

robotron

Z 1013

Mikrorechnerbausatz

Handbuch Teil I

VEB Robotron-Elektronik Riesa

ROBOTRON

Der Herausgeber ist jederzeit für Korrekturhinweise
fachlicher, stilistischer und redaktioneller Art dankbar.

Z 1013

Herausgeber:

VEB Robotron-Elektronik Riesa
Pausitzer Str. 60
Riesa
8400

© VEB Robotron-Elektronik Riesa

III-6-15 4255-86 AG 706-68-86

digitalisiert: U.Zander, 2011 <zander@felix.sax.de>

R O B O T R O N

Mikrorechnerbausatz Z 1 0 1 3

Handbuch Teil 1

Inhaltsverzeichnis

	Seite
2.	Grundbegriffe der Mikrorechentechnik 2
2.1.	Hardware oder Software 2
2.2.	Bestandteile eines Mikrorechners 3
2.2.1.	Zentrale Verarbeitungseinheit 3
2.2.2.	Speicher 4
2.2.2.1.	Programmspeicher (Nur-Lese-Speicher) 4
2.2.2.2.	Datenspeicher (Schreib-Lese-Speicher) 5
2.2.3.	Ein-/Ausgabe-Einheiten 6
2.2.4.	Verbindung der Funktionseinheiten 6
2.3.	Programmabarbeitung 7
2.3.1.	Ablauf in der CPU 7
2.3.2.	Holen der Befehle 7
2.3.3.	Darstellung von Informationen im Speicher 8
2.4.	Grundbegriffe der Software 10
2.4.1.	Darstellung von Zahlen 10
2.4.2.	Logische Operationen 13
2.4.3.	Arithmetische Operationen 16
3.	Hardware des Z1013 19
3.1.	Blockschaltbild 19
3.2.	Steuerung des Mikroprozessors 19
3.2.1.	Beschreibung der Steuersignale 19
3.2.2.	Takterzeugung 23
3.2.3.	Resetlogik 24
3.3.	Speichereinheiten 25
3.3.1.	Anschluß 25
3.3.2.	Zusammenarbeit mit der CPU 27
3.4.	Ein-/Ausgabe-Baugruppen 27
3.4.1.	Parallel Ein-/Ausgabe-Baustein U855-PIO 27
3.4.1.1.	Beschreibung der Steuersignale 27
3.4.1.2.	Programmierung 29
3.4.2.	Tastaturanschluß 32
3.4.3.	Magnetbandanschluß 32
3.4.4.	Bildschirmsteuerung 33
3.5.	Stromversorgung 34
3.6.	Bussystem 35

4.	Der Befehlssatz des Mikroprozessors U880	36
4.1.	Befehlsschlüssel	37
4.1.1.	1-Byte Befehle	37
4.1.2.	2-Byte Befehle	38
4.1.3.	3-Byte Befehle	39
4.1.4.	4-Byte Befehle	40
4.2.	Adressierung	40
4.2.1.	Registeradressierung	40
4.2.2.	Direktwertadressierung	41
4.2.3.	Registerindirektadressierung	41
4.2.4.	Indexierte Adressierung	42
4.3.	Maschinenbefehle und ihre Bedeutung	42
4.3.1.	Ladebefehle	42
4.3.2.	Byte- und Doppelbyte-Zähl-Befehle	43
4.3.3.	Arithmetische Befehle	44
4.3.4.	Vergleichsbefehle	46
4.3.5.	Logische Befehle	46
4.3.6.	Spezielle arithmetische Hilfsoperationen	47
4.3.7.	Befehle zur Bitmanipulation	48
4.3.8.	Verschiebepbefehle	48
4.3.9.	Sprungbefehle	51
4.3.10.	Kelleroperationen	53
4.3.11.	Unterprogrammoperationen	55
4.3.12.	Ein- und Ausgabebefehle	57
4.3.13.	Gruppenoperationen für Lade-, Vergleichs- und Ein-/Ausgabe-Befehle	58
4.3.14.	Austauschbefehle	60
4.3.15.	CPU-Steuerbefehle	61
4.3.16.	Bedeutung der Flags	61
4.4.	Unterbrechungsorganisation	63

Bestandteile des Handbuches:

Handbuch Teil I
Handbuch Teil II
Anlagenteil

2. Grundbegriffe der Mikrorechentechnik

2.1. Hardware oder Software?

Dieses Kapitel ist vor allem für den Leser gedacht, der in der Mikrorechentechnik nicht bewandert ist. Es werden hier einige Grundbegriffe erläutert, die das Verständnis der nachfolgenden Kapitel erleichtern sollen.

Der erste Begriff, der zu klären wäre, ist der des Mikrorechners.

Ein Mikrorechner ist ein komplexes System verschiedener Funktionseinheiten auf der Basis mikroelektronischer Schaltkreise, die auf bestimmte Art miteinander in Verbindung treten und durch ihr Gesamtverhalten eine vorgegebene Aufgabe (Programm) lösen. Ein Programm stellt dabei eine Folge von Anweisungen (Befehlen) dar.

Gekennzeichnet wird ein Mikrorechner im wesentlichen durch seine Hard- und Software. Unter Hardware wird dabei sowohl die Gesamtheit der mechanischen und elektronischen Bauelemente, wie integrierte Schaltkreise, Transistoren, Widerstände usw., als auch die Art und Weise der Verschaltung dieser Bauelemente verstanden.

Als Software eines Rechners werden seine Programme, z. B. Betriebsprogramm und BASIC-Interpreter, bezeichnet. Das Betriebsprogramm (oder auch Betriebssystem) enthält die Programme, die die Zusammenarbeit der einzelnen Systemkomponenten organisieren bzw. überhaupt ermöglichen. Worin unterscheidet sich aber nun ein Mikrorechner von einer herkömmlichen Schaltung?

Um eine bestimmte Steuerungsaufgabe lösen zu können oder immer wiederkehrende Berechnungen zu realisieren, muß nicht immer ein Mikroprozessor verwendet werden. Vielfach ist es einfacher, eine Schaltung mit einfachen Logikschaltkreisen aufzubauen. Eine solche Schaltung hätte überdies den Vorteil, schneller als ein Mikroprozessor zu arbeiten. Aber bereits einfache Änderungen der Aufgabenstellung würden einen neuen Schaltungsentwurf erfordern, der mit einem bestimmten Arbeitsaufwand realisiert werden müßte. Komplexere Aufgabenstellungen ließen sich auf diese Art überhaupt nicht realisieren, da der Aufwand zu hoch werden könnte. Die Lösung einer Aufgabe mit Hilfe eines Mikrorechners ist weitaus einfacher.

Der Mikroprozessor ist in der Lage, alle Verknüpfungsmöglichkeiten der Logikschaltkreise nachzubilden und damit jedes gewünschte Verhalten zu realisieren. Die übrigen Funktionseinheiten des Mikrorechners enthalten dann in den Speichereinheiten den Lösungsablauf der Aufgabe in Form von Anweisungen für den Mikroprozessor, die Ausgangsdaten sowie konstante Werte. Über andere Funktionseinheiten werden Signale aufgenommen sowie Steuersignale wieder abgegeben. Die erreichbare Arbeitsgeschwindigkeit ist kleiner als bei reinen Logikschaltungen. Da aber nicht die maximal erreichbare Arbeitsgeschwindigkeit, sondern die für den jeweiligen Prozeß

oder die Steuerung benötigte Geschwindigkeit entscheidend ist, ist dieser Nachteil nur in wenigen Fällen von Bedeutung. Eine Änderung der Aufgabenstellung führt meist nur zu einer Änderung der Anweisungen für den Mikroprozessor. Diese Änderung ist schnell realisierbar. Mit dem Mikroprozessor lassen sich ohne technische Veränderungen vielerlei Aufgabenstellungen lösen, es ist meist nur erforderlich, andere Anweisungen zu erarbeiten.

2.2. Bestandteile eines Mikrorechners

2.2.1. Zentrale Verarbeitungseinheit

Die Zentrale Verarbeitungseinheit, im Englischen als "Central process unit" (CPU) bezeichnet, ist der wichtigste Bestandteil eines Mikrorechners. Eine solche CPU ließe sich aus diskreten Elementen, d. h. Transistoren, Widerständen und Kondensatoren aufbauen, würde aber einen sehr großen Aufwand erfordern. Mit der Entwicklung der Mikroelektronik konnte diese Funktionseinheit in Form einer integrierten Schaltung als sogenannter Mikroprozessor bereitgestellt werden und damit zu einer wesentlichen Vereinfachung im Schaltungsentwurf beitragen. Am Beispiel des Mikroprozessors U880, der im MRB Z1013 Verwendung findet, sollen einige wichtige Bestandteile erläutert werden.

Dazu gehören:

- CPU-Steuerung/Befehlsdekodierung
Hier werden anhand eines vorgegebenen Befehls bestimmte Signale erzeugt. Bestimmte Zustände, die von der CPU-Steuerung erkannt werden, sowie der zugeführte Takt, erzeugen zeitlich festgelegte Signalfolgen, die sowohl den Ablauf innerhalb der CPU steuern, die aber auch als Steuersignale in allen angeschlossenen Funktionseinheiten ausgewertet werden können und die gesamten Abläufe eines Mikrorechners koordinieren (siehe Zeitdiagramme Anlage 10).
- Arithmetisch-logische Einheit (ALU)
In der ALU können Daten entsprechend eines Befehle verknüpft werden. Zu diesen Operationen mit Daten gehören: Addition, Subtraktion, UND-Verknüpfung (Konjunktion), ODER-Verknüpfung (Disjunktion) sowie eine Reihe weiterer Operationen wie Verschiebungen und Bitmanipulationen. Eine Veränderung der Daten ist nur in der ALU möglich, erforderlichenfalls müssen diese erst in die ALU geholt und danach zurücktransportiert werden.
- Registersatz (Zwischenspeicher)
In der CPU existieren Zwischenspeicher, die als Register bezeichnet werden. Hier können Zwischenergebnisse aufbewahrt und in der ALU miteinander verknüpft werden. Einige Register

besitzen spezielle Bedeutung, wie z. B. der sogenannte Kellerzeiger (Stackpointer SP), Befehlszähler (PC), Refreshregister und Interruptregister (s. auch Abschn. 4.). Bestimmte Register sind doppelt vorhanden und können durch einen Befehl umgeschaltet werden. Ein Register wird benutzt, um den Zustand der CPU während der Befehlsabarbeitung zu speichern. Es wird als Flag-Register bezeichnet (die Bezeichnung "Flag" sollte als Anzeiger verstanden werden). In einem Register wird der gelesene Befehl zwischengespeichert, bis die durch ihn veranlaßte Operation beendet ist. Dieses Register heißt demzufolge Befehlsregister. Die Arbeit der CPU wird durch eine Reihe von Systemsignalen gekennzeichnet, die als Anschlüsse herausgeführt wurden und das Zusammenwirken mit den angeschlossenen Funktionseinheiten steuern.

2.2.2. Speicher

In der Mikrorechenteknik haben sich zur Speicherung von Informationen Halbleiterspeicher weitgehend durchgesetzt. Es sind integrierte Schaltungen in unterschiedlichen Gehäusegrößen, je nach Kapazität des Speichers. Speicher werden zu verschiedenen Zwecken benötigt, z. B. um der CPU die abzuarbeitenden Befehle zur Verfügung zu stellen. Da die Register der CPU meist nicht ausreichen, alle Zwischenergebnisse aufzubewahren, müssen diese ebenfalls in den Speicher gebracht werden.

Als Modell eines Speichers mag ein langer Schrank mit vielen Fächern dienen. Diese Fächer sind einzeln nummeriert. Diese Numerierung soll bei Null beginnen und lückenlos bis zu einem Endwert erfolgen. Jedes Fach entspricht einem Speicherplatz und kann eine Information enthalten. Die maximale Anzahl der Fächer bestimmt die Kapazität dieses Speichers.

Die Zeit, die vom Anlegen einer Speicherplatzadresse bis zur Bereitstellung der gespeicherten Daten benötigt wird, wird Zugriffszeit genannt.

Zwischen der Speicherung von Daten und Programmen bestehen einige Unterschiede. Programme werden meist in Speichern aufbewahrt, die auch nach Abschalten der Versorgungsspannung ihren Inhalt behalten. Allerdings können diese Speicher nur gelesen werden, zum Beschreiben dieser Speicher sind spezielle Einrichtungen notwendig.

2.2.2.1. Programmspeicher (Nur-Lese-Speicher)

In einem Programmspeicher sind die Anweisungen für einen Mikroprozessor enthalten. Diese Anweisungen gehen auch nach Ausschalten des Rechners nicht verloren, sie sind nicht flüchtig. Diese Speicher bezeichnet man als Nur-Lese-Speicher (Read only memory - ROM), die Informationen werden einmal eingegeben und stehen ständig zur Verfügung.

Je nach Eingabe der Information unterscheidet man: ROMs, die bereits während der Herstellung ihre Informationen erhalten und ROMs, die nachträglich elektrisch programmiert werden können (ein einmaliger Vorgang, da die Struktur des Speichers verändert wird). Diese letztgenannten Speicher heißen PROM (Programmable ROM). Eine weitere Speicherart kann sowohl programmiert als auch wieder gelöscht werden. Das Löschen erfolgt mit ultraviolettem Licht (UV-Licht) und löscht immer den gesamten Speicher. Diese Speicherart nennt man EPROM (Erasable PROM). Das Einschreiben der Programme in den EPROM geschieht mit speziellen Funktionseinheiten, sogenannten EPROM-Programmiergeräten.

In den EPROM wird die zu speichernde Information mittels einer Programmierspannung als Ladungsmenge eingegeben. Nach Erreichen einer vorgegebenen Ladung ist der Baustein programmiert. Die Bestrahlung mit UV-Licht hat zur Folge, daß die gespeicherte Ladungsmenge wieder abgebaut wird.

Nach dem Löschen ist der EPROM wieder programmierbar.

2.2.2.2. Datenspeicher (Schreib-Lese-Speicher)

Zur Aufbewahrung von Zwischenergebnissen oder anderen veränderbaren Informationen werden Schreib-Lese-Speicher verwendet.

Da diese Speicher wahlweise gelesen oder beschrieben werden können, nennt man sie Speicher mit wahlfreiem Zugriff (Random access memory - RAM). Mit Abschalten der Stromversorgung verlieren RAMs ihren Inhalt, sie sind also nicht zur Aufbewahrung von Informationen verwendbar, die immer verfügbar sein müssen. Es werden zwei grundsätzliche Typen unterschieden: statische und dynamische RAMs.

In den statischen RAMs werden Transistorkombinationen zur Aufbewahrung der Informationen verwendet. Eine solche Transistorkombination kann zwei verschiedene Zustände annehmen und behält eine somit eingetragene Information bis zum Abschalten oder Überschreiben mit einer neuen Information.

Dynamische RAMs speichern die Information als Ladung eines kleinen Kondensators ab. Diese Ladung muß auf Grund der Selbstentladung periodisch erneuert werden; dieser Vorgang wird mit REFRESH (Auffrischen) bezeichnet. Das Auffrischen wird bereits erreicht, wenn der Speicher gelesen wird. Die CPU U880 unterstützt diesen Vorgang durch Aussenden einer REFRESH-Information, um die zeitlichen Bedingungen zum Auffrischen unter allen Umständen zu gewährleisten. Werden die Zellen der dynamischen RAMs nicht spätestens nach 2 Millisekunden aufgefrischt, geht ihre gespeicherte Information verloren.

Trotz des nicht unerheblichen Mehraufwandes werden dynamische RAMs verwendet, da sie bei gleichen Abmessungen der Bausteine eine größere Speicherkapazität und kleinere Leistungsaufnahme gegenüber statischen RAMs aufweisen.

2.2.3. Ein/Ausgabe-Einheiten

Unter externen Geräten sollen im folgenden alle Geräte verstanden werden, mit denen Informationen in den Mikrorechner eingegeben oder vom Mikrorechner ausgegeben werden. Damit ist es möglich, sowohl Daten als auch Programme in den Mikrorechner zu bringen und die Ergebnisse für den Nutzer sichtbar zu machen. Solche Geräte sind Lochbandleser und -stanzer, Magnetbandtechnik, Tastaturen, Bildschirm usw.

Die Verbindung dieser Geräte mit dem Mikroprozessor erfolgt über sogenannte E/A-Funktionseinheiten, in denen spezielle integrierte Schaltungen enthalten sind. Diese Funktionseinheiten steuern selbständig die Arbeit der Geräte und treten mit der CPU nur zur Informationsübermittlung in Kontakt.

Damit wird die CPU entlastet und die Programmabarbeitung wesentlich effektiver.

Weiterhin können auch Funktionseinheiten angeschlossen werden, die beliebig zur Verfügung gestellte Meldesignale aus zu überwachenden Prozessen aufnehmen und sie für den Mikroprozessor aufbereiten. Gleichermäßen ist die Abgabe von Steuersignalen zur Beeinflussung bestimmter zu steuernder Prozesse möglich. Als integrierte Schaltkreise werden dazu im MRB Z1013 parallele E/A-Schaltkreise (Parallel Input Output-PIO) vom Typ U855 verwendet.

2.2.4. Verbindung der Funktionseinheiten

Der Mikroprozessor (CPU) sendet Signale ab und wertet bestimmte empfangene Signale aus. Diese Signale werden i. a. in allen angeschlossenen Funktionseinheiten benötigt.

Die Leitungen zur Übermittlung von Daten, Adressen und Systemsignalen, wie z. B. LESEN, SCHREIBEN, werden entsprechend ihrer Funktion zu Leitungsbündeln zusammengefaßt.

Da diese Leitungen die Daten und Informationen zwischen den einzelnen Funktionseinheiten transportieren, wurde der Begriff "Bus" für ein solches Leitungsbündel geprägt.

Demzufolge bezeichnet man die Datenleitungen als DATENBUS, die Adreßleitungen werden als ADRESSBUS und die Systemleitungen als STEUERBUS bezeichnet. Gelegentlich steht der Begriff "SYSTEMBUS" auch für alle Leitungen innerhalb des Mikrorechnersystems.

Durch Verwendung eines einheitlichen Systembusses ist es möglich, beliebige Funktionseinheiten einem bestehenden System hinzuzufügen, d. h. das System ständig zu erweitern.

Voraussetzung ist die Übereinstimmung der elektrischen Anschlüsse der jeweiligen Einheiten.

2.3. Programmabarbeitung

2.3.1. Ablauf in der CPU

Wie bereits gesagt, benötigt die CPU zur Lösung ihrer Aufgaben Anweisungen, die den gesamten Ablauf des Mikrorechners steuern. Diese Anweisungen oder Befehle findet die CPU in den angeschlossenen Speichereinheiten in einer ganz bestimmten, für sie verständlichen Form, die als Maschinencode bzw. MC bezeichnet wird. Alle Anweisungen an die CPU müssen also in Form dieses MC vorliegen bzw. sind in diese Form zu bringen. Im Anhang befindet sich eine Übersicht, in der die Befehle des U880 sowie deren Darstellung im Maschinencode enthalten sind. Um eine bestimmte Anweisungsfolge abzuarbeiten, ist es notwendig, der CPU mitzuteilen, in welchem Speicherbereich diese Befehle zu finden sind. Die CPU liest in diesem Bereich den Speicher und versucht die gelesenen Informationen als Befehl auszuführen. Dazu werden die gelesenen Informationen ins Befehlsregister transportiert und steuern von hier den Ablauf in der CPU. In Abhängigkeit vom konkreten Befehl werden entweder zusätzliche Informationen aus dem Speicher gelesen, werden Daten zum oder vom Speicher transportiert oder bestimmte logische Verknüpfungen in der ALU vorgenommen. Alle diese Aktivitäten der CPU sind mit dem Aussenden bestimmter Steuersignale verbunden, die die jeweilige Art der Operation anzeigen.

Einen Überblick über die Steuersignale bei einigen ausgewählten Operationen gibt Anlage 10. Zwischenzeitlich, während der Befehlsverarbeitung, sendet die CPU mit Hilfe des REFRESH-Registers eine Information zum Auffrischen des Speicherinhaltes

eventuell angeschlossener dynamischer Speicher aus. Während des Refresh-Zyklus wird der Befehl ausgeführt.

War der eben abgearbeitete Befehl ein Verarbeitungs- oder Transportbefehl, dann wird die Verarbeitung mit dem im Speicher folgenden Befehl fortgesetzt. Einige Befehle verändern aber diese Abarbeitungsreihenfolge, sie teilen der CPU mit, in welchem Speicherbereich der nächste Befehl zu finden ist.

Nach erfolgter Programmabarbeitung kann die CPU anhalten oder ein Steuerprogramm bearbeiten, mit dem z. B. die nächste Aufgabe ausgewählt werden kann.

2.3.2. Holen der Befehle

Für die Organisation der Befehlsabarbeitung besitzt die CPU ein besonderes Register, den Befehlszähler (PC). Der Befehlszähler besitzt 16 Bitstellen entsprechend der Anzahl der Adreßleitungen. Beim Betätigen der RESET-Taste wird dieses Register auf Null gesetzt. Damit liest die CPU den ersten abzuarbeitenden Befehl auf dem Platz Null.

Aus diesem Grund muß ein Programm ab dieser Stelle beginnen.

Um ein solches Programm ab Null nach dem Einschalten zur Verfügung zu stellen, ist ein nicht flüchtiger Speicher, z. B. ein EPROM notwendig. Befindet sich in diesem Speicherbereich kein Programmspeicher, so findet die CPU zufällige Bitkombinationen,

die als Befehl aufgefaßt und abgearbeitet werden.

Es kann also immer nur ein Programm nach dem Einschalten gestartet werden. Das wird im Normalfall ein Steuerprogramm sein, mit dem andere Programme aktiviert werden können. Soll ein anderes Steuerprogramm verwendet werden, ist der entsprechende Programmspeicher auszuwechseln. Da Programme auch in den Schreib-Lese-Speicher (RAM) geladen werden können, kann der Speicherbereich ab Null als RAM ausgelegt werden. Dann muß aber durch die Hardwareschaltung das Erreichen eines Steuerprogramms

sichergestellt werden, welches in einem beliebigen Speicherbereich stehen kann. Es können nun beliebige Betriebsprogramme in den Bereich ab Null geladen und verwendet werden, ohne jedesmal den Speicher auszuwechseln zu müssen. Damit ist ein solches System jeder Aufgabenstellung anpaßbar.

Der Bereich ab Null ist noch aus einem anderen Grunde besonders

für Betriebsprogramme geeignet. Er enthält einige ausgewählte Adressen, die sowohl von Programmen (sogenannte RESTART-Befehle) als auch im Resultat von externen Ereignissen (sogenannten Programmunterbrechungen) benötigt werden. Das Lesen der Befehle oder auch anderer Informationen geschieht durch Aussenden einer Adresse, begleitet von bestimmten Steuersignalen.

Durch eine Speicherverwaltung werden aus bestimmten Stellen dieser Adresse die Auswahl der entsprechenden Speichereinheit sowie eines Speicherbereiches vorgenommen. Der niederwertige Teil der Adresse wird verwendet, um in dem betreffenden Speicherbereich den konkreten Platz zu adressieren.

War die dort vorgefundene Information ein Befehl für die CPU so wird automatisch der Befehlszähler entsprechend der Befehlslänge erhöht (inkrementiert) und damit die neue Befehlsadresse bereitgestellt. Wurde der Befehl als ein Verzweigungsbefehl erkannt, wird im Befehlszähler die neue Adresse bereitgestellt und dann erneut durch Aussenden dieser Adresse ein bestimmter Speicherplatz ausgewählt.

2.3.3. Die Darstellung von Informationen im Speicher

Bisher wurde immer nur allgemein von "Informationen" gesprochen, die in einem "Speicher" zu finden sind. Diese Informationen waren sowohl Daten als auch Befehle, die in unterschiedlichen Speichertypen aufbewahrt wurden (ROM bzw. EPROM oder RAM).

Hinsichtlich ihrer Darstellung im Speicher unterscheiden sich diese Informationen auch nicht; es wäre auch möglich, Daten als Befehle zu betrachten und umgekehrt. Bei einer Abarbeitung durch die CPU kommen dabei selten sinnvolle Ergebnisse zustande.

Es wurde bereits der Speicher mit einer endlichen Anzahl Fächer eines Schranken verglichen, in denen Informationen abgelegt werden können. Durch eine Adresse wird die Nummer eines konkreten Faches bereitgestellt.

Wenn Informationen sowohl gelesen als auch abgelegt werden können, entspricht das dem Prinzip des Schreib-Lese-Speichers. Als Information kann das Vorhandensein eines Zeichens, einer Markierung oder dergleichen gedeutet werden. Ist diese Markierung dauerhaft (eingraviert), so handelt es sich um einen Nur-Lese-Speicher.

In einem Kasten können auch mehrere Informationen enthalten sein. Analog dazu sind die Speicher im Mikrorechner aufzufassen.

Eine Funktionseinheit "Speicher" besteht aus einer bestimmten Anzahl adressierbarer Plätze, wobei jeder Platz zwei verschiedene Zustände annehmen kann. Diese beiden Zustände werden durch die Dualziffern "0" und "1" repräsentiert. Die Auswahl dieser Plätze erfolgt über sogenannte Adreßleitungen, die Anzahl der Leitungen richtet sich nach der Kapazität des Speichers.

Der einfachste Aufwand ergibt sich bei der Festlegung der Speicherkapazität, d. h. der Anzahl der adressierbaren Speicherplätze, als ein Vielfaches einer Potenz zur Basis 2.

Eine Adreßleitung kann zwei Zustände annehmen, entweder hohen Spannungspegel, sogenannten "H-Pegel" (logisch "1"), als auch niedrigen Spannungspegel, sogenannten "L-Pegel" (logisch "0"). Damit wären zwei verschiedene Speicherplätze adressierbar.

Zwei Adreßleitungen können zusammen bereits 4 Zustände annehmen, damit sind 4 verschiedene Speicherplätze (mit den Adressen 00, 01, 10 und 11) adressierbar. Demzufolge werden bei 10 Adreßleitungen $2^{10} = 1024 = 1\text{K}$ Speicherplätze adressiert. Damit ergibt sich:

Kapazität = 2^n (n = Anzahl der Adreßleitungen)

Die CPU U880 hat 16 Leitungen für die Bildung von Adressen zur Verfügung, d. h. sie kann maximal $2^{16} = 65536 \hat{=} 64\text{K}$ Speicherplätze adressieren.

Eine weitere Eigenschaft einer Speichereinheit ist die Aufrufbreite. Hierunter wird verstanden, wieviel Speicherplätze gleichzeitig mit einer Adresse angesprochen werden können, um die Information parallel zu verarbeiten. Diese Aufrufbreite ist verschieden: bei ROMs und EPROMs beträgt sie 8 Stellen, bei statischen RAMs 1, 4 oder 8 Stellen und bei dynamischen RAMs im allgemeinen eine Stelle. Um die mögliche Verarbeitungsbreite des Mikroprozessors U880 mit 8 Datenleitungen zu nutzen, müssen in einer Speichereinheit mehrere Speicherschaltkreise kombiniert werden, um damit die gewünschte Aufrufbreite zu realisieren. Das geschieht, indem die Adreßanschlüsse von acht Speicherschaltkreisen mit den jeweiligen Adreßleitungen der CPU verbunden werden. Jeder der Speicherschaltkreise wird an einer Datenleitung angeschlossen, die Auswahl erfolgt für alle acht Schaltkreise mit einem gemeinsamen Auswahlsignal.

2.4. Grundbegriffe der Software

2.4.1. Darstellung von Zahlen

Das Wesentliche bei der Programmabarbeitung besteht in der Veränderung der eingegebenen Zahlen, um die gewünschten Ergebnisse zu erhalten. Das gewohnte Dezimalsystem ist für die Zahlendarstellung im Mikrorechner nicht geeignet; die zwei möglichen Zustände führen auf ein anderes Zahlensystem, das sogenannte Dualsystem. Dieses kennt nur die Ziffern 0 und 1, welche mit dem "L"- und "H"-Pegel der Informationsspeicherung identisch sind.

Da aber die Bildung von Zahlen sowohl im Dezimalsystem als auch im Dualsystem nach gleichen Gesetzmäßigkeiten verläuft, ist eine Umrechnung unproblematisch und kann durch entsprechende Programme vom Mikroprozessor vorgenommen werden. Diese Zahlenbildung kann mit folgender Gleichung beschrieben werden:

$$Z = d * x^n + d * x^{n-1} + \dots + d * x^1 + d * x^0$$

wobei bedeuten:

Z = Zahlenwert
d = Ziffern innerhalb des Wertebereichs im Zahlensystem
x = Basis des Zahlensystems
n = ganzzahliger Exponent

Im Dezimalsystem kann d die Ziffern 0 ... 9 annehmen, x ist dann gleich 10. Im Dualsystem ist d entweder 0 oder 1, die Basis ist gleich 2.

Ein Beispiel soll das verdeutlichen.

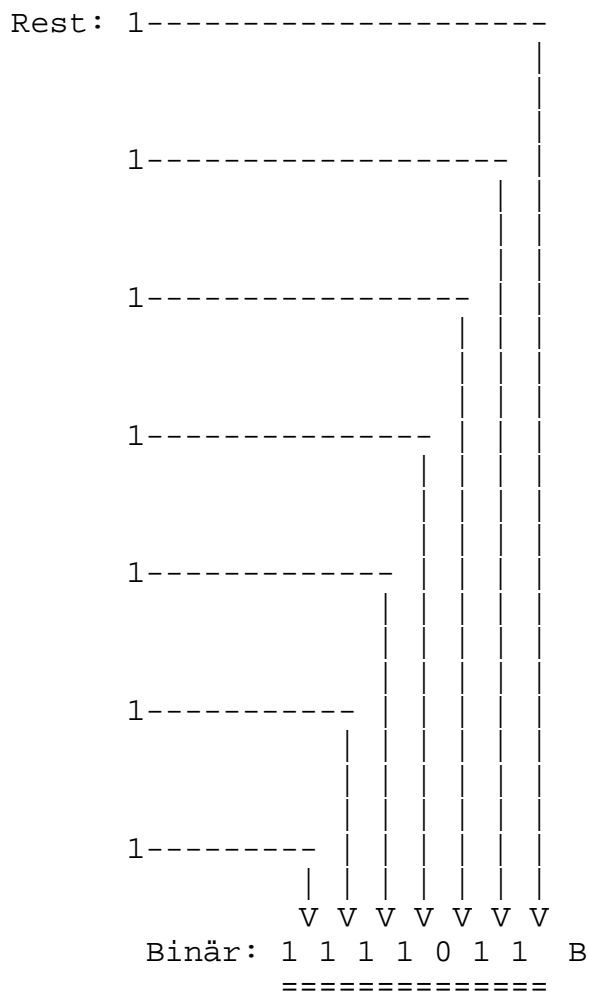
$$\begin{aligned} 123 &= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 && \text{(dezimal)} \\ &= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 11111011\text{B} && \text{(dual)} \end{aligned}$$

Das "B" hinter der Dualzahl soll zur Unterscheidung zur Dezimalzahl, die ohne Kennzeichnung geschrieben wird, dienen. "B" bedeutet "binär", abgeleitet von den zwei Zuständen. Nun wäre eine solche Umrechnung per Hand kompliziert. Es gibt jedoch ein einfaches Umrechnungsverfahren, das am deutlichsten durch ein Beispiel wird.

```

123 : 2 = 61
-----|
|
V
61 : 2 = 30
-----|
|
V
30 : 2 = 15
-----|
|
V
15 : 2 = 7
-----|
|
V
7 : 2 = 3
-----|
|
V
3 : 2 = 1
-----|
|
V
1 : 2 = 0
-----|

```



Die Speichereinheiten in U880 Systemen, wie dem Z1013, besitzen in der Regel eine Aufrufbreite von 8 Bit. Das heißt, auf einem Speicherplatz sind gleichzeitig 8 Bits, die zu einem Byte zusammengefaßt werden, gespeichert. Ein Byte kann demzufolge $2^8 = 256$ verschiedene Werte annehmen.

Für ein Byte ergeben sich die folgenden Wertigkeiten für die einzelnen Bits:

Byte:

								Wertigkeit oder
								Exponent zur
								Basis 2
7	6	5	4	3	2	1	0	
128	64	32	16	8	4	2	1	Zahlenwert
höherwertiges				niederwertiges				Halbbyte
								(BCD-Ziffer)

Die in der Bytedarstellung eingetragenen Ziffern geben die Numerierung der einzelnen Bits an. Das Bit 0 besitzt die niedrigste Wertigkeit, das Bit 7 die höchste.

Eine vorzeichenlose ganze Zahl mit der Bitfolge 01001010B kann auch in der Form:

$$Z = 0 \cdot 128 + 1 \cdot 64 + 0 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 74$$

geschrieben werden.

Sollen auch negative Zahlen dargestellt werden, besitzt das Bit 7 die Funktion des Vorzeichens.

Eine Dualzahl 10110110B kann als

$$Z = -1 \cdot 128 + 0 \cdot 64 + 1 \cdot 32 + 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = -74$$

aufgefaßt werden, diese Art bezeichnet man als Zweierkomplement. Damit ergibt sich ein Zahlenbereich für ganze vorzeichenlose Zahlen von 0 bis 255 und für vorzeichenbehaftete Zahlen von -128 über 0 bis +127.

Sollen größere Zahlen dargestellt werden, müssen 2 und mehr Bytes dafür genutzt werden. Die Zusammenfassung von 2 Bytes wird als Wort bezeichnet, analog dazu 4 Bytes als Doppelwort.

Die einzelnen Bytes des Maschinenkodes werden als Dualzahlen, d. h. als Ziffernfolgen von "0" oder "1" dargestellt. Insbesondere bei großen Programmen ergibt sich damit ein sehr großer Schreibaufwand, um diese Dualzahlen zu notieren.

Deshalb hat sich ein anderes Zahlensystem, das sogenannte Hexadezimalsystem für die Darstellung von Zahlen und Programmen

bei Mikrorechnern durchgesetzt. (Die Bezeichnung Hexadezimalsystem ist umgangssprachlich, exakt heißt es Sedezimalsystem.)

Im Hexadezimalsystem werden 4 benachbarte Dualziffern zusammengefaßt und durch eine Hexadezimalziffer dargestellt. Mit vier Dualziffern können 16 verschiedene Zustände dargestellt werden. Die Zahlen "0" bis "9" sind gleich den Dezimalzahlen, größer als "9" werden die ersten Buchstaben des Alphabets verwendet. Die folgende Tabelle enthält eine Gegenüberstellung von Dual-, Dezimal- und Hexadezimalziffern.

DUAL	DEZ	HEX	DUAL	DEZ	HEX
0 0 0 0	0	0	1 0 0 0	8	8
0 0 0 1	1	1	1 0 0 1	9	9
0 0 1 0	2	2	1 0 1 0	10	A
0 0 1 1	3	3	1 0 1 1	11	B
0 1 0 0	4	4	1 1 0 0	12	C
0 1 0 1	5	5	1 1 0 1	13	D
0 1 1 0	6	6	1 1 1 0	14	E
0 1 1 1	7	7	1 1 1 1	15	F

Da in einem Byte (mit 8 Bits) zwei sogenannte Halbbytes zu je 4

Bits enthalten sind, kann ein Byte mit 2 Hexadezimalziffern dargestellt werden.

Die binäre Darstellung der Dezimalzahlen von 0 bis 9 nennt man auch BCD-Zahlen. Auch mit dieser Zahlendarstellung kann gerechnet werden. Dabei muß aber eine Dezimalkorrektur vorgenommen werden. Warum und wie, wird bei der Erläuterung des DAA-Befehls im Befehlssatz genauer erklärt. Zur besseren Unterscheidung zu den Dezimalzahlen werden die Hexadezimalzahlen in Protokollen oder Drucklisten durch ein nachgestelltes Zeichen "H" gekennzeichnet.

Nehmen wir z. B. ein Byte in Binärdarstellung:

$$\begin{array}{rclcl} 0111 & 1011 & = & 7BH & = 123 \\ 1. & 2. & & \text{Halbbyte} & \end{array}$$

Dabei sind die Wertigkeiten der einzelnen Bits in einem Halbbyte:

$$\begin{array}{rcl} 3 & 2 & 1 & 0 & \text{Wertigkeit} \\ \text{-----} & & & & \\ 8 & 4 & 2 & 1 & \text{Zahlenwert} \end{array}$$

Die Umwandlung einer Hexadezimalzahl in die entsprechende Dezimalzahl geschieht am einfachsten auf folgende Weise:

$$\begin{aligned} 7BH &= 7 \times 16^1 + B \times 16^0 \\ &= 7 \times 16^1 + 11 \times 16^0 \\ &= 123 \end{aligned}$$

Die Umrechnung Dezimal- in Hexadezimalzahl erfolgt nach einem analogen Schema wie die Umrechnung Dezimal- in Dualzahl, z. B.

	Dez.	Hex.	
45 346	: 16 = 2 834	Rest 2	2 -----
2 834	: 16 = 177	2	2 -----
177	: 16 = 11	1	1 -----
11	: 16 = 11	11	B -----
			V V V V
			B 1 2 2 H
			=====

Um den Vorteil dieser Schreibweise deutlich werden zu lassen, hier zum Vergleich diese Zahl in Binärdarstellung:

$$1011 \ 0001 \ 0010 \ 0010B.$$

2.4.2. Logische Operationen

Mit den Dualzahlen lassen sich verschiedene logische Operationen durchführen. Bei den logischen Verknüpfungen werden die Dualzahlen als vorzeichenlose, ganze Zahlen aufgefaßt.

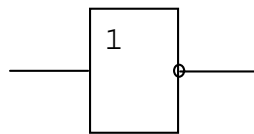
Die wichtigsten dieser Operationen sind:

- Komplementbildung (NEGATION):

Eine Dualzahl wird in ihr Komplement überführt, indem alle Bitstellen einzeln auf den entgegengesetzten Wert gebracht werden.

```
Zahl:      0 1 0 0 1 0 1 0
-----
Ergebnis: 1 0 1 1 0 1 0 1
```

Diese Operation wird nur mit einer Dualzahl durchgeführt. In Stromlaufplänen finden Sie dafür das folgende Sinnbild:

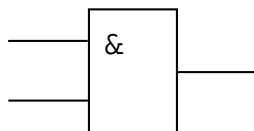


- UND-Verknüpfung (KONJUNKTION, AND)

Eine Konjunktion wird mit zwei Dualzahlen durchgeführt. Dabei bleibt nur in der Bitposition eine "1" stehen, in welcher in der ersten und in der zweiten Dualzahl eine "1" stehen.

```
1. Zahl: 0 1 0 0 1 0 1 0
2. Zahl: 0 0 0 1 1 1 1 1
-----
Ergebnis: 0 0 0 0 1 0 1 0
```

Sinnbild:



Zur besseren Darstellung der logischen Operationen ist es üblich, sich eine beliebige Bitposition auszuwählen und in einer Wertetabelle alle möglichen Kombinationen und deren Ergebnisse zu erfassen. Die Wertetabelle der Konjunktion besitzt danach folgendes Aussehen (gleiche Bitposition vorausgesetzt):

1. Zahl	2. Zahl	Ergebnis
0	0	0
0	1	0
1	0	0
1	1	1

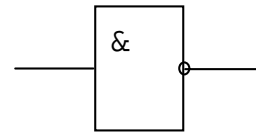
Besonders bei komplizierten Verknüpfungen stellt die Wertetabelle ein sehr einfaches Hilfsmittel dar.

- NICHT-UND-Verknüpfung (NAND)

Diese Verknüpfung stellt eine Konjunktion mit anschließender Negation dar.

1. Zahl	2. Zahl	Ergebnis
0	0	1
0	1	1
1	0	1
1	1	0

Sinnbild:

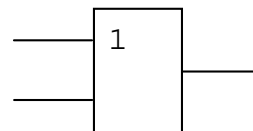


- ODER-Verknüpfung (DISJUNKTION, OR)

Zwei disjunktiv verknüpfte Dualzahlen liefern im Ergebnis eine "1", wenn in der ersten oder zweiten Dualzahl in der jeweiligen Bitposition eine "1" steht.

1. Zahl	2. Zahl	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	1

Sinnbild:

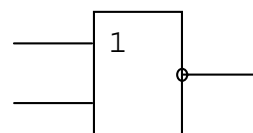


- NICHT-ODER-Verknüpfung: (NOR)

Diese Verknüpfung stellt eine Disjunktion mit anschließender Negation dar.

1. Zahl	2. Zahl	Ergebnis
0	0	1
0	1	0
1	0	0
1	1	0

Sinnbild:



- Exklusiv-ODER bzw. (ANTIVALENZ, EXOR)

In der jeweiligen Bitposition der Ergebnisse wird eine "1" eingetragen, wenn sich in dieser Bitposition die beiden Dualzahlen unterscheiden.

1. Zahl	2. Zahl	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	0

Sind beide Dualzahlen gleich, wird das Ergebnis auf Null gesetzt. Das wird besonders verwendet, um einen bestimmten Zwischenspeicher, z. B. das A-Register der CPU, zu löschen, indem der Inhalt des A-Registers mit sich selbst durch einen XOR-Befehl verknüpft wird (XOR A).

2.4.3. Arithmetische Verknüpfungen

Zu den arithmetischen Operationen gehören Addition und Subtraktion. Die Multiplikation zweier Dualzahlen kann durch fortlaufende Addition einer Dualzahl bei gleichzeitiger Verringerung der anderen Dualzahl, bis diese Null ist, vorgenommen werden. Auch eine teilweise Addition, kombiniert mit Verschiebung von Ergebnis und Operand ist üblich. Die Division kann analog dazu als eine fortlaufende Subtraktion einer Dualzahl von einer anderen durchgeführt werden. Dabei wird der Dividend solange vom Divisor subtrahiert und der Quotient jeweils um 1 erhöht, bis der Divisor kleiner als der Dividend geworden ist. Der Quotient als Ergebnis enthält damit die Anzahl der benötigten Subtraktionsschritte, im Divisor ist der Rest enthalten.

Nachfolgend die arithmetischen Operationen im einzelnen:

Es empfiehlt sich, die Beispiele mit anderen Zahlen selbst noch einmal nachzuvollziehen.

- ADDITION:

Die Addition zweier Dualzahlen liefert folgendes in der Wertetabelle sichtbare Ergebnis. Dabei wird der Übertrag in der letzten Spalte in der nächsthöheren Bitposition ausgewertet.

0	+	0	=	0	
0	+	1	=	1	
1	+	0	=	1	
1	+	1	=	0	Übertrag 1

Sollen z. B. die Zahlen 26 und 43 miteinander addiert werden, ergibt das folgende Rechnung:

26:	0	0	0	1	1	0	1	0	
43:	0	0	1	0	1	0	1	1	
Überträge:	1	1	1		1				

Ergebnis:	0	1	0	0	0	1	0	1	= 69

Werden zwei Zahlen addiert, deren Ergebnis den Zahlenbereich überschreitet, kommt es zum Überlauf, d. h. das ermittelte Ergebnis ist falsch. An der Addition der Zahlen 69 und 73 soll das im Rechenschema gezeigt werden.

69:	0	1	0	0	0	1	0	1	Zahlenbereich:
73:	0	1	0	0	1	0	0	1	-128 <= x <= 127
Überträge:	1						1		

Ergebnis:	1	0	0	0	1	1	1	0	= -114

Im Ergebnis entsteht die Zahl -114, obwohl die Addition dieser Zahlen zu dem Ergebnis 132 führen müßte. Dieser Überlauf ist dadurch charakterisiert, daß ein Übertrag in die

Vorzeichenstelle ein-, aber kein Übertrag aus der Vorzeichenstelle herausläuft.

- SUBTRAKTION

Die Subtraktion zweier Dualzahlen verläuft ähnlich der der Dezimalzahlen, d. h. wenn die Subtraktion einen negativen Wert in der Bitposition ergibt, muß von der höherwertigen Stelle etwas "geborgt" werden, es entsteht ein Übertrag.

Daraus resultiert folgende Wertetabelle:

1. Zahl		2. Zahl	=	Ergebnis	
0	-	0	=	0	
0	-	1	=	1	(0-1 => 10-1 => 1+Übertrag)
1	-	0	=	1	
1	-	1	=	0	'-> 'geborgte 1'

Subtrahiert man die Dualzahl 26 von der 43, so ergibt sich folgendes Rechenschema:

43:		0	0	1	0	1	0	1	1	
26:		0	0	0	1	1	0	1	0	
Überträge:				1						

Ergebnis:		0	0	0	1	0	0	0	1	= 17

Subtrahiert man die Zahlen in anderer Weise, d. h. die Dualzahl 43 von der 26, so kann man auch einen Vorzeichenwechsel beobachten.

26:		0	0	0	1	1	0	1	0	
43:		0	0	1	0	1	0	1	1	
Überträge: 1<=		1	1		1	1	1	1		

Ergebnis:		1	1	1	0	1	1	1	1	= -17

Da hier aber ein Übertrag sowohl in die Vorzeichenstelle hinein als auch ein Übertrag aus der Vorzeichenstelle heraus erfolgt, handelt es sich um keinen Überlauf und das Ergebnis ist korrekt. Dieser herauslaufende Übertrag wird bei Zahlen im Wort- oder Doppelwortformat weiterverwendet.

- ZWEIERKOMPLEMENT:

Eine Ergebnisdarstellung wie im vorangegangenen Subtraktionsbeispiel wird Zweierkomplement genannt. Jede Zahl kann in ihr Zweierkomplement überführt werden, wenn diese Zahl zuerst in ihr Komplement umgewandelt (negiert) wird und anschließend zur niederwertigsten Bitposition eine "1" addiert wird.

```

-17:          1  1  1  0  1  1  1  1
Negation:     0  0  0  1  0  0  0  0
Addition:                    1
-----
Ergebnis:    0  0  0  1  0  0  0  1 = 17

```

Das Zweierkomplement wird verwendet, um eine Subtraktion auf eine Addition zurückzuführen.

Die Addition einer Zahl und ihres Zweierkomplements liefert als Ergebnis immer eine Null.

Abschließend noch ein Beispiel zur Multiplikation, die hier als eine fortlaufende Addition betrachtet werden soll.

Es werden die Dualzahlen 9 und 5 miteinander multipliziert:

```

Ausgangswerte:  5:  0  0  0  0  0  1  0  1
                 9:  0  0  0  0  1  0  0  1
                                                    Multiplika-
                                                    tor 5

          9:  0  0  0  0  1  0  0  1           5
+9:         0  0  0  0  1  0  0  1           4
-----
          =  0  0  0  1  0  0  1  0           3
+9:         0  0  0  0  1  0  0  1
-----
          =  0  0  0  1  1  0  1  1           2
+9:         0  0  0  0  1  0  0  1
-----
          =  0  0  1  0  0  1  0  0           1
+9:         0  0  0  0  1  0  0  1
-----
Ergebnis:      =  0  0  1  0  1  1  0  1 = 45

```

Die Multiplikation mit teilweiser Addition und Verschiebung kann analog zur Multiplikation von Dezimalzahlen dargestellt werden:

```

Ausgangswerte:      0  0  0  0  1  0  0  1  *  0  1  0  1
-----
                   0  0  0  0  1  0  0  1  |  1
                   0  0  0  0  0  0  0  0  |  0 = 5
                   0  0  0  0  1  0  0  1  |  1
                   0  0  0  0  0  0  0  0  |  0
-----
Ergebnis:  0  0  0  0  0  1  0  1  1  0  1  = 45

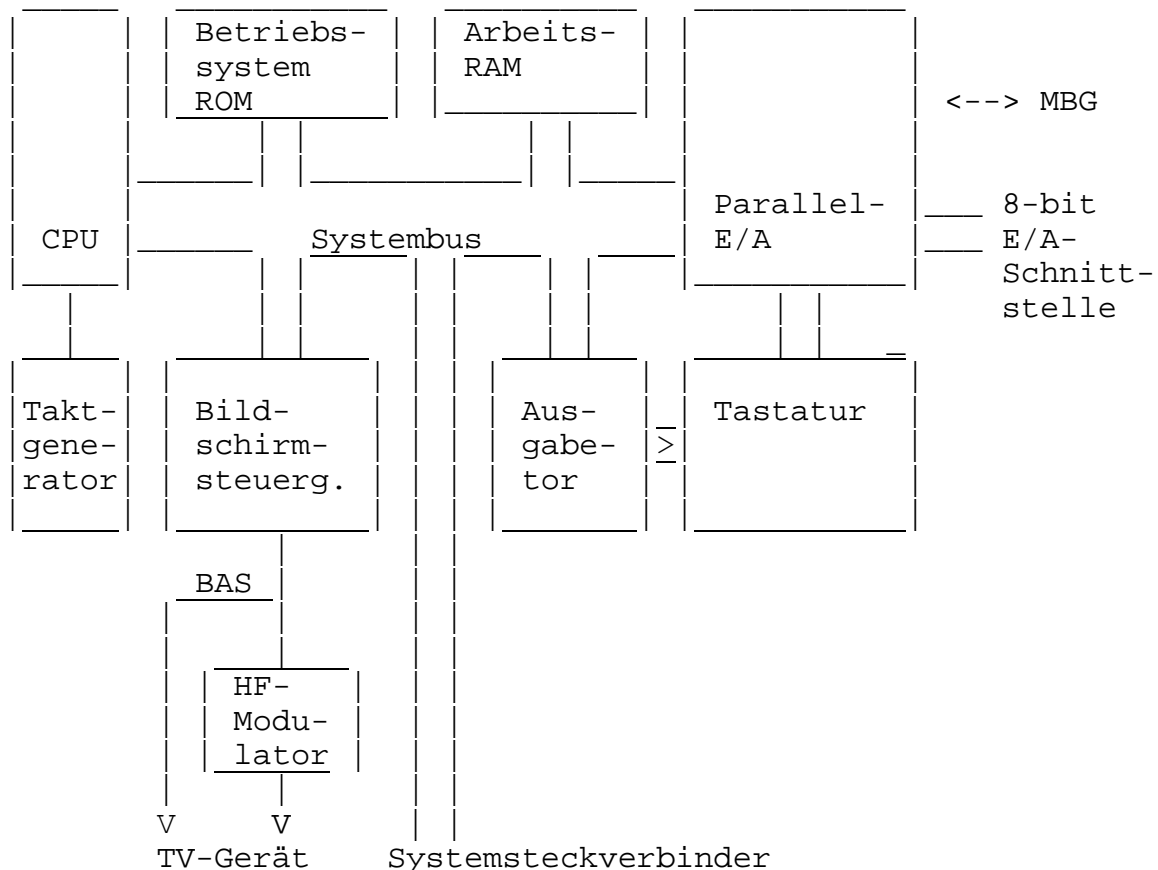
```

Bei der teilweisen Addition kann ebenfalls ein Übertrag auftreten, d. h. der Zahlenbereich überschritten werden. Wenn die Kontrolle nicht in jedem Zwischenschritt vorgenommen wird, ist mit fehlerhaften Ergebnissen zu rechnen.

3. Hardware des Z1013

Am konkreten Beispiel des MRB Z1013 soll in diesem Kapitel die Arbeitsweise eines Mikrorechners erläutert werden. Grundlage dafür bilden Stromlaufpläne des Z 1013, die Sie in der Anlage 16 finden.

3.1. Blockschaltbild



3.2. Steuerung des Mikroprozessors

3.2.1. Beschreibung der Steuersignale

Um den ordnungsgemäßen Betrieb der CPU zu gewährleisten, sind bestimmte Steuersignale notwendig. Andere Signale werden von der CPU gebildet und kennzeichnen bestimmte Zustände während der Abarbeitung von Befehlen. Im folgenden werden alle Steuersignale der CPU und des Systembusses beschrieben.

Signale, die mit einem Schrägstrich beginnen, sind sogenannte LOW-aktive Signale, die normalerweise H-Pegel führen und bei ihrer Aktivierung L-Pegel zeigen.

Nach dem Signal steht in Klammern ein "A" für von der CPU ausgesandte, ein "E" für von der CPU empfangene und ein "B" für

Signale, die sowohl von der CPU empfangen als auch ausgesandt werden können.

(A-Ausgabe, E-Eingabe, B-bidirektional, d. h. sowohl Ein- als auch Ausgabe)

- A0 bis A15 (A)

Sie bilden den 16-Bit-Adreßbus. Sie werden in der CPU gebildet und bei der Arbeit mit den Speichern als Speicheradresse sowie die Leitungen A0 bis A7 bei der E/A-Arbeit als E/A-Adresse verwendet.

- A0 bis A6 (A)

Sie dienen zum Auffrischen dynamischer Speicher.

- D0 bis D7 (B)

Diese Leitungen stellen den 8 Bit-Datenbus dar. Die Signale können sowohl von der CPU gebildet werden (bei Ausgabe oder Speicherschreiben) oder sie werden von den ausgewählten Funktionseinheiten erzeugt (bei Eingaben oder Speicherlesen).

- /MREQ (A)

Dieses Signal wird benötigt, um eine auf dem Adreßbus ausgesandte Adresse zur Speicheradresse zu erklären und einen Speicherzugriff durchzuführen.

- /IORQ (A)

Das Signal kennzeichnet die auf dem Adreßbus anliegende Adresse als Adresse einer E/A-Funktionsgruppe. Dabei werden nur die Adreßleitungen A0 bis A7 in die Auswahl einbezogen.

- /M1 (A)

Es charakterisiert den Maschinenzyklus 1. Dieses Signal wird von der CPU ausgesendet und kennzeichnet in Verbindung mit dem Signal /MREQ, daß vom Speicher ein Befehl geholt wird. In Verbindung mit dem Signal /IORQ wird gekennzeichnet, daß von einem interrupterzeugenden Baustein (s. 4.4.) der sogenannte Interruptvektor gelesen wird (Vektorlesen).

- /RD (A)

Es wird in den angeschlossenen Funktionseinheiten ausgewertet und legt die Richtung des Datentransportes als "Lesen", d. h. zur Eingabe in die CPU fest.

- /WR (A)

Es kennzeichnet die Richtung des Datentransportes für die angeschlossenen Funktionseinheiten als "Schreiben", d. h. die CPU sendet Daten aus.

- /RFSH (A)

Zeigt den angeschlossenen Speichereinheiten in Verbindung mit /MREQ, daß auf dem Adreßbus eine Refresh-Information verfügbar ist. Diese Refresh-Information besteht aus einer 7-Bit-Adresse (A0 bis A6), die festlegt, welche Speicherzellen in den dynamischen Speichern; aufgefrischt werden sollen. Das Adreßbit 7 kann durch den Programmierer gesetzt oder rückgesetzt werden und ist Bestandteil der Refresh-Information. Auf dem höherwertigen Teil des Adreßbusses wird der Inhalt des I-Registers ausgesandt.

- /HALT (A)

Wird von der CPU ausgesandt, wenn der soeben gelesene Befehl den Operationskode 76H hatte. Die Abarbeitung wird unterbrochen, der Befehlszähler zeigt auf den nächsten Befehl. Die Refresh-Steuerung wird aufrechterhalten, eine Fortsetzung der CPU-Arbeit ist nur nach Reset oder Interrupt möglich.

- /WAIT (E)

Wird von der CPU zu bestimmten Zeiten abgetastet. Ist dieses Signal Low, wird die Arbeit der CPU angehalten, die Informationen auf dem Systembus bleiben erhalten. Anwendung findet dieses Signal vor allem bei der Anpassung der Verarbeitungsgeschwindigkeit von langsamen Funktionseinheiten, indem die Arbeitsgeschwindigkeit der CPU durch solche WAIT-Zyklen der entsprechenden Funktionseinheit angepaßt wird. Während des WAIT-Zustandes findet kein Refresh-Zyklus statt.

- /INT (E)

Wird von der CPU am Ende eines Befehls abgetastet und signalisiert, daß eine angeschlossene Funktionseinheit das gerade abzuarbeitende Programm unterbrechen möchte, damit von der CPU die Ursache dieser Unterbrechung analysiert und bearbeitet werden kann. Die Ursachen dieser Unterbrechung können ein notwendiger Datentransport zwischen CPU und Interface-Baustein sein oder eine Ereignismeldung aus einem zu überwachenden Prozeß. Die Funktionseinheiten sind untereinander über eine sogenannte Prioritätskette miteinander verbunden, um die jeweils wichtigste Unterbrechung vorrangig zu behandeln. Die CPU kann ihrerseits die Annahme einer Unterbrechung sperren, um z. B. bestimmte Programmabschnitte störungsfrei abzuarbeiten.

Nach Freigabe des Unterbrechungseinganges wird die dort eventuell gespeicherte Unterbrechung ausgewertet.

- /NMI (E)

Dieser Eingang stellt äquivalent zum INT-Signal eine Unterbrechungsmöglichkeit der laufenden CPU-Arbeit dar.

- /NMI (E)

Dieser Eingang stellt äquivalent zum INT-Signal eine Unterbrechungsmöglichkeit der laufenden CPU-Arbeit dar, die allerdings nicht gesperrt werden kann. Die Abarbeitung des Unterbrechungsbehandlungsprogramms beginnt ab der Adresse 66H, nachdem zuvor die Fortsetzungsadresse des gerade laufenden Programmes gerettet wurde.

- /RESET (E)

Unterbricht jede weitere Arbeit der CPU, stellt einen Anfangszustand ein und gibt mit dem Übergang nach H-Pegel die CPU wieder frei. Da das Reset-Signal meist manuell erzeugt wird, wird durch die Schaltung eine Verkürzung dieses Signals vorgenommen, um angeschlossenen dynamischen Speichern die Refresh-Informationen zu garantieren.

- C (E)

Stellt den der CPU zugeführten Systemtakt dar. Dieser Takt ist gleichzeitig in allen Funktionseinheiten verfügbar und sichert die Synchronität aller Baugruppen.

- /BUSRQ (E)

Diese Leitung wird am Ende eines Befehls durch die CPU abgetastet. Dieses Signal kennzeichnet, daß eine angeschlossene Funktionseinheit den Systembus benötigt, um ihrerseits die Vorgänge im Mikrorechner zu steuern. Die CPU unterbricht das laufende Programm und setzt ihre Ausgänge in den hochohmigen Zustand.

Gleichzeitig wird ein Quittungssignal von der CPU aktiviert, welches den hochohmigen Zustand anzeigt. Dieser bleibt solange bestehen, wie das Signal BUSRQ aktiv ist, d. h. L-Pegel führt. Danach wird das Quittungssignal von der CPU abgeschaltet, alle Ausgänge nehmen wieder ihr erforderliches Potential ein und die Abarbeitung wird fortgesetzt.

Während des hochohmigen Zustandes kann die CPU keine Refresh-Informationen aussenden.

- /BUSAK (A)

Ist das Quittungssignal der CPU, welches den hochohmigen Zustand kennzeichnet und damit der den Systembus anfordernden Funktionseinheit den Zugriff erlaubt.

Weiterhin umfaßt der Systembus folgende Signale, die nicht von der CPU ausgesandt oder empfangen werden:

- /MEMDI

Stellt ein Systemsignal dar, mit dem angeschlossene Funktionseinheiten den Zugriff auf Speichereinheiten auf der Leiterplatte der Grundausbaustufe verhindern können.

Dieses Signal wird erzeugt, wenn Speichererweiterungen die festgelegten Speicheradressen des Grundgerätes ebenfalls verwenden. Es wird verhindert, daß nicht mehr als eine Speichereinheit den Datenbus benutzen kann.

- /IODI

Stellt analog zum MEMDI-Signal eine Möglichkeit dar, bestimmte Adreßbereiche auszublenden und Konflikte auf dem Datenbus bei der E/A-Arbeit zu verhindern.

- /IEI und /IEO

Werden zur Bildung der Prioritätskette der interrupterzeugenden Funktionseinheiten benötigt. Jeweils der Ausgang (IEO) der höheren Priorität wird dem Eingang (IEI) der nächstfolgenden Prioritätsstufe zugeführt (vergleiche auch Abschnitt 4.4 Interruptbehandlung).

Ein Interrupt kann von einer Funktionseinheit nur ausgelöst werden, wenn das zugeführte Signal IEI H-Pegel führt. Gleichzeitig wird das abgegebene Signal IEO auf L-Pegel gehalten. Damit wird sichergestellt, daß immer nur die in der Prioritätskette am weitesten am Anfang eingereihte Funktionseinheit eine Unterbrechung auslösen kann.

- /BAI und /BAO

Stellen analog zu den Signalen IEI und IEO die Signale einer Prioritätskette dar, die alle Funktionseinheiten verbindet, die eine Anforderung auf den Systembus (BUSRQ) stellen können. Für die Benutzer des MRB Z1013 werden diese Signale kaum Bedeutung haben.

- RDY

Stellt ein ähnliches Signal wie WAIT dar, um langsame Funktionseinheiten an die CPU anzupassen. Es kennzeichnet die Kommunikationsbereitschaft einer Funktionseinheit und kann mit der WAIT-Leitung verbunden werden. Im Gegensatz zu den meisten anderen Steuersignalen ist es nicht Low-aktiv.

3.2.2. Takterzeugung

Der Taktgenerator wird durch drei Gatter von A6, dem Kondensator C7.1 und den Widerständen R38 und R39 gebildet. Stabilisiert wird die Taktfrequenz durch den Schwingquarz Q1. Dieser schwingt mit einer Frequenz von 8 MHz. Der Takt wird dem Binärteiler A3 zugeführt, an dessen Ausgängen die Taktfrequenzen von 4 MHz, 2 MHz und 1 MHz anliegen. Der Z 1013.01 arbeitet standardmäßig mit 1 MHz Systemtakt, der Z 1013.12 mit 2 MHz.

Hinweis: Das Umrüsten des Z 1013.01 auf 2 MHz führt zum Erlöschen der Garantie. Die Taktfrequenz 4 MHz ist nicht zugelassen!

Je nach Lage von E1 erhält die CPU den Takt mit der Frequenz entsprechend folgender Zuordnung:

Lage	Systemtakt
E1.1	1 MHz
E1.2	2 MHz

Mittels des Widerstandes R52 erfolgt noch die erforderliche Pegelanpassung zur Speisung der CPU (A7) und des E/A-Schaltkreises A45.

Dieser Takt realisiert die Synchronität aller Zeitabläufe.

3.2.3 RESET-Logik

Um einen definierten Anfangszustand der CPU zu erreichen, ist die RESET-Steuerung erforderlich. RESET kann von 3 Stellen ausgelöst werden:

1. Taste TA1 auf der Leiterplatte (RESET-Taste)
2. Externe Tastatur über den Steckverbinderanschluß X2:A02
3. A20 des Systemsteckverbinders X1

Eine spezielle Schaltung sorgt dafür, daß der Datenbustreiber A1 inaktiv wird, d. h. er wird vom Prozessor getrennt. Unmittelbar an der CPU werden die Datenleitungen über die Widerstände R44 ... R51 auf Masse, d. h. L-Pegel gelegt.

Da die CPU nach aktiven RESET den Befehlszähler auf die 0000H einstellt, werden nun auf dieser Adresse die Daten 00H gelesen. Das bedeutet für den Prozessor die Ausführung eines sogenannten Leerbefehls (NOP, s. 4.3.15). Bei dessen Ausführung wird der Befehlszähler um eins erhöht. Auf diese Art und Weise zählen die Adressen hoch, bis die Adresse des Betriebssystems erreicht wird und das Signal /CS aktiviert wird, das den Datenbus mit Hilfe der Logik wieder frei gibt. Als nächstes wird jetzt der erste Befehl des Betriebssystemprogrammes gelesen und dieses wird abgearbeitet.

Damit die Länge des Reset-Impulses von der Länge der Betätigung unabhängig wird, wurde ein Monoflop verwendet. Damit wird eine zeitgerechte Auffrischung der dynamischen Speicher gewährleistet. Einige periphere Schaltkreise besitzen keinen Reset-Anschluß. Sie werten das alleinige Auftreten des Signale /M1 als Resetimpuls. Damit auch diese Schaltkreise in einen definierten Anfangszustand versetzt werden können, wurden die Signale /RESET und /M1 zum Signal /PM1 verknüpft, welches die Rücksetzfunktion ausführt.

3.3 Speichereinheiten

3.3.1. Anschluß

Der Anschluß der Speicherschaltkreise ist abhängig vom Typ. Im MRB Z1013 werden drei Arten verwendet.

In einem PROM U 2616 bzw. ROM U 2316 (A14) ist das Monitorprogramm enthalten. Dieser Schaltkreis besitzt eine Kapazität von 2048 (=2K) Speicherplätzen, wobei bei jedem Zugriff acht Bit parallel gelesen werden. Um diese 2 KByte zu adressieren, sind 11 Adreßleitungen (A0 ... A9) notwendig.

Die verwendeten statischen Schreib-Lese-Speicher besitzen eine Kapazität von 1024 (=1K) Plätzen, wobei jeweils 4 Bit gleichzeitig angesprochen werden. Erst zwei dieser Schaltkreise besitzen deshalb eine Kapazität von 1 KByte, wobei 10 Adreßleitungen (A0 bis A9) ausreichen.

Mit Hilfe dieser 11 bzw. 10 Adreßbits wird jeweils nur ein Byte ausgewählt. Die verbleibenden Adreßleitungen werden nun dazu verwendet, um einen oder mehrere Speicherschaltkreise auszuwählen, damit nur eine Information, und zwar die richtige, bearbeitet werden kann. Die Auswahl des betreffenden Speicherschaltkreise erfolgt mit dem Adreßdekoder A23, der aus einem Bereich von 8 KByte für jeden einzelnen 1 KByte-Bereich eine Auswahlleitung bereitstellt. Mit dem Gatter A 24/25 wird dieser Bereich auf den oberen Adreßraum eingestellt. Dazu werden mit A25 die betreffenden Adreßleitungen mit dem Speicherauswahlsignal MREQ verknüpft und damit der Adreßdekoder freigegeben, d. h. konkret

MREQ ADR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	1	1	1														Diese Leitungen werden an alle Speicherschaltkreise gelegt
				0	0	0											----> /DK10 = E000H = RAM
				0	0	1											----> /DK11 = E400H
																	.
																	.
				0	1	1											----> /DK13 = EC00H = Bildwiederholungspeicher
				1	0	0											----> /DK14 = F000H = Mit Dioden D9
				1	0	1											----> /DK15 ODER verknüpft für 2K-Monitor

Die so gebildeten Leitungen zur Bausteinauswahl (chip-select, CS) werden an den CS-Eingang der Speicherschaltkreise geführt und geben diese frei.

Die Bildung der Auswahlsignale kann über das Signal MEMDI am Steckverbinder X1 von außerhalb verhindert werden. Das wird dann sinnvoll sein, wenn der MRB Z1013 als Bestandteil eines Mikrorechnersystems betrieben wird und in diesen Adreßbereichen bereits Speichereinheiten angeschlossen sind.

Bei der Verwendung der dynamischen 16 KByte Speicher U 256 bzw. K 565 RU3 oder K 565 RU6 (A33 bis A40) ergibt sich folgende Adreßauswertung:

```
ADR: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
      0 0 Diese Adreßbits werden intern zur Auswahl des
          Speicherplatzes ausgewertet
      Legt den Bereich ab Adresse 0000H fest.
```

Da diese Speicher pro Platz nur 1 Bit speichern, müssen hier 8 Schaltkreise parallel an den Datenbus angeschlossen werden. Der Anschluß ist außerdem komplizierter, weil diese Schaltkreise nur 6 Adreßeingänge haben. Die Übernahme der 14-stelligen Adresse erfolgt deshalb zeitlich gestaffelt.

Zuerst werden bei Auswahl dieses Speicherbereiches die sieben niederwertigen Adreßbits in den Speicherschaltkreis übernommen. Dazu wird das Signal RAS (ROW ADDRESS STROBE, Reihenadreßübernahmeimpuls) am Schaltkreis aktiviert. Anschließend werden die sieben höherwertigen Adreßbits auf die Schaltkreisanschlüsse geschaltet und mit dem Signal CAS (COLUMN ADDRESS STROBE, Spaltenadreßübernahmeimpuls) diese in die Schaltkreise eingetragen.

Die Erzeugung des Signals /RAS ist durch das Speicheranforderungssignal /MREQ gegeben. Das negierte Signal MREQ gibt ein Flip-Flop (A17) frei. Mit der nächsten steigenden Flanke des Systemtaktes wird das Flip-Flop umgeschaltet und steuert den Adreßleitungsumschalter A28/41 (Multiplexer) und gibt ein zweites Flip-Flop frei, das mit der absteigenden Flanke des Systemtaktes das Signal CAS erzeugt und, sofern ein Zugriff in diesen Adreßbereich (A14=A15=0) erfolgt, das Signal /CAS an den Schaltkreisanschlüssen aktiviert. War der Zugriff zum Speicher nicht in dem Bereich der dynamischen RAMs oder wurde das Signal /REFRESH als Zeichen der Speicherauffrischungszeit aktiv, so wird zwar das Signal /RAS gebildet, aber kein Signal /CAS. Damit werden in den dynamischen RAMs nur Auffrischungsaktivitäten ausgelöst.

In Verbindung mit den Signalen /RD oder /WR, die ebenfalls an die Speicherbausteine geführt werden, werden entweder Daten oder Befehle gelesen und auf den Datenbus geschaltet oder die auf dem Datenbus vorhandenen Daten im Speicher eingetragen.

Im Anhang ist ein Schema der Speicherverteilung innerhalb des gesamten Adreßraumes zu finden (Anlage 2).

3.3.2. Zusammenarbeit mit der CPU

Wenn die CPU auf den Speicher zugreifen möchte, sei es, um Befehle oder Daten zu holen oder um etwas abzuspeichern, ist das durch folgende Signale gekennzeichnet:

- /MREQ (Speicheranforderung) wird Low, d. h. aktiv und zeigt damit den Zugriff auf den Speicher an.
- A0 bis A15 (Adressen) geben den konkret adressierten Speicherplatz an.
- /WR (Schreiben) wird aktiv, wenn die CPU etwas in den Speicher schreiben möchte.
- /RD (Lesen) wird beim Lesen von Daten oder Befehlen aktiv.
- /M1 (Befehlsholezyklus) kennzeichnet in Verbindung mit /MREQ und /RD das Holen des Operationskodes eines Befehls (s. Kapitel 4)
- D0 bis D7 enthalten entweder die abzuspeichernde oder gelesene Information.

Die detaillierten Zeitabläufe können der Anlage 10 entnommen werden, wo die Taktdiagramme für den Speicher-Schreib- und Speicher-Lese-Zyklus angegeben werden.

3.4. Ein- und Ausgabebaugruppen

3.4.1. Parallel E/A-Baustein U 855 PIO

3.4.1.1. Beschreibung der Steuersignale

Aus der Bezeichnung des Bausteins geht eigentlich seine Verwendung bereits hervor. Er dient bevorzugt zur parallelen Ein- bzw. Ausgabe, d. h. zum Beispiel, daß alle acht Bit des Datenbusses gleichzeitig ausgegeben werden können.

Man kann natürlich auch Daten seriell, d. h. bitweise nacheinander aus- oder eingeben. Dazu ist aber ein gesondertes Programm notwendig.

Im MRB Z1013 kommt ein Baustein U 855 zum Einsatz. Ein Teil davon wird von den E/A-Baugruppen des Z1013 selbst genutzt (s. 3.4.2., 3.4.3.). Über den anderen Teil können Sie frei verfügen. Dazu müssen Sie allerdings die Anschluß- und Funktionsweise einer PIO kennen. Das soll Inhalt dieses Abschnittes sein.

Die Anschlußbelegung des U 855 finden Sie in der Anlage 9. Es ist zu erkennen, daß die PIO rechnerseitig an den Datenbus angeschlossen wird und prozesseitig zwei Kanäle A und B, auch Tore oder Ports genannt, besitzt.

Außerdem verfügt er über eine Reihe von Steuersignalen, deren Bedeutung hier kurz erläutert werden soll:

Es gelten die gleichen Vereinbarungen wie im Abschnitt 3.2.1.

- B/A SEL (E)
Liegt dieser Eingang auf "L", so wird das Tor A, liegt er auf "H", dann das Tor B freigegeben. Üblicherweise wird hieran die Adreßleitung der CPU A1 geführt.
- C/D SEL (E)
Der U 855 ist ein programmierbarer E/A-Baustein, d. h. es muß vor der eigentlichen Nutzung für den Datentransfer zwischen Rechner und Prozeß mitgeteilt werden, was er machen soll. Dazu gibt es eine Reihe von Steuerwörtern, die die Befehle der PIO darstellen. Diese Programmierung der PIO wird im allgemeinen als Initialisierung bezeichnet. Lesen Sie dazu den Abschnitt 3.4.1.2.
Erhält dieser Eingang "L"-Begel, so sind die auf dem Datenbus befindlichen Informationen Daten, bei "H"-Pegel Steuerwörter. Üblicherweise liegt C/D SEL an der Adreßleitung A0.
- /CS (E)
Hiermit wird die PIO für den Datentransfer freigegeben. Die Bildung dieses Bausteinauswahlsignals erfolgt analog zur CS-Dekodierung für die Speichereinheiten, nur daß für die E/A-Dekodierung die Adressen A0 bis A7 (Niederwertiger Teil des Adreßbusses) ausgewertet werden. Es können also maximal 256 (2 hoch 8) E/A-Tore angeschlossen werden.
- /IORQ (E)
Dient in Verbindung mit den anderen Signalen zur Kennzeichnung der E/A-Anforderung. Dieser Eingang wird direkt an den entsprechenden Ausgang der CPU gelegt.
- /M1 (E)
Mit aktiven M1 bei nicht aktiven RD und IORQ wird die PIO in einen definierten Anfangszustand zurückgesetzt. Geschieht dies nicht, arbeitet die PIO unkontrolliert. Anschließend muß die Initialisierung erfolgen.
Außerdem synchronisiert dieses Signal in Verbindung mit IORQ die Interruptbehandlung durch die CPU. Damit beide Funktionen gewährleistet werden können, muß dieses M1 aktiv bei aktivem RESET der CPU oder bei Aussendung des CPU-M1 sein. Diese ODER-Verknüpfung wird durch die Bildung des /PM1 realisiert, welches an das PIO-M1 angeschlossen wird.
- RD (E)
Wird die Datenbusinformation in den PIO geschrieben, muß RD inaktiv sein. Ist RD auf "L", legt die PIO die vom Prozeß gelesenen Daten, entsprechend den mit B/A SEL ausgewählten Tor, auf den Datenbus.

Mit dem im vorigen Abschnitt zu den C/D-SEL- und B/A-SEL-Signalen Gesagten ergibt sich folgende Adreßverteilung:

```

ADR: 7  6  5  4  3  2  1  0
                                C/D SEL
                                0, wenn Information Daten
                                1, wenn Information Steuerworte
                                B/A SEL
                                0 , wenn Tor A
                                1 , wenn Tor B
beliebig, z.B.                  0 0 0 0 0 0 ==> IOSEL0, PIO
                                0 1 0 ==> IOSEL2, Tastaturspaltentreiber

```

Damit ergeben sich die Adressen:

```

Tor A (Anwenderport) - Daten:      00H
                    - Steuerwort:  01H
Tor B (Systemport)  - Daten:      02H
                    - Steuerwort:  03H

```

Im Z1013 sind diese Adressen nicht eindeutig, da die Adreßbits A7, A6, A5 auch 111 sein könnten. Da diese nicht ausgewertet werden, spielt das aber keine Rolle.

Die Arbeitsweise der PIO wird durch die Steuerworte festgelegt, die im folgenden erläutert werden sollen.

1. Betriebsartenauswahl

```

Bit:          7   6   5   4   3   2   1   0
Belegung:
              beliebig
              0 0 Betriebsart   0 :   Byteausgabe
              0 1              1 :   Byteeingabe
              1 0              2 :   Byteein/ausgabe
              1 1              3 :   Bitein/ausgabe

```

Betriebsart 0:

Die durch die CPU bereitgestellten Daten werden während des durch sie veranlaßten Ausgabezyklus in das angesprochene Ausgaberegister geschrieben. Mit RDY zeigt die PIO dem Prozeß an, daß Daten zur Übernahme in PIO bereitstehen. Dieses RDY wertet das periphere Gerät aus, übernimmt daraufhin die Daten und teilt mit STB der PIO die Datenübernahme mit. Anschließend löst die PIO ein Interrupt aus, um der CPU das Ende der Datenausgabe zu melden.

Betriebsart 1:

Die PIO teilt mit H-Pegel an RDY dem externen Gerät die Bereitschaft zur Datenübernahme mit. Mit STB=L schreibt das Gerät die Daten in das entsprechende Eingaberegister. RDY=L sperrt eine weitere Eingabe bis die Daten durch eine Interruptbehandlung von der CPU übernommen werden.

Betriebsart 2:

Diese bidirektionale Betriebsart ist nur mit dem Kanal A möglich. Mit Kanal B ist dann nur noch Betriebsart 3 möglich, da für die Abwicklung des Datentransfers alle vier Quittungssignale ARDY, ASTB, BRDY und BSTB benötigt werden. ARDY und ASTB steuern die Ausgabe, die beiden anderen die Eingabe.

Betriebsart 3:

In dieser Betriebsart kann innerhalb eines Tores jedem Bit eine beliebige Datenflußrichtung zugeordnet werden. Auf diese Weise können Stellsignale und Statusmeldungen für Prozeß-Steuerungen aus- bzw. eingegeben werden.

2. Ein-/Ausgabe Maskenwort

Soll die Bitstelle eine Eingabeleitung sein, muß an dieser Stelle eine 1 stehen, bei Ausgabe eine 0. Da dieses Steuerwort kein eigenes Kennzeichen besitzt, muß es unmittelbar auf das Betriebsauswahlsteuerwort folgen. Ist in dieser Betriebsart 3 festgelegt worden, liest die PIO das nächste Byte immer als E/A-Maskenwort.

3. Interruptvektor

Bit: 7 6 5 4 3 2 1 0
Belegung: 0 Kennzeichen
niederwertiger Teil des Interruptvektors
(s. 4.4.)

4. Interruptsteuerwort

Bit: 7 6 5 4 3 2 1 0
Belegung: 0 1 1 1 Kennzeichen
0 , nächstes Steuerwort ist kein Maskenwort
1 , nächstes Steuerwort wird als Maskenwort erkannt
0 , Interrupt bei H -> L Flanke
1 , Interrupt bei L -> H Flanke
0 , die im folgenden Steuerwort festgelegten Interrupt auslösenden Bits sind ODER-verknüpft, d. h. eine dieser Leitungen kann bereits Interrupt auslösen
1 , UND verknüpft, d. h. alle festgelegten Stellen müssen gleichzeitig die mit Bit 5 festgelegte Interruptbedingung erfüllen
0 , Interrupt freigeben
1 , Interrupt gesperrt

5. Interruptmaskenwort

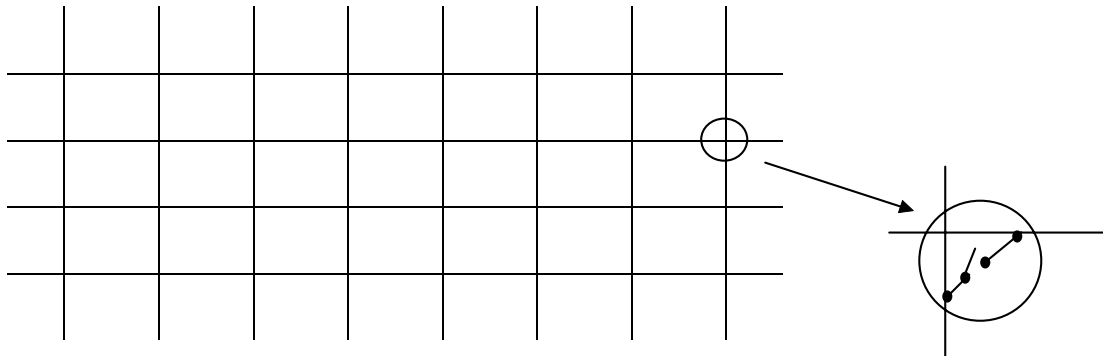
Die Bitstelle der Eingabeleitungen, die Interrupt auslösen sollen, werden durch eine 0 gekennzeichnet, die keinen Interrupt auslösen sollen, durch eine 1.

6. Interrupt Ein/Aus

Bit:	7	6	5	4	3	2	1	0	
Belegung:		beliebig			0	0	1	1	Kennzeichen
		0			,Interrupt gesperrt				
		1			,Interrupt freigegeben				

3.4.2. Tastaturanschluß

Elektrisch stellt die Tastatur nichts anderes als eine Matrix von Schaltern in folgender Anordnung dar:



Die Zeilen dieser Anordnung sind mit den Widerständen R11 bis R14 auf "H"-Pegel gelegt. Diese Leitungen sind mit dem Tor B, Bit 0 bis 3, des PIO verbunden, welche für Eingabe programmiert sind. Wird keine Taste gedrückt, liest die PIO auf allen vier Leitungen eine 1.

Die acht Spaltenleitungen der Tastatur sind an ein separates Ausgabator, das durch die Bausteine A47 (Speicher für Spaltennummer) und A46 (1 aus 8 Spaltenleitungen) gebildet wird, angeschlossen. Die Adresse dieses Tores ist 08H. Die Spaltennummer steht im niederwertigen Halbbyte des Datenbusses binär verschlüsselt. Bei einer Ausgabe werden diese vier Bits entschlüsselt und legen so eine Spalte auf "L"-Potential. Wird in dieser aktivierten Spalte nun eine Taste betätigt, wird der L-Pegel auf die entsprechende Zeilenleitung durchgereicht. Der Rechner liest jetzt eine 0 in der entsprechenden Bitstelle.

Aus der ausgegebenen Spaltennummer und der eingelesenen Zeilennummer ermittelt das Tastaturbedienprogramm des Betriebssystems den rechnerinternen Code der gerade betätigten Taste. Der Z1013 benutzt den sogenannten ASCII-Code (s. Anlage 7).

3.4.3. Magnetbandanschluß

Von der auf der Leiterplatte installierten PIO wird eine Bitleitung (PB 7) zur Ausgabe eines seriellen Datenstromes genutzt. Die erforderliche Parallel/Serienwandlung wird softwaremäßig realisiert. Das ausgegebene Signal wird über einen Spannungsteiler R27/28 zur Pegelanpassung abgeschwächt; mit einem Kondensator C1.9 werden die Flanken verrundet, damit ein etwa sinusförmiges Signal in Magnetbandgerät aufgezeichnet werden kann.

Das Ausgangssignal eines Magnetbandgerätes wird gleichspannungsfrei einem Operationsverstärker A48 zugeführt. Das auf TTL-Pegel verstärkte Signal wird an einen Anschluß der PIO (PB 6) geleitet. Durch entsprechende Software wird dieser Anschluß ständig abgefragt und aus dem ankommenden seriellen Datenstrom durch Serien/Parallelwandlung die ursprüngliche Information wieder zurückgewonnen.

3.4.4. Bildschirmsteuerung

Die Bildschirmsteuerung wandelt die vom Rechner auszugebende Information in ein CCIR-kompatibles Fernsehsignal, indem sie zusätzlich die notwendigen Synchron- und Dunkeltastimpulse erzeugt. Um diesen Vorgang prinzipiell zu verstehen, sind einige Bemerkungen über den Aufbau des Fernsehsignals notwendig.

Beim Schreiben eines Fernsehbildes läuft ein Elektronenstrahl, auf den die Bildinformation aufmoduliert wurde, über einen fluoreszierenden Schirm. Für eine Zeile benötigt er eine Zeit von 64 μ s. Das entspricht einer Zeilenfrequenz von 15,625 kHz. Ein Zeilensynchronimpuls veranlaßt den Strahlrücklauf, wobei der Strahl dunkelgesteuert wird. Um ein Flimmern der Anzeige zu vermeiden, muß das ganze Bild mit einer Frequenz von mindestens 25 Hz wechseln.

Da beim Fernsehen in dieser Zeit zwei Halbbilder geschrieben werden, im Z1013 aber ein Bild zweimal, ergibt sich hier eine Bildwechselfrequenz von 50 Hz.

Ein sogenannter Bildsynchronimpuls löst dann jeweils einen Strahlrücklauf zum oberen Bildrand aus. Die Bildschirmsteuerung des MRB Z1013 arbeitet nach folgendem Prinzip:

Die gesamte Erzeugung des fernsehgerechten Signals, des sogenannten BAS-Signals, wird durch die Zählkaskade ohne Mitarbeit der CPU gesteuert. Die Kaskade A3, A4, A5 und A12 wird mit dem 8 MHz-Takt des Taktgenerators gespeist. Eine Teilung durch 2 hoch 9 liefert z. B. die Zeilenfrequenz.

Aus dem Bildaufbau wissen wir bereits, daß eine Zeile aus 32 (=2 hoch 5) Zeichen besteht. Um diese abzuzählen, werden die 5 niederwertigen Adressen des BildwiederholSpeichers (BWS) A30/31 genutzt. Die höherwertigen Adreßeingänge zählen die Zeichenzeilen eines Bildes. Da die Zählkaskade immer zyklisch durchzählt, wird auch der BWS zyklisch ausgelesen.

Das aus dem BWS gelesene Byte, das den ASCII-Kode entsprechend Anlage 7 des darzustellenden Zeichens enthält, steht als höherwertiger Adreßteil am Zeichengenerator A44. Mit den drei Ausgängen des Linien pro Zeichenzählers, die an die niederwertigen Adreßeingänge von A44 gehen, werden nacheinander die Bildpunktzeilen an den nachfolgenden Parallel/Serien-Wandler A21/22 übergeben. Hier wird das übernommene Bitmuster mit dem 8 MHz-Takt seriell herausgeschoben. Dieser seriell Datenstrom

bildet die Bildinformation des Bild-, Austast- und Synchronsignals (BAS-Signal).

Mit den Gattern der Schaltkreise A9, A10, A13 und A20 werden aus dem Zählfolgen entsprechend der Fernsehnorm die Synchronimpulse dekodiert.

Außerdem wird durch diese Schaltung gesichert, daß für der Strahlrücklauf das Signal dunkelgesteuert wird, da dieser sonst auf dem Bildschirm sichtbar wäre. Diese Impulse werden mit der Bildinformation gemischt und ergeben so das BAS-Signal.

In einem HF-Modulator wird das BAS-Signal auf eine HF-Trägerfrequenz, die auf den Fernsehkanal 3 abgestimmt ist, aufmoduliert. Der Ausgang dieses Modulators kann nun direkt mit dem Antenneneingang des Fernsehgerätes verbunden werden.

Wie gelangen aber nun in diese selbständig arbeitende Einheit die darzustellenden Daten? Über die Adreßmultiplexer (A29, A42, A18) kann die CPU einen Platz im BWS adressieren. Dazu wird mit einem Speicherbereichauswahlsignal der Multiplexer umgeschaltet. Über den Datentreiber A43 kann die CPU den BWS beschreiben oder lesen.

Damit ist auch deutlich gemacht, daß der BWS wie ein normaler Speicher behandelt werden kann. Die Anfangsadresse ergibt sich analog zu dem ROM-Auswahlsignal zu EC00H. Welche Position die einzelnen Speicherplätze auf dem Bildschirm einnehmen, ist in der Anlage 8 schematisch dargestellt.

3.5. Stromversorgung

Für den Betrieb des MRB Z1013 sind drei verschiedene Versorgungsspannungen nötig.

Zur Versorgung aller Logikschaltkreise wird eine Spannung von +5 V, die im folgenden mit 5P bezeichnet wird und etwa mit 1 A belastbar ist, verwendet. Die beiden anderen Spannungen von +12 (12P) und -5 V (5N) werden für die Speichereinheiten sowie einige Spezialfälle benötigt. Sie werden nicht so stark belastet.

Um diese Spannungen zu erzeugen, besitzt der MRB Z1013 ein eigenes Netzteil. Eine zugeführte Wechselspannung von ca. 12 V wird mittels Dioden in Einweggleichrichtung gleichgerichtet. An den Ladekondensatoren C2.1, C3.1 und C5.1 sind jeweilige Rohspannungen verfügbar. Eine Ausnahme bildet die Erzeugung der Rohspannung für die 12P. Hier wird mit einer Spannungsverdopplerschaltung gearbeitet.

Die Erzeugung der 5P wird mit einem integrierten Festspannungsregler A2 vorgenommen, der auf einem Chip alle benötigten Bauteile enthält und kaum eine Außenbeschaltung benötigt. Lediglich ein Kondensator am Ausgang ist erforderlich. Da eine

starke Belastung dieses Bauelementes erfolgt, wird eine angemessene Kühlung benötigt.

Die Spannung 5N wird mittels einer Z-Diode D4 stabilisiert. Diese einfache Widerstands/Z-Dioden-Kombination ist bei dem geringen Leistungsbedarf ausreichend.

Um die Spannung 12P zu erzeugen, wird eine verdoppelte und anschließend mit einer Widerstands/Z-Dioden-Kombination stabilisierte Spannung der Basis eines Transistors V2 zugeführt. Dadurch ist am Emitter dieses Transistors eine stabilisierte Spannung verfügbar, die stärker belastet werden kann.

3.6. Bussystem

Die wichtigsten Signale des Mikrorechners Z1013 sind an den Rand der Leiterplatte geführt und dort für den Anschluß von Steckverbindern vorbereitet. Dabei haben diese Anschlüsse folgende Bedeutung:

X1: Systembus (Steckverbinder: StL 304-58 TGL 29331/03)
Enthält alle Signale des Systembusses und ist elektrisch kompatibel zum K1520-Systembus. (Anlage 6)

X2: Prüfkamm und Tastaturanschlußpunkte (hier wird entsprechend den Hinweisen von Pkt.1.2.4.1. und 1.4. der Bedienungsanleitung das Tastaturbandkabel oder die Buchsenleiste BuL 202-26 TGL 29331/04 angelötet)

X3: Wechsellspannungszuführung (Flachsteckverbinder)

X4: PIO Kanal A (Steckverbinder: BuL 402-15 TGL 29331/04)
Hier werden die Anschlüsse des Kanals A der PIO herausgeführt. Außer den Steuerleitungen ARDY und /ASTB des Kanals A wurden auch die des Kanals B (BRDY und /BSTB) auf den Steckverbinder gelegt, um die Betriebsart bidirektionale E/A realisieren zu können.

X5: Anschluß Magnetbandgerät (Diodenbuchse)

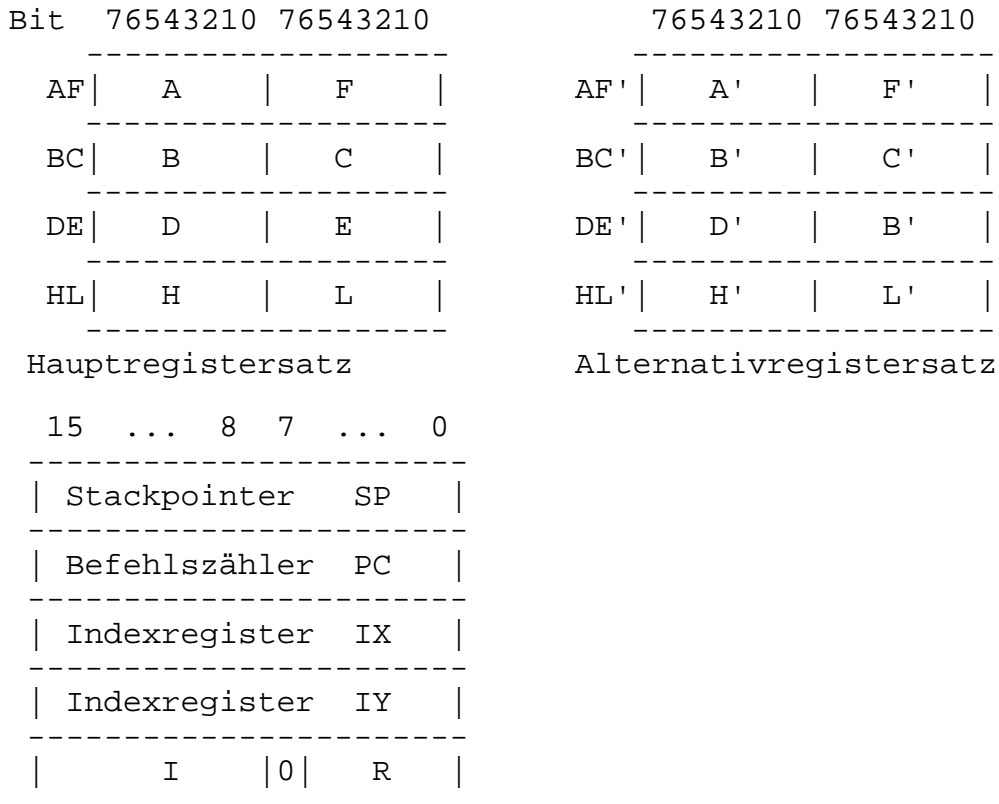
X6: HP-Ausgang des Modulators (Koaxialbuchse)

Die genaue Zuordnung der einzelnen Signale zu den jeweiligen Anschlüssen ist der Anlage 6 zu entnehmen.

4. Der Befehlssatz des Mikroprozessors U880

Dieser Abschnitt soll das Verständnis der Arbeitsweise und der Programmierung des Mikrorechnerbausatzes Z1013 erleichtern. Anhand von Beispielen erfolgt eine Erläuterung der verschiedenen Mikroprozessor-Befehle und deren Wirkungsweise und Anwendungsmöglichkeiten. Der folgende Überblick soll prinzipielle Eigenschaften und Besonderheiten des Mikroprozessors U 880 aufzeigen:

- 64 K Byte Adreßraum für Speicher
- 256 Ein-/Ausgabekanäle
- 3 Doppelregister mit Alternativregistersatz
- 2 Indexregister mit je 16 Bit Breite
- 1 Refreshregister (ermöglicht das automatische Auffrischen externer dynamischer RAM-Speicher)
- 1 maskierbarer, 1 nichtmaskierbarer Interrupt
- Architektur des Mikroprozessors U 880:



Speicheradressen:

0000H		maximal 64 KByte =65535 Speicher- plätze
0001H		
.		
.		
.		
0FFFFEH		
0FFFFFH		

Flagregister F:

S Z X H X P/V N C
S Signum (Vorzeichen)
Z Zero
H Half-Carry
P/V Parity/Overflow (Parität, Überlauf)
C Carry
X nicht verwendet

Kanaladressen:

0...255 256 Eingänge

0...255 256 Ausgänge

Befehlsvorrat:

158 Grundbefehle

4.1. Befehlsschlüssel

Der Mikroprozessor erhält seine Befehle vom Speicher über den 8 Bit-Datenbus binär verschlüsselt zugeführt. Für den Programmierer ist diese Darstellung im Binärcode meistens zu detailliert und erschwert die Programmierung; es werden deshalb Hexadezimalcodes mit entsprechend zugehöriger Mnemonik des Maschinenbefehls verwendet. Als Mnemonik bezeichnet man Pseudonamen der Befehle. Diese Pseudonamen bzw. Abkürzungen geben gleichzeitig Auskunft über die Funktion der Befehle und erleichtern damit die Programmierung. Die Übersetzung in den Maschinencode erfolgt dann anhand der Tabelle der Befehlsliste oder durch ein Programm, genannt Assembler.

Bemerkung: Bei hexadezimaler Verschlüsselung muß der führenden Ziffer, falls diese ein Buchstabenzeichen ist, eine Null vorangestellt werden. Dadurch wird eine Verwechslung mit Bezeichnern vermieden. Bei der Programmierung des U 880 werden die Maschinenbefehle im allgemeinen hexadezimal verschlüsselt, wobei das 'H' und die führende '0' vor Buchstabenzeichen innerhalb des Maschinenkodes weggelassen werden.

z.B. 00 = NOP
40 = LD C,H
FF = RST 38H

Wie bereits erwähnt, können Maschinenbefehle aus einem oder mehreren Bytes bestehen. Man spricht dann von 1-Byte-, 2-Byte-Befehlen und so weiter. Jedem Byte ist in Abhängigkeit seines Platzes im Befehl und seiner Kodierung eine bestimmte Bedeutung zugeordnet worden.

4.1.1. 1-Byte-Befehle

Diese Befehle bestehen nur aus dem Operationskode (im weiteren als OPC bezeichnet). Da es sich fast ausschließlich um Registeroperationen, also Operationen innerhalb der Prozessorregister handelt, ist in diesem Byte auch die notwendige Registeradresse enthalten. Bei der Behandlung der Adressierung wird noch einmal näher darauf eingegangen.

1. Byte
OPC

Beispiel:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	47	LD B,A	;Lade B mit A
1001	23	INC HL	;HL:=HL+1
1002	40	LD B,B	
1003	81	ADD C	

Die Form des Beispiels soll als Normativ eines Programmprotokolls dienen. Es empfiehlt sich, diese übersichtliche Darstellung bei der Erstellung von Programmen zu nutzen, da sich in dieser tabellenartigen Zusammenstellung alle Angaben widerspiegeln, die zu einem Programm gehören. Die letzte Spalte bietet die Möglichkeit, einen Kommentar unterzubringen, um noch nach längerer Zeit den Inhalt eines Programms nachvollziehen zu können. Kommentarzeilen werden mit einem Semikolon gekennzeichnet. Der Inhalt sollte kurz, aber eindeutig sein.

Wird nur ein Byte als Operationskode verwendet, würden sich 256 Möglichkeiten ergeben. Man nutzt davon aber nur 252. Die verbleibenden vier Kombinationen sind für folgende Aufgaben reserviert worden:

Beim U 880 werden zwecks Erweiterung und Ausbau des Befehlsumfangs vier Hexadezimalcodes (0CBH, 0DDH, 0EDH, 0FDH) der Befehlsliste als sogenannte "Signalbytes" festgelegt. Diese Signalbytes stehen grundsätzlich an erster Stelle des Befehls (1.Byte). Sie kennzeichnen aber keinen konkreten Befehl, sondern kündigen eine spezielle Gruppe von Befehlen an. Die Konkretisierung des Befehle erfolgt durch zusätzlich ein oder mehrere Bytes. Auf diese Art ist es möglich, durch ein Signalbyte weitere 256 Befehle zu kennzeichnen und somit den Befehlsumfang stark zu erweitern. Der U 880 bietet folgende Erweiterungen des Befehlsschlüssels mit den Signalbytes:

Signalbyte CB: Bitmanipulationen, Verschiebebefehle

Signalbyte DD: Umschaltung von HL nach IX

Signalbyte ED: Blocktransport- und Suchbefehle

Signalbyte FD: Umschaltung von HL nach IY

4.1.2. 2-Byte-Befehle

Diese Befehle können jetzt einen zweiten OPC, einen Direktwert oder eine Sprungweite (s. auch Sprungbefehle) im 2. Byte des Befehle enthalten.

1. Byte	2. Byte	
OPC	OPC	
OPC	n	n=Direktwert
OPC	c	c=Sprungweite

Ist das 2. Byte ein OPC, so stellt das 1. Byte das Signalbyte dar.

Beispiele:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	DD 23	INC IX	;IX:=IX+1
1002	FD 23	INC IY	;IY:=IY+1
1004	CB 40	BIT 0,B	
1006	3E 10	LD A,10H	;Lade A mit 16
1008	36 FF	LD(HL),0FFH	;Lade den von HL adres- ;sierten Speicherplatz ;mit 255
100A	28 04	JRZ 06	;Springe, wenn Z=1 ist, ;um 6 Byte nach vorn, ;PC: PC +6 -2

Bei der Berechnung der Sprungweite wird die aktuelle Position des Befehlszählers (PC), der ja bereits auf den nächsten Befehl zeigt, durch Subtraktion einer 2 berücksichtigt. Im Quellkode beziehen sich die Sprungweiten immer auf den Befehlsanfang, also den PC-Stand, der zum betreffenden Befehl gehört. Es empfiehlt sich, bei der Programmerstellung im Quellkode symbolische Sprungmarken zu verwenden, denen bei der Übersetzung in Maschinenkode ein konkreter Wert zugewiesen wird. Wie das gemacht wird, zeigen spätere Beispiele.

4.1.3. 3-Byte-Befehle

Diese Befehle enthalten einen Operationskode und einen 16-Bit-Direktwert (nn). Dieser Direktwert stellt einen normalen Datenwert oder eine Adresse dar, wie er z.B. in Lade- oder Sprungbefehlen benötigt wird. Ebenfalls ist eine Kombination von Signalbyte und OPC sowie von Signalbyte und 8-Bit-Direktwert (n) für einige Befehle möglich.

1. Byte	2. Byte	3. Byte	
OPC	N(nn)	H(nn)	(bei Ladebefehlen)
OPC	N(nn)	H(nn)	(Sprungadresse)
OPC	OPC	n	(wobei der 1. OPC ein Signalbyte DD oder FD sein kann)

Die Angabe "N(nn)" bezeichnet den niederwertigen Teil, d.h. die letzten 8 Bits des 16-Bit-Direktwertes, "H(nn)" demgemäß den höherwertigen Teil, d.h. die ersten 8 Bit.

Wert:		H		N	
		15.....8		7.....1	

Beispiele:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	21 00 F7	LD HL,0F700H	;Lade HL mit 0F70CH
1003	C3 2D F0	JMP 0F02DH	;Springe zur Adresse ;0F02DH
1006	CD B7 E1	CALL 0F1B7H	;Sprung ins Unterprogr.
1009	DD 46 3F	LD B,(IX+3FH)	
100C	FD 72 00	LD (IY+0),D	

4.1.4. 4-Byte-Befehle

In den 4-Byte-Befehlen sind im wesentlichen Kombinationen bereits bekannter Befehle mit einem oder zwei Signalbyte enthalten. Eine Übersicht über diese Befehle ist der Anlage zu entnehmen. Zwei Beispiele sollen hier genügen.

Beispiele:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	DD 21 00 00	LD IX,0	;Lade IX mit 0
1004	DD CB 0F 1E	RR (IX+0FH)	

Die im Anhang enthaltene Befehlsliste für den Mikroprozessor U 880 enthält übersichtlich alle Befehlsschlüssel hexadezimal kodiert mit Hinweisen für die Verwendung des Signalbytes bzw. der Verschiebung 'd' (siehe auch 'Indexierte Adressierung'). Diese Befehlsliste dient bei der manuellen Programmierung im Maschinenkode als rationelles Hilfsmittel.

4.2. Adressierung

4.2.1. Registeradressierung

Die Angabe von Registeroperanden (also von Registern des Notizblockspeichers im U 880) erfolgt implizit im Operationsbyte durch sogenannte Kurzadressen. Für die Adressierung von maximal acht allgemeinen Registern sind im OPC-Byte zweimal 3-Bit-Stellen (sowohl für das Quellregister als auch für das Zielregister) notwendig. Bei der Angabe der Kurzadressen für U880-Register gilt also:

Register	A	B	C	D	E	H	L	(HL)
Kurzadresse	7	0	1	2	3	4	5	6
(binär)	111	000	001	010	011	100	101	110
Doppelregister		BC	DE	HL	AF	oder	SP	

(binär) 00 01 10 11

Die Angabe von Operanden, die sich im Hauptspeicher befinden, kann auf verschiedene Weise erfolgen. Der Zugriff auf eine bestimmte Speicherstelle (beim U 880 ist eine 16-Bit-Adresse erforderlich) erfolgt durch Bereit stellen der entsprechenden Adresse zu dem Zeitpunkt, wo der Maschinenbefehl diese benötigt.

4.2.2. Direktwertadressierung

Der Zugriff zum Speicher erfolgt mittels der im Befehl komplett angegebenen Speicheradressen. Diese steht im mnemonischen Befehl immer in Klammern.

z. B.: Der unbekannte 8-Bit-Wert K soll mittels direkter Adressierung vom Speicherplatz mit der Adresse 3000H in das A-Register geholt werden. Danach soll diese Konstante auf den Speicherplatz mit der Adresse 3010H gebracht werden.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	3A 00 30	LD A,(3000H)	;A:=(3000)
1003	32 10 30	LD (3010H),A	;(3010H):=A=K

4.2.3. Registerindirektadressierungng

Bei der indirekten Adressierung erfolgt der Speicherzugriff durch Angabe eines Doppelregisters im Maschinenbefehl, wobei im Doppelregister die Adresse für den Speicherzugriff geladen sein muß (Registerpaare HL, BC, DE sind möglich).

z. B.: Der unbekannte 8-Bit-Wert K soll wie im oben genannten Beispiel mittels indirekter Adressierung von Speicherplatz 3000H in den Speicherplatz 3010H umgespeichert werden. Zunächst lädt man die Adresse des Speicherplatzes, von der der Wert K geholt werden soll, in das Registerpaar BC und die Zieladresse in das Registerpaar HL.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	01 00 30	LD BC,3000H	;Quelladresse
1003	21 10 30	LD HL,3010H	;Zieladresse
1006	0A	LD A,(BC)	;A:=(3000H)
1007	77	LD (HL),A	;(3010H):=A=K

Anmerkung: Zur indirekten Adressierung wird gern das Doppelregister HL zur Adressenbereitstellung eingesetzt; man schreibt auch:

(HL)= M (englisch: Memory = Gedächtnis, Speicher)

Dann würde die letzte Zeile des obigen Beispiels lauten:

```
1007      77          LD M,A          ;= LD (HL),A
```

4.2.4. Indexierte Adressierung

Soll der Speicherzugriff mittels indexierter Adressierung erfolgen, so wird im Maschinenbefehl ein sogenanntes Indexregister (IX, IY) und eine Verschiebung 'd' als vorzeichenbehaftete Konstante angegeben. Die Adressenbildung für den Speicherzugriff erfolgt durch Summieren der im jeweiligen Indexregister enthaltenen Grundadresse und der Verschiebung 'd'. Die Verschiebung 'd' kann Werte von -128 bis +127 annehmen. Bei Verwendung einer negativen Verschiebung wird diese von der Grundadresse subtrahiert.

z. B.: Der 8-Bit-Wert K soll wie im oben genannten Beispiel mittels indexierter Adressierung umgeladen werden. Ins Indexregister IX wird die Adresse 3000H geladen.

Befehlszähler	Maschinenkode	Quellkode	Kommentar
1000	DD 21 00 30	LD IX,3000H	
1004	DD 7E 00	LD A,(IX+0)	;A:=(3000H),d=0
1007	DD 77 10	LD (IX+16),A	;(3010H):=A, ;da IX=3000H und d=10h=16

Die indexierte Adressierung wird verwendet, um einen einfachen Zugriff zu in Tabellenform gespeicherten Daten zu erhalten. Dazu wird der Anfangspunkt der Tabelle in das Indexregister geladen, die Verschiebung 'd' entspricht dann der konkreten Tabellenposition.

4.3. Maschinenbefehle und ihre Bedeutungen

Im folgenden Abschnitt werden die entsprechenden Befehlsgruppen näher erläutert und an Beispielen die Funktionsweise untermauert.

4.3.1. Ladebefehle

Bei den Ladebefehlen werden prinzipiell die Byte- und Doppelbyte-Ladebefehle unterschieden. Die Datenbewegung erfolgt stets nur zwischen Speicher und Prozessor bzw. innerhalb des Prozessors zwischen den Registern. Die Flags werden nicht

beeinflusst (Ausnahmen bilden nur die Befehle LD A,I und LD A,R).

Allgemeiner Aufbau der Ladebefehle:

```
[MARKE:] LD Ziel , Quelle [;KOMMENTAR]
```

Die eingeklammerten Angaben sind wahlfrei, sie können vorhanden sein, müssen aber nicht.

Beispiele für Byte-Ladebefehle:

(d.h. die transportierten Daten umfassen ein Byte)

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	3E 3A	LD A,3AH	;A:=3AH
1002	57	LD D,A	;D:=3AH
1003	46	LD B,(HL)	
1004	1A	LD A,(DE)	
1005	02	LD (BC),A	
1006	DD 70 00	LD (IX+0),B	;B wird auf ;IX+d adressierten Speicherplatz gela- ;den
1009	DD 4E 7F	LD C,(IX+127)	
100C	ED 57	LD A,I	;das Interruptregister ;I wird in A geladen, Stand ;von IFF 2 in das Flag P/V
100E	ED 5F	LD A,R	;das Refreshregister ;wird in das A-Register geladen

Beispiele für Doppelbyte-Ladebefehle (d. h. die transportierten Daten umfassen 2 Byte):

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	01 34 12	LD BC,1234H	;B:=12H, C:=34H
1003	2A 34 12	LD HL,(1234H)	
1006	ED 43 34 12	LD (1234H),BC	
100k	DD F9	LD SP,IX	

Es ist verständlich, daß Doppelbyte-Ladebefehle auf Grund des 8-Bit-Batenbusses nur byteweise abgearbeitet werden, wobei zunächst das niederwertige und danach das höherwertige Byte geladen wird.

Einen Sonderfall der Doppelbyte-Ladebefehle bilden die sogenannten Kelleroperationen, diese werden aber später beschrieben.

4.3.2. Byte- und Doppelbyte-Zähl-Befehle

Diese Befehlsgruppe dient dem Erniedrigen bzw. Erhöhen von Registerinhalten oder Speicherinhalten um jeweils den Wert 1.

Beispiele für Zählbefehle:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	3C	INC A	;A:=A+1
1001	34	INC (HL)	;(HL):=(HL)+1
1002	03	INC BC	;BC:=BC+1
1003	DD 23	INC IX	
1005	0D	DEC C	;C:=C-1
1006	DD 35 00	DEC (IX+0)	;(IX+0):=(IX+0)-1
1009	DD 2B	DEC IX	;IX:=IX-1

Die Byte-Zählbefehle beeinflussen das Z-Flag. Ist das Resultat des Befehls im behandelten Byte identisch 0, so wird das Z-Flag auf "1" gesetzt, sonst bleibt es "0". Die Doppelbytezählbefehle beeinflussen keine Flags.

Beispiel: Laden von 3 Speicherstellen mit 00,01,02 ab Adresse 3000H

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	3E 03	LD A,3	;Anzahl Speicherpl.
1002	21 00 30	LD HL,3000H	;1. Adresse laden
1005	75	M1: LD (HL),L	;Speicherpl. laden
1006	23	INC HL	;Adresse und Wert ;um 1 erhöhen
1007	BD	CMP L	;Vergleich L mit A
1008	20 FB	JRNZ M1-#	;Rücksprung zu M1, ;wenn L noch nicht 3

4.3.3. Arithmetische Befehle

Beim U 880 ist nur die Addition und die Subtraktion von Bytes und Doppelbytes (16-Bit-Worte) möglich.

Beispiele für arithmetische Befehle:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	80	ADD B	;A:=A+B
1001	66 3E	ADD 3EH	;A:=A+3EH
1003	86	ADD (HL)	;A:=A+(HL)
1004	88	ADC B	;A:=A+B+CY
1005	CE 0F	ADC 0FH	;A:=A+0FH+CY
1007	92	SUB B	;A:=A-D
1008	DD 9E 04	SBC (IX+4)	;A:=A-(IX+4)-CY

Die arithmetischen Operationen laufen prinzipiell in folgender Form ab:

A := A + Operand s

Hierbei ist die Konstruktion "==" als "ergibt sich aus" zu interpretieren: A ergibt sich aus A plus Operand s. Diese Befehlsgruppe beeinflusst das Flagregister vollständig. Die Wirkungsweise der Flag-Bits lässt sich günstig bei Zahlenbereichsüberschreitungen nach Ausführung der Operationen zeigen:

Beispiel:

```

A:= -128
ADD B      ;(B=127,-128,-127)

B = 127   B = -128   B = -129

A alt    = 1000 0000   1000 0000   1000 0000
+s       = 0111 1111   1000 0000   1000 0001
-----
A alt+s  = 1111 1111  10000 0000  10000 0001

CY-Flag
neu      = 0          1          1
Z neu    = 0          1          0
-----
A neu    = 1111 1111   0000 0000   0000 0001

```

Beim Überlauf wird das CY-Flag=1, d. h. das Ergebnis der Addition ist falsch.

Für die Doppelbytearithmetik gilt mit gewissen Besonderheiten das oben genannte.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	09	ADD HL,BC	;HL:=HL+BC
1001	ED 7A	ADC HL,SP	;HL:=HL+SP+CY
1003	19	SBC HL,DE	
1004	DD 09	ADD IX,BC	

Falls bei der ADD-Operation ein Überlauf entsteht, wird das Carry-Flag gesetzt (auf "1"). Bei ADC und SBC wird das Flagregister vollständig neu bestimmt. Anwendungsgebiet dieser Befehlsgruppe ist insbesondere die sogenannte Adreßarithmetik, d.h. wenn mit Adressen gerechnet wird.

4.3.4. Vergleichsbefehle

Einen Sonderfall der Arithmetik-Befehle bilden die Vergleichsbefehle. Es wird eine Subtraktion des A-Registerinhaltes mit dem jeweiligen Operanden ausgeführt, allerdings werden im Ergebnis der Operation nur die Flags geändert, der A-Registerinhalt bleibt unverändert.

Beispiele für Vergleichsbefehle:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	B8	CMP B	;A-B=?
1001	FE 38	CMP 38H	;A-38H=?
1003	FD BE 05	CMP (IX+5)	;A-(IX+5)=?

Die Flagbeeinflussung funktioniert nach folgender Tabelle:

	Z	CY
ist A>s , dann ist A-s>0 :	0	0
ist A=s , dann ist A-s=0 :	1	0
ist A<s , dann ist A-s<0 :	0	1

(s ist der entsprechende Operand)

In der Praxis schließen sich i.a. an Vergleichsbefehle entsprechende Verzweigungsoperationen oder bedingte Unterprogrammaufrufe in Abhängigkeit vom Z- oder/und vom CY-Flag an (siehe unter 'Bedingte Sprungbefehle').

4.3.5. Logische Befehle

Die logischen Befehle des U 880 umfassen das UND, das ODER und das EXKLUSIV-ODER. Die Wirkungsweise wurde im Punkt 2.4 ausführlich erläutert. Die logische Verknüpfung erfolgt stets mit dem A-Register und kann mit B, C, D, E, H, L, (HL), A, n, (IX+d) und (IY+d) erfolgen. Das Ergebnis der logischen Operation steht nach deren Ausführung immer im A-Register.

Nachfolgend einige Beispiele:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	A0	AND B	;A:=A UND B
1001	AE	XOR (HL)	;A:=A XOR (HL)
1002	F6 55	OR 55H	;A:=A OR 55H
Spezialfall:			
1004	AF	XOR A	;CY:=0, Z:=0, ;P:=1, A:=00H !!!

Das Flag-Register wird bei diesen Operationen neu bestimmt:

Carry-Flag ist stets 0
Zero-Flag entsprechend dem A-Registerinhalt
Sign-Flag entsprechend dem Bit 7 des A-Registers
Paritäts-Flag entsprechend der Anzahl der 1-Bits des
Ergebnisses (P=1, wenn die Anzahl der 1-Bits geradzahlig
ist)

4.3.6. Spezielle arithmetische Hilfsoperationen

DAA ; Dezimalkorrektur bei BCD-Verarbeitung

CPL ; Komplementieren des A-Registers (Einernkomplement)
A:=/A entspricht einer bitweisen Negation

NEG ; Negieren des A-Registers (Zweierkomplement)
A:=-A entspricht A:=/A+1

CCF ; Komplementieren des Carry-Flags, CY:=/CY

SCF ; Setzen des Carry-Flags, CY:=1

Für eine effektive Verarbeitung von Dezimalzahlen können diese direkt durch Addition und Subtraktion verarbeitet werden. Dabei können, wie dem nachfolgenden Beispiel zu entnehmen ist, unkorrekte Ergebnisse auftreten (Pseudoergebnisse). Deshalb wurde der DAA-Befehl zur Erkennung von Pseudoergebnissen und zur anschließenden Korrektur nach arithmetischen Operationen bereitgestellt. Der DAA-Befehl kann nach folgenden Befehlen verwendet werden:

ADD, ADC, INC, SUB, SBC, DEC, NEG

Für die Funktion des DAA-Befehls übernimmt das Carry-Flag CY für das höherwertige Halbbyte des A-Registers {H(A)} und das Half-Carry-Flag H für das niederwertige Halbbyte des A-Registers {N(A)} die Überwachungsfunktion, d. h. ein Übertrag aus dem niederwertigen Halbbyte steht im H-Flag, der aus dem höherwertigen im CY-Flag.

Ein Beispiel soll die Wirkungsweise des DAA-Befehls verdeutlichen: In jedem Halbbyte steht eine Dezimalzahl im zugelassenen Wertebereich von 0.. .9.

Die Korrektur nach Additionen erfolgt in der Form:

$N(A) > 9$ oder $H = 1 \implies A:=A+06H$
 $H(A) > 9$ oder $CY= 1 \implies A:=A+60H$

Bei Subtraktionen erfolgt die Korrektur in folgender Form:
(N-Flag = 1)

$N(A) > 9$ oder $H = 1 \implies A:=A-06H$
 $H(A) > 9$ oder $CY= 1 \implies A:=A-60H$

Zahlenbeispiel: Addition zweier zweistelliger Dezimalzahlen
 A:=99, B:=39, A:=A+B ?

```

  A:   1001   1001   = 99     99
  B:   0011   1001   = 39     +39
-----
A:=A+B: 1101   0010   = D2 falsch !!
         CY=0   H=1!

DAA: +06H  0000   0110   = 38
+60H      0110   0000   = 60
-----
  A:   0011   1000   = 38
         CY=1   H=0   = 138
  
```

Ergebnis: Es entsteht eine dreistellige BCD-Zahl, die sich zu zwei Stellen aus dem A-Registerinhalt und als dritte Stelle aus dem Carry-Bit (CY=1) ergibt:
 (CY, H(A), N(A))=(138)

Auf die anderen Hilfsoperationen soll hier nicht weiter eingegangen werden. Ihre Bezeichnung erläutert die Funktion genügend.

4.3.7. Befehle zur Bit-Manipulation

Diese Befehlsgruppe erlaubt den Einzelbit-Test, das definierte Setzen bzw. Rücksetzen ausgewählter Bits von Registern des Prozessors oder der durch den Inhalt der Register HL, IX und IY adressierten Speicherplätze.

Der Aufbau und die Funktion eines Bit-Test-Befehls läßt sich wie folgt allgemein beschreiben:

```

BIT i,s          ;Bit-Test-Operation, Testergebnis ist über
                  Z-Flag auswertbar; Z:=1, wenn das Bit i des
                                Registers 's' =0 ist,
                                Z:=0, wenn das Bit i =1 ist
                                0 <= i <= 7
Operanden 's' = A, B, C, D, E H, L, (HL), (IX+d), (IY+d)
  
```

Befehle zum Rücksetzen einzelner Bits haben den Aufbau 'RES i,s' und Befehle zum Setzen 'SET i,s' . Sie beeinflussen keine Flags. Mit dem Befehl 'RES 7,A' wird das Bit 7 des A-Registers auf den Wert 0 gesetzt, mit 'SET 4,B' das Bit 4 auf den Wert 1.

4.3.8 Verschiebepfehle

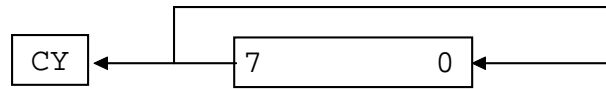
Mit den Verschiebepfehlen erfolgt im entsprechenden Register entweder eine Rechts- oder eine Linksverschiebung um eine Dualstelle, d. h. um ein Bit. Die Besonderheiten der verschiedenen Verschiebepfehle gehen aus den Darstellungen der einzelnen Befehle hervor.

Für alle Operanden 's' gilt:

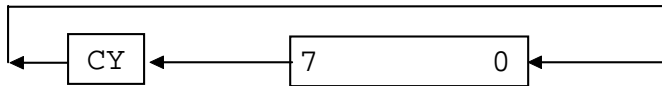
s = A, B, C, D, E, H, L, (HL), (IX+d), (IY+d)

Linksverschieben um eine Dualstelle

RLC s ; Linksverschieben mit Setzen des Carry-Flags



RL s ; Linksverschieben über das Carry-Flag



SLA s ; Linksverschieben durch Carry-Flag

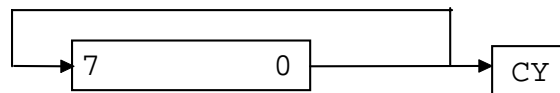


RLCA ; wie RLC A, aber außer dem Carry-Flag wird das Flagregister nicht beeinflußt.

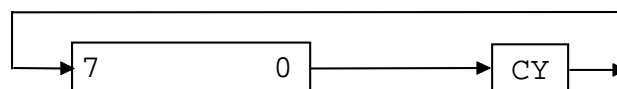
RLA ; wie RL A, aber außer dem Carry-Flag wird das Flagregister nicht beeinflußt.

Rechtsverschiebung um eine Dualstelle

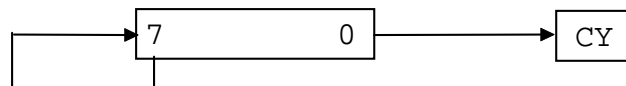
RRC s ; Rechtsverschiebung mit Setzen des Carry-Flags



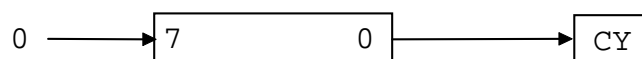
RR s ; Rechtsverschiebung über das Carry-Flag



SRA s ; Rechtsverschieben mit Setzen des Carry-Flags (arithmetische Rechtsverschiebung)



SRL s ; Rechtsverschieben mit Setzen des Carry-Flags (logische Rechtsverschiebung)



RRCA ; wie RRC A, außer Carry wird aber das Flagregister nicht beeinflußt.

RRA ; wie RR A, außer Carry wird aber das Flagregister nicht beeinflußt.

Beispiel: Ein beliebiger Inhalt des A-Registers soll nach rechts verschoben werden, bis Bit 0 des A-Registers = 1 ist. Wenn das A-Register gelöscht war, sollen keine Verschiebungen stattfinden.

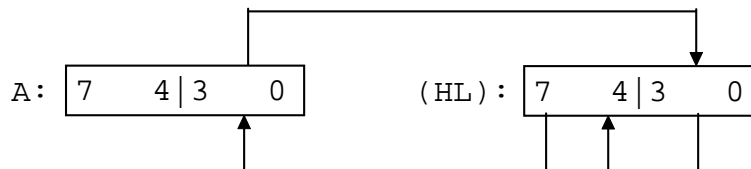
Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	B0	OR A	;Z = 1, wenn A = ist
1001	28 0D	JRZ END-#	;Abbruch des ;Programms, wenn A = 0
1003	CB 3F	M1: SRL A	;Rechtsverschiebung ;durch CY
1005	30 FC	JRNC M1-#	;Rücksprung zu M1 ;wenn CY=0 ist, also das herausge- ;schobene Bit eine 0 enthielt
1007	CB 17	RL A	;wenn Bit 1, ;muß es aus CY wieder in das A-Regi- ;ster zurück
.	.	.	.
1010	76	END: HALT	

Mit einer Verschiebung läßt sich z. B. ein Steuerbit aus dem Bit 0 oder Bit 7 ins CY-Flag schieben und abtesten, ohne die anderen Bits zu zerstören.

Weitere Anwendungsmöglichkeiten ergeben sich für die Arithmetik. Beispielsweise läßt sich der SRA-Befehl als Division durch 2 oder der SLA-Befehl als Multiplikation mit 2 einsetzen. Ein Sonderfall der Verschiebungsoperationen ist die Verschiebung um vier Dualstellen.

Linksverschiebung um vier Dualstellen

RLD ; Der Befehl wirkt zwischen A-Register und dem durch HL indirekt adressierten Speicheroperanden.

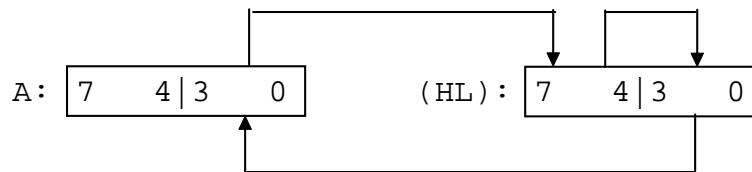


Zahlenbeispiel:

A:	3F		A:	33
		==> RLD ==>		
(HL):	3C		(HL):	CF

Rechtsverschiebung um vier Dualstellen

RRD ; der Befehl wirkt ebenfalls nur zwischen A und (HL)



4.3.9 Sprungbefehle

Grundsätzlich werden bedingte und unbedingte Sprünge unterschieden.

- Unbedingte Sprünge:

Bei Erkennen des Sprungbefehls wird vom Prozessor der Befehlszähler entweder mit der im Adreßteil des Befehls angegebenen Sprungadresse geladen (absolute Sprungadresse) oder der Befehlszähler wird um eine Konstante e verändert (relative Sprungweite). Die Konstante e wird im Befehl mit angegeben und liegt im Wertebereich von $-128 \leq 127$. Die Angabe der absoluten Sprungadresse kann indirekt oder indiziert erfolgen. Relative Sprungbefehle bzw. indirekt adressierte Sprünge benötigen weniger Speicherraum, sind aber bei manueller Programmierung ohne Assembler schwer zu beherrschen.

Allgemein:

JMP sadr ; Befehlszähler PC nimmt den Wert 'sadr' an.
JR e ; Befehlszähler PC wird um den Wert e verändert.
PC:=PC+e (Vorwärts- und Rückwärtssprünge sind möglich)
JMP (xx) ; Befehlszähler PC nimmt den Wert an, der im Registerpaar xx enthalten ist.
Registerpaare 'xx' = HL, IX, IY

Die Berechnung der Sprungweite ist mit entsprechender Sorgfalt durchzuführen.

Die Sprungweite ist eine Differenz von Adressen: Zieladresse minus die Adresse des Befehls, der dem Sprungbefehl folgt. Dabei ist zu beachten, daß ein Relativsprungbefehl immer 2 Byte lang ist. Von der CPU wird prinzipiell nach dem Lesen des Befehls der PC auf den Beginn des physisch nächsten, abzuarbeitenden Befehls gestellt, es sei denn, bei der Abarbeitung des eben gelesenen Befehls stellt sich heraus, daß der PC geändert werden muß (bedingter Sprung). Relativsprünge beziehen sich dann dabei auf die Adresse des nachfolgenden Befehls. Dieser Umstand ist bei der Sprungweitenberechnung unbedingt zu beachten.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	21 00 10	LD HL,1000H	;Zähler laden
1003	75	M1: LD M,L	
1004	2C	INC L	
1005	20 FC	JRNZ M1-#	
1007	24	INC H	
.	.	.	;Programmfortsetzung

Sprungweitenberechnung für den Sprungbefehl auf PC = 1005H:

```

Zieladresse: 1003H
Absprungadresse: - 1007H
-----
Differenz: 0FFFCH

```

Es wird nur das niederwertige Byte der Differenz verwendet, also in diesem Falle 0FCH. Man erkennt außerdem noch folgendes:

Der Betrag der Sprungweite ist größer 7FH (Bit 7 = 1), das heißt, daß rückwärts gesprungen werden muß. Rückwärts in dem Sinne, daß der PC um den Absolutbetrag der Sprungweite 's' vermindert wird. Die Sprungweite 's' ist eine vorzeichen-behaftete Konstante.

- Bedingte Sprünge:

Bedingte Sprünge beziehen sich auf den Zustand der durch vorangegangene Operationen entsprechend veränderten Flags. Auch hier gibt es die Möglichkeit der absoluten und relativen Sprungmarkenangabe. Der Befehlszähler wird wie bereits oben genannt verändert.

```

JPcc adr ; wenn die Bedingung 'cc' erfüllt ist, nimmt
           ; der PC den Wert 'adr' an, PC:=adr

```

```

cc kann sein: C wenn Carry      gesetzt C=1 ==> JPC  adr
              NC wenn No Carry   "      C=0 ==> JPNZ adr
              Z wenn Zero        "      Z=1 ==> JPZ  adr
              NZ wenn No Zero    "      Z=0 ==> JPNZ adr
              P wenn Plus        "      S=0 ==> JPP  adr
              M wenn Minus       "      S=1 ==> JPM  adr
              PO wenn No Parity  "      P=0 ==> JPPO adr
              PE wenn Parity     "      P=1 ==> JPPE adr

```

```

JRcc      ; wenn die Bedingung 'cc' erfüllt ist, wird der
           PC um den Wert 'e' verändert (PC:=PC+e)

```

```

cc =  C , ==> JRC  e
      NC , ==> JRNC e
      Z , ==> JRZ  e
      NZ , ==> JRNZ e

```

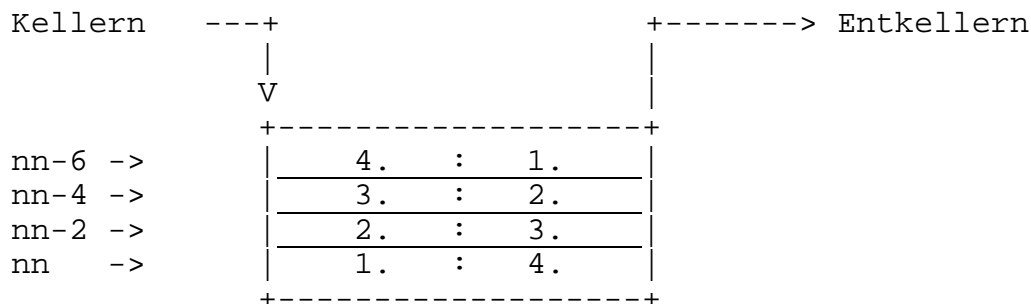
Ein Sonderbefehl ist der Zyklusbefehl mit relativer Adressierung.

DJNZ e ;Das B-Register wird dekrementiert (um 1 vermindert). Solange dessen Inhalt größer Null ist, erfolgt der relative Sprung (PC:=PC+e), sonst wird der diesem Befehl folgende aufgerufen.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	AF	TIME: XOR A	;A:=00H
1001	47	LD B,A	;B:=00H
1002	10 FE	DJNZ 2	;B:=B-1,B=0 ?
			;Rücksprung zum Befehl DJNZ, ;also um 2 Byte zurück
1004	C9	RET	;B:=0, Zeitschleife ;abgelaufen

4.3.10. Kellerooperationen

Die Kellerooperationen sind eine spezielle Gruppe von Transportoperationen. Die Inhalte der Doppelregister werden in einen Speicherbereich gebracht oder von dort geholt, der durch ein spezielles Register (Stackpointer) indirekt adressiert wird. Beim U 880 ist der Kellerspeicherbereich im gesamten Adreßbereich sowohl von der Lage als auch von der Größe frei wählbar. Allerdings kann nur ein RAM verwendet werden. Zu Beginn eines Programmes muß der Stackpointer (SP) festgelegt werden. Das geschieht mit dem Ladebefehl 'LD SP,nn'. Dieser Zeiger ist also eine Adresse und damit maximal 16 Bit lang. Sie stellt die Anfangsadresse (den Kontrollboden) des gewünschten Kellerspeicherbereiches dar.



Die Anwendung der Kellerooperationen liegt vor allem im Retten von Doppelregistern und anschließendem "Zurückholen" aus dem Keller zwecks Weiterverwendung in anderen Programmteilen. Das heißt, die geretteten Register können bei der Abarbeitung eines Programmteiles anders verwendet werden, denn ihre Inhalte wurden im Keller "sichergestellt".

Auf Grund des 8-Bit-Datenbusses wird deutlich, daß eine Kelleroperation immer in zwei Schritten abläuft. Beim Kellern wird zuerst der Stackpointer dekrementiert, das höherwertige Byte gekellert, der SP wieder dekrementiert und dann das niederwertige Byte in den Keller gebracht. Das Entkellern erfolgt derart, daß zuerst das niederwertige Byte gelesen und der SP inkrementiert wird. Anschließend wird das höherwertige Byte gelesen und der SP erneut inkrementiert. Nach Kellern und anschließendem Entkellern zeigt der Stackpointer wieder auf den gleichen Speicherplatz.

Der Kellerbereich wird auch noch von Unterprogrammssprüngen und der Interruptorganisation benötigt, um die erforderlichen Rückkehradressen aufzubewahren und bereitzustellen.

```
Kellern:      PUSH ss    ;ss= BC, DE, AF, HL, IX, IY
Entkellern:   POP  ss     ;ss= BC, DE, AF, HL, IX, IY
```

Beispiel:

Retten der Register BC, DB, AF und HL in einem Unterprogramm, da diese Register im Unterprogramm benötigt werden und danach ihre alten Inhalte wieder erhalten sollen.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	31 00 00	LD SP,0	;Kellerzeiger ;laden
1003	CD F1 10	CALL UP1	;Unterprogrammaufruf ;PC wird gekellert: (FFFFH):=H(PC)=10H ; (FFFBH):=N(PC)=06H ;SP: =FFFEH
1006	.	.	
	.	.	;Programmfortsetzung nach der Rückkehr
	.	.	;aus dem Unterprogramm
	.	.	
	.	.	
	.	.	
	.	.	
10F1	C5	UP1: PUSH BC	;Unterprogramm UP1 ;SP:=FFFCH (FFFDH):=B (FFFCH):=C
10F2	D5	PUSH DE	;SP=0FFFAH (FFFBH):=D (FFFAH):=E
	.	.	
	.	.	
	.	.	
	.	.	;Unterprogramm
	.	.	
1107	D1	POP DE	;SP:=FFFCH
1108	C1	POP BC	;SP:=FFFEH
1109	C9	RET	;Rücksprung zur ;Adresse, die jetzt im Keller oben ;an steht, in diesem Falle 1006H ;SP:=0000H

Man erkennt, daß das "Zurückholen" in genau umgekehrter Reihenfolge erfolgen muß wie das "Retten". Die Ursache liegt darin begründet, daß der zuletzt gerettete Wert zuerst wieder zurückgeholt wird und demzufolge der zuerst gerettete Wert als letzter im Stack verfügbar ist. Im Keller werden also alle geretteten Werte "aufeinandergestapelt" und von diesem Stapel in umgekehrter Reihenfolge wieder herausgeholt.

4.3.11. Unterprogrammoperationen

Unterprogramme sind Programmteile, die sich bei der Abarbeitung eines Programmes oft wiederholen. Um unnötige Programmlängen zu vermeiden, werden solche Teile nur einmal ins Programm eingefügt und bei Bedarf als UP aufgerufen. Das heißt: Das Hauptprogramm wird an diesen Stellen verlassen, das Unterprogramm abgearbeitet und dann das Hauptprogramm mit dem Befehl fortgesetzt, der der Aufrufstelle folgt.

Der Aufruf von Unterprogrammen kann sowohl unbedingt als auch bedingt in Abhängigkeit vom Flagregister erfolgen. Im wesentlichen ist mit dem Unterprogrammaufruf die definierte Änderung des Befehlszählers ähnlich den Sprungbefehlen verbunden, allerdings muß bei Rückkehr aus dem Unterprogramm der Befehlszähler PC die Adresse des Befehls enthalten, der als unmittelbarer Nachfolger des Unterprogrammaufrufes gilt, um genau an dieser Stelle im Programm weiterarbeiten zu können. Dies wird realisiert, indem vor dem Sprung ins Unterprogramm die alte PC-Adresse, die auf den folgenden Befehl zeigt, gekellert wird und erst dann der Befehlszähler PC die Unterprogrammadresse erhält. Bei Rückkehr aus dem Unterprogramm wird diese im Keller gesicherte Adresse wieder in den Befehlszähler geladen. Das erledigt der Prozessor allein durch den Befehl 'RET'.

CALL adr ;Unterprogrammaufruf, es wird die Adresse des CALL-Befehls +3 in den Keller gebracht (also der nachher notwendige PC-Stand), der PC selbst nimmt den Wert adr an (16-Bit-Adresse). Der Stackpointer wird um 2 erniedrigt.

RET ;Unterprogrammrückkehr, aus dem Keller wird die sichergestellte Adresse in den Befehlszähler PC geholt, das Hauptprogramm kann weitergehen; der SP wird wieder um 2 erhöht.

CACC adr ;bedingter Unterprogrammaufruf, wenn die Bedingung cc erfüllt ist, sonst nachfolgender Befehl.

RCC ;bedingte Unterprogrammrückkehr, wenn die Bedingung cc erfüllt ist, sonst nachfolgender Befehl.

Mögliche Bedingungen: cc= NZ, Z, C, NC, P, M, PO, PE

Die Bedingungen der Unterprogrammrufer sind die gleichen wie bei den bedingten Sprungbefehlen.

Einen Sonderfall des Unterprogrammrufer bilden die Restart-Befehle. Hier entfällt die Angabe einer 2-Byte Sprungadresse. Es erfolgen Sprünge zu festen Adressen (0000H, 0008H, 0010H, 0018H, 0020H, 0028H, 0030H, 0038H).

RST hh ;Unterprogrammrufer, der aktuelle PC des RST-Befehle (konkret der PC-Stand nach dem RST-Befehl) wird gekellert. Der PC nimmt den Wert 00hhH an.
hh=0H, 8H, 10H, ... , 30H, 38H

Beispiel zur Unterprogrammtechnik:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	3E FF	START: LD A,0FFH	;Start des Haupt- ;programms
1002		.	
		.	
100E	77	LD (HL),A	
100F	CD 34 13	CALL UP1	;1. Unterprogrammrufer ;UP1 beginnt bei ;Adr.1334H
1012	23	INC HL	;Fortsetzung des ;Hauptprogramms
1013	7E	LD A,(HL)	
		.	
103F	88	ADC B	
		.	
125E	CD 34 13	CALL UP1	;2. Unterprogrammrufer
1261	23	INC HL	;Fortsetzung des ;Hauptprogramms
		.	
		.	
126A			
126B	DD CB 00 DE	SET 3,(IX+0)	
		.	
		.	
1333	76	HALT	
1334	C5	UP1: PUSH BC	;Beginn des UP
1335	CB 4F	BIT 1, A	
		.	
		.	
1353	C1	POP BC	
1354	C9	RET	;Rückkehr ins ;Hauptprogramm

Hinweis:

Es ist unbedingt zu beachten, daß im Unterprogramm die Anzahl der PUSH-Befehle identisch der Anzahl der POP-Befehle sein

muß, da sonst bei der Entkellerung bei Unterprogrammrückkehr ein falscher Wert in den Befehlszähler kommt und Programm undefiniert und nicht ab der Aufrufstelle weiterläuft.

Eine Verschachtelung mehrerer Unterprogramme (ein UP ruft ein weiteres UP auf) ist möglich, die Anzahl der Schachtelungen ist theoretisch unbegrenzt.

4.3.12 Ein- und Ausgabebefehle

Wie bereits erwähnt, sind beim U 880 maximal 256 Ein- und Ausgabekanäle (Peripherieadressen) adressierbar. Zur Adressierung bei Ein- und Ausgaben wird der niederwertige Teil des Adreßbusses verwendet. Im Befehl selbst wird die Geräteadresse (Kanaladresse) entweder direkt ausgegeben oder indirekt über das C-Register adressiert. Bei direkter Kanaladresse erfolgt der Datentransport immer zwischen dem A-Register und der Peripherie, bei indirekter Kanaladresse (C-Register muß vorher mit der Kanaladresse geladen werden) kann der Datentransport zwischen einem der Register A, B, C, D, E, H, L, und der Peripherie erfolgen.

```
IN n      ;Eingabe des Kanals n in das A-Register, A:=(n)
IN r      ;Eingabe des Kanals, dessen Adresse im C-Register
          enthalten ist, nach dem Register 'r'.

OUT n     ;Ausgabe des A-Registers nach Kanal 'n', (n):=A
OUT r     ;Ausgabe des Registers 'r' nach dem Kanal, dessen
          Adresse im C-Register enthalten ist.
```

Register 'r' = A, B, C, D, E, H, L

Einen Sonderfall stellt der Eingabebefehl 'INF' dar. Der Inhalt des von Register C adressierten Kanals wird gelesen und danach entsprechend die Flags gesetzt.

Beispiel: Das Ausgaberegister mit der Adresse 10H soll definiert mit 00, 01 und 02 geladen werden.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	0E 10	LD C,10H	;C:=10H Kanaladresse
		1002 A8 XOR A	;des Registers A:=00H
1003	ED 79	OUT A	; (C):=00H
1005	3C	INC A	;A:=A+1 = 01H
1006	ED 79	OUT A	; (C):=01H
1008	3C	INC A	;A:=A+1 = 02H
1009	ED 79	OUT A	; (C):=02H
	.	.	
	.	.	
1178	76	END: HALT	

4.3.13. Gruppenoperationen für Lade-, Vergleichs- und Ein-/Ausgabe-Befehle

Gruppenoperationen in diesem Sinne sind hardwaremäßig im Prozessor U 880 installierte Befehlsabläufe, die vom Programmierer lediglich die Generierung bestimmter Register als Parameter verlangen. Diese Befehle vermeiden nicht nur das umständliche Programmieren mit einzelnen Befehlen, sondern führen besonders zur Programmbeschleunigung und Einsparung von Programmspeicherplätzen.

- Einfache Gruppenoperationen

LDI ; inkrementierendes Laden
- die Zieladresse ist in DE zu laden
- die Quelladresse ist in HL zu laden
- die Blocklänge in BC (=Anzahl der im Speicher aufeinanderfolgenden 'zu transportierenden Bytes).

Befehlsablauf: (DE):=(HL)
DB:=DE+1
HL:=HL+1
BC:=BC-1
P/V=0 bei BC-1=0
1 bei BC-1<>0

LDD ; dekrementierendes Laden
wie LDI, nur DE:=DE-1 und HL:=HL-1

CPI ; inkrementierendes Vergleichen (Suchbefehl)
- die Anfangsadresse ist in HL zu laden
- die Blocklänge ist in BC zu laden

Befehlsausführung: A-(HL)=?
HL:=HL+1
BC:=BC-1
Z=1 bei A=(HL)
0 bei A<>(HL)
P/V=0 bei BC-1=0
1 bei BC-1<>0

CPD ; dekrementierendes Vergleichen
wie CPI, nur HL:=HL-1

INI ; inkrementierende Eingabe
- Zieladresse ist in HL zu laden
- Kanaladresse ist in C zu laden
- Blocklänge in B (max. 256 Byte)

Befehlsausführung: (HL):=(C)
HL:=HL+1
B:=B-1
Z=1 bei B=0
0 bei B<>0

```

IND      ; dekrementierende Eingabe
         wie INI, nur HL:=HL-1

OUTI     ; inkrementierende Ausgabe
         - Zieladresse ist in C zu laden
         - Kanaladresse ist in Hl zu laden
         - Blocklänge in B

         Befehlsausführung: (C):=(HL)
                           HL:=HL+1
                           B:=B-1
                           Z=1 bei B=0
                           0 bei B<>0

OUTD     ; dekrementierende Ausgabe
         wie OUTI, nur HL:=HL-1

```

Bei den einfachen Gruppenoperationen kann der Programmierer gezielt die Flags testen und entsprechende Rücksprünge bzw. die weitere Programmbearbeitung fordern.

Als mögliche Anwendungen ergeben sich:

```

LDI, LDD zum Blocktransport
CPI, CPD zur Byte- oder Blocksuche
INI, IND Blockeingabe
OUTI, OUTD Blockausgabe

```

Die Blocklänge ist dabei in BC bzw. B anzugeben.

- Repetierende Gruppenoperationen (sich automatisch wiederholende Gruppenoperationen)

Diese Befehle basieren auf den einfachen Gruppenoperationen. Es werden lediglich die Testung der Flags und entsprechende Rücksprünge zur Wiederholung des Befehls automatisch ausgeführt. Allerdings ist bei den Ein- und Ausgabegruppenoperationen zu beachten, daß die peripheren Geräte eine gleiche Verarbeitungsgeschwindigkeit haben müssen wie der Prozessor, oder hardwaremäßig über die WAIT-Leitung an der CPU eine Angleichung der Verarbeitungsgeschwindigkeit erfolgen muß.

```

LDIR ; wie LDI, Befehl wird wiederholt bis BC:=00H
LDDR ; wie LDD, Befehl wird wiederholt bis BC:=00H
OPIR ; wie CPI, Befehl wird wiederholt bis BC:=00H oder
A=(HL)
CPDR ; wie CPD, wird wiederholt bis BC:=00H oder A=(HL)
INIR ; wie INI, wird wiederholt bis B:=0
INDR ; wie IND, wird wiederholt bis B:=0
OTIR ; wie OUII, wird wiederholt bis B:=0
OTDR ; wie OUTD, wird wiederholt bis B:=0

```

Beispiel: Ein Speicherbereich von Adresse 0D00H bis Adresse 0FFFH soll definiert mit "0E3H" geladen werden.

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	21 00 0D	LD HL, 0D00H	;Quelladresse
1003	3E E3	LD A, 0E3H	
1005	77	LD (HL),A	; (HL):=0E3H, ;1. Speicherplatz geladen
1006	11 01 0D	LD DE,0D01H	;1. Zieladresse
1009	01 FE 02	LD BC,2FEH	;Blocklänge -1 ;da ein Byte schon geladen wurde -
1000	ED B0	LDIR	;Befehl bewirkt ;das sogenannte Durchschleifen des ;Wertes '0E3H' bis zur Adresse 0FFFH

4.3.14. Austauschbefehle

EXAF ; Wechsel des Doppelregisters AF gegen AF'

EXX ; Wechsel des Doppelregistersatzes:
BC gegen BC'
DE gegen DE'
HL gegen HL'

Angewendet wird der Registerwechsel beispielsweise zur Rettung bei Interrupt oder Unterprogrammaufruf, da diese Variante schneller ist als das Kellern bzw. Entkellern der Register.

EX DE, HL ; Tausch der Doppelregisterinhalte
E-->L und L-->E
D-->H und H-->D

EX (SP), xx; xx=HL, IX, IY
Tausch der jeweils obenanstehenden Stackinhalte gegen die Inhalte der Doppelregister xx.
(SP) gegen L, N(IX) oder N(IY)
(SP+1) gegen H, H(IX) oder K(IY)

Beispiel:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar
1000	08	INT1: EXAF	;Registerwechsel, da im ;Interruptprogramm ein Großteil der ;Register benötigt wird
1001	D9	EXX	
1002	3E E3	LD A,3FH	
1004		.	
		.	
		.	
101B	D3 04	OUT 04	
101D	D9	EXX	
101E	08	EXAF	
101F	FB	EI	
1020	ED 4D	RETI	

4.3.15. CPU-Steuerbefehle

- NOP ; Es wird keine Operation ausgeführt, nur der Befehlszähler um ein Byte weitergestellt.
- HALT ; Prozessor führt intern NOP-Operationen zur Aufrechterhaltung des Refresh-Zyklus durch.
Der Befehlszähler erhält die Adresse des nachfolgenden Befehls. Dieser Befehl wird aber erst nach der vollständigen Ausführung einer Interruptroutine ausgeführt. Eine Fortsetzung ist auch mit Reset möglich.

4.3.16. Bedeutung der Flags

Ein Großteil aller Mikroprozessorbefehle beeinflusst definiert entweder das gesamte Flag-Register bzw. nur einzelne Flag-Bits. Die genaue Kenntnis der Bedeutung der Flag-Bits bzw. die Art der Beeinflussung durch verschiedene Befehle ist die wichtigste Voraussetzung für die fehlerfreie bzw. optimale Programmerstellung. Die allgemeine Bedeutung der Flag-Bits ist folgender Zusammenstellung zu entnehmen:

- S: Vorzeichen (signum)
Ist eine Kopie des Bits A7;
es repräsentiert das echte Vorzeichen des Resultates nach arithmetischen Operationen mit Operanden in Zweierkomplementdarstellung.
S=1, wenn das Ergebnis <0 ist, Abfrage mit cc=M (JPM)
S=0, wenn das Ergebnis ≥ 0 ist, Abfrage mit cc=P (JPP)
- Z: Null (zero)
Bei einer Null im A-Register (alle 8 Bitpositionen = 0) nach arithmetischen und logischen Operationen (auch nach CMP) wird Z=1.
- H: Halbbyteübertrag (half-carry)
Wird bei einem Übertrag von Bit A3 nach Bit A4 auf "1" gesetzt. (Wird vom DAA-Befehl intern ausgewertet.)
- N: Subtraktion
N=1, wenn der Befehl eine Subtraktion (auch CMP) war.
Wird vom DAA-Befehl ebenfalls herangezogen.
- CY: Übertrag (Carry)
Übertrag aus dem A-Register nach einer Addition, Mittels SCF und CCF ist das CY-Flag auch setz- und komplementierbar.
- P/V: Paritäts/Überlauf (parity/overflow)
Dieses Flag hat verschiedene Funktionen.
- 1.) Es zeigt nach logischen und Verschiebeoperationen und nach dem Befehl 'IN r' die Parität P an:

P=1 bei gerader Anzahl vorhandener Bit-Einsen, die Abfrage kann mit JPPE erfolgen Abfrage einer ungeraden Anzahl mit JPP0

- 2.) Es zeigt im Gegensatz zu Carry den echten Überlauf V des Resultates nach einer arithmetischen Operation mit Operanden an.
Abfrage mit JPPE
- 3.) Das Flag sichert nach LD A,I und LD A,R das Bit aus IFF2, damit es mit 'LD A,I' wieder in IFF2 gesetzt werden kann. In IFF2 wird eine angemeldete Unterbrechung gespeichert, bis sie vom Prozessor bearbeitet wird.
- 4.) Es fixiert nach LD- und CMP-Gruppenoperationen die Aussage, ob der Bytezähler in BC nach Erniedrigung ungleich Null ist.
D. h. P/V=0 bei BC-1=0
P/V=1 bei BC-1<>0

Die Abfrage der Flag-Bits bei bedingten Operanden erfolgt nach den im jeweiligen Befehl anzugebenden Bedingungen (siehe Sprungbefehle) wie folgt:

Bedingung	Frage	
NZ :	Z =0?	==> JPNZ , JRNZ
Z :	Z =1?	==> JPZ , JRZ
NC :	Cy=0?	usw.
C :	Cy=1?	
PO :	P/V=0?	
PE :	P/V=1?	
	P : S=0?	
	M : S=1?	

Die komplette Analyse aller Flag-Bits ist jederzeit möglich durch Kellern von AF und anschließendem Laden der durch den SP angezeigten letzten zwei Speicherplätze in ein Doppelregister.

Beispiel:

Befehls- zähler	Maschinen- kode	Quellkode	Kommentar

-			
1000	F5	PUSK AF	;Kellern von AF
1001	E1	POP HL	
			;H:=(SP+1)=A,L:=(SP)=F
1002	LD A,L	;A:=F	

Die im Anhang beigefügte Befehlsliste enthält Hinweise auf die Beeinflussung der Flag-Bits durch die jeweiligen Befehle sowie die Kodierung der einzelnen Befehle zur manuellen Programmübersetzung.

4.4. Unterbrechungsorganisation

Um auf Signale aus der Umwelt des Prozessors reagieren zu können, kann mit einem Programm eine Leitung (also ein Signal) zyklisch abgefragt werden. Das setzt aber voraus, daß dieses Programm immer dieses Signal abfragt. Eine Reaktion des Rechners erfolgt erst, wenn das Signal durch die Abfrage erkannt wird. Der U 880 verfügt über die Möglichkeit eines Interrupts. Das heißt, daß ein Signal von einem externen Gerät eine Unterbrechung des laufenden Programms zu einem beliebigen Zeitpunkt verursachen kann, um den Prozessor zur sofortigen Abarbeitung eines Programms zu zwingen, das diese Unterbrechung entsprechend behandelt (Interruptbehandlungsroutine).

Für die effektive und schnelle Bearbeitung von Unterbrechungswünschen der Peripherie stehen beim U 880 ein maskierbarer Interrupt und ein nichtmaskierbarer Interrupt zur Verfügung (der maskierbare Interrupt kann verboten werden).

Wird von der Peripherie ein Signal für den nichtmaskierbaren Interrupt erzeugt (L-Pegel am Eingang NMI), so fügt der Prozessor eine RST-Operation mit der festen Adresse 0066H ein. Es liegt beim Programmierer, inwieweit er in diesem Unterprogramm für eine Rettung bestimmter Registerinhalte sorgt. Es gibt keinen Befehl, der den nichtmaskierbaren Interrupt verbietet. Die Anwendung liegt meist bei der Datenrettung vor Erkennen eines Spannungsausfalls des Rechnersystems. Der maskierbare Interrupt kann in 3 unterschiedlichen Betriebsarten auftreten, die in der CPU durch einen Befehl eingestellt werden können.

Für das Sperren bzw. Freigeben aller drei maskierbaren Interruptarten steht jeweils ein Befehl zur Verfügung.

Interrupt sperren: DI
Interrupt erlauben: EI

Trifft eine Anforderung an den maskierbaren Interrupt ein (L-Pegel am Eingang INT), wenn er durch 'DI' gesperrt ist, so nimmt der Prozessor diese Anforderung zur Kenntnis und arbeitet diese Anforderung sofort nach dem Wiederfreigeben der Interrupterlaubnis durch den Befehl 'EI' ab.

Maskierbare Interruptarten

Diese 3 Interruptarten werden durch einen der Befehle 'IM0', 'IM1' oder 'IM2' gesetzt.

IM0: In dieser Betriebsart wird nach akzeptiertem Unterbrechungswunsch der Interruptquelle der nächste abzuarbeitende Befehl vom Datenbus geholt und in den Programmablauf eingeschoben.
(Es bietet sich hier an, RST-Befehle zu verwenden. Das sind 1-Byte-Befehle, die einem Unterprogrammprung entsprechen.) Das heißt aber auch, daß durch das interrupt-

anfordernde Gerät der Befehl auf den Datenbus gelegt wird.

IM1: Nach dem Akzeptieren des Unterbrechungswunsches wird der Befehl 'RST 38H' automatisch ausgeführt. Gegebenenfalls muß im Interruptbehandlungsprogramm eine Abfrageroutine eingeleitet werden, welche den "Interrupt-Anmelder" feststellt und danach entsprechende Programme aktiviert.

IM2: Diese Betriebsart stellt für den Prozessor die leistungsfähigste Unterbrechungsbehandlung dar. Man nennt sie auch Vektorinterrupt, weil die Organisation der Behandlungsroutinen über Zeiger auf einen Adreßvektor realisiert wird.

Bei Unterbrechungsanforderung stellt jede Interruptquelle den für die gewünschte Interruptroutine notwendigen niederwertigen "Zeigerteil" durch Aussenden eines 8-Bit-Wertes auf den Datenbus bereit (sogenanntes Vektorlesen). Dieser Wert bildet den niederwertigen Teil, der Inhalt des I-Registers den höherwertigen Teil einer Adresse einer bestimmten Speicherzelle. Diese und die nachfolgende Speicherzelle beinhalten dann die Adresse des Interruptbehandlungsprogramms. Dies setzt aber natürlich das vorbereitende Laden des I-Registers und der entsprechenden Register der Peripheriebausteine voraus (sogenannte Initialisierung).

Beispiel: 3 Peripheriebausteine verlangen jeweils eine andere Interruptroutine. Die Startadressen für die jeweiligen Unterprogramme lauten:

```
INT1:=0FF0H
INT2:=0FFFH
INT3:=2F00H
```

Aufbau der Startadressentabelle:

```
Zeiger1: 0C00 F0H
          0C01 0FH
Zeiger2: 0C02 FFH <=== N(INT2):=0FFFH
          0C03 0FH <=== H(INT2):=0FH
Zeiger3: 0C04 00H ==> 0FFFH=Startadresse
          0C05 2FH
          0C06
```

- Das I-Register muß mit "0CH" geladen werden. Es stellt den absoluten Zeigerteil dar.
- Dem Peripheriebaustein 1 muß für das entsprechende Register der variable Zeigerteil mit dem Wert "00" mitgeteilt werden. Entsprechend gilt "02H" und "04H" für Baustein 2 und 3.

Durch Aneinandersetzen des I-Registers als High-Teil und des variablen Zeigerteils als Low-Teil entsteht der gesamte Zeiger.

Das heißt zum Beispiel:

Peripheriebaustein 2 meldet einen Interrupt an. Der Interrupt 2 wird zu gegebener Zeit akzeptiert und der Peripheriebaustein muß jetzt den variablen Zeigerteil im Falle des Zahlenbeispiels 02H, auf den Datenbus legen (Vektorlesen der CPU). Jetzt erfolgt die Bestimmung des gesamten Zeigers.

I-Register 0CH (High-Teil)

var. Zeiger 02H (Low-Teil)

Zeiger 2 0C02H

In der Startadrestabelle ist unter der Adresse 0C02H ein 0FFH und unter 0C03H ein 0FH eingetragen, d. h. die Startadresse der zum Peripheriebaustein 2 gehörenden Interruptbehandlungsroutine lautet 0FFFH.

Auf diese Art, läßt sich nicht nur sehr schnell aus vielen Interruptroutinen die für den Peripheriebaustein notwendige herausfinden und aktivieren, es wird auch möglich, für ein und denselben Peripheriebaustein durch Umprogrammieren des "variablen Zeigerteils" eine andere Interruptroutine aufzurufen. Das Akzeptieren einer Interruptanforderung setzt stets das Retten der Fortsetzungsadresse voraus. Dies geschieht wie beim Unterprogrammruf durch Kellern des Befehlszählers. Bei Rückkehr aus der Interruptroutine wird der Befehlszähler wieder entkellert und es erfolgt somit die Fortsetzung der Programmabarbeitung an der zuvor verlassenen Stelle. Das Kellern und Entkellern führt der Prozessor automatisch aus. Auch hier gilt zu beachten, daß im Interruptprogramm die Anzahl der Keller- und Entkelleroperationen gleich sein muß.

Problematisch ist es, wenn mehrere Peripheriebausteine gleichzeitig eine Unterbrechung anmelden. Vorrang hat stets der Baustein, welcher in der Prioritätskette an "weitesten vorn" steht. Diese Kette wird durch die Signale 'IEI' und 'IEO' gebildet, die durch alle Bausteine hindurchgeführt werden. Während der Abarbeitung einer Interruptbehandlungsroutine bleibt es dem Programmierer überlassen, ob er durch Setzen des EI-Befehls die Unterbrechung der Interruptbehandlungsroutine vorzeitig durch einen Peripheriebaustein höherer Priorität zuläßt oder nicht, da die Annahme einer Unterbrechung weitere Unterbrechungsannahmen ausschließt. Am Ende einer Interruptbehandlungsroutine muß stets, sofern nicht schon erfolgt, mit dem Befehl 'EI' die Unterbrechungserlaubnis wieder freigegeben werden. EI wird immer erst nach Ausführung des auf EI folgenden Befehls wirksam.

- Rückkehrbefehle aus dem Interruptprogramm:

```
RETI      ; Rückkehr aus dem maskierbaren Interruptprogramm  
RETN     ; Rückkehr aus dem nichtmaskierbaren Interruptprogramm
```

Hinweis: RETI bewirkt beim Peripheriebaustein, welcher die Interruptroutine ausgelöst hat, das Wiederschließen der Prioritätskette (IEO=high). Somit können nach RETI auch die Peripheriebausteine niedriger Priorität einen Interrupt auslösen.