADD	2	B1
9	SPE	6	C1
10	STP		
11	UND	4	MASKE
12	SPE	7	C2
13	LAD	0	A1
14	ADD	5	EINS
15	SPR	8	

Arbeitsspeicher-belegung:

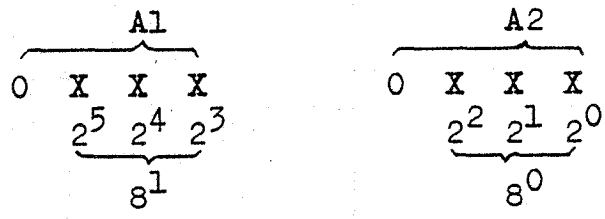
ASP-Adresse	Inhalt	
	dual	sympolisch
0	beliebig	A1
1	beliebig	A2
2	beliebig	B1
3	beliebig	B2
4	0III	MASKE
5	000I	EINS
6	beliebig	C1
7	beliebig	C2

Lehrerdemonstration:

Die Aufgabe zeigt, dass die Binärstellenzahl eines Datenworts (in diesem Fall 4 Bit) nicht notwendigerweise eine Beschränkung der darstellbaren Zahlenmenge zur Folge hat.

Die hier behandelte Verknüpfung und Zusammenfassung zweier Datenworte (Doppelwortverarbeitung) ist in fast allen grösseren Datenverarbeitungsanlagen vorgesehen und wird dort als das sogenannte "Doppelwortformat" bezeichnet. Sie dient neben der Vergrösserung des mit Festpunktzahlen darstellbaren Zahlenbereichs zur Erhöhung der Rechengenauigkeit. Die Rechengenauigkeit wird dabei durch Vermehrung der möglichen Stellen nach dem Komma erhöht. Festpunktzahlen muss man sich nämlich nicht unbedingt als ganze Zahlen vorstellen, das Komma kann an einer beliebigen, aber festen Stelle stehen. Ausserdem wird die Rechengenauigkeit bei der sogenannten "Gleitpunktrechnung" erhöht, wenn man die Stellenzahl der Mantisse erhöht. (Bei der Gleitpunktrechnung werden Zahlenwert (Mantisse) und Hochzahl (Exponent) getrennt gespeichert und getrennt bearbeitet.

An dieser Stelle kann auch das Oktalsystem angesprochen werden, wenn man die 3 besetzten Bits von A2 als 8^0 und A1 als 8^1 bezeichnet (binär codiertes Oktalsystem):



z.B. 0 I 0 I / 0 I 0 0 = 54 (oktal)

Das Löschen von Bitstellen mit dem UND-Befehl bzw. die Intersektion sollte als grundsätzliche Programmiermethode erwähnt werden.

Kommentar:

Es ist durchaus möglich, ein einfaches Beispiel zur Darstellung von Zahlen, getrennt nach Betrag (Mantisse) und Hochzahl (Exponent) zu schreiben.

Das vorliegende Beispiel ist ein erster Schritt zu einer solchen Methode.
In diesem Fall müssten die Mantissen (1. Datenwort) in Abhängigkeit von den zwei Exponenten (2. Datenwort) verschoben (normiert) werden und dann könnte z.B. eine Addition der Mantissen durchgeführt werden. Damit könnten sogar die Prinzipien der Gleitkommadarstellung demonstriert werden.

Duale Multiplikation

Aufgabenstellung:

Zwei positive Zahlen sollen multipliziert werden.
Zur Erleichterung der Aufgabe werden folgende Vereinbarungen festgelegt:

A	mal	B	=	C
0xxx		0xxx	=	xxxx

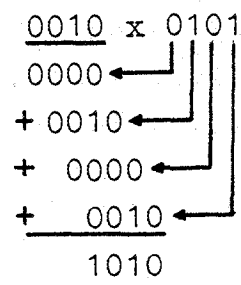
↑ kein Vorzeichen

Das 1. Bit des Ergebnisses soll nicht als Vorzeichen gewertet werden, so dass die Zahlen von 0 bis 15 darstellbar sind.

Lösungsweg:

Die Multiplikation könnte einfacherweise auf eine fortgesetzte Addition zurückgeführt werden. Ein derartiges Programm (oder eine elektronische Schaltung) addiert A so lange auf, bis B durch jeweilige Subtraktion von Eins auf Null herabgezählt ist. Bei sehr grossen Zahlen müsste man dabei jedoch sehr lange Rechenzeiten einkalkulieren. Üblicherweise wird deshalb ein geeigneterer Multiplikationsalgorithmus verwendet. Wie beim Multiplizieren von Hand besteht das Programm aus einer Folge von Additionen und Verschiebungen wie aus dem folgenden Beispiel ersehen werden kann.

Zum Beispiel:



2 x 5 = 10

Wann wird der Multiplikand (0010) addiert?

- wenn der Multiplikator an der betroffenen Bitstelle mit "Eins" besetzt ist.

Wann wird das "halbfertige" Ergebnis verschoben?

- unabhängig von der Besetzung des Multiplikators nach jeder Bitstelle bis zur letzten Multiplikatorstelle, wo keine Verschiebung mehr stattfindet.

Programm:

Programmbeschreibung: 1.) Dateneingabe

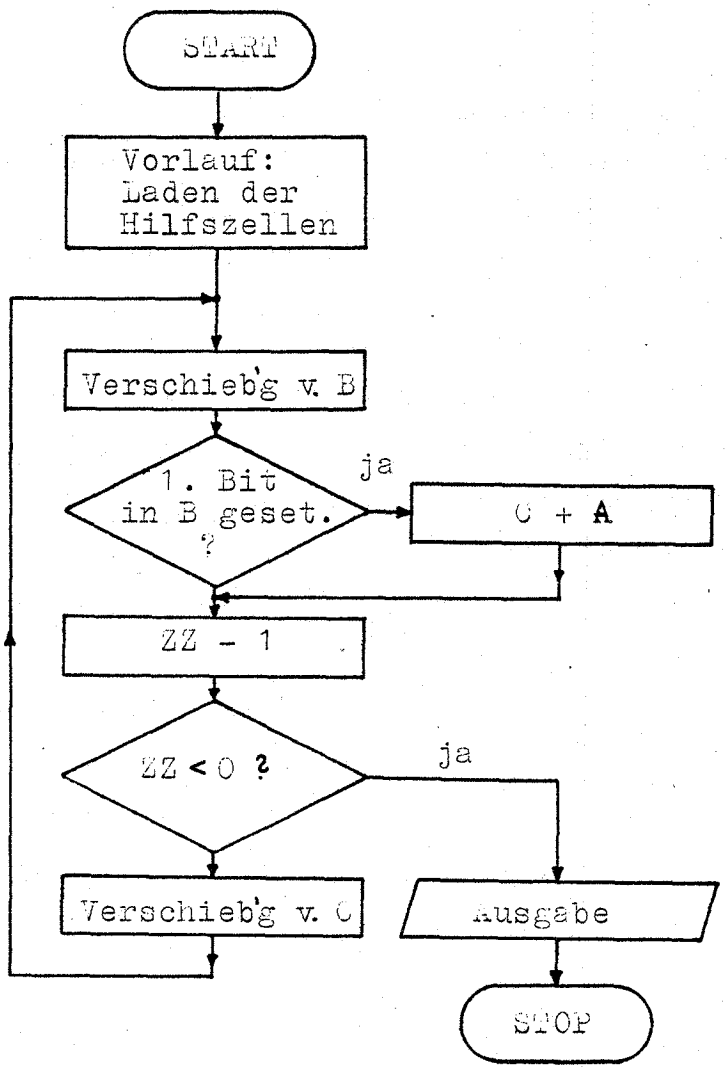
Die zu multiplizierenden Zahlen und die Konstanten des Programms werden durch Vorwahl der Adresse (ANF0 drücken!) und durch anschließende Eingabe des Inhalts in die gewählte Adresse (ANF1 drücken!) mittels indirektem Speicherzugriff (Substitution, siehe auch Seite 63-65) eingegeben.

2.) In einem Programmvorlauf müssen die Hilfszellen des Programms geladen werden, z.B. muß die Ergebniszelle C vor jedem Programmlauf auf Null gesetzt werden.

3.) Die Abfrage der Bitstellen des Multiplikators erfolgt durch den SAM-Befehl. Die 2. bis 4. Bitstelle müssen durch Verschieben in die 1. Bitstelle gebracht werden.

4.) Die Anzahl der Verschiebungen wird von einem Zähler begrenzt, der auf Null herabgezählt wird und bei Null (SGN-Befehl) gestoppt wird.

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse	
		dez.	symb.
ANF0→0	EIN	7	Adresse
ANF1→1	EIN	(7)	Inhalt
2	STP		
3	LAD	0	NULL
4	SPE	1	C
5	LAD	4	DREI
6	SPE	5	ZZ
7	LAD	3	B
8	VLL		
9	SPE	7	HZB
10	SAM	20	
11	LAD	5	ZZ
12	ADD	6	KM1
13	SPE	5	ZZ
14	SGN	24	
15	LAD	1	C
16	VLL		
17	SPE	1	C
18	LAD	7	HZB
19	SPR	8	
20	LAD	1	C
21	ADD	2	A
22	SPE	1	C
23	SPR	11	
24	AUS	1	C

Arbeitsspeicher-belegung:

ASP-Adresse	Inhalt	
	dual	symbolisch
0	0000	NULL
1	belieb.	C
2	0XXX	A
3	0XXX	B
4	00II	DREI
5	belieb.	ZZ (Zählzelle)
6	IIII	KM1 (Konstante minus 1)
7	belieb.	HZB (Hilfszelle für B)

Bedienung:

NETZ EIN, RÜCKSETZEN
Programm, wie angegeben, stecken.
Arbeitsspeicherbelegung durch Eingabe erzeugen.
Zahlen A und B eingeben.
Das Programm ist in allen Betriebsarten ablauffähig.

Lehrerdemonstration:

Das Programm dient zur Demonstration des Multiplikationsalgorithmus. Dieser Algorithmus muß nicht notwendigerweise durch eine technische Schaltung (Ablaufsteuerung) verwirklicht werden, er kann genau so gut durch ein Programm nachgebildet werden.

Es zeigt sich, daß diese wenigen Maschinenbefehle, die nur Mikroschritte darstellen, ausreichend sind für den Aufbau immer komplizierterer Abläufe.
Damit kann auch erklärt werden, weshalb DV-Anlagen in der Lage sind, mit den 4 Grundrechenarten ADD, SUB, MLT und DIV Aufgaben aus dem gesamten Bereich der höheren Mathematik zu berechnen.
Ebenso, wie die Multiplikation auf Mikrobefehle zurückgeführt wird, werden höhere Funktionen mit den 4 Grundrechenarten bewältigt, indem man noch kompliziertere Programme aufstellt und sich der Methoden der numerischen Mathematik (Reihenentwicklungen usw.) bedient.

Der Multiplikationsalgorithmus kann in der Betriebsart "8 Hz" besonders gut beobachtet werden, wenn man die 3 Zellen für A, B und C im Arbeitsspeicher betrachtet. Darüberhinaus ist die Programmierung beispielgebend für die Anwendung von Maschinenbefehlen.

Kommentar:

Die Nachbildung komplizierterer Abläufe (Befehle, wie z.B. Multiplikation, MLT) wird bei kleineren und mittleren DV-Anlagen häufig angewendet. Da jedes erstmals aufgestellte Programm übersetzt werden muß (Assembler oder Compiler), kann das Übersetzerprogramm die Aufgabe haben, für den MLT-Befehl (Multiplikationsbefehl) oder das Multiplikationszeichen, "*", automatisch das Multiplikationsprogramm einzufügen (zu "generieren"). Da ein Programm jedoch im

allgemeinen eine Vielzahl solcher MLT, DIV, ...-Befehle enthält, werden die zugehörigen Simulationsprogramme in das Betriebssystem eines Rechners übernommen, wo sie dann nur einmal gespeichert sind. Ein jeweils auftretender MLT-Befehl führt dann zu einem Sprung in das Betriebssystem. Das heißt also, jede gute DV-Anlage wird von Anfang an mit einem Programmpaket, dem Betriebssystem geladen, das einen Teil des Arbeitsspeichers (eventuell auch Externspeicher) belegt und damit zwar den verfügbaren Speicherplatz verringert; mit Hilfe der in ihm enthaltenen Simulationen und Organisationsroutinen (z.B. für die Ein- und Ausgaben der verschiedenen angeschlossenen Geräte) jedoch,erweitert es die Möglichkeit solcher Rechner erheblich, bzw. es ermöglicht erst den Betrieb der Anlage (Programme zum Laden, Einlesen von Anwenderprogrammen,und Übersetzerprogramme). Ein solches gut ausgestattetes Betriebssystem steigert ohne großen Kostenaufwand die gesamte Leistungsfähigkeit eines Rechners. Es gilt lediglich die Einschränkung, daß der erhöhte Komfort solcher Rechneranlagen durch verlängerte Bearbeitungszeiten erkauft werden muß, was aber bei den hohen Verarbeitungsgeschwindigkeiten moderner Rechner häufig nicht mehr von Bedeutung ist.

Eine Zeitlang verlief die Rechnerentwicklung in Richtung auf leistungsfähigere vielfältige Einzelbefehle. Heute zeichnet sich eine neue Richtung ab: Beschränkung auf wenige einfache Befehle, wie bei diesem Modell. Die dabei nötig werdenden Simulationsprogramme für kompliziertere und vielseitigere Befehle und Makrobefehle werden dann in hochintegrierter Form in Festkörperbausteinen (Chips) gespeichert. Da es möglich ist, mehrere tausend solcherBefehle auf einem Chip ($1x2cm^2$) zu speichern, ist die Aussicht auf ganze Betriebssysteme, die in dieser Form aufgebaut zur Hardwarelieferung gehören würden, in den Bereich des Möglichen gerückt.

Codierung, Umcodierung

Aufgabenstellung:

Nach der Darstellung des BCD-Codes, der aus dem Dualcode direkt abgeleitet ist, sollen andere Codes vorgestellt werden und die Übergänge zwischen Codes demonstriert werden.

Oder: Nach der Darstellung der Codes wird die Aufgabe gestellt, ein Umcodierungsprogramm zu entwerfen, da die Programmierung in diesem Fall sehr leicht ist.

Lösungsweg:

Zahlen werden im BCD-Code eingegeben.
Die Umcodierung erfolgt durch Programme, welche die einfachen Zusammenhänge zwischen den Codes, die aus der Codetabelle ersichtlich sind, realisieren.

	Dezimalziffer		
Codewort	BCD	Stibitz (Exzess-3)	Aiken
0000	0		0
000I	1		1
00I0	2		2
00II	3	0	3
0I00	4	1	4
0I0I	5	2	
0II0	6	3	
0III	7	4	
I000	8	5	
I00I	9	6	
I0I0		7	
I0II		8	5
II00		9	6
II0I			7
III0			8
IIII			9

Bedienung: NETZ EIN, RÜCKSETZEN

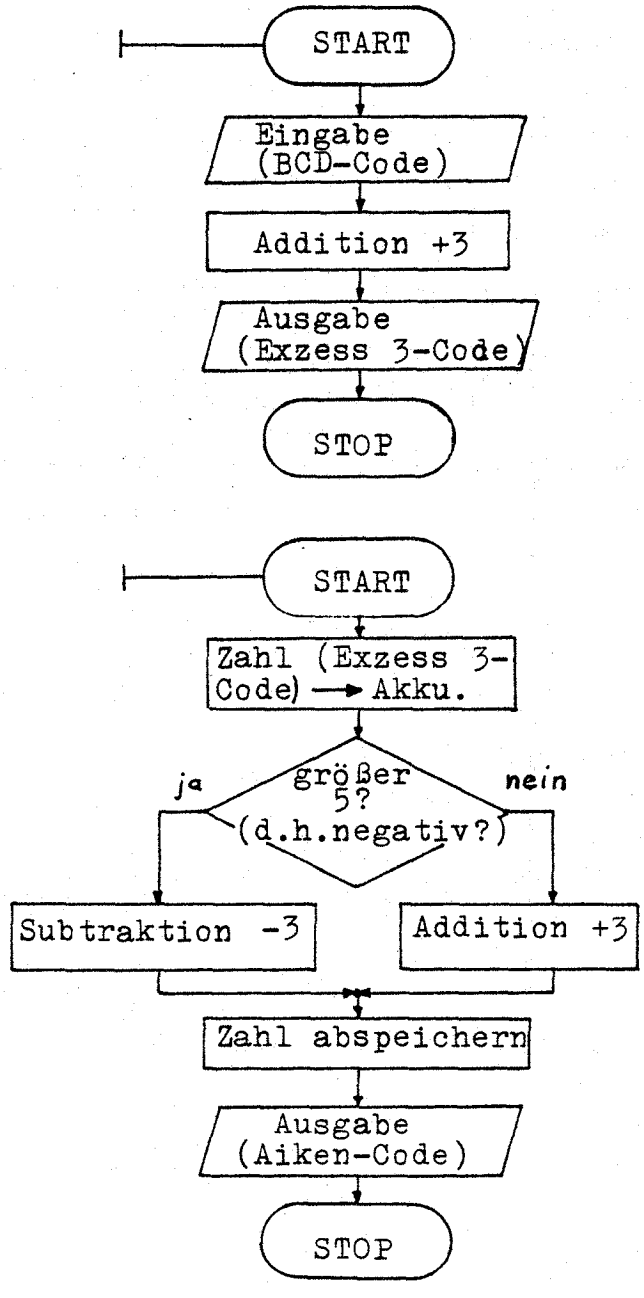
Betriebsart: PROGRAMM 40kHz

Die Zahlen OOII (3) in Adresse 2 und IIOI (-3) in Adresse 3 (EIN-Befehl und Kippschalter) eingeben.

Programm wie angegeben stecken.

Programm:

Programmablaufplan



Befehlsfolge

Befehls-Adr.	Operation	Adresse	
		(Kippschalter einstel.!)	
0	EIN	0	(belieb.)
1	ADD	2	(OOII)
2	SPE	1	
3	AUS	1	
4	STP		
5	LAD	1	
6	SAM	11	
7	ADD	3	(IIOI) (-3)
8	SPE	4	
9	AUS	4	
10	STP		
11	ADD	2	(OOII) (+3)
12	SPR	8	

Demonstration: Gewünschte BCD-Zahl über Kippschalter eingeben. Programmabschnitte über die Funktionstaste START auslösen.
Die Ergebnisse werden mit der Tabelle verglichen.

Kommentar: Vermittelt werden soll die Erkenntnis, daß es nur auf die Vereinbarung ankommt, wie eine binäre Kombination zu verstehen ist. An diese Übung müssen sich Demonstrationen anschliessen, welche die Vorteile der Codearten aufzeigen, z.B. die Eigenschaft der Selbstkomplementierung (Durchführung der Subtraktion).

Das Programm kann auch erweitert werden durch eine Kontrolle der eingegebenen BCD-Zahlen auf ihre Gültigkeit (Zahlen größer als die duale IOOI sind nicht zugelassen, Pseudotetraden).

Anmerkung: Da die Frage nach der Darstellung (Codierung) von Buchstaben im Rechner beantwortet werden muß, ist festzustellen, daß Buchstaben im Rechner grundsätzlich wie Zahlen, nämlich binär codiert, dargestellt und wie Dualzahlen behandelt werden.

Um alle Buchstaben und Sonderzeichen darzustellen, sind jedoch mehr als 4 Bit je Zeichen erforderlich, meist 6 oder 8 Bit (Byte) je Zeichen.
Mit den zur Verfügung stehenden 4 Bit, die 16 Verschlüsselungen zulassen, können nur ein Teil des Alphabets, z.B. die Buchstaben A, B, C ... bis P oder z.B. die Zahlen 1 bis 9 zusammen mit den Buchstaben A bis F dargestellt werden. Diese Buchstaben werden vom Rechner wie Zahlen behandelt und lediglich vom Ausgabegerät, z.B. nach Umschaltung auf Buchstabenausgabe, als Buchstaben ausgedruckt.

Paritybiterzeugung

Aufgabenstellung:

Die Quersumme eines Binärworts "A" soll per Programm auf Parität ergänzt werden. D.h. bei ungerader Quersumme der "Einsen" eines Binärwortes soll eine bestimmte Bitstelle mit "Eins" besetzt werden; wenn die Quersumme gerade ist, soll die Bitstelle gleich "Null" bleiben.
In unserem Falle soll jeweils ein drei Bit breites Datenwort (XXX0) ← Paritybit auf Parität ergänzt werden.

Lösungsweg:

Die Lösung kann durch folgende Fragen erarbeitet werden:

1.) Wie kann die Quersumme des Binärworts gebildet werden?
   - Addition der Binärstellen
   - Addition von +1, falls eine Binärstelle gleich "Eins" ist.

2.) Wie stellt man fest, ob eine "Eins" in einem Binärwort gesetzt ist?
   - Verschieben und auf 1. Bit (SAM) prüfen, oder
   - Maske (UND) bilden und auf Null (SGN) abfragen.

3.) Wie stellt man fest, ob die Quersumme gerade oder ungerade ist?

- Letztes Bit einer Dualzahl betrachten.

4.) Wie kann die Prüfung des Datenworts abgeschlossen werden (Erkennung des Endes)?

- Zähler für jeden Abfragezyklus.

5.) Wie kann das Paritybit in das Datenwort eingetragen werden?

- Einodern (ODR).

Bedienung:

NETZ EIN
Programm, wie angegeben, stecken.
Arbeitsspeicherbelegung durch Eingabe von Adresse und Inhalt in der Betriebsart PROGR.40kHz oder Befehl erzeugen.
Z.B. Eingabe der Konstante K0 (0000) in Zelle 1:

1.) Kippschalter auf 000I stellen und ANF0 drücken.
2.) Kippschalter auf 0000 stellen und ANF1 drücken.

Programmablauf in verschiedenen Betriebsarten beobachten.

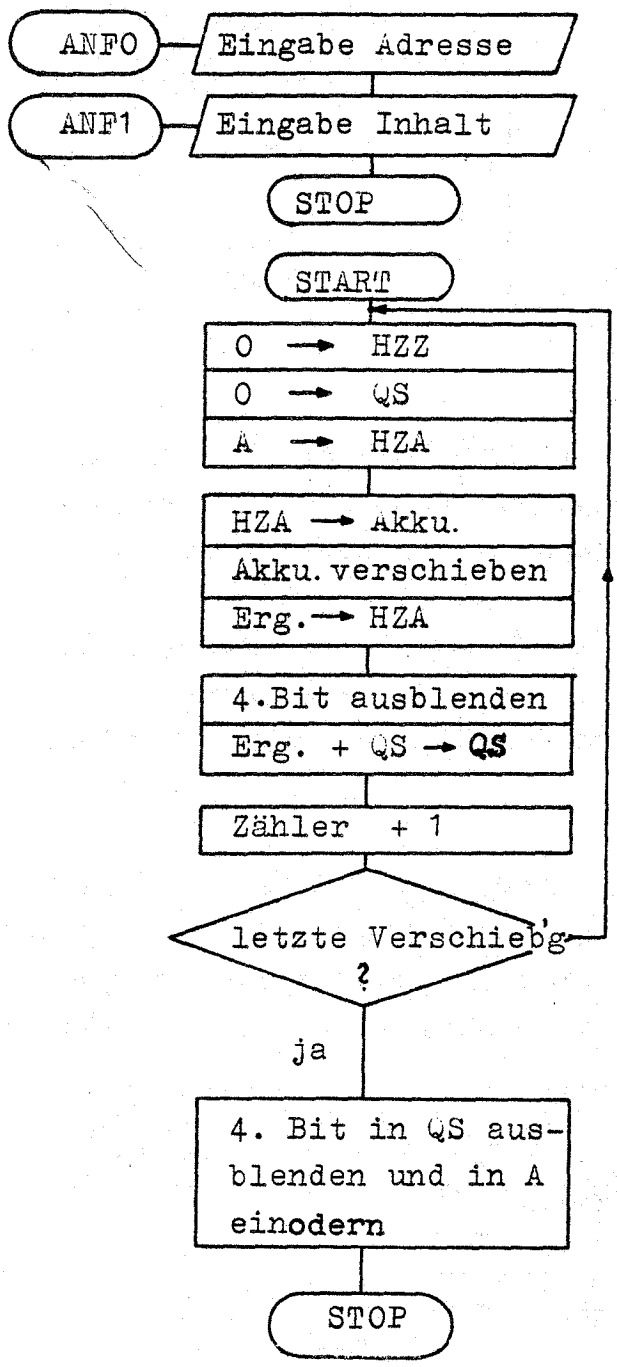
Anmerkung:

Die auf der nächsten Seite verwendeten Abkürzungen haben folgende Bedeutung:

K0 ... Konstante 0
HZZ ... Hilfszelle Zähler
QS ... Quersumme
HZA ... Hilfszelle A
M ... Maske
K1 ... Konstante 1

Programm:

Programmablaufplan



Befehlsfolge

Befehls-Adr.	Operation	Adresse dez.	Adresse symb.
0	EIN	7	
1	EIN	(7)	
2	STP		
3	LAD	1	KO
4	SPE	6	HZZ
5	SPE	4	QS
6	LAD	0	A
7	SPE	5	HZA
8	LAD	5	HZA
9	VLR		
10	SPE	5	HZA
11	UND	2	M/K1
12	ADD	4	QS
13	SPE	4	QS
14	LAD	6	HZZ
15	ADD	2	K1
16	SPE	6	HZZ
17	KPL		
18	ADD	3	Z
19	SAM	21	
20	SPR	8	
21	LAD	4	QS
22	UND	2	M/K1
23	ODR	0	A
24	SPE	0	A
25	STP		

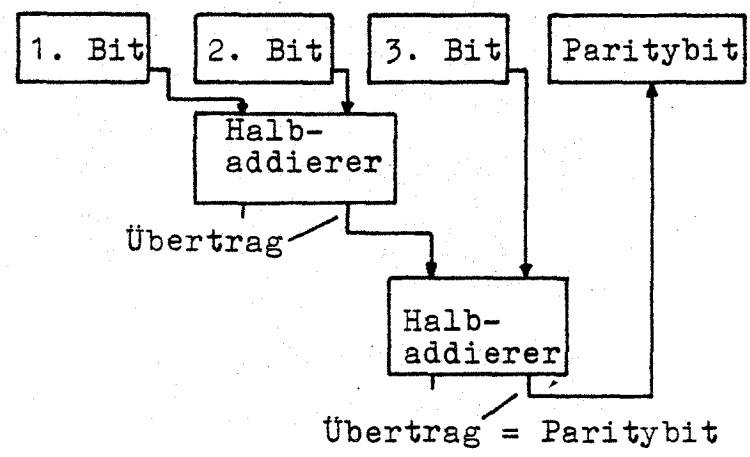
Das Ergebnis kann in der Arbeitsspeicherzelle 0 betrachtet werden.

Arbeitsspeicher-belegung:

ASP-Adresse		Inhalt	
dez.	symbol.		
0	A	XXX0	← Datenwort
1	K0	0000	} Paritybit-erzeugung
2	M/K1	000I	} Konstanten
3	Z	00II	} Konstanten
4	QS	bel.	} Hilfs-zellen
5	HZA	bel.	} Hilfs-zellen
6	HZZ	bel.	} Hilfs-zellen

Lehrerdemonstration: Es besteht die Möglichkeit, lediglich Daten einzugeben und (in der Betriebsart PROGR.40kHz die Ergebnisse zu demonstrieren und zu kommentieren. Andererseits können die Programmschritte aus dem Abschnitt "Lösungsweg" (Betriebsart BEFEHL) vorgeführt werden.

Kommentar: Die Bildung eines Paritybits wird natürlich meistens per "hardware", d.h. durch z.B. elektronische Bauelemente, vorgenommen, aber in ganz ähnlicher Weise:



Die Paritybitbildung durch ein Programm (d.h. per "software") ist ein Beispiel für die häufig angewendete Methode "hardware"-Strukturen durch "software" zu erzeugen.

Der Sinn und Zweck des Paritybits besteht darin, eine Fehlerkontrolle z.B. im Anschluß an Datentransfers durchführen zu können: Hat sich ein Bit geändert, so kann dieser Fehler an Hand des Paritybits erkannt werden. Eine Erweiterung des vorliegenden Programms könnte nun darin bestehen, eine Kontrolle von mit Paritybit versehenen Daten vorzunehmen und einen Fehler anzuzeigen.

An dieser Stelle wäre auch der Begriff "Redundanz" zu erklären. Die Redundanz kennzeichnet einen "Überschuß an Information, der nicht unbedingt nötig ist, so daß z.B. eine Nachricht noch aus ihren Bruchstücken erkannt werden kann. Das erwähnte Paritybit (es gibt natürlich auch ein Unparitybit) stellt eine solche Redundanz dar. Redundanzen werden in der Technik entweder aus Kostengründen vermieden oder aber zur Informationssicherung verwendet.

Darüber hinaus könnte das Beispiel auch Anlaß sein, auf die weiterführenden Fragen der Informationssicherung einzugehen. Z.B. wären dann fehlerkorrigierende Codes zu erläutern. Das sind binäre Verschlüsselungen von Ziffern und Buchstaben mit größerer als notwendiger Bitstellenzahl, so daß eine Fehlererkennung und Fehler-

korrektur möglich wird, weil von den insgesamt möglichen Binärkombinationen einer Bitstellenzahl nur eine Auswahl (Untermenge) von "gleichabständigen" Kombinationen verwendet wird. Da in den meisten Fällen nur ein Bit eines Datenwortes fehlerhaft sein wird, ist dann eine eindeutige Fehlererkennung bzw. sogar eine Fehlerkorrektur möglich. Der dabei entscheidende Begriff ist die sog. "Hamming - Distanz", die den Bereich der nicht zugelassenen fehlerhaften Binärkombinationen um eine echte Kombination herum bis zu der benachbarten zugelassenen Kombination kennzeichnet.

Mathematische Aufgabe 1

Auf die Ausarbeitung komplizierter mathematischer Aufgaben wurde hier kein besonderer Wert gelegt, da es nicht die Aufgabe eines Schulungsrechners sein kann, die Mathematik zu erleichtern, sondern die Prinzipien von Digitalrechnern und die Prinzipien der Programmierung darzulegen.

Aufgabenstellung:

$$x = \frac{a + b}{2}$$

Das Programm zur Mittelwertbildung ist aufzustellen und zu demonstrieren (a und b sind positive Zahlen).

Lösungsweg:

Es muß eine eindeutige Verarbeitungsvorschrift in Form einer Befehlsfolge (Programm) aufgestellt werden, nach welcher die Rechnerautomatik die gestellte Aufgabe schrittweise lösen kann. Daraus folgt: Zergliederung (Analyse) der Aufgabe und Aufbau (Synthese) eines Algorithmus unter Benützung eines übersichtlichen Programmablaufplans, aus dem das Maschinenprogramm leicht abgeleitet werden kann.

Bedienung:

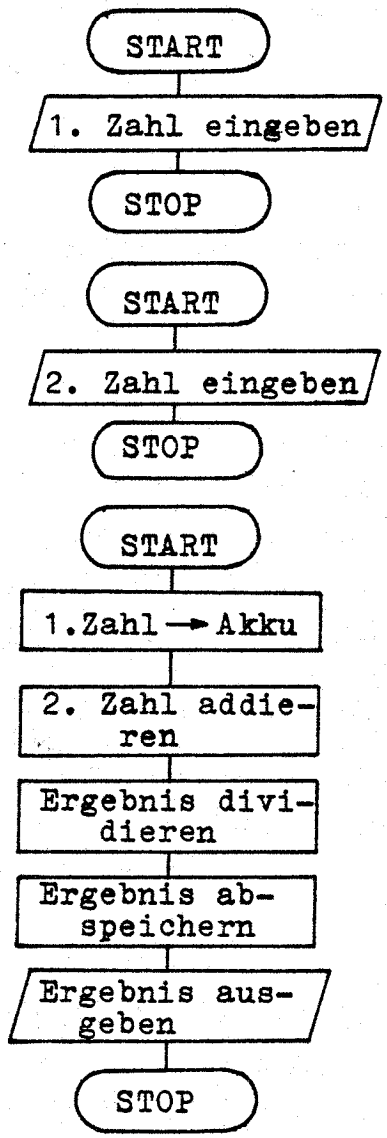
NETZ EIN, RÜCKSETZEN
Betriebsart: BEFEHL (zur Demonstration des Programmablaufs), PROGR 40kHz (beim Rechnen weiterer Zahlenbeispiele, zur Betrachtung des Ergebnisses).
Programm, wie angegeben, stecken.

Programm:

Programmbeschreibung:

Die Zahlen a und b (Adressen 0 und 1) müssen zunächst über Kippschalter und Eingabeprogramm eingegeben werden. Zur Verarbeitung muß eine der Zahlen in den Akkumulator gebracht werden, wo die Addition und die Division durch 2 (in diesem Fall Verschiebung um 1 Stelle nach rechts) erfolgt. Das Ergebnis wird ausgegeben.

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse (Inh.)
	(Kippschalter einstellen)	
0	EIN	0 (1. Zahl)
1	STP	
	(Kippschalter einstellen)	
2	EIN	1 (2. Zahl)
3	STP	
4	LAD	0 (1. Zahl)
5	ADD	1 (2. Zahl)
6	VLR	
7	SPE	2 (Ergebnis)
8	AUS	2
9	STP	

Lehrerdemonstration: Das Ineinandergreifen und die Notwendigkeit der einzelnen Schritte wird erklärt. Das Auftreten von Rundungsfehlern bzw. die Verminderung der Rechengenauigkeit bei beschränkter Stellzahl in Digitalrechnern wird demonstriert und besprochen.

Mathematische Aufgabe 2

Aufgabenstellung: $x = \sum_{\nu=0}^{n} a_\nu$

Eine Vielzahl von Zahlen (positiv und negativ), die in einem Arbeitsspeicherbereich stehen (hineingebracht wurden) ist zu addieren. In diesem Fall die ersten 4 Zahlen.

Lösungsweg: Die Programmierung wird wesentlich erleichtert und universell gehalten, wenn man in einer Schleife programmiert. Dazu muß eine Adreßrechnung durchgeführt werden. Das heißt, der Befehl ADD muß sich bei jedem Durchlauf in der Schleife auf die nächst höhere (bzw. nächst niedrige) Adresse (nächste Zahl) beziehen. Dies wird durch die sogenannte Substitution ermöglicht. Der Befehl"ADD" bezieht sich dann nicht auf die direkt angesprochene Adresse, sondern auf die Adresse, die in der zunächst angesprochenen hinterlegt ist. Die indirekt hinterlegte Adresse kann so nach jedem Durchlauf erhöht oder erniedrigt werden.

Bedienung: NETZ EIN, RÜCKSETZEN
Betriebsart: BEFEHL
(zur Demonstration des Programmablaufs) bzw.
PROGRAMM 8 Hz zur Beobachtung des Schleifendurchlaufs (Befehlsspeicheradressen verfolgen) oder PROGRAMM 40 kHz zum Rechnen von Beispielen.
Programm wie angegeben stecken.

Programm:

Programm-beschreibung:

1) Eingabe der Konstanten und Summanden des Programms entsprechend der "Arbeitsspeicherbelegung" mit dem Eingabeprogramm. (vergl. S.67) 1. Eingabe der Adresse durch ANF0. 2. Eingabe des Inhalts durch ANF1.

2) Belegung einer Hilfszelle mit der Zahl 1, in der die Adreßrechnung durchgeführt wird.

```
LAD   6   (000I)
SPE   7   (Hilfszelle)
```

3) Diese Hilfszelle wird bei der Addition substituiert angesprochen.

```
ADD  (7)
```

Dadurch wird nicht der Inhalt der Adresse 7 addiert, sondern der Inhalt der Adresse, die in der Adresse 7 gefunden wird. In diesem Fall wird zunächst die Adresse 1, d.h. die 2. Zahl addiert.

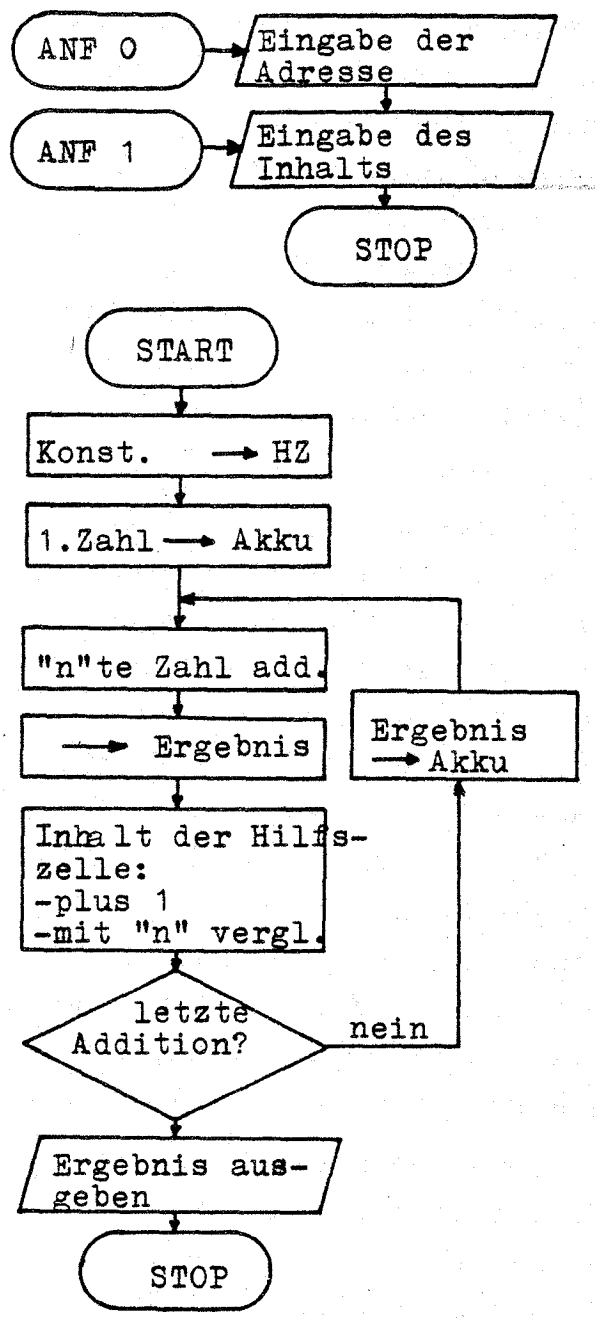
4) Erhöhung des Inhalts der Hilfszelle um 1 (nächste Zahl)

```
LAD   7   (Hilfszelle)
ADD   6   (000I)
SPE   7   (Hilfszelle)
```

(Programmierter Zähler)

5) Abfrage ob schon die "n"te Zahl, d.h. die letzte Zahl, addiert wurde. Dazu wird die Hilfszelle mit der Zahl "n" subtrahiert (durch Komplementbildung und Addition) und auf Null abgefragt. Bei Null erfolgt kein weiterer Durchlauf der Schleife (Rücksprung) mehr. (Programmierte Zählerabfrage).

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse dezimal	(Inhalt) symbolisch
0	EIN	7	Adresse
1	EIN	(7)	Inhalt
2	STP		(Eingabe entsprechend ASP-Belegung)
3	LAD	6	Konstante 1 000I
4	SPE	7	Hilfszelle
5	LAD	0	1. Zahl
6	ADD	(7)	1./2./3. Zahl
7	SPE	4	Ergebnis
8	LAD	7	Hilfszelle
9	ADD	6	Konstante 1 000I
10	SPE	7	
11	KPL		
12	ADD	5	"n"
13	SGN	16	
14	LAD	4	Ergebniszelle
15	SPR	6	
16	AUS	4	Ergebnis
17	STP		

ASP-Belegung:

ASP-Adresse	Inhalt dual	symbolisch
0	beliebig	1. Zahl
1	"	2. "
2	"	3. "
3	"	4. "
4	"	Ergebnis
5	00I0	"n" (2≤n≤4)
6	000I	Konstante 1
7	beliebig	Hilfszelle

Lehrerdemonstration: Das wesentlichste Lehrziel dieser Aufgabe ist die Erläuterung der Adreßrechnung und die daraus resultierende Möglichkeit der Abarbeitung eines ganzen Datenbereichs in einer einzigen Schleife mit Hilfe eines programmierten Zählers, der bei jedem Durchlauf abgefragt werden muß. Daraus wird auch ersichtlich, daß Daten, die in gleicher Weise bearbeitet werden sollen, in Blöcken eingegeben werden müssen.

Das Programm zeigt außerdem, daß ein solches Programm universell ist, das heißt, der Laufbereich kann vor jedem Programmstart ohne eine Änderung des Programms selbst durch eine einfache Dateneingabe geändert werden. (Ebenso können natürlich auch die zu bearbeitenden Daten durch Eingabe geändert werden).
Der Laufbereich "n" könnte nun auch in Erweiterung der bisherigen Aufgabe einer Plausibilitätskontrolle unterworfen werden, da "n" zumindest nicht ≤ 0 sein darf. Es würde sich in diesem Fall ein Sprung auf den Programmanfang (Bef.-Nr. 2) empfehlen.

Diese Lehrinhalte können mit den verschiedenen Betriebsarten hervorragend demonstriert werden. So läßt sich der Schleifendurchlauf in der Betriebsart "8 Hz" (pro Sekunde 2 Befehle) an der Lampenleiste des Befehlsspeichers (die Lampen laufen durch bzw. springen entsprechend der Programmierung) leicht verfolgen. Die 5 Schleifendurchläufe dauern dann 20 Sek., wobei sich die Zählzelle alle 4 Sekunden um 1 erhöht. In der Betriebsart 2 Hz wird dieser Zeitablauf um den Faktor 4 gedehnt.

Kommentar:

Die dargestellte Bearbeitung eines Arbeitsspeicherbereichs ist dann besonders sinnvoll, wenn es sich wie bei großen Rechnern um große Datenmengen handelt, hier wird dann auch der Vorteil einer DV-Anlage ausschlaggebend (Ausblick auf die kommerzielle DV). Hier wurde nur das Prinzip gezeigt.

Bei mathematischen Aufgaben folgen auf wenige Eingabedaten umfangreiche arithmetische Bearbeitungen, die mit nur einem Ergebnis bzw nur wenigen Ausgabedaten abschließen.
In der kommerziellen Datenverarbeitung der folgenden Beispiele hingegen werden umfangreiche Datenmengen eingegeben, nur knapp bearbeitet, wie z.B. sortiert oder umgebucht, und dann werden meist wieder große Datenmengen ausgegeben. Daraus folgt die besondere Bedeutung der Ein-/Ausgabegeräte, welche die Zentraleinheit, den Rechner, häufig in den Hintergrund treten lassen.

Beispiel aus Gruppe 2 (EDV) Sortieraufgabe

Aufgabenstellung: Die größte Zahl der 4 im Arbeitsspeicher von Adresse 0 bis 3 stehenden Zahlen ist herauszufinden und an die Spitze (Adresse 0) zu stellen.

Lösungsweg: Es wird ein Programm erarbeitet, das einen fortschreitenden Vergleich jeweils zweier Zahlen durch Subtraktion durchführt und eine vom Ergebnis abhängige Umspeicherung der Zahlen ermöglicht.

Bedienung: Betriebsart: BEFEHL oder PROGRAMM 0,5 Hz zur Demonstration des Programmablaufs, PROGRAMM 40 kHz oder 8 Hz zur Durchführung der Aufgabe. Programm, wie angegeben, stecken.

Programm:

Programmbeschreibung: Der Vergleich jeweils zweier Zahlen kann in einer Schleife durchgeführt werden, wenn man die Adresse im Befehl, der den Vergleich ausführt, fortlaufend erhöht und damit in jedem weiteren Durchlauf durch die Schleife die nächsthöhere Zahl anspricht.

1.) Laden einer Hilfszelle mit der ersten Adresse, in welcher später die Adreßerhöhung durch Aufaddieren der Zahl 1 durchgeführt wird.

```
LAD   5   (000I)
SPE   6   (Hilfszelle)
```

Die verwendete Hilfszelle ist eine sogenannte "Zeigerzelle", da sie auf die zu bearbeitende Adresse zeigt.

2.) Vergleich durch Subtraktion (Komplementbildung und Addition durchführen).

```
LAD   0   (1. Zahl)
KPL
ADD  (6)  (2./3./4. Zahl)
            1. 2. 3. Schleifendurchlauf
```

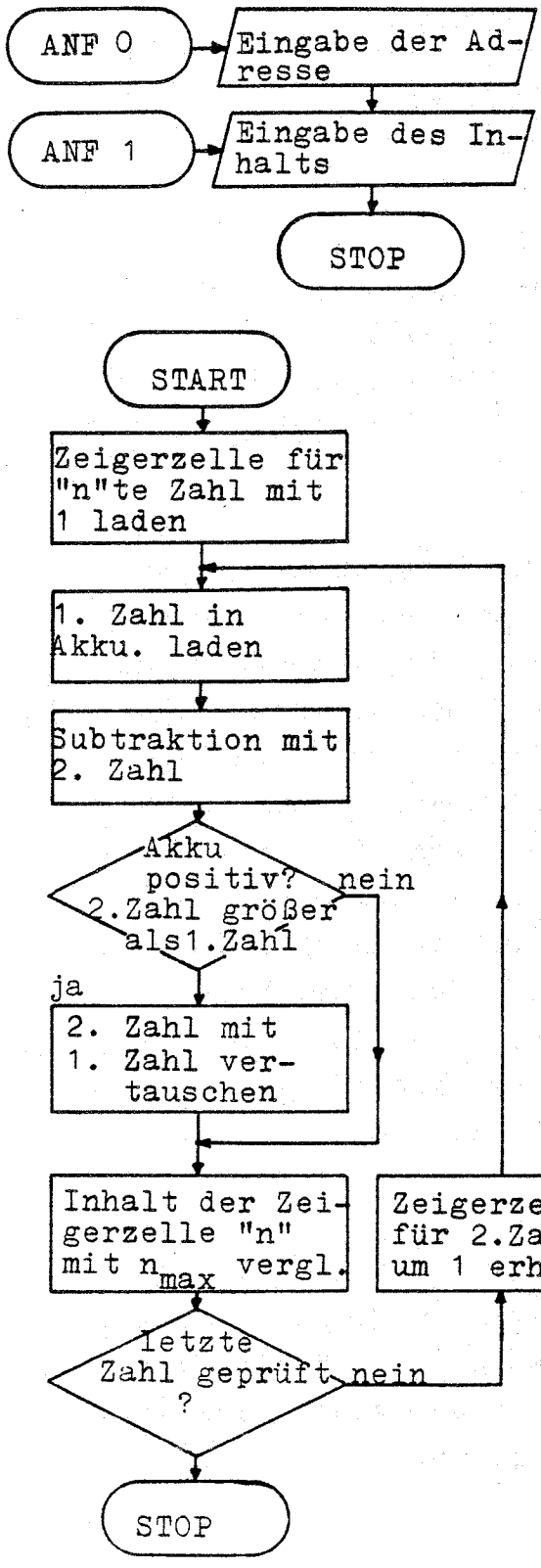
Da die Hilfszelle 6, die nach jedem Durchlauf erhöht wird, substituiert angesprochen wurde, addiert man jeweils diejenige Zahl, deren Adresse in der Hilfszelle zu finden ist, also fortlaufende Zahlen.

3.) Die Umspeicherung der zu vergleichenden Zahlen hängt vom Ergebnis ab und muß mit einer zweiten Hilfszelle zur Zwischenspeicherung durchgeführt werden. Auch hier muß die 2. (bzw. 3./4.) Zahl substituiert angesprochen werden.

4.) Die Adresse in der Hilfszelle 6 wird nach jedem Durchlauf geprüft, ob sie schon ihren Höchstwert erreicht hat (Ende des Laufbereichs), durch Subtraktion mit der Zahl, die den Laufbereich kennzeichnet, in diesem Fall dezimal 3:

```
LAD   6   (Hilfszelle
KPL
ADD   4   (00II)
SGN  23
```

Programmablaufplan



Befehlsfolge

Befehl Adr.	Operation	Adresse dez.	Adresse symbolisch	Inhalt dual
0	EIN	7	Adresse	
1	EIN	(7)	Inhalt	
2	STP			
3	LAD	5	Konst. 1	000I
4	SPE	6	HZ-Zeiger	
5	LAD	0	1. Zahl	
6	KPL			
7	ADD	(6)	2./3./4.Zahl	
8	SAM	15		
9	LAD	0	1. Zahl	
10	SPE	7	HZ Umsp.	
11	LAD	(6)	2./3./4.Zahl	
12	SPE	0	1. Zahl	
13	LAD	7	HZ Umsp.	
14	SPE	(6)	2./3./4.Zahl	
15	LAD	6	HZ Zeiger	
16	KPL			
17	ADD	4	n_{max}	
18	SGN	23		
19	LAD	6	HZ Zeiger	
20	ADD	5	Konst. 1	
21	SPE	6	HZ Zeiger	
22	SPR	5		
23	STP			

Anmerkung:

Das so abgefaßte Programm bezieht sich auf Zahlen ohne Vorzeichen, absolute Zahlen (0...15)

Arbeitsspeicher-belegung:

Adresse	Inhalt (dual)
0	1. Zahl (beliebig)
1	2. Zahl "
2	3. Zahl "
3	4. Zahl "
4	Laufbereich, n_{max} (OOII)
5	Konst. 1 (OOOI)
6	Hilfszelle Zeiger, HZ Zeiger
7	Hilfszelle Umspeichern, HZ Umsp.

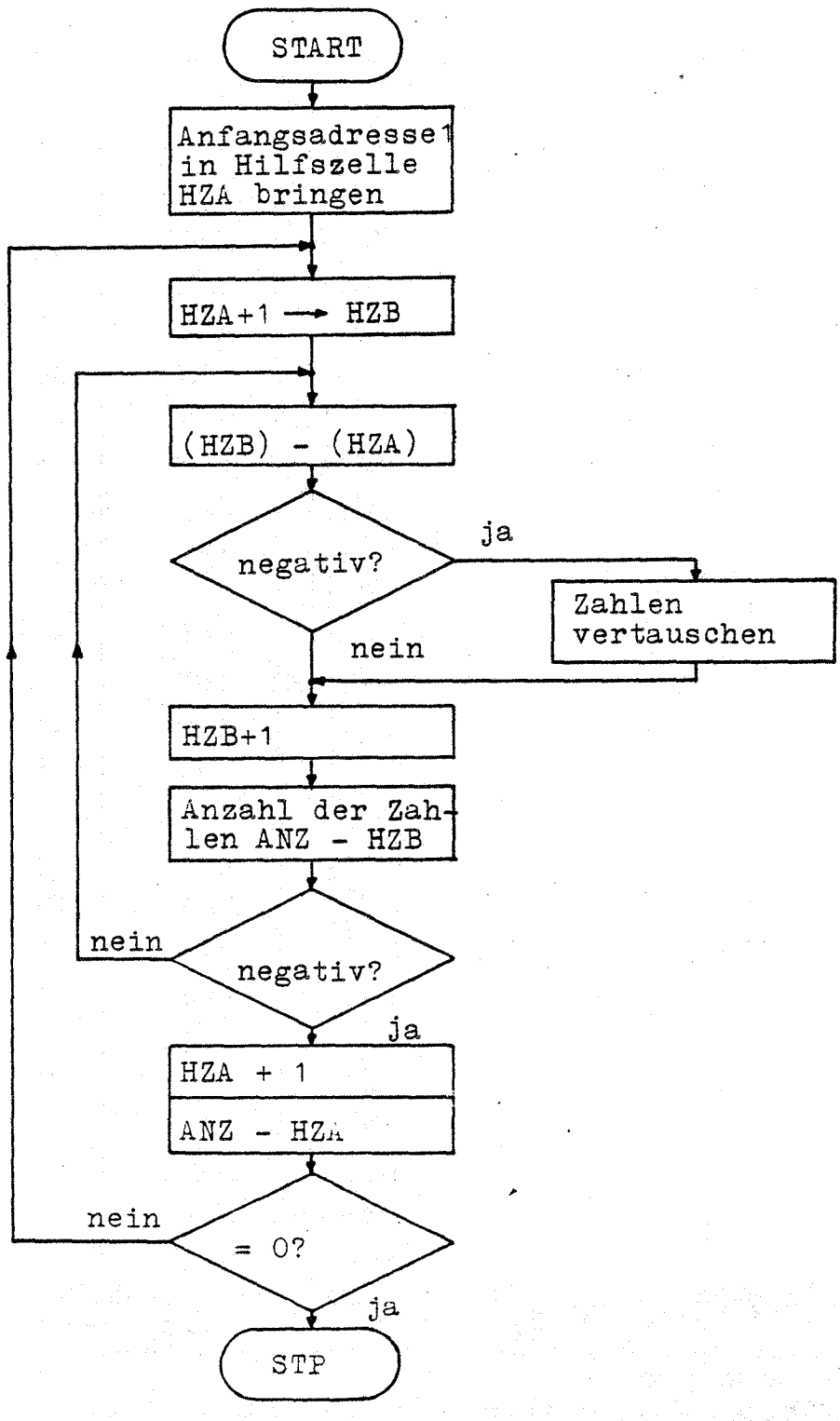
Lehrerdemonstration: Die Prinzipien der Programmierung entsprechend der Programmbeschreibung werden in ihrer Auswirkung erläutert und begründet. Hinweis auf das schrittweise (serielle) Abarbeiten eines Problems bei jedem Digitalrechner entsprechend dem schrittweisen Vergleich. Demonstrieren der Universalität solcher Programme durch Eingabe neuer Daten (Zahlen) oder durch Änderung des Laufbereichs (aufgrund des kleinen Arbeitsspeichers ist in diesem Fall nur eine Verringerung von 4 auf 3 oder 2 Zahlen möglich).

Kommentar: Die Aufgabe kann erweitert werden zur vollständigen Sortieraufgabe, d.h. Ordnen der Zahlen nach ihrer Größe. Dieses Programm ist um eine zweite übergeordnete Schleife erweitert.

Vollständige Sortieraufgabe

Diese Aufgabe ist lediglich eine Erweiterung der vorhergehenden Übung. Daher werden hier nur die Lösungen der Aufgabe, nämlich Programmablaufplan und Maschinenprogramm, angegeben. Das Programm sortiert Zahlen ohne Vorzeichen (0 ... 15) nach ihrer Größe.

Programmablaufplan



Maschinenprogramm: Befehlsfolge

Befehls-Adr.	Operation	Adresse dez.	Adresse symbolisch
0	EIN	7	
1	EIN	(7)	
2	STP		
3	LAD	0	Konst. 1
4	SPE	6	HZ Zeiger
5	LAD	6	HZ Zeiger
6	ADD	0	Konst. 1
7	SPE	7	Adr.d.m.Zahl
8	LAD	(6)	n.Zahl
9	KPL		
10	ADD	(7)	m.Zahl
11	SAM	18	
12	LAD	(6)	n.Zahl
13	SPE	5	HZ Umspeichern
14	LAD	(7)	m.Zahl
15	SPE	(6)	n.Zahl
16	LAD	5	HZ Umspeichern
17	SPE	(7)	m.Zahl
18	LAD	7	Adr.d.m.Zahl
19	ADD	0	Konst. 1
20	SPE	7	Adr.d.m.Zahl
21	KPL		
22	ADD	4	$n_{max} = m_{max}$
23	SAM	25	
24	SPR	8	
25	LAD	6	HZ Zeiger
26	ADD	0	Konst. 1
27	SPE	6	HZ Zeiger
28	KPL		
29	ADD	4	n_{max}
30	SGN	2	
31	SPR	5	

Bedienungshinweis: Vor dem ersten Programmstart muß die Konstante 1 (Inhalt = 0001) in Adresse 0 eingegeben werden, ebenso n_{max} in Adresse 4. Bei jedem weiteren Start kann n_{max} neu eingestellt werden.

Anmerkung: Das so abgefaßte Programm sortiert Absolutzahlen(0...15)

Überwachung eines Gleichstrommotors auf Grenzwertüberschreitung

*

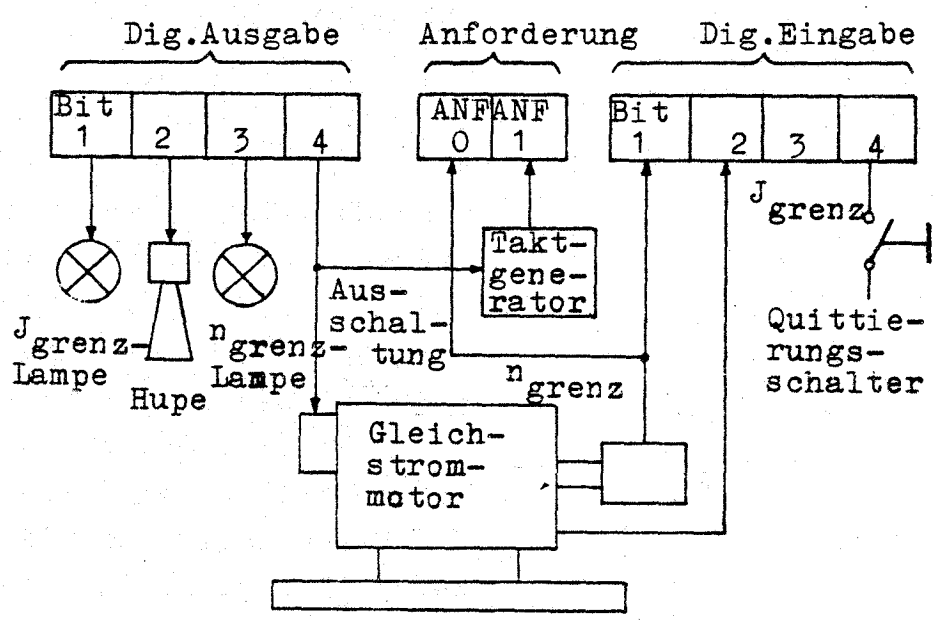
Aufgabenstellung:

Ankerstrom und Drehzahl eines Antriebsmotors sind auf Grenzwerte zu überwachen. Bei Überschreiten des zulässigen Stromwertes ist zunächst ein optisches Signal zu setzen, dann eine Hupe einzuschalten und nach Ablauf einer weiteren Zeit abzuschalten. Das akustische Signal soll quittiert werden können.
Beim Auftreten einer Grenzdrehzahlüberschreitung soll sofort abgeschaltet und ein optisches Signal gesetzt werden.

Lösungsweg:

Die Signale zweier Meßinstrumente mit Grenzwertmelder für Strom und Drehzahl werden auf die statische Digitaleingabe geschaltet und dort durch zyklische Meßstellenabfrage in Zeitintervallen überwacht. Das Grenzdrehzahlsignal wird zusätzlich durch Alarmeingabe (Anforderung) überwacht und wird dadurch sofort wirksam.

Anlagenkonfiguration:



*
Vor der Bearbeitung dieses Beispiels sollten die ab Seite 51 behandelten Prinzipien, Ausblenden und Setzen von Bitstellen, in Erinnerung gebracht werden.

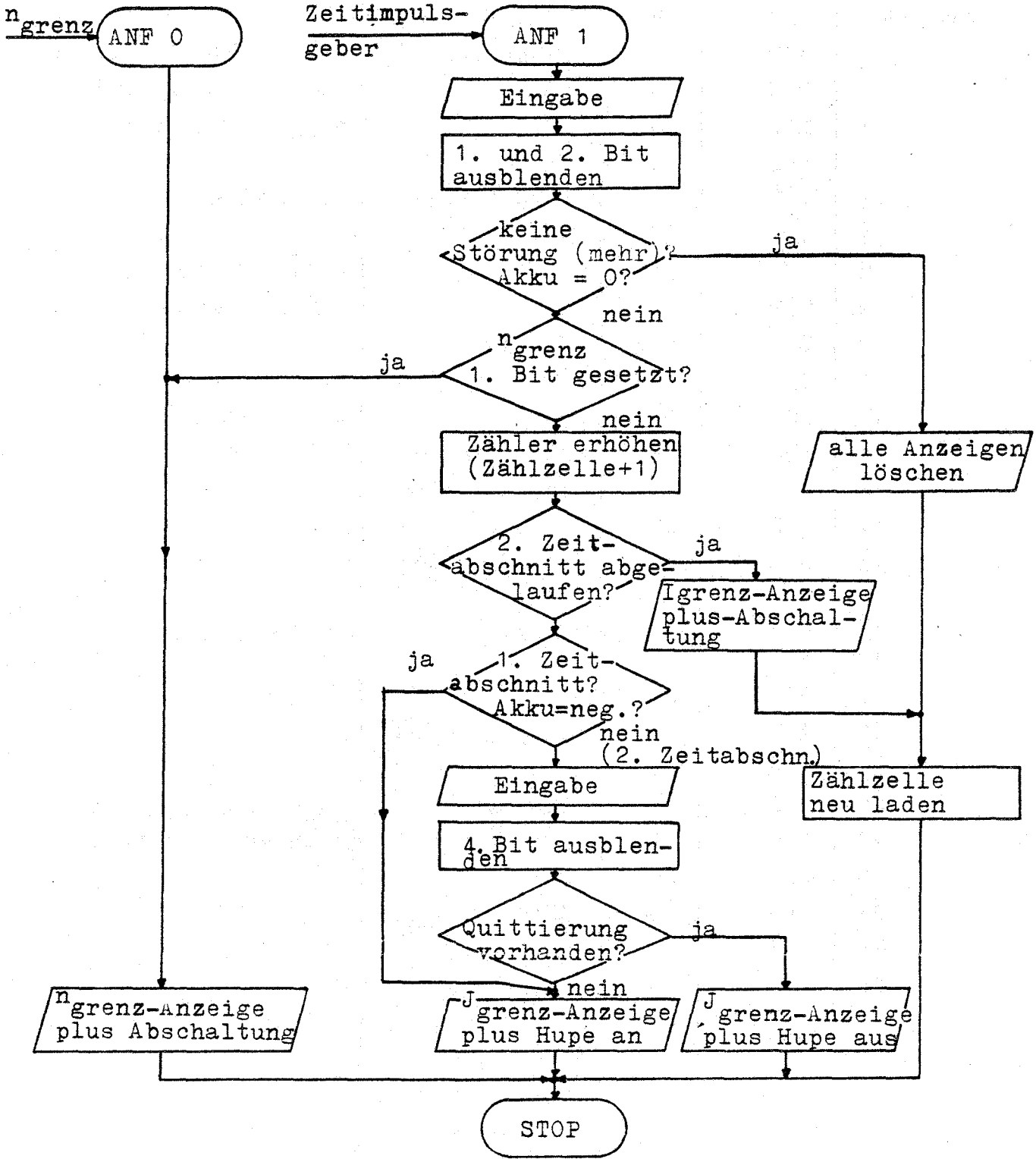
Als Quittierschalter wird der am Modell vorhandene Kippschalter verwendet. Das Bit 4 der Ausgabe schaltet über Relais den Netzanschluß des Motors und den Taktgenerator aus. Bit 1 und 3 können je eine Lampe einschalten, können aber auch durch die Lämpchen der Ausgabe direkt abgelesen werden. Bit 3 schaltet einen elektrischen Wecker oder Hupe ein (mit externer Stromversorgung, da zwischen den Buchsen der Ausgabe lediglich ein Kontakt geschlossen wird.) Die Grenzwerte selbst müssen über sogenannte Relaismelder gebildet und auf die Eingabe bzw. Anforderung geschaltet werden.

Programm:

Programmbeschreibung: Die zyklische Meßstellenabfrage wird von der Anforderung "ANF1" ausgelöst (Anschl. eines Zeitimpulsgebers, eines Taktgenerators, an "ANF1"). "ANF1" löst eine Eingabe aus mit anschließender Untersuchung der Eingabebitstellen. Bei n_{grenz} (Grenzdrehzahl) wird sofort abgeschaltet (AUS 00II). Bei J_{grenz} (Grenzstrom) wird eine Zählzelle aufaddiert. Nach "n" Abfragen der Störung J_{grenz} ("n" x Zeitintervall der zyklischen Abfrage) wird zur optischen Anzeige (Bit 1) eine Hupe (Bit 2) eingeschaltet. Eine Abfrage des Quittierungsschalters der Eingabe ermöglicht eine Abschaltung der Lärmbelästigung des Personals durch die Hupe.
Wird bei einer folgenden zyklischen Abfrage festgestellt, daß keine Störung mehr vorhanden ist, so wird die Zählzelle neu auf "n" eingestellt und die optischen Anzeigen werden gelöscht. Die Abschaltung erfolgt, wenn die Störung J_{grenz} das "n" plus 8te mal registriert wurde. Die Alarmeingabe

von n_{grenz} über die Anforderung 0 (ANF0) löst auch zwischen den Abfrageintervallen ein sofortiges Wirksamwerden des Rechners und Abschalten des Motors aus.

Programmablaufplan:



Befehlsfolge

Befehls-Adr.	Operation	Adresse dez.	Adresse symb. (dual)	Kommentar
ANF0 → 0	SPR	19		
ANF1 → 1	EIN	5	HZ EIN	
2	UND	3	Maske (II00)	1.u.2. Bit ausblenden
3	SGN	15		keine Störung mehr?
4	SAM	19		n_{grenz} vorhanden?
5	LAD	6	HZZ	} Zeitzähler
6	ADD	1	EINS (000I)	
7	SPE	6	HZZ	
8	SUL	21		2.Zeitabschn. Ende?
9	SAM	13		1.Zeitabschnitt?
10	EIN	5	HZ EIN	
11	UND	1	EINS (000I)	4. Bit ausblenden
12	SGN	26		Quittierung?
13	AUS	4	J_{grenz}-Anzeige (I000)	
14	SPR	27		
15	AUS	0	Anzeigen löschen (0000)	
16	LAD	7	Z	} Zähler neu laden
17	SPE	6	HZZ	
18	SPR	27		
19	AUS	2	n_{grenz}, Absch.	
20	SPR	27		
21	LAD	4	(I000)	} I000 wird durch "Einodern" erzeugt
22	ODR	1	(000I)	
23	SPE	5	(I00I)	
24	AUS	5	Absch., J_{grenz}	
25	SPR	16		
26	AUS	3	J_{grenz}, Hupe (II00)	
27	STP			

Arbeitsspeicherbelegung:

Vor dem Stecken des Programms muß der Arbeitsspeicher wie folgt durch "Eingabe" belegt werden.

Adresse	Inhalt	Verwendung (symb. Adresse)
0	0000	NULL/Anzeigen löschen
1	000I	EINS/Ausblenden
2	00II	Abschaltung, n_{grenz}
3	II00	Maske bzw. J_{grenz} + Hupe
4	I000	J_{grenz}-Anzeige
5	belieb.	Hilfszelle EIN (HZ EIN)
6	neg. Zahl	Hilfszelle Zählen (HZZ)
7	"	Anzahl Z der Zeitintervalle des 1. Zeitabschnitts (negativ einzugeben)

Lehrerdemonstration:

In der Betriebsart PROGRAMM 40 kHz können lediglich die Reaktionen auf Störungen, optische und akustische Anzeigen bzw. Abschaltung beobachtet werden. Bei Verwendung der langsameren Betriebsarten mit 8, 2 und 0,5 Hz können sämtliche Vorgänge durch Kippschalter bzw. Tastendruck simuliert werden:

- n_{grenz} durch Kippschalter 1
- J_{grenz} durch Kippschalter 2
- Zeitimpulsgeber durch Taste ANF1
- Alarmeingabe n_{grenz} durch Taste ANF0

Dabei können sämtliche Vorgänge leicht verfolgt und erklärt werden.

Demonstriert werden folgende Prinzipien:

- Zyklische Meßstellenabfrage
- Alarmeingabe (→ Echtzeitrechnerprinzip)
- Dialog: Rechner - Bediener (Quittierungsschalter)
- Abgestufte Reaktion des Rechners bei Überwachungsaufgaben (optische, dann akustische Anzeige, schließlich Abschaltung)
- Eingriff des Rechners in den Prozeß (Abschaltung), also Online-Rechner-Prinzip
- Variable Einstellbarkeit der Reaktionszeit des Rechners (Anzahl der Zeitintervalle)

Kommentar:

Bei den erweiterten Möglichkeiten größerer Prozeßrechner würde man die Meßwerte für Strom und Drehzahl über Analog-Digital-Wandler erfassen und die Grenzwertkontrolle durch Vergleich mit abgespeicherten Grenzwerten realisieren.
Außerdem ist üblicherweise (und sinnvollerweise) die Anzahl der vom Rechner gleichzeitig überwachten Meßwerte wesentlich größer. Das Beispiel ist jedoch zur Demonstration der prinzipiellen Funktion geeignet.

Anschluß eines Analog-Digitalwandlers mit externgesteuerter Eingabe

Aufgabenstellung:

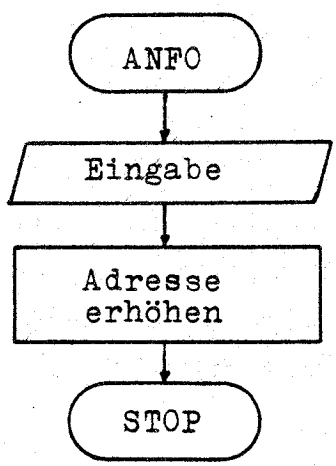
Die vom Analog-Digitalwandler umzuschlüsselnden analogen Meßwerte sind erst nach Abschluß der Umschlüsselung in den Rechner einzugeben.

Lösungsweg:

Der Ausgang des Analog-Digitalwandlers wird an das Eingaberegister angeschlossen.
Die Aufforderung zur Meßwerteingabe erfolgt vom externen Gerät über die Anforderung ANFO, die den Start des vorbereiteten Meßwerteingabeprogrammes auslöst.

Lösung:

Programmablaufplan



Befehlsfolge

Befehls-Adr.	Operation	Adresse
0	EIN	(7)
1	LAD	7
2	ADD	6 (000I)
3	SPE	7
4	STP	

Bedienung:

Betriebsart: PROGR. 40 kHz

Voraussetzung für den ordnungsgemäßen Ablauf ist die Herstellung eines Anfangszustandes: Eingabe von 0000 in Zelle 7 und 000I in Zelle 6 z.B. durch die anschließend an das Programm steckbaren Befehle:

5	EIN	7
6	EIN	6

Außerdem stellt das Programm nicht sicher, daß eine maximale Adresse nicht überschritten werden kann. (In diesem Fall darf Adresse 5 nicht überschritten werden.)
Dazu wäre folgende Änderung und Erweiterung nötig:

```
   0   LAD   7                 }  Adreß-
   1   ADD   6   (0001)        }  erhöhung
   2   SPE   7                 }
   3   KPL                     }  Abfrage ob Adreß-
   4   ADD   5   (0I0I)        }  bereich überschrit-
   5   SGN   2                 }  ten wurde.
   6   EIN  (7)
   8   STP
   9   EIN   5   (0I0I)        }  Herstellung
  10   EIN   6   (000I)        }  des Anfangs-
  11   EIN   7   (0000)        }  zustandes
```

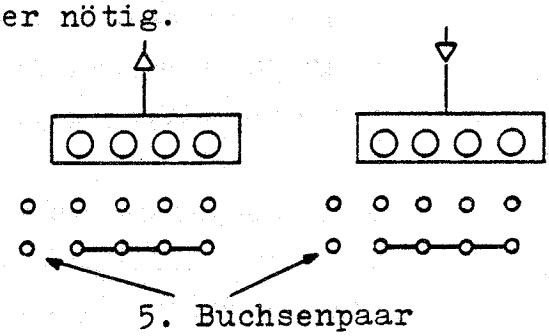
Der Bereich des Datenspeichers von Adresse 0 bis 4 stellt bei diesem Programm einen Umlauf-Pufferspeicher dar für die Eingabe der verschlüsselten Meßwerte vom Analog-Digital-Umsetzer (ADU).

Das Programm kann selbstverständlich auch ohne Analog-Digital-Umsetzer vorgeführt werden. Es genügt, diesen durch Kippschaltereinstellung und Tastendruck auf ANFO zu simulieren.

Kommentar:

Erweiterungsmöglichkeiten im Bezug auf übliche Anwendungen bei der Meßwerterfassung sind: Die eingebenen Meßwerte werden einer Grenzwertkontrolle unterzogen. Aus zwei Meßwerten kann die Summe oder der Mittelwert gebildet werden. Oder aber die Meßwerte werden nach einer einfachen Formel umgerechnet.

An dieser Stelle ist ein Hinweis auf das fünfte Buchsenpaar am Eingabe- bzw. Ausgaberegister nötig.



Dieses Buchsenpaar ist nicht verdrahtet und hat damit noch keine Bedeutung. Es wurde für folgende Möglichkeit vorgesehen: Realisierung des Anforderungs-Quittungsprinzips durch Aufschalten der Quittierung auf das 5. Buchsenpaar. Die Quittierung selbst muß dabei von den Leuchtpfeilen für Eingabe und Ausgabe abgenommen werden.
(20 V/20 mA verfügbar).
Dazu muß man sich Folgendes vor Augen halten: Die Arbeitsgeschwindigkeit angeschlossener Ein-/Ausgabegeräte ist um mehr als den Faktor 1000 kleiner als die Arbeitsgeschwindigkeit von Rechnern. Aufgrund dieser unterschiedlichen Arbeitsgeschwindigkeiten ist eine Entkopplung der Geräte vom Rechner nötig. Es arbeiten z.B. ein Eingabegerät und ein Ausgabegerät am Modell. Das Eingabegerät melde über ANF 1, daß eine Information zur Eingabe ansteht. Im selben Augenblick stellt das Ausgabegerät die Anforderung ANF0. Da ANF0 und ANF1 prioritiert werden (ANF0 hat Vorrang vor ANF1, ANF1 wird deshalb noch nicht wirksam, wird aber gespeichert) verzögert sich die Übernahme der Information vom Eingabegerät. Es darf deshalb

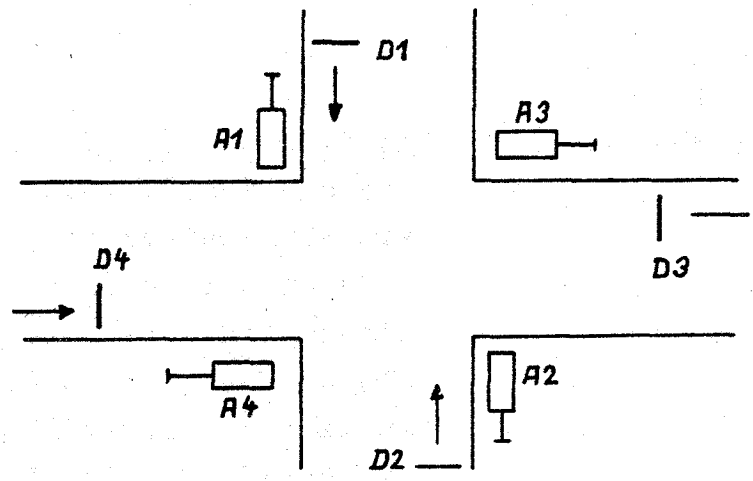
z.B. so lange keinen neuen Wert verschlüsseln, bis es seine Quittierung über die erfolgte Eingabe erhalten hat. Damit wird das Anforderungs-Quittungssystem realisiert.

Beim Ausgabegerät sieht dieses Prinzip umgekehrt aus. Der Rechner gibt eine Information ins Ausgaberegister aus, startet mit Hilfe eines Impulses auf dem 5. Buchsenpaar (abgeleitet vom Leuchtpfeil für Ausgabe) das Ausgabegerät. Nun muß er aber mit weiteren Ausgaben so lange warten, bis sich das Gerät über ANFO als klar zur nächsten Informationsübernahme zurückmeldet. In der Zwischenzeit können dann andere Programme laufen, wie z.B. ein Eingabeprogramm des anderen Gerätes. Dadurch bearbeitet der Rechner zwar nur immer ein bestimmtes Programm, die angeschlossenen Geräte aber sind dauernd beschäftigt und arbeiten gleichzeitig (Simultanarbeit von Geräten). Dieses Beispiel kann ebenfalls leicht programmiert werden und kann in den Zeitlupenbetriebsarten anschaulich verfolgt werden.

Ampelsteuerung

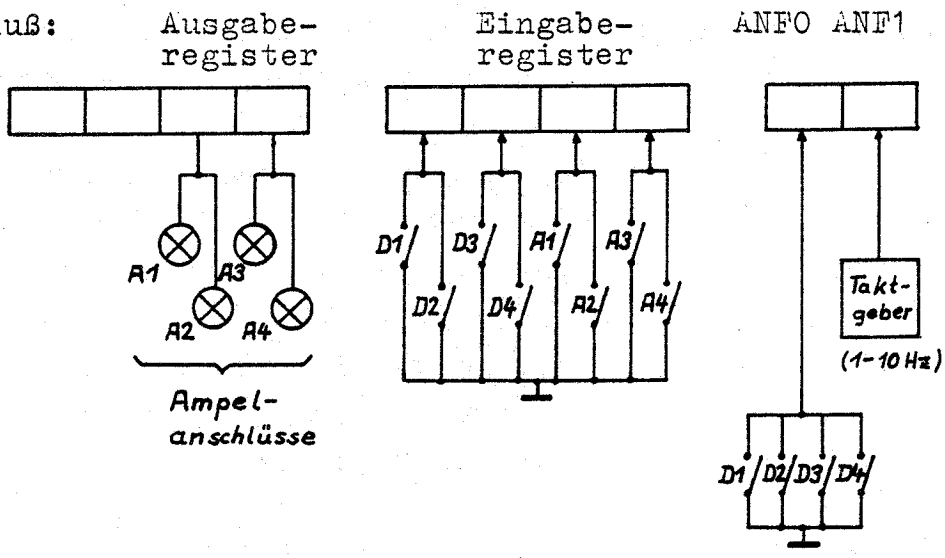
Aufgabenstellung: Die 4 Ampeln einer Kreuzung sollen so gesteuert werden, daß der normale Wechsel zwischen "grün" und "rot" in Abhängigkeit von der Verkehrsbelastung verändert wird. Und zwar soll die Grünphase, die normalerweise aus "n" Zeitintervallen besteht, von jedem Fahrzeug, das in der Grünphase fährt, um 1 Zeitintervall verlängert werden bis maximal auf das Doppelte.

Lösungsweg:



Die 4 Ampeln A 1 bis A 4 enthalten der Einfachheit halber nur eine Lampe für Rotlicht. Nicht "rot" bedeutet "grün". Auch bei üblichen Verkehrsrechnern werden nur die Signale für Grün- und Rotphase gegeben; das Gelblicht wird dann von der lokalen Ampelsteuerung selbst, da immer gleich lang, gebildet)
Die 4 Detektoren D 1 bis D 4 (Mikrorelais) melden das Verkehrsaufkommen.
Ein Taktgeber (Blinkrelais) gibt dem Rechner Zeitimpulse vor. (Zeitimpulsgeber sind in vielen DV-Anlagen zur Bildung der Uhrzeit enthalten)

Rechneranschluß:

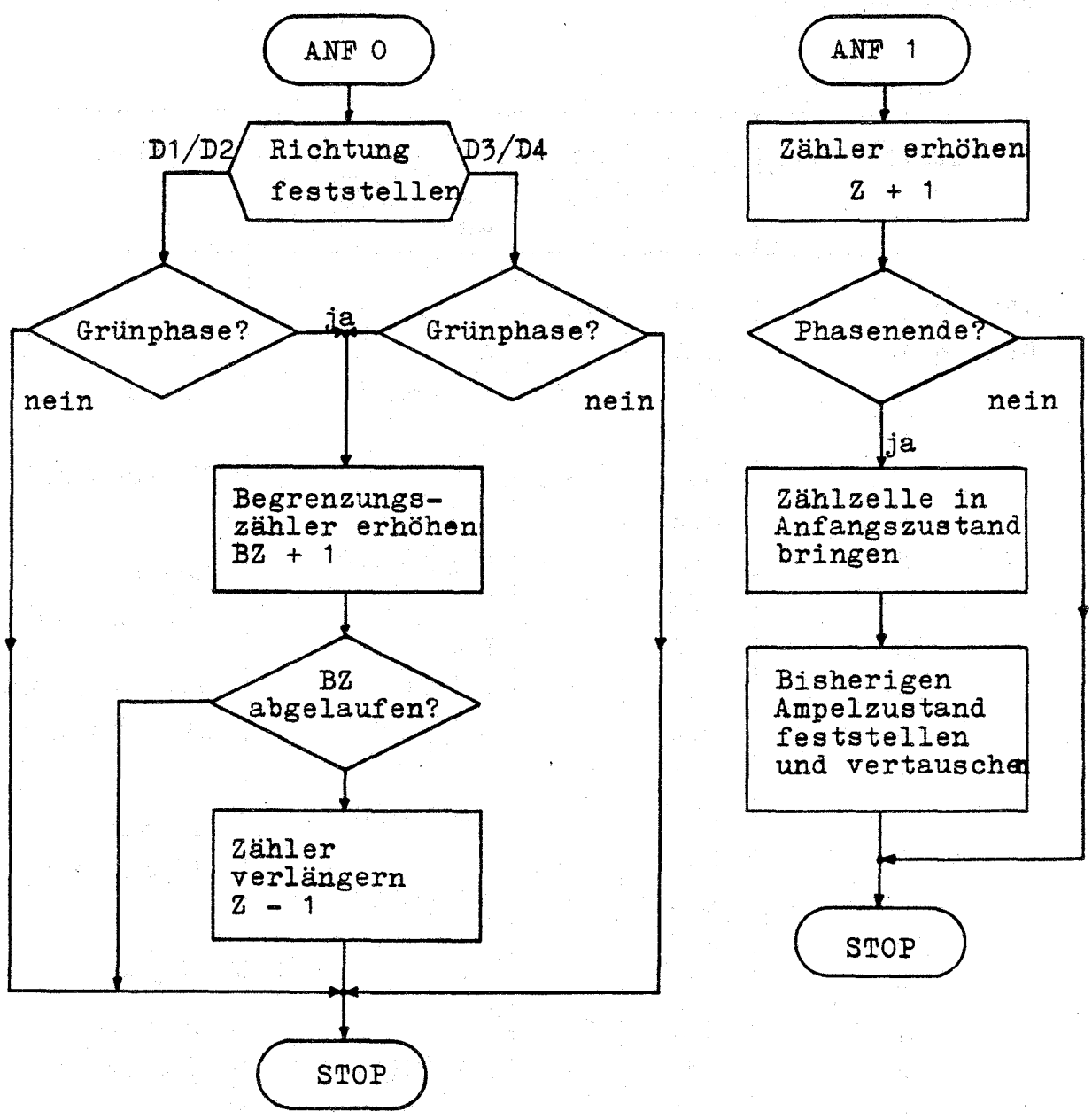


Programm

Programmbeschreibung:

Mit den Impulsen der Anforderung 1 (ANF1) wird ein Zählerprogramm gestartet, das beim Zählerstand "n" (in diesem Fall 8, da dabei ein Überlauf auftritt, der leicht ausgewertet werden kann) die Ampeln umschaltet und einen Programmanfangszustand herstellt. Jedes Auto löst beim Überfahren der Detektoren "ANF0" aus, welche ein zweites Programm startet. Da A 1 ... A 4 und D 1 ... D 4 am Eingaberegister angeschlossen sind, kann das Programm den momentanen Ampelzustand und die beiden Straßen unterscheiden und richtig reagieren. Fällt die Anforderung noch in die Grünphase, so wird der Zähler durch Subtraktion mit 1 um 1 verlängert. Ein weiterer Zähler (Begrenzungszähler) sorgt dafür, daß diese Verlängerung nur "n" mal möglich ist.

Programmablaufplan:



Maschinenprogramm:

Befehlsfolge

	Befehls-Adr.	Operation	Adresse dez.	Adresse symbolisch (dualer Inhalt)	Kommentar
ANF0 →	0	SPR	16		
ANF1 →	1	LAD	4	Z	
	2	ADD	1	K1 (000I)	Zähler
	3	SPE	4	Z	
	4	SUL	6		Phasenende?
	5	STP			
	6	LAD	0	K0 (0000)	Zähler
	7	SPE	4	Z	neu einstellen
	8	SPE	5	BZ	
	9	EIN	6	HZEIN	Ampelzustand
	10	UND	1	K1 (000I)	feststellen
	11	SGN	14		
	12	AUS	2	K2 (00I0)	Ampeln
	13	STP			Umschalten
	14	AUS	1	K1 (000I)	
	15	STP			
	16	EIN	6	HZEIN	Straßenrich-
	17	SAM	28		tung feststel.
	18	UND	2	K2 (00I0)	Ampelzustand?
	19	SGN	27		
	20	LAD	5	BZ	Begrenzungs-
	21	ADD	1	K1 (000I)	zähler
	22	SPE	5	BZ	
	23	SUL	27		
	24	LAD	4	Z	Zähler
	25	ADD	3	KM1 (IIII)	verlängern
	26	SPE	4	Z	
	27	STP			
	28	UND	2	K2 (00I0)	
	29	SGN	20		
	30	STP			

Arbeitsspeicherbelegung:

Adresse	Inhalt (dual)		
0	K0	(0000)	
1	K1	(000I)	
2	K2	(00I0)	
3	KM1	(IIII)	≙ - 1
4	Z	(0000)	
5	BZ	(0000)	
6	HZEIN	(bel.)	

Der Arbeitsspeicher muß vor dem ersten Programmstart durch Eingabe, wie angegeben, belegt werden, damit das Programm die nötigen Konstanten zur Verfügung hat.

Lehrerdemonstration:

Bei der Simulation der Ampelanlage durch Kippschaltereingabe und Tastendruck auf die Anforderung muß folgendes berücksichtigt werden:

1.) Da der Ampelzustand dem Rechner mitgeteilt werden muß, ist es nötig, das Bit 3 (A1, A2) des Ausgaberegisters über eine Steckverbindungsleitung mit Bit 3 (A1, A2) des Eingaberegisters zu verbinden. Ebenso ist mit Bit 4 (A3, A4) zu verfahren. Außerdem gilt: A1 (bzw. A2) = $\overline{A3}$ (bzw. $\overline{A4}$).

2.) Da ein Detektorsignal (D1...D4) immer mit einer Anforderung verknüpft ist, muß vor dem Drücken der Taste ANF 0 mindestens Bit 1 oder Bit 2 der Eingabe auf I gestellt werden und danach wieder zurückgesetzt werden.

Es werden im wesentlichen die gleichen Prinzipien wie in der vorhergehenden Aufgabe gezeigt. Besonders hervorzuheben ist, daß der "Rechner" hier weniger mit Zahlen als mit Prozeßzuständen arbeitet, die durch einzelne Bits signalisiert werden.