Assembler

ASSEMBLER

Inhaltsverzeichnis

	ASSEMBLER	

ANHANG

0. Einleitung

Das Nixdorf-System 820 stellt ein kompaktes und leistungsfähiges Angebot an kleinen und mittleren Datenverarbeitungsanlagen dar, das sich bereits in vielen Wirtschaftszweigen hervorragend zur direkten Erfassung und Verarbeitung von Daten bewährt hat.

Das reichhaltige und preisgünstige Angebot an peripheren Geräten ermöglicht es dem Anwender, diesen Computer überall dort in seinem Unternehmen einzusetzen, wo die Organisation eine schnelle Verarbeitung von Daten zur Erlangung einer höheren Effektivität erforderlich macht.

Um das Programmieren einfacher zu gestalten, wurde der Nixdorf-Assembler entwickelt. Das Programmieren in der Assemblersprache erfordert wesentlich weniger Aufwand als das Programmieren in der Maschinensprache.

Der Nixdorf-Assembler übersetzt in Assemblersprache geschriebene Programme in Maschinensprache. Das Quellprogramm wird auf Lochkarten gespeichert. Das übersetzte Programm (Objektprogramm) kann auf den Datenträgern Lochkarte und Lochstreifen gespeichert werden. Der Assembler erstellt ein Protokoll, das die Befehle in Assembler- und Maschinensprache enthält.

T E I L A

Formaler Aufbau der Assembler Sprache

1. Nixdorf Assembler System Serie 820

1.1. Einleitung

Das Nixdorf Assembler System setzt die eigentliche Assembler Sprache "Symbolsprache" und das Übersetzungsutility "ASSEMBLER" voraus. Der erste Teil dieses Handbuches befaßt sich mit der Assembler Sprache. Der zweite Teil behandelt die Vorschriften zum Assemblieren und den Assembliervorgang.

1.2. Die Assembler Sprache

Die Assembler Sprache ist eine symbolische Programmiersprache, die es gestattet, mit symbolischen Merkmalen (Adressen) zu arbeiten. Die A n w e i s u n g e n dieser Sprache (Statements), werden durch einen leicht verständlichen symbolischen Code dargestellt, dessen Mnemotechnik der englischen Sprache entnommen ist.

Der Aufbau der Assembler Sprache läßt sich unter zwei Gesichtspunkten erläutern:

- formal und
- logisch.

Der formale Teil befaßt sich mit dem Aufbau der D a t e n s ä t z e, während der logische Teil die Funktionen dieser Datensätze behandelt.

2. Formaler Aufbau der Assembler Sprache

2.1. Allgemeines zur Syntax

Die Syntax zur Assembler Sprache legt die Regeln fest, wie sich aus Elementen (2.2.) und zulässigen Zeichen (2.3.) gültige Ausdrücke (2.4.) und Datensätze (2.6.) bilden lassen. Die Syntax bestimmt also alle formalen Möglichkeiten, die sich aus der Kombination der Elemente ergeben dürfen.

2.2. Elemente der Assembler Sprache

Elemente sind Bausteine der Sprache, mit denen sich alle Funktionen ausdrücken lassen. Es gibt folgende Elemente:

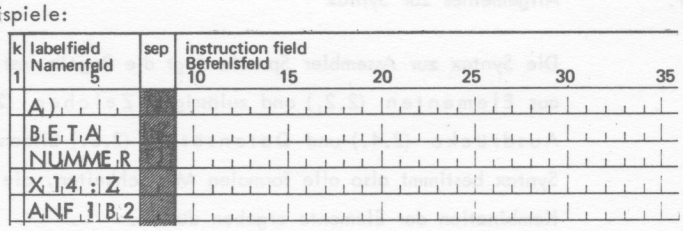
- Name
- Zahl
- Funktionszeichen.

2.2.1. Name

Ein Name darf nur aus Ziffern (2.3.1.), Buchstaben (2.3.2.) und einem Teil der Sonderzeichen (2.3.3.) bestehen. Die im Namen zugelassenen Sonderzeichen müssen ≤ 3.15 (ALC-Code) sein und dürfen keine besonderen Funktionen (Funktionszeichen) ausführen. Der Formalismus der Syntax bestimmt, daß ein Name mit

einem Buchstaben beginnen muß und mindestens aus zwei, maximal aus sechs Buchstaben, Ziffern oder Sonderzeichen bestehen darf.

Beispiele:



2.2.2. Zahl

Eine Zahl besteht aus einer Folge von Ziffern. Eine Zahl in dezimaler Schreibweise darf maximal aus sechs Ziffern bestehen und den Wert ≤261 143 annehmen.

Eine sedezimale Zahl wird durch das Wertigkeitszeichen '.' unterteilt und enthält maximal fünf Ziffern, z.B. 3.15.15.15.15 . Diese sedezimale Zahl entspricht dem dezimalen Wert 261 143.

2.2.3. Funktionszeichen

Zu den Funktionszeichen gehören:

- Trennzeichen
- Verknüpfungszeichen
- Kennzeichen
- Wertigkeitszeichen
- Ersetzungszeichen
- Abbruchzeichen.

2.2.3.1. Trennzeichen

Trennzeichen sind Komma und Blank ',','␣'. Beide Zeichen sind gleichwertig. Sie trennen z.B. den Operationscode (2.8.1.2.) von den Operanden (2.8.1.4.) und die Operanden untereinander.

2.2.3.2. Verknüpfungszeichen

Die Verknüpfungszeichen Plus und Minus '+', '-' sind arithmetische Bindeglieder. Verknüpfungszeichen verbinden Namen (2.2.1.) und Zahlen (2.2.2.) miteinander und stehen bündig, d.h. ohne Blank, zwischen zwei Elementen (2.2.).

2.2.3.3. Kennzeichen

Es gibt drei Kennzeichen. Sie sind Sonderzeichen, die eine besondere syntaktische Funktion haben.

- Stern '*'
- Zwischenraum, Blank '␣'
- Gleichheitszeichen '='.

2.2.3.3.1. Das Kennzeichen Stern '*' in Spalte 1 leitet die Pseudobefehle "Setzen Befehlszähler" (2.7.4.) und "Wertzuweisungen" (2.7.3.) ein.

2.2.3.3.2. Das Kennzeichen Blank '␣' wird nur dann als Kennzeichen interpretiert, wenn es auf dem Codierformular bzw. der Lochkarte in Spalte 1 steht.

		ASSEMBLER		

2.2.3.3.3. Das Kennzeichen Gleichheitszeichen '=' wird nur dann als Kennzeichen interpretiert, wenn es auf dem Codierformular bzw. der Lochkarte in Spalte 1 steht.

2.2.3.4. Wertigkeitszeichen

Das Wertigkeitszeichen Punkt '.' wird auch als Sedezimalpunkt bezeichnet. Er steht bündig, d.h. ohne Blank, zwischen den Stellen einer Sedezimalzahl und trennt sie voneinander.

2.2.3.5. Ersetzungszeichen

Das Ersetzungszeichen '/' tritt in drei Varianten auf:

- in L i t e r a l e n
- in Befehlen
- in Spalte 10.

2.2.3.5.1. Ersetzungszeichen in Literalen

L i t e r a l e werden vom A S S E M B L E R in Tabellenform abgespeichert, und zwar drei Zeichen in einem Befehlswort. Enthält ein Literal ein Zeichen, das nicht auf der Lochertastatur vorhanden ist, so muß es umschrieben und von Ersetzungszeichen bündig eingeschlossen werden. Das Symbol des Zeichens ist der ALC - Code Tabelle im Handbuch zu entnehmen. Nicht darstellbar in einem Literal sind alle Zeichen mit dem Wert > 3.15, da pro Zeichen nur 6 bit zur Verfügung stehen.

Beispiel:

```
            5‰PROVISION
  |1|   5     |  10   15   20   25   30   35
1 | |TEXT     |▒▒|*5/YPML/PROVISION/YECC*
```

Bei dem Endezeichen YECC kann das zweite Ersetzungszeichen fortfallen, da es durch den Ende-Stern ersetzt wird.

2.2.3.5.2. Ersetzungszeichen in Befehlen

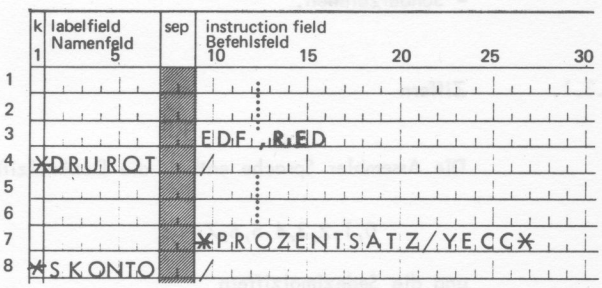
Wird ein Ersetzungszeichen im Operandenteil eines Befehls angegeben, so wird das Ersetzungszeichen durch die aktuelle Adresse des Befehls ersetzt. Das Ersetzungszeichen kann durch ein Verknüpfungszeichen und einen Ausdruck ergänzt werden.

Beispiele:
```
BR,/+3
BR /+DIFF+7
SST /+PAUL
OPX,/+SPATIC-2
```

2.2.3.5.3. Ersetzungszeichen in Spalte 10

Befinden sich vor dem Ersetzungszeichen in Spalte 10 ein Kennzeichen '*' in Spalte 1 und ein Name in den Spalten 2 bis 7, so ist der Name die symbolische Adresse des vorangegangenen Befehles. Wird die Anweisung nach einem Literal gegeben, so ist der Name die symbolische Adresse des letzten Zeichentriples.

Beispiel:

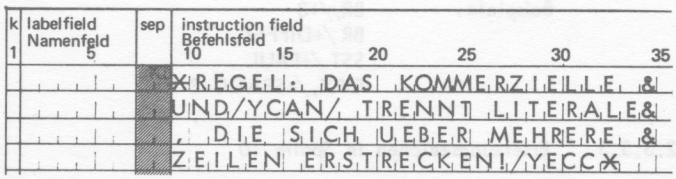


Der Befehl EDF,RED erhält die symbolische Adresse DRUROT. SKONTO ist die symbolische Adresse des Befehles, der aus den Zeichen T,Z und YECC (3.15) erzeugt wird.

2.2.3.6. Abbruchzeichen

Als Abbruchzeichen dient das kommerzielle UND '&'. Es zeigt an, daß sich ein Literal über mehr als eine Zeile des Codierformulars bzw. einer Lochkarte erstreckt.

Beispiel:



2.3. Zeichenvorrat

Der Zeichenvorrat umfaßt:

- Ziffern,
- Buchstaben und
- Sonderzeichen.

2.3.1. Ziffern

Die Assembler Sprache enthält die Dezimalziffern

0,1,2,3,4,5,6,7,8,9

und die Sedezimalziffern

0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15.

2.3.2. Buchstaben

Zu den Buchstaben der Assembler Sprache zählen alle Zeichen des lateinischen Alphabets. Es werden jedoch nur die Großbuchstaben verwendet.

A, B, C, D, E, F, G, H, I, J, K, L, M,
N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

	ASSEMBLER	

2.3.3. Sonderzeichen

Alle Zeichen, die nicht Ziffern oder Buchstaben sind, werden zu den Sonderzeichen gerechnet. Der Vorrat an Sonderzeichen ist abhängig von dem verwendeten Kugelkopf des Serialdruckers. Folgende Sonderzeichen befinden sich auf dem Kugelkopf Nr. 1000, der für Deutschland und Österreich gültig ist:

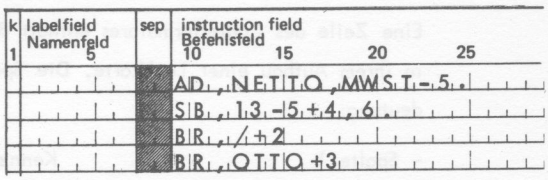
Symbol	Zeichen	Symbol	Zeichen	Symbol	Zeichen
YBLK	␣	YNMB	#	YEQL	=
YEND	-	YPNT	.	YULN	—
YPLS	+	YCM	,	YPOP	(
YMIN	-	YSEM	;	YPCL	)
YITS	◇	YCOC	:	YDLR	$
YITSM	◇̲	YEXM	!	YCAN	&
YSTR	*	YQEM	?	YPD	⁄
YSTRM	*̲	YQOM	"	YPCT	%
YCMT	,	YPRG	§	YPML	‰
YCA	@	YHS	⌑	YAPH	'
YM	m	YL	L	YBAR	/
YT	t	YDIA	ø	YECC	
YMCB	m^3	YST	st	YMSQ	m^2
YKG	kg	YG	g	YINN	1/

2.4. Ausdrücke

Ausdrücke setzen sich zusammen aus Werten, die durch Verknüpfungszeichen verbunden sind. Werte können dargestellt werden durch:

- Namen, EING+NETTO
- Zahlen, 17+5
- Ersetzungszeichen /+2

Beispiele:



Auch ein Name oder eine Zahl ohne Verknüpfungszeichen (entspricht einem Verknüpfungszeichen mit der Zahl Null) ist ein Ausdruck:

Beispiel:

NAME (±0)
DIFF (±0)
123 (±0)

2.5. Formatvorschriften

Die Syntax der Assembler Sprache legt bestimmte Formatvorschriften fest, die auf dem Codierformular angegeben sind. Dadurch ist der Aufbau der Datensätze festgelegt. Weiterhin bestimmen sie, wie viele und welche Stellen zu einem Datensatz gehören, und wo die Angabe der Zeichen zu erfolgen hat.

2.6. Datensätze

Ein Datensatz besteht aus einer Folge von Zeichen, die zu einer Einheit zusammengefaßt sind. Diese Einheit entspricht formal einer Zeile des Codierformulars bzw. dem Inhalt einer 80-spaltigen Lochkarte.

Gliederung der Datensätze

Eine Zeile des Codierformulares enthält 80 Spalten. Sie entspricht in ihrem Aufbau einer Lochkarte. Die Spalten haben folgende Bedeutung:

- Spalte 1	Kennzeichenfeld
- Spalte 2 bis Spalte 7	Namenfeld
- Spalte 8 bis Spalte 9	Trennfeld
- Spalte 10 bis Spalte 74	Befehlsfeld
- Spalte 75 bis Spalte 80	Folgenummernfeld

2.6.1. Kennzeichenfeld

Im Kennzeichenfeld dürfen 3 verschiedene Angaben gemacht werden. Erlaubt sind die Kennzeichen:

- Stern '*'
- Gleichheitszeichen '='
- Blank '␣'

Ein Stern '*' in Spalte 1 besagt, daß dieser Datensatz als Pseudobefehl (2.7.) ohne Formularsteuerungen interpretiert wird.

Ein Gleichheitszeichen '=' in Spalte 1 veranlaßt einen Seitenvorvorschub im Assemblerprotokoll.

Ein Blank'␣' in allen 80 Spalten (Blank-Karte) veranlaßt einen Zeilenvorschub im Assemblerprotokoll.

2.6.2. Namenfeld

Das Namenfeld kann durch die Angabe eines Namens dem Befehlsfeld eine symbolische Adresse zuweisen.

2.6.3. Trennfeld

Das Trennfeld teilt Namen- und Befehlsfeld voneinander. Es muß immer Blanks enthalten.

2.6.4. Befehlsfeld

Das Befehlsfeld beinhaltet alle Angaben, die eine Anweisung zwingend benötigt. Zusätzlich kann im Befehlsfeld ein Kommentar stehen. Dieser muß durch mindestens zwei Trennzeichen (2.2.3.1.) von der Anweisung getrennt sein. Beginnt das Befehlsfeld mit 1 Blank (Spalte 10), so wird der gesamte Inhalt des Befehlsfeldes als Kommentar interpretiert.

2.6.5. Folgenummernfeld

Das Folgenummernfeld soll zur Identifikation der Lochkarte dienen, durch Numerierung der Anweisungen in den Spalten 75 bis 80.

2.7. Pseudobefehle

Pseudobefehle sind Datensätze (2.6.), die den Ablauf des ASSEMBLERS steuern. Sie sind nur während des Assemblierens wirksam und erscheinen daher lediglich im Quellprogramm.

Der Nixdorf-Assembler kennt vier Pseudobefehle von denen einer im Programm zwingend notwendig ist (END-Anweisung). Die übrigen drei Pseudobefehle dienen dem Programmierer zur Vereinfachung des Programmierens.

Die vier Pseudobefehle sind:
- END-Anweisung
- Formularsteuerung
- Wertzuweisung
- Setzen Befehlszähler

2.7.1. END-Anweisung

Der Pseudobefehl END-Anweisung muß immer als letzter Befehl gegeben werden. Durch ihn wird die Assemblierung beendet. Formal besteht die END-Anweisung aus

2 Sternen in den Spalten 1 und 2.

```
* *
```

2.7.2. Formularsteuerung

Mit den Befehlen der Formularsteuerung kann der Programmierer den Ausdruck des Assemblerprotokolls übersichtlich gestalten. Er hat die Möglichkeit, eine Steuerung pro Zeile und pro Seite vorzunehmen.

Wird eine Formularsteuerung pro Zeile gewünscht, dürfen keine Eintragungen in den Spalten 1 bis 80 des Codierformulars vorgenommen werden.

Die Formularsteuerung auf dem Anfang der nächsten Seite verlangt ein Gleichheitszeichen '=' in Spalte 1.

2.7.3. Wertzuweisungen

Bei diesem Pseudobefehl wird einem Namen ein Wert zugeordnet, der eine Konstante oder eine Adresse sein kann. Symbolische Angaben sind erlaubt, sofern sie vorher definiert worden sind.

Gekennzeichnet wird die Wertzuweisung durch einen Stern '*' in Spalte 1. Das Namenfeld muß einen Namen beinhalten. Ab Spalte 10 wird der Wert eingetragen, der dem Namen zugeordnet werden soll.

2.7 4. Setzen Befehlszähler

Dieser Pseudobefehl verändert den Befehlszähler, d.h. der Befehlszähler steht nach Ausführung des Befehls auf der Adresse, die im Befehlsfeld angegeben wird.

Die Angabe kann relativ erfolgen. Voraussetzung ist, daß die symbolischen Namen vorher definiert werden müssen.
Tritt bei dieser Anweisung ein Fehler auf, wird der Inhalt der Lochkarte und ein Fehlerschlüssel ausgedruckt. Ein Fehler kann sowohl im ersten als auch im zweiten Assembler-Durchlauf auftreten.

	ASSEMBLER	

Das Programm wird unterbrochen und die rote Lampe leuchtet auf. Jetzt ist die Möglichkeit gegeben, die letzte - als falsch erkannte Karte - zu korrigieren und erneut einzugeben. Wird die Karte nicht korrigiert, wird mit der letzten aktuellen Adresse des Befehlszählers weiter assembliert. Durch Druck auf die Taste C wird die Assemblierung fortgesetzt.

Elemente dieses Pseudobefehls sind:

- Spalte 1	Kennzeichen Stern '*'
- Spalten 2 bis 9	Blank '␣'
- ab Spalte 10	Ausdruck, Stern '*', Zahl (modulo).

Beispiele:

*	NETTO
*	238
*	*8

Das letzte Beispiel zeigt einen Sonderfall. Die hinter dem Stern in Spalte 10 angegebene Zahl wird nicht selbst als Adresse zugewiesen, sondern zur Konstruktion einer Adresse benutzt.

Die aktuelle Adresse, das ist die Befehlswortadresse des letzten Zeichentriples eines Literals oder des Befehls vor einer Befehlszähleränderung, wird so lange erhöht, bis sie durch die ab Spalte 11 angegebene Zahl teilbar ist (modulo). Der ASSEMBLER weist dem folgenden Befehl die errechnete Befehlswortadresse zu.

Beispiele:

1. Aktuelle Adresse 3.4.7
   Setze Befehlszähler *1.
   Zugewiesene Adresse 3.5.0

2. Aktuelle Adresse 0.4.7
   Setze Befehlszähler *5
   Zugewiesene Adresse 0.4.11

2.8. Befehle

Ein Befehl besteht formal aus einem oder mehreren Datensätzen. Ist der Befehl in einem Datensatz enthalten, ist das entscheidende Element der Operationscode (2.8.1.2.). Belegt ein Befehl mehrere Datensätze, ist der Befehl eine Konstante. Die Anzahl der Datensätze richtet sich nach der Länge der Konstante.

2.8.1. Der Befehl als ein Datensatz

Die folgenden Elemente bestimmen den Aufbau des Datensatzes:

- Symbolische Adresse
- Operationscode (OP-Code)
- OP-Code Ergänzung
- Operanden.

2.8.1.1. Symbolische Adresse

Die Adreßangabe mit einem symbolischen Namen steht im Namenfeld. Sie ist ein Name und daher an seine syntaktischen Vorschriften gebunden. Der Name sollte Bezug haben zu dem Programm, der Routine, der Tabelle usw., die er kennzeichnet.

Beispiele:	CASVAR	Lade Kernspeicher aus Cassette (Utility)
	PAUL	Produktions-Auftrags-Liste (Programm Name)
	LKLES	Lochkarte lesen (Unterprogramm Name)
	STOLZI	Storno Liste Zinsvergütung (Routine)

	ASSEMBLER	

2.8.1.2. Operations-Code

Der Operations-Code (OP-Code) ist eine symbolische Verschlüsselung, die aus maximal 6 Zeichen besteht. Die Zeichen sind meistens Buchstaben. In einigen OP-Codes ist das letzte Zeichen eine Ziffer. Eine Aufstellung der OP-Codes ist der "Liste der Befehle" zu entnehmen. Die Assembler Sprache ist format- und reihenfolgefrei. Auf den OP-Code bezogen bedeutet es, daß er an einer beliebigen Stelle im Befehlsfeld angegeben werden kann, sofern das erste Element in Spalte 10 beginnt.

Beispiele:

AD,	SM1,
MVH,	DC2,
POSX,	BUFX2,
SGNIN,	ALOUT8,
SORTMX,	

Die Symbole der OP-Codes sind in der "Liste der Befehle" im Programmier Handbuch aufgeführt.

2.8.1.3. Operations-Code Ergänzung

Die OP-Code Ergänzung ist wie der OP-Code eine symbolische Verschlüsselung. Sie besteht meistens aus Buchstaben. In einigen Fällen ist das letzte Zeichen eine Ziffer. Die Ergänzungen stehen bei den OP-Codes und sind aus der "Liste der Befehle" im Programmier Handbuch zu entnehmen.

Ob eine OP-Code Ergänzung geschrieben wird oder mehrere, ist abhängig vom OP-Code.

Es gibt OP-Codes

- ohne Ergänzung
- mit einer zwingenden Ergänzung
- mit einer wahlweisen Ergänzung
- mit mehreren wahlweisen Ergänzungen.

Beispiele:

ohne Ergänzung

```
POSX,
EFK,
RBOUT,
WTC
```

zwingende Ergänzung

```
RDX,PC
XPN,PC
RDX,PT
```

wahlweise Ergänzung

```
SM1,ZERO
SML,ONE
EDF,RED
```

mehrere wahlweise Ergänzungen

```
EDF,ZERO,REDL,SGN
PPCP,XR,SEC
```

2.8.1.4. Operanden

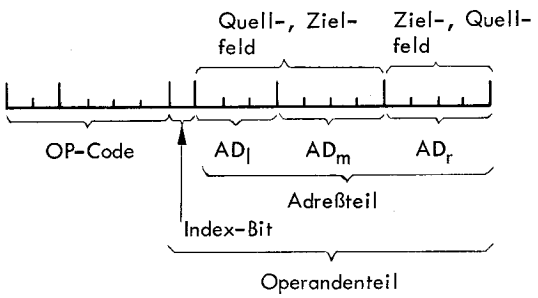
Die Operanden werden als Ausdrücke im Befehlsfeld angegeben.
Die Anzahl der Operanden ist abhängig vom OP-Code.

Es gibt OP-Codes

- ohne Operanden
- mit einem Operanden
- mit zwei Operanden.

Da die Reihenfolge der Elemente innerhalb des Befehlsfeldes beliebig ist, muß bei der Angabe von zwei Operanden unterschieden werden, welcher Operand das Quellfeld und welcher das Zielfeld beinhaltet.

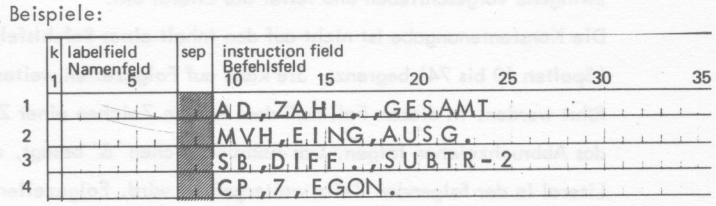
Quell- und Zielfeld wiederum stehen in Abhängigkeit der Befehle einmal im Adreßteil rechts oder links und mitte des Maschinenwortes.



Soll die Adresse eines Feldes im Adreßteil links und mitte des Maschinenwortes gespeichert werden, wird dem Operanden ein Punkt '.' hinzugefügt. Diese Angabe sagt dem ASSEMBLER, daß der Operand im Adreßteil des Maschinenwortes um 4 Stellen nach links versch-

ben wird. Wird ein Operand mit zwei Punkten '..' geschrieben, wird ein shift um 8 Stellen durchgeführt.

Beispiele:



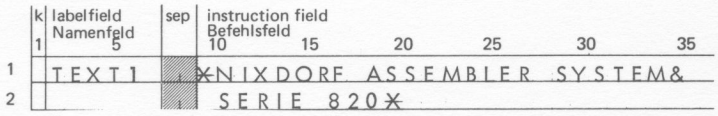
2.8.2. Der Befehl mit einem bzw. mehreren Datensätzen (Konstante, Literal)

Eine Konstante ist im Unterschied zu einem variablen Symbol der Ausdruck für eine feststehende Größe. Konstanten werden oft als Kenndaten bezeichnet und stellen eine alphanumerische Zeichenkette, ein Literal, dar. Sie sind während eines Programmablaufes immer gleichbleibend und stehen unverändert im Kern- oder Festspeicher. Auf eine Konstante kann im Programm wiederholt Bezug genommen werden.

2.8.2.1. Elemente der Konstante

Die Konstante setzt sich aus folgenden Elementen zusammen:

- Name
- Stern '*' in Spalte 10
- Konstantenangabe
- ggf. Abbruchzeichen '&'
- Endestern '*'



Der Name unterliegt dem Formalismus der Syntax und ist die symbolische Anfangsadresse des Literals. Ein Stern '*' in Spalte 10 ist zwingend vorgeschrieben und leitet das Literal ein.
Die Konstantenangabe ist nicht auf den Inhalt eines Befehlsfeldes (Spalten 10 bis 74) begrenzt. Sie kann auf Folgezeilen weitergeführt werden. In diesem Fall muß dem letzten Zeichen einer Zeile das Abbruchzeichen folgen. Das Abbruchzeichen '&' besagt, daß das Literal in der folgenden Zeile weitergeführt wird. Folgezeilen dürfen keine Namen (Spalten 2 bis 7) beinhalten. Der Endestern '*' schließt ein Literal formal ab.

T E I L B

Logischer Aufbau der Assembler Sprache

3. Der logische Aufbau der Assembler Sprache

Die Assembler Sprache besteht logisch aus einer Anzahl von Anweisungen, die mit Hilfe eines Übersetzers, dem ASSEMBLER, in den Maschinencode umgewandelt werden. Mit diesen übersetzten Anweisungen ist der Computer in der Lage, alle gestellten Aufgaben zu lösen.

3.1. Anweisungen

Anweisungen haben Funktionen, die verschiedene Steuerungen bewirken. Eine Anweisung ist ein Glied in einer Kette logisch zusammenhängender Anweisungen, deren Gesamtheit einen logischen Ablauf, ein Programm darstellt. Sie teilen sich in zwei Gruppen.

- Befehle und
- Pseudobefehle.

Die Befehle sind mnemonische Ausdrücke, denen beim Assemblieren ein Maschinencode zugeordnet wird. Sie sind Bestandteil des Objektprogramms.

Pseudobefehle sind ebenfalls mnemonische Ausdrücke, die allerdings nur während des Assembliervorgangs wirksam sind. Sie haben keinen Maschinencode und sind daher kein Bestandteil des Objektprogramms.

3.2. Befehle

Befehle werden in einem Maschinenwort dargestellt. Das Maschinenwort, auch Befehlswort genannt, besteht aus 18 bit, die in 6 bit für den Operationsteil (OP-Teil) und 12 bit für den Operandenteil aufgeteilt werden (siehe Aufbau des Maschinenwortes Seite 19).

Aufbau der Maschinenworte als Befehlsworte

Der OP-Teil belegt die bit 18 bis 13 und kann 64 Werte annehmen, die sedezimal von 0.0. bis 3.15 geschrieben werden. Es sind nicht alle 64 Werte belegt, d.h. nicht alle Werte ergeben gültige OP-Codes.

Der Operandenteil zerfällt in vier Gruppen

- 1 Indexbit (AD_i = bit 12)
  mit den Werten 0 bis 1
- 3 bit Adreßteil links (AD_l = bit 11 bis 9)
  mit den Werten 0 bis 7
- 4 bit Adreßteil mitte (AD_m = bit 8 bis 5)
  mit den Werten 0 bis 15
- 4 bit Adreßteil rechts (AD_r = bit 4 bis 1)
  mit den Werten 0 bis 15

Die Adreßteile links, mitte, rechts können zusammen Werte von 0 bis 2047 annehmen.

4. Der ASSEMBLER

Das Nixdorf Assembler System 820 ist ein 2 Phasen Übersetzer mit wahlweiser Lochkarten- und Lochstreifenausgabe während des zweiten Durchlaufes. Alle ASSEMBLER-Angaben werden in englischer Sprache ausgedruckt.

Sämtliche Protokolle erfolgen seitengerecht und fortlaufend auf Standardpapier (DIN A 4 quer, 8 Zoll hoch).
Jede Seite beginnt mit einer Überschrift.
Links: wird die Art des Protokolls ausgedruckt, z.B. ASS-PASS I, ASS-PASS II usw.
Rechts: der Hinweis PAGE mit der jeweiligen Seitennummer.
Blank-Karten dienen zur Gestaltung des Druckbildes. Eine Blank-Karte entspricht einem Zeilenvorschub.
Das Ende eines Protokolls wird immer durch END abgeschlossen.

4.1. Erster Durchlauf

4.1.1. Einleitung

Im ersten Durchlauf werden alle Programmkarten eingelesen. Jede Lochkarte beinhaltet ein Statement, einen Datensatz. Der Aufbau der Karte muß dem formalen Aufbau des Codierformulars entsprechen, und zwar:

S 1				Kennzeichenfeld
S 2	bis	S 7		Namenfeld
S 8	bis	S 9		Trennfeld
S 10	bis	S 74		Befehlsfeld
S 75	bis	S 80		Folgenummernfeld

	ASSEMBLER	

Der ASSEMBLER übersetzt auch symbolische Programme der bisherigen Version, sofern nur 1 Statement pro Karte in den Spalten 1 bis 40 vorhanden ist.
Eine Besonderheit bei der Textverarbeitung ist zu beachten.
Sollte es einmal vorkommen, daß Lochkarten, die nur einen Befehl enthalten, mit dem bisherigen ASSEMBLER assembliert werden, ergeben sich Schwierigkeiten bei der Textverarbeitung. In diesem Fall muß in der Spalte 51 ein zusätzliches Abbruchzeichen gelocht werden. Diese Karten können auch bei dem jetzigen ASSEMBLER benutzt werden. Es erscheint lediglich das zusätzliche Abbruchzeichen als Kommentar.

Die folgenden wesentlichen Merkmale kennzeichnen den ersten ASSEMBLER-Durchlauf:

festgelegte Arbeitsabläufe

- Formale Prüfung aller Statements mit Fehlerprotokoll
- Anlegen des Adreßbuches
- Ausdruck der nicht definierten Namen

wahlweise Arbeitsabläufe

- Prüfung der Folgenummern
- Auflistung des symbolischen Programmes.

4.1.2. Festgelegte Arbeitsabläufe

4.1.2.1. Formale Prüfung aller Statements mit Fehlerprotokoll

Während des ersten Durchlaufes erkennt der ASSEMBLER alle

formalen Fehler. Fehlermeldungen erfolgen über einen Schlüssel. Eine Liste der Fehlermeldungen befindet sich im Anhang.

Eine Fehlermeldung setzt sich aus dem Buchstaben "E" (bedeutet ERROR) und maximal sechs Zahlen zusammen.

Beispiel:

E3,5,7

In der Fehlerliste kann unter E3,5 und 7 der Typ des Fehlers festgestellt werden.

Tritt in einem Statement ein Fehler mehrfach auf, wird er so oft gemeldet, wie er auftritt. Für jedes Statement können bis zu 6 Schlüssel ausgedruckt werden.

4.1.2.2. Anlegen des Adreßbuches

Weiterhin wird im ersten Durchlauf eine Tabelle für die Zuordnungen von Namen und Adressen aufgebaut. Es wird ein Adreßbuch angelegt.

4.1.2.3. Ausdruck der nicht definierten Namen

Enthält ein Programm undefinierte Namen, werden sie in der Liste UNDEFINED SYMBOL ausgedruckt. Anhand dieser Liste kann eine schnelle Fehlerbeseitigung erfolgen.

	ASSEMBLER	

4.1.3. Wahlweise Arbeitsabläufe

4.1.3.1. Prüfung der Folgenummern

Wird die Prüfung der Folgenummern gewünscht, muß die aufsteigende Reihenfolge gewährleistet sein. Blanks in den Spalten 75 bis 80 setzen den internen Zähler auf Null. Dadurch ist es möglich, zusammengesetzte Programme mit Folgenummernprüfung zu assemblieren.

4.1.3.2. Auflistung des symbolischen Programms

Der wahlweise Ausdruck des symbolischen Programms gibt dem Programmierer die Möglichkeit, die Testzeit eines Assemblerprogramms erheblich zu verkürzen. Da im ersten Durchlauf alle formalen Fehler erkannt werden, kann der Programmierer bei den ersten Übersetzungen auf das Protokoll des symbolischen Programms verzichten. Er erhält nach diesen "schnellen" ersten Durchläufen in jedem Falle ein Fehlerprotokoll, das sich so oft wiederholt, bis das Programm formal fehlerfrei ist. Jetzt empfiehlt sich der Ausdruck des symbolischen Programms.

4.1.4. Maschinenbedienung

Der Ablauf des ASSEMBLERS, bzw. die Ausführung bestimmter Routinen während des Assemblierens, wird durch bestimmte Tasten der numerischen Eingabetastatur gesteuert. Ein Übersichtsblatt der Tastatur ist dem Anhang zu entnehmen.

Besondere Funktionen der Tasten

Den folgenden Tasten werden vom Assembler besondere Funktionen zugeordnet:

Code		Bedeutung
2.9		Programmwahl Anwahl des Programms mit der Zehnertastatur. Anschließend Taste 2.9 1 1. ASS.-Durchlauf 2 2. ASS.-Durchlauf 3 1. ASS.-Durchlauf Nachtrag 4 Ausgabe Adreßbuch
2.2		Start der Programme (Taste ←—)
1.1	gesetzt:	STOP-Taste In dieser Stellung kann eine neue Programmwahl erfolgen.
1.2	gesetzt:	keine Ausgabe des Objektprogramms auf Lochkarte.
1.3	gesetzt:	kein Ausdruck des Maschinencode-Programms im Protokoll.
1.4	gesetzt:	keine Prüfung der Folge-Nummer.
1.6	gesetzt:	keine Auflistung des symbolischen Programms
1.9	gesetzt:	Ausgabe des Objektprogramms auf Lochstreifen. Taste 1.2 darf nicht gesetzt sein.

	ASSEMBLER	

Bemerkungen:

STOP-Taste Ist die STOP-Taste eingerastet, wird nach Druck auf die Taste 2.2 jeweils eine Lochkarte verarbeitet. Beim Ausdruck der "nicht definierten" Namen und bei der Ausgabe des Adreßbuches bewirkt die STOP-Taste einen Halt.

Rote Lampe Im Falle eines Lesefehlers beim Kartenleser und -stanzer leuchtet die rote Lampe. Durch Drücken der C-Taste wird die rote Lampe gelöscht und der Vorgang wiederholt.

Gelbe Lampe Die gelbe Lampe zeigt an, daß die Kartenzuführung des Lesers/Stanzers leer ist. Karten nachlegen und Taste I drücken beseitigen den Fehler.

END Nach Ausdruck der Anweisung END erfolgt vom Programm ein automatischer Funktionstasten-Auswurf. An dieser Stelle, ebenfalls beim Programmanfang (vor der Programmwahl), haben die Tasten 1.1 und 1.2 Sonderfunktionen.

Taste 1.1 bewirkt 1-zeiligen Papiervorschub,
Taste 1.2 bewirkt 4-zeiligen Papiervorschub.

Mit diesen Sonderfunktionen kann zum Anfang einer neuen Seite transportiert werden.

4.1.5. Bedienungsanleitung zum ASS-PASS I

Arbeitsgang	Taste	Kommentar
1	V	muß zum Assemblieren immer gesetzt sein.
2	—	Quellprogramm in den Kartenleser legen
3	C ⇦,	Einschalten (Monitorebene)
3.1.	(1.1)	Zeilenschaltung 1-zeilig
3.2.	(1.2)	Zeilenschaltung 4-zeilig
4	1	Internationale Zehnerblock-Tastatur
5	(2.9)	Es erfolgt der Ausdruck 01 ASS-PASS I Das Programm verharrt in einer Warteschleife. Es kann bei Arbeitsgang 4 erneut begonnen werden.
6	$\overleftarrow{(2.2)}$	Start des Programms (MLAR) Das Quellprogramm wird eingelesen.
6.1.	C	Rote Lampe leuchtet bei Lesefehler; ggf. korrigieren
6.2.	I	(Taste neben grüner Lampe, auch "F" genannt) Gelbe Lampe leuchtet, wenn das Kartenzuführungsfach leer ist.

Besonderheiten der Funktionstasten sind zu beachten (4.1.4.).

	ASSEMBLER	

4.2. Zweiter Durchlauf

4.2.1. Einleitung

Im zweiten Durchlauf werden erneut alle Karten des Quellprogramms eingelesen. In dieser Phase werden die OP-Codes und Operanden übersetzt. Es werden die folgenden Funktionen ausgeführt:

festgelegte Arbeitsabläufe

- Formale Prüfung aller Statements mit Fehlerprotokoll
- Kennzeichen Stern * bei Befehlen mit modifizierbaren Operanden
- Kennzeichen Blockumschaltung BL, wenn die Adresse eines anderen Blocks angesprochen wird.

wahlweise Arbeitsabläufe

- Symbolischer Ausdruck der Statements
- Auflistung des Objekt-Programms mit Adreßangabe
- Stanzen des Objekt-Programms auf Lochkarten (Objektkarten).
- Prüfung der Folgenummern.

4.2.2. Festgelegte Arbeitsabläufe

4.2.2.1. Formale Prüfung aller Statements mit Fehlerprotokoll

analog ASS→PASS I (4.1.2.1.).

4.2.2.2. Kennzeichen Stern

Der Stern besagt, daß die Adreßteile dieser Befehle modifizierbar sind. Alle Befehle, die mit einem Stern versehen sind, werden vom ASSEMBLER auf etwaige Blockumschaltungsbefehle untersucht.

4.2.2.3. Kennzeichen Blockumschaltung

Kennzeichen BL wird nur in Verbindung mit dem Kennzeichen Stern gedruckt. Es wird geprüft, ob sich die Adresse im gleichen Block befindet. Wird eine Adresse in einem anderen Block angesprochen, wird dem Kennzeichen Stern ein rotes BL hinzugefügt. Der Programmierer muß vor diesem Befehl eine Blockumschaltung einfügen. Auch wenn ein Blockumschaltbefehl gegeben wurde, druckt der ASSEMBLER das Kennzeichen BL aus.

4.2.3. Wahlweise Arbeitsabläufe

4.2.3.1. Symbolischer Ausdruck der Statements

Wird diese Auflistung gewünscht, erscheint auf dem Protokoll die gesamte Information der Lochkarte, also Name, Operation, Operanden und Bemerkung. Die Auflistung erfolgt einzeilig in Schwarz.

4.2.3.2. Auflistung des Objekt-Programms mit Adreßangabe

Dieser Ausdruck umfaßt zwei Angaben

- absolute Speicheradressen
- Interncode der Statements.

	ASSEMBLER	

Die absolute Speicheradresse wird Rot ausgeschrieben, daneben der Interncode. War ein Statement fehlerhaft, wird der Interncode nicht gedruckt. Dafür erscheint an seiner Stelle der Fehlerschlüssel in Rot.

Der Interncode der Statements gibt in übersichtlicher Schreibweise den Inhalt der 18 bit eines Maschinenwortes an.

Er ist aufgeteilt in:

6 bit	OP-Code	Angaben von 0.0 bis 3.15
1 bit	Indexbit	Angabe 1 oder Blank
3 bit	AD_l	Angabe von 0 bis 7
4 bit	AD_m	Angabe von 0 bis 15
4 bit	AD_r	Angabe von 0 bis 15.

Aufgelistet sieht die Zeile folgendermaßen aus:

```
1  0  0   0  1   0  3  0
1  0  1   2 15   2  1  2
└──┘       └──┘      └──┘
Adresse    Befehl im Interncode
```

Unmittelbar hinter dem AD_r steht bei den Befehlen ein roter Stern *, bei denen im Operanden ein symbolischer Name als Adresse eines Befehls angegeben ist. Ausgenommen sind Namen, die in einer Wertzuweisung definiert wurden.

4.2.3.3. Stanzen des Objekt-Programms auf Lochkarten

Ist ein Programm formal und logisch fehlerfrei oder möchte man den augenblicklichen Stand festhalten, gibt der ASSEMBLER die Mög-

lichkeit, das Programm auf Lochkarten zu stanzen. Die gewonnenen Lochkarten, die Objektkarten genannt werden, haben einen anderen Aufbau als die Lochkarten des Quellprogramms.

4.2.3.3.1. Aufbau der Objektkarten

In einer Objektkarte werden 8 Befehle im Maschinencode mit Angaben zur Adressierung und der Prüfsumme abgelocht.

Die Spalten haben folgende Bedeutung:

Spalte 1	Blocknummer 0 - 7	
Spalten 2 - 6	Blockadresse 0 - 2047	
Spalten 7 - 15	1. Befehl	8 Befehle
Spalten 16 - 24	2. Befehl	
Spalten 25 - 33	3. Befehl	
Spalten 34 - 42	4. Befehl	
Spalten 43 - 51	5. Befehl	
Spalten 52 - 60	6. Befehl	
Spalten 61 - 69	7. Befehl	
Spalten 70 - 78	8. Befehl	
Spalten 79 - 80	Prüfsumme modulo 100 der Spalten 1 - 78	

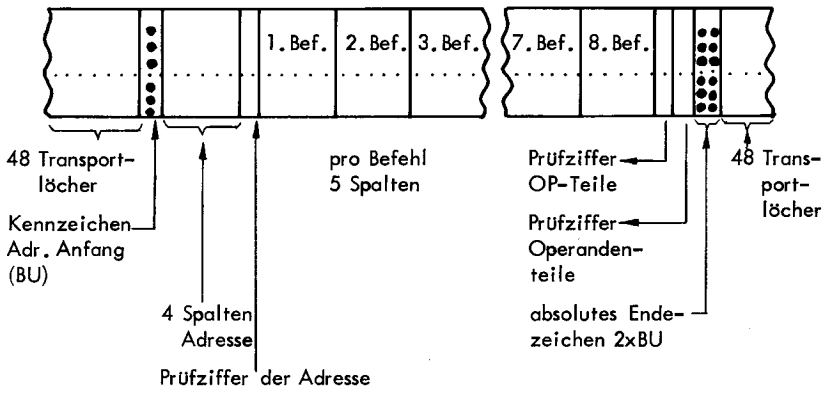
Neu in diesem ASSEMBLER ist, daß in der Prüfsumme außer den Befehlen auch die Blocknummer und die Blockadresse berücksichtigt wird.

Die letzte Ziffer der Blockadresse reicht von 0 bis 15. Bei früheren Versionen war nur 0 und 8 möglich. Das hat zur Folge, daß jetzt die Unterteilung von 0 bis 7 und 8 bis 15 bei der Auflistung fortfällt und die Befehle fortlaufend ab jeweiliger Adresse gedruckt werden.

Neun Spalten stehen jedem Befehl zur Verfügung. Es wird immer linksbündig gestanzt. Bei modifizierbaren Befehlen, also den Befehlen, die bei der Auflistung des Maschinencodes durch einen roten Stern '*' gekennzeichnet sind, wird der OP-Teil links um 4 erhöht. Bei der Auflistung wird der echte OP-Code geschrieben.

4.2.3.3.2. Aufbau des Objektstreifens

Der Objektstreifen hat Ähnlichkeit mit der Struktur der Objektkarte. Der Streifen (7 Kanal) enthält pro Satz maximal 8 Befehle, die mit einer Anfangsadresse in sedezimaler Schreibweise gelocht werden.



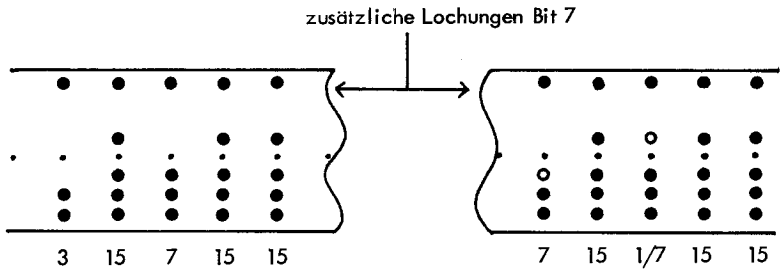
Die vier Spalten der Adresse werden aufgeteilt in:

Blocknummer	Spalte 1
Blockadresse	Spalten 2-4

Die 5 Spalten pro Befehl werden unterteilt in:

Operationscode	2 Spalten (3.15 bzw. 7.15)
Operandenteil	3 Spalten (7.15.15 bzw. 1/7.15.15)

Ist ein Befehl modifizierbar, wird der OP-Teil links um 4 erhöht.



Bei einem indizierten Befehl wird zusätzlich bit 4 (Wertigkeit 8) gelocht.

Vorteil des Objektstreifens: Das Programm kann mit Lochstreifen schneller eingelesen werden als mit Lochkarten. Als Datenträger ist der Streifen handlich, billig und läßt sich platzsparend aufbewahren.

4.2.3.3.3. Benutzung früherer Objektprogramme

Objektprogramme, die durch frühere ASSEMBLER erzeugt wurden, können mit dem jetzigen Lade-Programm nicht verarbeitet, d.h. nicht eingelesen werden.

Es gibt zwei Möglichkeiten, um trotzdem mit diesen Programmen arbeiten zu können:

- Das Programm in einen Speicher mit alter Lese-Routine einlesen. Anschließend den Speicher in einen anderen Rechner stecken. Umgekehrt kann in gleicher Weise verfahren werden.
- Mit einer kleinen Routine können alte Objektprogramme gedoppelt werden. Es entstehen Objektkarten ohne Prüfsumme. Das neue Lade-Programm verarbeitet auch diese Karten.

ASSEMBLER

Auf diesem Umweg können auch von Hand abgelochte Objektprogramme eingelesen werden.

	k	label field Namenfeld	sep	instruction field Befehlsfeld
	1	5		10 15 20 25 30 35
1				PBG RD SPEICHER Bereich für den Inhalt
2				PCH RD 80
3				PBG PC SPEICHER einer Lochkarte
4				PCH PC 78
5				BR /-4

4.2.3.4. Prüfung der Folgenummern

Die Prüfung erfolgt wie im ASS-PASS I in aufsteigender Reihenfolge (4.1.3.1.)

4.2.4. Anmerkung zum ASS-PASS II

Beim ASS-PASS II kann ebenso wie beim ASS-PASS I ein schneller Durchlauf erfolgen, d.h. assemblieren ohne jeden Ausdruck.

Ist der Durchlauf beendet, wird es dem Programmierer durch die Nachricht END bekannt gegeben. Es folgt ggf. der Ausdruck fehlerhafter Statements, die mit Adresse und Fehlerschlüssel ausgegeben werden. War z.B. ein Name nach dem 1. Durchlauf als UNDEFINED SYMBOL gemeldet worden, wird er im 2. Durchlauf überall dort herausgeschrieben, wo dieser Name tatsächlich aufgetreten ist.

4.2.5. Bedienungsanleitung zum ASS-PASS II

Arbeitsgang	Taste	Kommentar
1	V	muß zum Assemblieren immer gesetzt sein.
2	—	Quellprogramm in den Kartenleser legen
3	C⇦,	Einschalten (Monitorebene)
3.1.	(1.1)	Zeilenschaltung 1-zeilig
3.2.	(1.2)	Zeilenschaltung 4-zeilig
4	2	Internationale Zehnerblock-Tastatur
5	(2.9)	Es erfolgt der Ausdruck 02 ASS-PASS II. Das Programm verharrt in einer Warteschleife. Es kann bei Arbeitsgang 4 erneut begonnen werden.
6	$\overleftarrow{(2.2)}$	Start des Programms (MLAR) Das Quellprogramm wird eingelesen.
6.1.	C	Rote Lampe leuchtet bei Lesefehler; ggf. korrigieren
6.2.	I	(Taste neben grüner Lampe, auch " F-Taste" genannt) Gelbe Lampe leuchtet, wenn das Kartenzuführungsfach leer ist.

Besonderheiten der Funktionstasten sind zu beachten (4.1.4.).

4.3. Nachtrag zum ersten Assembler Durchlauf

4.3.1. Einleitung

Nachträge haben die Aufgabe, zusätzliche Angaben zum Programm zu geben. Nachträge können in beliebiger Anzahl gemacht werden.

4.3.2. Festgelegte und wahlweise Arbeitsabläufe

Die Arbeitsabläufe sind mit dem ASS-PASS I identisch (4.1.2. bis 4.1.3.2.).

4.3.3. Namendefinition

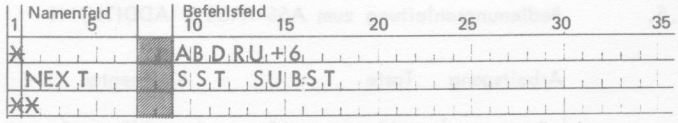
Hauptsächlich ist der Nachtrag dazu gedacht, nicht definierte Namen, die im Anschluß an den ASS-PASS I ausgedruckt werden, nachträglich zu definieren. Man erspart dabei einen nochmaligen ersten Durchlauf.

Wie ein Nachtrag gemacht wird, läßt sich am besten an einem Beispiel erläutern.

k	labelfield Namenfeld	sep	instruction field Befehlsfeld
1	5		10 15 20 25 30 35
1	ABDRU		TT TAB9
2			LF LEP UP1
3			SM1 WPW
4			BR1 NEXT
5			SM1 R11
6			BR1 /-5
7	(NEXT)		SST SUBST

Der letzte Befehl SST, SUBST soll den fehlenden Namen NEXT im Nachtrag erhalten.

Zunächst muß der Befehlszählerstand auf die gewünschte Adresse gesetzt werden. Als Ausgangspunkt dient der letzte oder nächstfolgende symbolische Name. Mit der Adreßrechnung läßt sich jede beliebige Adresse ansprechen. Im Beispiel erhält der Befehl SST, SUBST die Adresse ABDRU+6.



Der anschließend definierte Name wird in das Adreßbuch übernommen. Das Endesymbol * * schließt den Nachtrag ab.

4.3.4. Programmteile als Nachtrag

Es gibt eine weitere Möglichkeit den Nachtrag zum ASS-PASS I zeitsparend einzusetzen.
Alle größeren Programme bestehen aus mehreren, in sich abgeschlossenen, logisch vollständigen Programmteilen. Diese Programmteile können einzeln übersetzt, ggf. korrigiert und zum Test mit dem bereits bestehenden Programm verknüpft werden.

Von einem Programm sind z.B. bereits 6000 Befehle assembliert und liegen als Objektkarten vor. Dieser Teil ist mit A bezeichnet. Ein weiterer Programmabschnitt B liegt als Quellprogramm vor. Dieser Teil soll assembliert und zusammen mit Teil A getestet werden.

In folgender Reihenfolge wird verfahren:

- Schneller ASS-PASS I mit dem Quellprogramm A
- Nachtrag zu ASS-PASS I mit dem Quellprogramm B
- ASS-PASS II mit Anfangsadresse von Programmteil B

Ein schneller ASS-PASS I mit dem Programmteil A ist notwendig, um ein Adreßbuch zu erhalten. Denn in den meisten Fällen wird zwischen den beiden Programmteilen A und B in beiden Richtungen gesprungen.

	ASSEMBLER	

4.3.5. Bedienungsanleitung zum ASS-PASS I ADDITIONS

Arbeitsgang	Taste	Kommentar
1	V	muß zum Assemblieren immer gesetzt sein.
2	—	Quellprogramm in den Kartenleser legen
3	C⇦ ,	Einschalten (Monitorebene)
3.1.	(1.1)	Zeilenschaltung 1-zeilig
3.2.	(1.2)	Zeilenschaltung 4-zeilig
4	3	Internationale Zehnerblock-Tastatur
5	(2.9)	Es erfolgt der Ausdruck 03 ASS-PASS I ADDITIONS Das Programm verharrt in einer Warteschleife. Es kann bei Arbeitsgang 4 erneut begonnen werden.
6	⟵ (2.2)	Start des Programms (MLAR) Das Quellprogramm wird eingelesen.
6.1.	C	Rote Lampe leuchtet bei Lesefehler; ggf. korrigieren
6.2.	I	(Taste neben grüner Lampe, auch "F-Taste" genannt) Gelbe Lampe leuchtet, wenn das Kartenzuführungsfach leer ist.

Besonderheiten der Funktionstasten sind zu beachten (4.1.4.1.).

4.4. Ausgabe des Adreßbuches

4.4.1. Einleitung

Der Ausdruck des Adreßbuches soll dem Programmierer die Testarbeit erleichtern. Der Ausdruck ist in zwei Gruppen unterteilt:

- NAMED CONSTANT Liste 1
- LABEL Liste 2

Die Listen erscheinen auf getrennten Seiten.

4.4.1.1. NAMED CONSTANT

Die Ausgabe des Adreßbuches wird durch den Ausdruck

04 SYMBOL - TABLE
NAMED CONSTANT

eingeleitet.
In dieser Liste werden alle Namen, die in Wertzuweisungen definiert worden sind, in der Reihenfolge ihres Auftretens ausgedruckt.

Die Namen werden Rot und die Werte Schwarz einzeilig untereinander geschrieben.
Der Ausdruck des Wertes wird als Befehlswort angegeben.

Die Wertzuweisung
NAME 13
erscheint in der Liste als
NAME 0 0 0 013

Ist diese Liste beendet erfolgt ein Zeilenvorschub bis zur nächsten Seite.

	ASSEMBLER	

4.4.1.2. LABEL

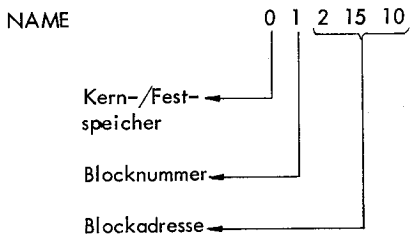
Diese Liste wird mit dem Ausdruck

SYMBOL - TABLE
LABEL

eingeleitet.

Sie enthält alle definierten Namen des Programmes. Die Namen werden in der Reihenfolge ihres Auftretens Rot ausgedruckt. Anschließend werden die Adressen Schwarz ausgeschrieben. Die Adresse enthält die Angabe Kernspeicher = 0 oder Festspeicher = 1, die Blocknummer 0 bis 7 und die Blockadresse 0 bis 2047.

Die Namen werden einzeilig untereinander geschrieben.



Das Symbol NAME ist im Kernspeicherblock 1 auf die Adresse 2.15.10 definiert.

Der Ausdruck END schließt die Ausgabe des Adreßbuches ab.

4.4.1.3. Anwendungsbeispiele

Es werden drei Beispiele behandelt, die sich auf die Anwendung der Assemblerlisten beziehen

- Verbindung von Teilprogrammen
- Benutzung vorgegebener Programmteile die gefädelt sind
- Überprüfung des Definitionsteils

4.4.1.3.1. Verbindung von Teilprogrammen

Ein Programmteil A liegt in Objektkarten vor und ist bereits gefädelt. Teil B soll assembliert werden und zusammen mit Teil A getestet werden. Teil B beinhaltet Sprungbefehle nach Teil A.

Bevor Teil B assembliert wird, müssen alle angesprochenen Adressen aus Teil A in Form von Wertzuweisungen definiert werden.

Ist Teil B formal fehlerfrei und als Objektprogramm vorhanden, kann getestet werden.

Kommt im Verlauf des Programmteils B der ASSEMBLER auf einen Sprung in den Teil A, setzt die Wertzuweisung den Befehlszähler auf die definierte Adresse im angegebenen Block.

4.4.1.3.2. Benutzung vorgegebener Programmteile die gefädelt sind

Es kommt oft vor, daß in sinnverwandten Programmen die gleichen Routinen benutzt werden. Diese Routinen sind meistens in sich geschlossene, logische Programmteile, die sich mit einigen Parametern leicht in ein Programm einfügen lassen.

Die Routine verlangt ganz bestimmte Anfangs- und Absprungadressen, denn sie ist absolut adressiert und liegt in der Programmbibliothek als Objektprogramm vor oder ist bereits gefädelt.

Der Programmierer muß also einen Bereich bis zu der vorgegebenen Adresse freihalten. Es genügt ein Blick ins Adreßbuch, um festzustellen, wie weit das eigene Programm gehen darf, wo die einzufügende Routine beginnt und endet und wo das eigene Programm mit der entsprechenden Speicherwortzuweisung erneut anfängt.

Muß das eigene Programm, das vor der Routine beginnt, nachträglich um einige Befehle erweitert werden, müßten sich die folgenden Adressen verschieben. Da die Routine unverschiebbar ist, müssen Befehle, die bisher vor der Routine standen, hinter die Routine gesetzt werden. Diese Korrekturen lassen sich mit Hilfe des Adreßbuches schnell und sicher durchführen.

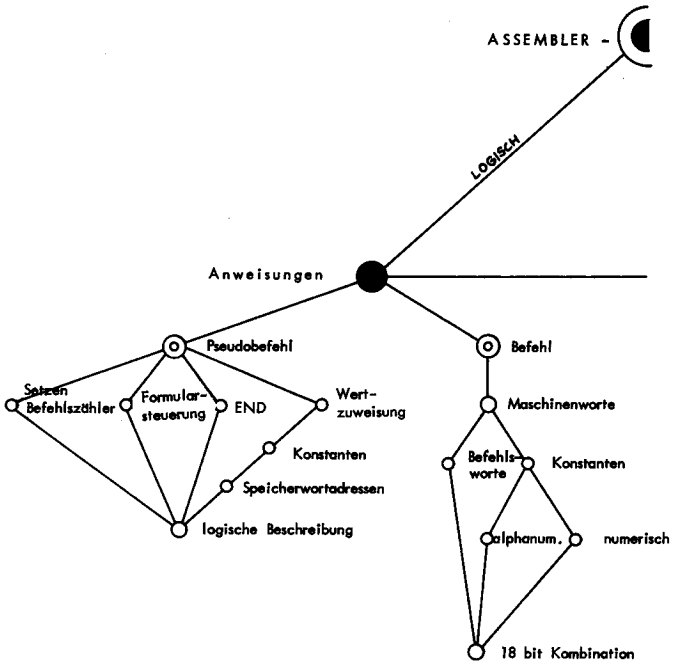
4.4.1.3.3. Überprüfung des Definitionsteils

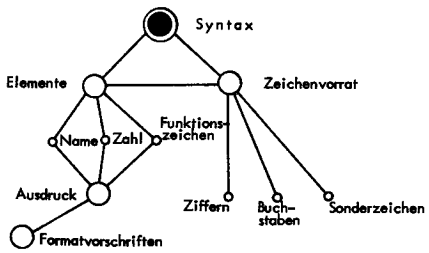
Unter A.3.1. ist das Beispiel mit Befehlsdarstellung mittels einer Wertzuweisung erläutert.

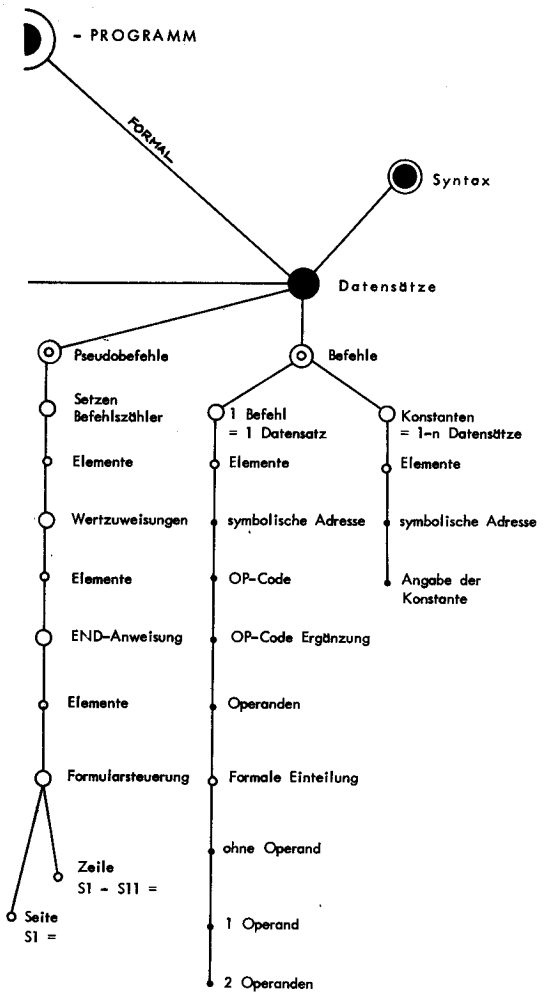
In der Liste 1 NAMED CONSTANT findet man schnell unter dem definierten Namen DVB den Aufbau des Befehls. Hier läßt sich leicht die richtige Zuordnung prüfen.

A N H A N G

ASSEMBLER


ASSEMBLER -
LOGISCH
Anweisungen
Pseudobefehl
Befehl
Setzen Befehlszähler
Formular-steuerung
END
Wert-zuweisung
Konstanten
Speicherwortadressen
logische Beschreibung
Maschinenworte
Befehls-worte
Konstanten
alphanum.
numerisch
18 bit Kombination


Syntax
Elemente
Zeichenvorrat
Name
Zahl
Funktions-zeichen
Ausdruck
Ziffern
Buch-staben
Sonderzeichen
Formatvorschriften


- PROGRAMM
FORMAL
Syntax
Datensätze
Pseudobefehle
Befehle
Setzen Befehlszähler
Elemente
Wertzuweisungen
Elemente
END-Anweisung
Elemente
Formularsteuerung
Zeile S1 - S11 =
Seite S1 =
1 Befehl = 1 Datensatz
Elemente
symbolische Adresse
OP-Code
OP-Code Ergänzung
Operanden
Formale Einteilung
ohne Operand
1 Operand
2 Operanden
Konstanten = 1-n Datensätze
Elemente
symbolische Adresse
Angabe der Konstante

A 2. Adreßrechnung

Die Adreßrechnung des ASSEMBLERS erfolgt modulo 11 bit, d.h. die größte darstellbare Adresse innerhalb eines Blockes ist 7.15.15.

Wird in einem Befehl eine symbolische Adresse angegeben, die nicht in einer Wertzuweisung definiert worden ist, druckt der ASSEMBLER im Protokoll einen roten Stern '*' hinter den Maschinencode des Befehls. Diese Befehle werden später vom Lochkarten-Lade-Programm mit der Anfangs-Lade-Adresse modifiziert. Alle symbolischen Namen, die nicht in einer Wertzuweisung definiert worden sind, erhalten intern ein Kennzeichen.

Die Errechnung und Interpretation einer symbolischen Adreßangabe oder eines Ausdruckes geschieht in zwei 12 bit-Zellen. In diesem Zwischenspeicher mit 24 bit errechnet der ASSEMBLER den absoluten Wert einer Adreßangabe. Erst wenn dieser Vorgang abgeschlossen ist, wird eine Aufteilung vorgenommen. Bei einer Adreßangabe, die nicht in einer Wertzuweisung definiert ist, werden 11 bit, bei einer Konstanten 18 bit des Zwischenspeichers übernommen.

Beispiel:

	k	labelfield Namenfeld	sep	instruction field Befehlsfeld
	1	5		10 15 20 25
1	*	MIN1		A1.-A2

ASSEMBLER

A1 und A2 sind die Namen von zwei aufeinanderfolgenden Adressen. Die Differenz von A1 ./. A2 ist minus 1. Errechnet und definiert wird der Ausdruck folgendermaßen:

Der ASSEMBLER sucht im Adreßteil die Namen A1 und A2, errechnet die Differenz und speichert sie in 24 bit.

|1111|1111|1111|1111|1111|1111| 15.15.15.15.15.15.

Dieser Wert soll als Konstante definiert werden. Also werden von rechts nach links 18 bit abgetrennt.

|11|1111|1 111|1111|1111| 3.15.15.15.15.

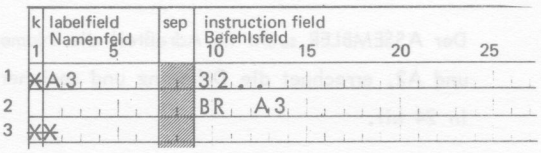
Auf dem Protokoll wird der Ausdruck als Befehlswort niedergeschrieben.

NAMED CONSTANT
MIN1 3 15 1 7 15 15

Die Adreßangabe mit dem Namen einer Wertzuweisung kann zu Fehlern führen, die dem Programmierer auf dem Assemblerprotokoll nicht angezeigt und darum schwer erkannt werden können.

Daher sollte jeder Programmierer wissen, daß die Adressen in 24 bit errechnet und anschließend 18 bit in das Maschinenwort übertragen werden.

Beispiel:



In einer Wertzuweisung wird A3 definiert, d.h. A3 wird ein Wert zugeordnet, der die Adresse 0 im Block 4 anspricht. Für 32.. kann man auch 2.0.0.0 schreiben. Da der symbolische Name A3 in einer Wertzuweisung definiert wurde, erhält er intern kein Kennzeichen.

Das Beispiel wird von dem ASSEMBLER folgendermaßen definiert :

```
NAMED        CONSTANT
A3                    0 2 0 0 0
ASS-PASS II
                        ✕ A 3        32..
0 00 00 1 2 000                      BR A 3

                        ✕ ✕

END
```

Das Beispiel zeigt, daß aus dem Befehl "BR" der Befehl "BR2" geworden ist.

ASSEMBLER

Mit den folgenden Arbeitsschritten verändert der ASSEMBLER den Befehl :

Übernahme des OP-Codes in einen 24 bit Zwischenspeicher.

|0000|0001|0000|0000|0000|0000| 0.1.0.0.0.0 OP-Code

Übernahme der definierten Wertzuweisung aus dem Adreßbuch in einen weiteren Zwischenspeicher.

|0000|0000|0010|0000|0000|0000| 0.0.2.0.0.0 32..

Verknüpfungen der beiden Zwischenspeicher im Sinne des LOGISCHEN ODER.

|0000|0001|0010|0000|0000|0000| 0.1.2.0.0.0

Von diesem Wert werden 18 bit von rechts nach links ins Maschinenwort übertragen, da intern kein Kennzeichen als Name gesetzt ist.

|01|0010|0 000|0000|0000| 1.2.0.0.0.

Die dritte Möglichkeit ist die Angabe eines Namen, der als Symbol für die Adresse eines Befehls steht.

k	labelfield Namenfeld	sep	instruction field Befehlsfeld
1	5		10 15 20 25
1	ANF.		MV EING.,SPEICH.
2			
3			
4			BR ANF

ANF ist der symbolische Name für einen Befehl und soll die Adresse 1.4.3.15. haben. Im folgenden Programmablauf soll auf diese Adresse gesprungen werden.

	ASSEMBLER	

Bei der Übernahme des Symbols ANF ins Adreßbuch wird intern ein Kennzeichen gesetzt und ANF die absolute Adresse 1.4.3.15. zugeordnet.

Der ASSEMBLER interpretiert den Befehl BR,ANF folgendermaßen:
Zunächst werden in gewohnter Weise der OP-Code und die absolute Adresse von ANF in Zwischenspeicher übertragen.

OP-Code BR
|0000|0001|0000|0000|0000|0000| 0.1.0.0.0. 0

Adresse von ANF
|0000|0000|0000|0100|0011|1111| 0.0.1.4.3.15

Aufgrund des internen Kennzeichens trennt der ASSEMBLER von rechts nach links 11 bit der Adreßangabe ab und verknüpft OP-Code und Adresse im Sinne des LOGISCHEN ODER. In diesem Fall kann niemals ein OP-Code zerstört oder verändert werden.

|0000|0001|0000|0000|0000|0000| 0.1.0.0.0. 0
|100|0011|1111| 4.3.15

Logische Verknüpfung ODER
|0000|0001|0000|0100|0011|1111| 0.1.0.4.3.15

Übernahme ins Befehlswort
|01|0000|0 100|0011|1111| 1.0.4.3.15

	ASSEMBLER	

A 2.1. Das Objektprogramm

Der ASSEMBLER des Nixdorf Systems 820 gibt die Möglichkeit, den Maschinencode eines Programmes

- im Protokoll zu drücken,
- in Lochkarten zu stanzen,
- in Lochstreifen zu stanzen.

A 2.1.1. Die Auflistung des Objektprogrammes ist unter 4.3.1.2 beschrieben.

A 2.1.2. Das Stanzen des Objektprogrammes in Lochkarten und der formale Aufbau der Objektkarten ist unter 4.3.1.3 ff behandelt worden. An dieser Stelle soll auf den Inhalt der Objektkarten und das Stanzen eingegangen werden.

Objektkarten werden aus einem Puffer gestanzt. In den Puffer wird als erstes die Nummer des Blockes übernommen, der gerade bearbeitet wird. Es folgt die Adresse des Befehls, der als erster in die Karte übernommen wird. Der ASSEMBLER prüft jeden Befehl daraufhin ab, ob seine interne Verschlüsselung Lochungen im Lochkartencode erzeugt. Ist das der Fall, wird der Befehl in den Puffer übernommen und der Befehlszähler um 1 erhöht. Erzeugt ein Befehl keine Lochung, das ist immer dann der Fall, wenn ein Befehl ein Error-Kennzeichen erhält, wird der Inhalt des Puffers in die Karte gestanzt.

Ist die Karte gestanzt, wird der Puffer gelöscht, der Befehlszähler um 1 erhöht und die Adresse einschließlich Blocknummer in den Puffer übertragen. Die letzte Befehlsadresse der gestanzten Karte und die Anfangsadresse der aufzubereitenden Karte ist in diesem Falle nicht fortlaufend. Stellt der ASSEMBLER fest, daß auch der erste Befehl der neuen Karte keine Lochungen erzeugt, wird automatisch der Befehlszähler erhöht und die neue Adresse in die Spalten 2 bis 7 des Puffers übertragen. Dieser Vorgang wiederholt sich solange, bis ein Befehl Lochungen im Lochkartencode erzeugt. Es gibt also niemals Objektkarten, die keinen Befehl beinhalten. Die Befehle stehen bündig hintereinander und es tritt niemals der Fall ein, daß zwischen zwei Befehlen 9 Spalten der Lochkarte freibleiben. Welche Auswirkungen dieses Verfahren hat, zeigt sich deutlich bei dem Lade-Programm.

ASSEMBLER

Die folgenden Karten sollen gelesen werden.

BL	Bl-Adr.	1. Bef. Adr.	2. Bef. Adr.	3. Bef. Adr.			12
0	00999	999	1000	1001			

BL	Bl-Adr.	1. Bef Adr.	2. Bef. Adr.	3. Bef. Adr.	4. Bef. Adr.		8. Bef. Adr.	48
0	01004	1004	1005	1006	1007		1011	

Die Karte mit der Blockadresse 999 enthält drei Befehle. Es folgen zwei Befehle, die keine Lochungen erzeugen. Die nächste Karte beginnt daher mit der Adresse 1004.

Liest das Lade-Programm die erste aufgezeichnete Karte, wird der erste Befehl in den Block 0 auf Adresse 999 abgespeichert, der zweite auf Adresse 1000, der dritte auf Adresse 1001. Der Befehlszähler wird wieder um 1 erhöht und steht auf Adresse 1002. Das Ladeprogramm erkennt, daß die Befehle 4 bis 8 dieser Karte nicht vorhanden sind. In diesem Falle wird der augenblickliche Speicherinhalt dieser Adressen nicht verändert. Die Blockadresse der nächsten Karte ist 1004. Der Befehlszähler wird auf 1004 gesetzt und die abgelochten Befehle wer-

den analog gespeichert. Die Befehle mit der Adresse 1002 und 1003 bleiben in ihrer ursprünglichen Form im Speicher erhalten.

A 2.1.3. Stanzen des Objektprogramms in Lochstreifen

Der formale Aufbau des Lochstreifens ist unter 4.2.3.3.2. beschrieben. Der Inhalt des Objektstreifens und das Stanzen entspricht den Angaben der Objektkarten.

A 3. Weitere Möglichkeiten des ASSEMBLERS

Der ASSEMBLER bietet folgende weitere Darstellungsmöglichkeiten innerhalb eines Datensatzes.

- Darstellung eines Befehls durch einen Namen
- Darstellung und Benutzung fremder Befehlssymbole
- Darstellung von numerischen Konstantentabellen mit festen Symbolen

A 3.1. Darstellung eines Befehls durch einen symbolischen Namen

Es ist möglich, mit einem symbolischen Namen einen vollständigen Befehl darzustellen. Wird in einem Programm ein Befehl häufig benutzt, kann anstelle des Befehls ein Name im Befehlsfeld angegeben werden.

Beispiel an einem Druckvorbefehl

	k	labelfield Namenfeld	sep	instruction field Befehlsfeld
	1	5		10 15 20 25
1				EDF REDL STR 4 FTS

Im Maschinencode wird dieser Befehl in 18 bit dargestellt und sieht folgendermaßen aus:

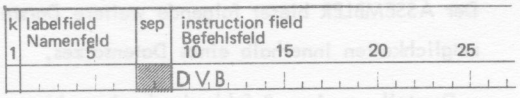
3.6.7.6.4

Dieser Befehl soll im Quellprogramm durch den Namen DVB ersetzt werden.

In einer Wertzuweisung muß DVB definiert werden.

	k	labelfield Namenfeld	sep	instruction field Befehlsfeld
	1	5		10 15 20 25
1	*	DVB		EDF REDL STR 4 FTS

Wird dieser Befehl im Quellprogramm benutzt, wird im Befehlsfeld DVB angegeben.



A 3.2. Darstellung und Benutzung fremder Befehlssymbole

Will ein Benutzer die Symbole der Nixdorf Assembler Sprache nicht verwenden, weil ihm die Symbole einer anderen Programmiersprache oder eines anderen Herstellers schon geläufig sind oder mehr zusagen, kann er seinen eigenen symbolischen Code benutzen.
Er schreibt anstelle von

"MV"	übertrage Zeichen	"MOVE"	oder
"CP"	vergleiche	"COMP"	usw.

In diesen Fällen ordnet er seine Symbolik dem internen Maschinencode des Nixdorf Systems zu. Das geschieht in Form einer Wertzuweisung.

Beispiel:

```
* MOVE    0.3.0.0.0
* COMP    0.13.0.0.0
```

In dieser Weise müssen alle Befehle des ASSEMBLERS in einer Befehlstabelle definiert werden.

A 3.3. Darstellung von numerischen Konstantentabellen

In einem Datensatz lassen sich eine oder zwei nume-

rische Konstanten darstellen. In Abhängigkeit der Definition befindet sich die Konstante rechts oder links im Befehlswort.

A3.3.1. Konstante als Wertzuweisung definiert

*A1 12 = 0.0.0.0.12

Die Konstante steht rechtsbündig im Befehlswort und kann maximal den Wert 261.143 annehmen.

A3.3.2. Konstantendefinition mit festen Symbolen

Mit den Symbolen

Z 0 bis Z 49, Wert 0 - 49
F 50 bis F 99, Wert 50 - 99
H100 bis H127 Wert 100 - 127

können die Konstanten 0 bis 127 im Assemblercode angelegt werden. Sie belegen sieben bit links im Befehlswort

Z12 = 0.6.0.0.0

A3.3.3. Kombination aus Wertzuweisung und Definition mit festem Symbol

Der ASSEMBLER bietet die Möglichkeit zwei unabhängige Konstanten in einem Datensatz zu speichern. Die Konstanten können einzeln angesprochen werden.

Die Konstante als Wertzuweisung definiert, kann mit dem Befehl ICA indirekte Konstante nach Speicherwort A, angesprochen werden.

Wurde die Konstante durch ein festes Symbol erzeugt, kann sie durch den Befehl 'OPX' OP- und AD_i-Teil nach Indexregister, angesprochen werden.

Beispiel

	k	labelfield Namenfeld	sep	instruction field Befehlsfeld
	1	5		10 15 20
1	X	B 1		7 4
2		B 2		Z 1 4 B 1

Die 18 bit Kombination mit der Adresse B2 ist jetzt folgendermaßen belegt:

B2 |00|0LLL|0|000|0L00|L0L0|

OPi l m r

0. 7. 0. 0. 4. 10.

ASSEMBLER

Adresse bezeichnet eine bestimmte Stelle im Speicher.
Einer solchen Adresse kann ein symbolischer Name zugeordnet werden. Der ASSEMBLER errechnet die relativen Adressen zu den vom Benutzer vereinbarten Anfangs-Assemblier-Adressen und weist sie den symbolischen Namen zu.

Adreßrechnung des ASSEMBLERS Nixdorf Serie 820 erfolgt modulo 11 bit. Innerhalb dieses Bereiches können Adressen durch Addition zugehöriger Blockadressen, bzw. Anfangsadressen bei Laderoutinen, modifiziert werden (relative Adressen).

Adreßteil ist der Teil eines Befehls, der angibt, an welcher Stelle die zu verarbeitende Information im Speicher zu finden ist.

Anweisung ⟶ Befehl, Statement.

ASSEMBLER (ASS) ist ein Teil der Software des Herstellers. Es ist das Übersetzungsprogramm, das die in Assembler-Sprache geschriebenen Quellprogramme in die eigentliche maschineninterne Sprache übersetzt, d.h. in Objektprogramme umwandelt.

Assembler ist eine symbolische Programmiersprache, die zur maschinennahen Programmierung dient. Unter Zuhilfenahme dieser Sprache können alle Operationsmöglichkeiten des Systems 820 realisiert werden.

Assemblieren ist der Ablauf des ASSEMBLERS. In zwei Phasen wird aus einem symbolisch formulierten Programm ein Maschinenprogramm erzeugt.

Ausdruck ist die Zusammenfassung von Namen und Zahlen durch Verknüpfungszeichen. Auch jeder einzelne Name oder Wert ohne Verknüpfungszeichen wird als Ausdruck bezeichnet.

Befehl ist die kleinste elementare Einheit eines Programms (Befehlswort). Die Mehrzahl der Befehle hat zwei Teile: den Operationsteil, der die verlangte Operation spezifiziert, und den Operandenteil, der meist in Form einer Adresse die Speicherstelle angibt, auf die sich der Operationsteil bezieht.

Blockadresse ist eine Adresse innerhalb eines bestimmten Speichermediums. Ein Block des Nixdorf Systems 820 umfaßt 2048 Befehlsworte (18 bit Maschinenworte), die mit den Blockadressen 0 bis 2047 angesprochen werden können.

Buchstaben sind im allgemeinen alle Zeichen des lateinischen Alphabets, also die Zeichen A bis Z.

Datensatz ist eine sachliche und logische Informationseinheit, die unter einem Ordnungsbegriff zusammengefaßt ist. Auf dem ASSEMBLER bezogen entspricht ein Datensatz dem Inhalt einer Zeile des Codierformulars oder dem Inhalt einer Lochkarte.
Eine Ausnahme bilden Konstanten, die als alphanumerische Texttabellen fortlaufend gespeichert werden und aus mehreren Datensätzen d.h. mehreren Lochkarten, bestehen können.

Elemente des ASSEMBLERS sind Name , Zahl und Funktionszeichen. Mit diesen Bausteinen lassen sich alle Funktionen der Syntax und des ASSEMBLERS ausdrucken.

Feste Symbole werden zur Konstantendefinition benutzt, wenn die Konstante den Operationsteil und den AD_i-Teil belegen soll Die Konstante kann maximal den Wert 127 annehmen.

Interncode bezeichnet die duale Verschlüsselung eines Maschinenwortes.

Konstante ⟶ Literal

Literal. Man spricht von numerischen und alphanumerischen Literalen. Sie sind während eines Programms unveränderlich und werden formal von den Zeichen ** eingeschlossen.

Maschinenbefehle setzen sich aus dem Operationsteil und dem Operandenteil zusammen. Der Operationsteil enthält den numerischen Operationscode. Der Operandenteil enthält Indexbit und Adreßteil. Der Adreßteil unterteilt sich in links, mitte und rechts und kann den Wert ≤ 2047 enthalten.
Merkmal ⟶ symbolische Adresse.

Maschinenwort. Im Speicher eines Computers werden Folgen von Zeichen (Zeichenketten) als Einheit betrachtet. Ein Wort ist eine solche Einheit. In Abhängigkeit der physikalischen Struktur der Speicher (Festspeicher, Kernspeicher) ergeben sich verschiedene Wortlängen.
Ein Wort im Festspeicher besteht aus 18 bit und kann einen Befehl (Befehlswort) oder drei alphanumerische Konstanten (Literale) zu je 6 bit oder zwei numerische Konstanten zu 7 bit und 11 bit enthalten.

Ein Wort im Kernspeicher hat unterschiedliche Länge, bedingt durch den Verwendungszweck des Speichers als Befehls- bzw. Datenspeicher.
Das Betriebsprogramm ist in der Lage, den Kernspeicher als Befehls-

speicher oder als Datenspeicher einzuteilen. Außerdem besteht die Möglichkeit, den Kernspeicher als Befehls- und Datenspeicher einzuteilen.

Als Befehlsspeicher wird der Kernspeicher in 18 bit-Worte (Befehlswort analog Festspeicher) und als Datenspeicher in Worte zu 64 bit (16 Stellen á 4 bit = 1 Speicherwort) unterteilt.

Ein Speicherwort kann 16 numerische Zeichen oder 8 alphanumerische Zeichen im 8 bit-Code bzw. 10 alphanumerische Zeichen im 6 bit-Code beinhalten. Bei alphanumerischer Speicherung kann die Speicherwortgrenze überschritten werden.

Name. Die Syntax bestimmt, daß ein Name mit einem Buchstaben beginnen muß und mindestens aus zwei Zeichen, maximal sechs Zeichen besteht. Namen können durch Verknüpfungszeichen mit Namen und Zahlen verbunden werden, d.h. die Adressen der Namen werden modifiziert.

Objektkarten werden während des zweiten Assembler-Durchlaufs erstellt. Jede Karte enthält maximal acht Befehle im Interncode mit der Anfangsadresse des ersten Befehls und einer Prüfziffer.

Objektprogramm wird das durch den ASSEMBLER übersetzte Quellprogramm genannt. Das Objektprogramm besteht ausschließlich aus Befehlen der Maschinensprache und wird daher oft als Maschinenprogramm bezeichnet.

Objektstreifen werden während des zweiten Assembler-Durchlaufs erstellt. Der Streifen hat die gleichen Funktionen wie die Objektkarten.

Operanden sind Ausdrücke , die das Rechenwerk entgegen nimmt, um sie

mit einer bestimmten Operation zu verknüpfen oder zu verarbeiten. Der Operand wird innerhalb eines Befehls durch den Adreßteil definiert.

Operation ist der durch den Operationsteil eines Befehls definierte Ablauf, den der Computer auszuführen hat. Die Operationen werden unterschieden in Eingabe-, Verarbeitungs-, Übertragungs- und Ausgabeoperationen.

Operationscode (OP-Code) des ASSEMBLERS ist ein mnemotechnischer Ausdruck, der die Art einer Operation bestimmt und im Operationsteil eines Befehls eingetragen wird.

Operationsteil eines Befehls beinhaltet den Operationscode, der die operative Funktion des Befehls spezifiziert.

Programm ist eine logische Folge von Befehlen, die in ihrer Gesamtheit ein Problem zu lösen vermag.

Pseudobefehle sind unechte, vorgetäuschte Befehle, die vom ASSEMBLER nicht in Maschinenworte übersetzt werden. Es sind Anweisungen, die meist nur zur Zeit des Assemblierens wirksam, d.h. vorhanden sind.

Quellprogramm ist der Name für ein Programm, das in einer symbolischen Sprache geschrieben ist und mit Hilfe eines Übersetzers in ein Objektprogramm umgewandelt wird.

Ein Quellprogramm wird auch als symbolisches Programm, Primärprogramm, Ursprungsprogramm oder source program benannt.

Sonderzeichen sind alle Zeichen, die nicht Ziffern oder Buchstaben sind.

Statement ⟶ Befehl.

Symbolische Adresse steht innerhalb eines Befehls. Die Adresse ist nicht in Form einer Maschinenadresse angegeben, d.h. als numerischer Wert, sondern trägt symbolischen Charakter. Die symbolische Adresse, auch Merkmal genannt, wird mit Hilfe des ASSEMBLERS in eine echte Maschinenadresse umgewandet.

Symbolisches Programm ist in einer symbolischen Programmiersprache geschrieben (Assembler). Die Symbolsprache besteht aus leicht merkbaren, mnemonischen Ausdrücken, die es ermöglichen, durch ein Umwandlungsprogramm (ASSEMBLER) mit dem Computer sinngemäß zu arbeiten.

Syntax wird das System von den Regeln einer Sprache bezeichnet. Die Syntax legt fest, wie aus einer gegebenen Anzahl von Grundelementen die zulässigen bzw. gültigen Ausdrücke oder Befehle der Sprache zu bilden sind.

Trennzeichen sind die Zeichen Blank '␣' und Komma ',' . Sie sind absolut gleichwertig.

Wertzuweisung ist ein Pseudobefehl . Durch ihn wird einem Namen ein Wert zugeordnet, der sowohl eine Konstante als auch eine Adresse sein kann.
Wert ⟶ Zahl.

Zahl besteht aus einer oder mehreren Ziffern, maximal bis zu 261 143.

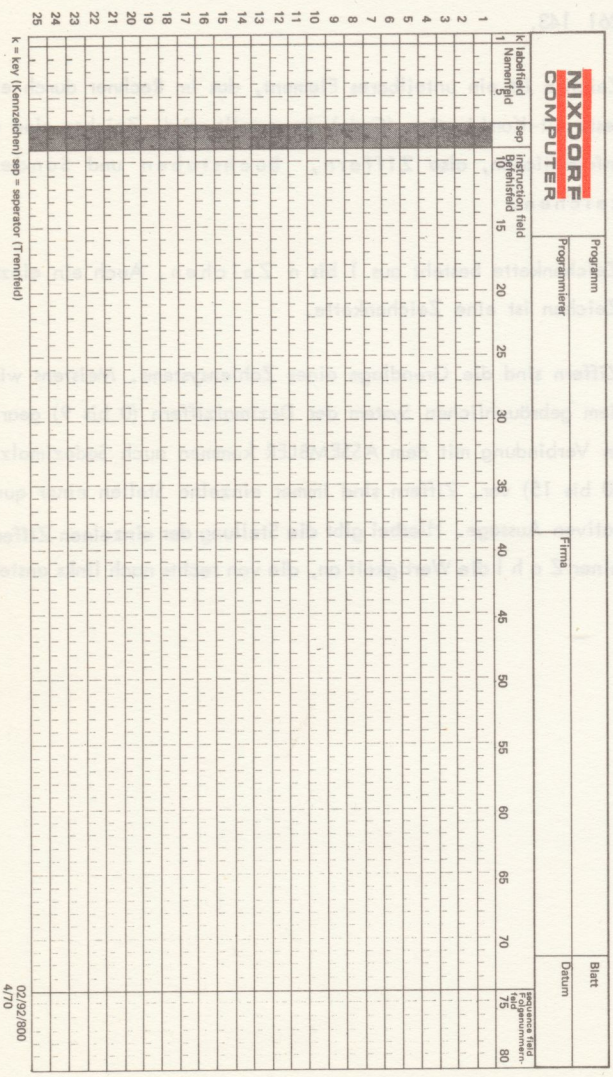
Zeichen ist ein unteilbares Element, das im Rechner durch eine feste Bit-Kombination (Code) dargestellt wird. Zeichen sind digitale Informationen, also Ziffern, Buchstaben und Sonderzeichen.

Zeichenkette besteht aus 1 bis n Zeichen. Auch ein einzelnes Zeichen ist eine Zeichenkette.

Ziffern sind die Grundlage eines Zahlensystems. Meistens wird mit dem gebräuchlichen System der Dezimalziffern (0 bis 9) gearbeitet. In Verbindung mit dem ASSEMBLER kommen auch Sedezimalziffern (0 bis 15) vor. Ziffern sind immer einzelne Stellen einer quantitativen Aussage. Hierbei gibt die Stellung der einzelnen Ziffern in einer Z a h l die Wertigkeit an, die von rechts nach links ansteigt.

Datenverarbeitungssystem 820

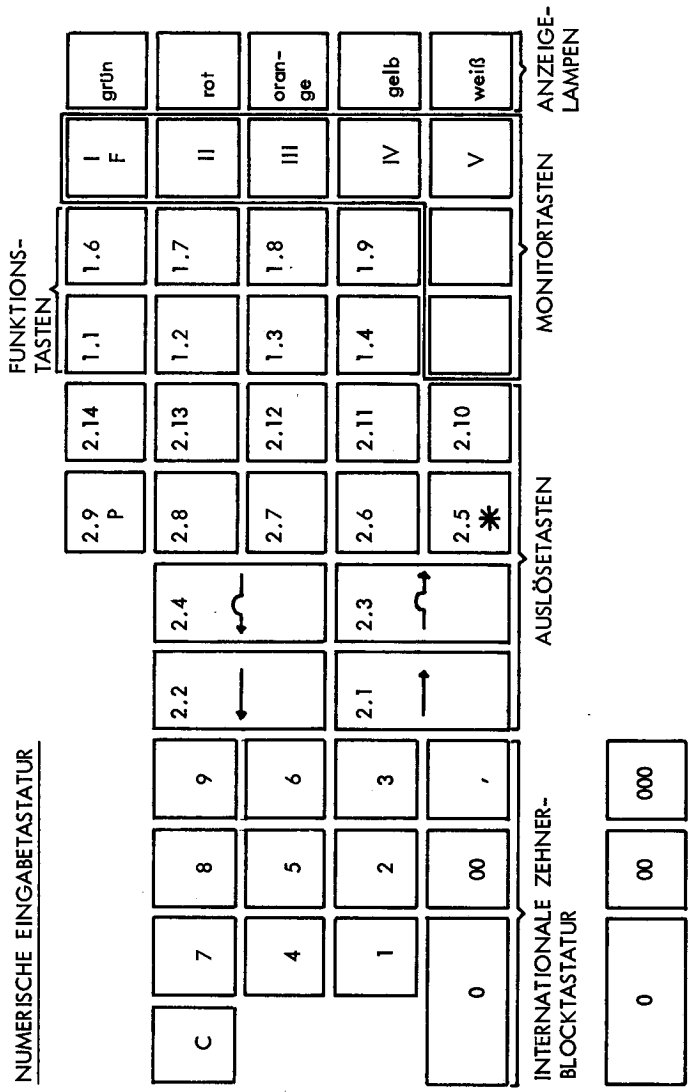
		ASSEMBLER	



NIXDORF SERIE 820

ASSEMBLER

ASSEMBLER

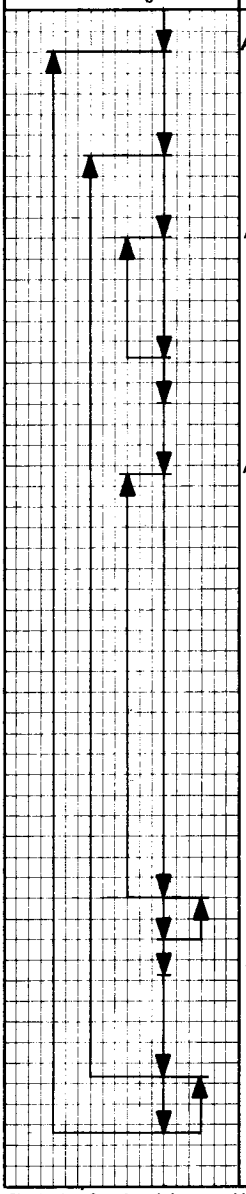

NUMERISCHE EINGABETASTATUR
FUNKTIONS-TASTEN
MONITORTASTEN
AUSLÖSETASTEN
INTERNATIONALE ZEHNER-BLOCKTASTATUR
ANZEIGE-LAMPEN
grün
rot
oran-ge
gelb
weiß
I F
II
III
IV
V
1.6
1.7
1.8
1.9
1.1
1.2
1.3
1.4
2.14
2.13
2.12
2.11
2.10
2.9 P
2.8
2.7
2.6
2.5 *
2.4
2.3
2.2
2.1
9
6
3
,
8
5
2
00
7
4
1
0
C
000
00
0

Anhang 7

Programmablaufplan

NIXDORF COMPUTER

Programm PROGRAMMBEISPIEL	NIXDORF, Geroldstraße 32,	Blatt 1
Programmierer Stolzenberger	Firma Tel.: 24802, App. 21	Datum 28.5.70

Darstellung	Marke	Text	Verweisung
	ANFANG	Fünfmaliger Zeilenvorschub Tabulation auf Position 20 Druck der Überschrift Löschen des Gesamtspeicherwortr Viermaliger Zeilenvorschub Konstante 5 nach Index	
	ANSCHR	Tabulation auf Position 10 Schreibmaschinenfreigabe bis Position 35 Einmaliger Zeilenvorschub Index Minus 1 Index, ungleich Null ? Fünfmaliger Zeilenvorschub Warten auf Eingabe der Artikelnummer (→)	
	ARTNR	Transport in das Arbeitsspeicherwort Transport in das Druckspeicherwort Druckvorbefehl und Druckbefehl für die Artikelnr. Warten auf Eingabe der Menge (→) Transport in das Arbeitsspeicherwort Transport in das Druckspeicherwort Druckvorbefehl und Druckbefehl für die Menge Warten auf Eingabe des Einzelpreises (→) Transport in das Zwischenspeicherwort Transport in das Druckspeicherwort Druckvorbefehl und Druckbefehl für den Einzelpr. Multiplikation von Menge und Einzelpreis Transport der Summe in das Druckspeicherwort Druckvorbefehl und Druckbefehl für die Summe Einmaliger Zeilentransport Addition der Summe in das Gesamtsummen-speicherwort Neue Eingabe verlangt (→) Absummierung verlangt (✱)	
	SUMDRU	Einmaliger Zeilenvorschub Transport Gesamtsumme nach Druckspeicherwort Druckvorbefehl und Druckbefehl für die Gesamt-summe Neue Anschrift verlangt (→) Neue Überschrift verlangt (←)	

Sinnbilder: Strecke - 1 Ausgang, Verzweigung - 2 Ausgänge (linker Ausgang - Bedingung erfüllt/rechter Ausgang - Bedingung nicht erfüllt)

NIXDORF SERIE 820

ASSEMBLER

NIXDORF COMPUTER

Programm: ASSEMBLER PROGRAMMBEISPIEL — Blatt: 1

Programmierer: Stolzenberger — Firma: NIXDORF, Geroldstraße 32, Tel.: 24802, App. 21 — Datum: 29.5.70

	k	labelfield Namenfeld	sep	instruction field Befehlsfeld		sequence field Folgenummernfeld
	1	5		10 15 20 25 30 35 40 45 50 55 60 65 70		75 80
1				AUFGABENSTELLUNG: EINFACHE PREIS-/MENGEN		10
2				BERECHNUNG		20
3						30
4		START		NOP 5,2,3	ANFANGSPARAMETER (KA USW.)	40
5	*			5	SETZE BEF.ZAEHLER ABSOLUT AUF 1. BEF.ADRESSE	50
6				BL 1	SPRUNG NACH BLOCK 1	60
7				BR ANFANG		70
8						80
9	*			START+40,76	SETZE BEF.-ZAEHLER (RELATIV)	90
10						100
11		ANFANG		LF,LEP,UP,SAVPL,5		110
12				TAB 20		120
13				BL 2		130
14				TT UEBERS		140
15				CLR GESSUM		150
16				LF,LEP,UP,SAVPL,4		160
17		ANSCHR		CX 5	SCHREIBMASCHINEN-FREIGABE ZUM DRUCKEN DER	170
18				TAB 10	ANSCHRIFT (5 ZEILEN)	180
19				TW 35		190
20				ZEIL1		200
21				CX,X1,8,-1		210
22				BXU,ANSCHR+1		220
23				2,14,2,8,5	LF LEP UP SAVPL 5	230
24		ARTNR		WT MRAR	WARTEN AUF EINGABE ARTIKEL-NUMMER	240
25				ACC ARB	EINGABE MIT TASTE PFEIL RECHTS	250

k = key (Kennzeichen) sep = seperator (Trennfeld)

NIXDORF SERIE 820

ASSEMBLER

NIXDORF COMPUTER

Programm: ASSEMBLER PROGRAMMBEISPIEL | Blatt: 2

Programmierer: Stolzenberger | Firma: NIXDORF, Geroldstraße 32, Tel.: 24802, App. 21 | Datum: 29.5.70

	k	label field Namenfeld	sep	instruction field Befehlsfeld		sequence field Folgenummernfeld
	1	5		10 15 20 25 30 35 40 45 50 55 60 65 70		75 80
1				MV DRUCK1 ARB	GANZZAHLIG UNGLEICH NULL	260
2				DVB 1	DRUCKVORBEFEHL	270
3				ED 15	DRUCKHAUPTBEFEHL AUS D1	280
4				BL 2	ABSOLUTER SPRUNG ZUM ANFANG DES NAECHSTEN	290
5				BR /-/	BLOCK (ADRESSE 0.0.0)	300
6		MENGE		WT,MRAR	WARTEN AUF TASTE PFEIL RECHTS	310
7				ACC ARB	EINGABE MENGE	320
8				MV ARB, DRUCK1		330
9				DVB 1		340
10				ED 30		350
11		EPREIS		WT,MRAR	WARTEN AUF TASTE PFEIL RECHTS	360
12				ACC ZWISP, 2	EINGABE EINZELPREIS MIT 2 NACHKOMMASTELLEN	370
13				MV ZWISP, DRUCK1		380
14				DVB 2		390
15				ED 2..45		400
16				MLH ARB ZWISP	MULTIPLIZIERE MENGE X EINZELPREIS	410
17				MV ZWISP, DRUCK1		420
18				DVB 3		430
19		GESPR		ED 2..65	DRUCKEN GESAMTPREIS IN ROT	440
20				ZEIL 1		450
21				AD ZWISP, GESSUM		460
22		WARTE		SM1 MRAR		470
23				BL 1		480
24				BR1 ARTNR+1	EINGABE NEUER WERTE	490
25				SM1 MFTS	TASTE STERN	500

k = key (Kennzeichen) sep = seperator (Trennfeld)

NIXDORF SERIE 820

ASSEMBLER

NIXDORF COMPUTER

Programm: ASSEMBLER PROGRAMMBEISPIEL — Blatt: 3

Programmierer: Stolzenberger — Firma: NIXDORF, Geroldstraße 32, Tel.: 24802, App. 21 — Datum: 29.5.70

	k	labelfield Namenfeld	sep	instruction field Befehlsfeld	sequence field Folgenummernfeld
1				BR 1 SUMDRU DRUCKEN DER ENDSUMME	510
2					520
3				BR WARTE	530
4		SUMDRU		LF LEP UP SAVPL 2	540
5				MVH GESSUM DRUCK 1	550
6				DVB 4	560
7				ED 2 65 DRUCKEN GESAMTSUMME	570
8		ENDE		SM1 MRAR	580
9				BL 1	590
10				BR 1 ANSCHR-2 NEUE ANSCHRIFT UND EINGABE NEUER WERTE	600
11				SM1 MLAR	610
12				BL 1	620
13				BR 1 ANFANG	630
14				BR /-6	640
15		UEBERS		*ASSEMBLER/YBLK/PROGRAMMBEISPIEL/YECC*	650
16	*	DVB 1		EDF BLACK BEFEHLS-	660
17	*	DVB 2		EDF ZERO BLACK 1 DARSTELLUNG	670
18	*	DVB 3		EDF ZERO RED 1 DURCH	680
19	*	DVB 4		EDF ZERO RED FTS 1 SYMBOLISCHE	690
20	*	ZEILI		LF LEP UP SAVPL 1 NAMEN (WERTZUWEISUNGEN)	700
21	*	GESSUM		5 DEFINIERE DATENSPEICHERWORT ABSOLUT	710
22	*	ARB		GESSUM+5 DEFINIERE DATENSPEICHERWORT RELATIV	720
23	*	ZWISP		ARB+ARB	730
24	*	DRUCK		1	740
25	**			END-ANWEISUNG	750

k = key (Kennzeichen) sep = seperator (Trennfeld)

```
                                        AUFGABENSTELLUNG: EINFACHE PREIS-/MENGEN-
                                                          BERECHNUNG

 START    NOP 5.2.3                     ANFANGSPARAMETER (KA USW.)
*         5                             SETZE BEF.ZAEHLER ABSOLUT AUF 1. BEF.ADRESSE
          BL 1                          SPRUNG NACH BLOCK 1
          BR ANFANG

*         START+4076                    SETZE BEF.-ZAEHLER (RELATIV)

 ANFANG   LF,LEP,UP,SAVPL,5
          TAB 20
          BL 2
          TT UEBERS
          CLR GESSUM.
          LF,LEP,UP,SAVPL,4
 ANSCHR   CX 5                          SCHREIBMASCHINEN-FREIGABE ZUM DRUCKEN DER
          TAB 10                        ANSCHRIFT (5 ZEILEN).
          TW 35
          ZEIL1
          CX,XI,8..-1
          BXU,ANSCHR+1
          2.14.2.8.5                    LF LEP UP SAVPL 5
 ARTNR    WT MRAR                       WARTEN AUF EINGABE ARTIKEL-NUMMER
          ACC ARB.                      EINGABE MIT TASTE PFEIL RECHTS
          MV DRUCK1 ARB.                GANZZAHLIG,UNGLEICH NULL.
          DVB1                          DRUCKVORBEFEHL
          ED 15                         DRUCKHAUPTBEFEHL AUS D1
          BL 2                          ABSOLUTER SPRUNG ZUM ANFANG DES NAECHSTEN
          BR /-/                        BLOCK (ADRESSE 0.0.0)
 MENGE    WT,MRAR                       WARTEN AUF TASTE PFEIL RECHTS
          ACC ARB.                      EINGABE MENGE
          MV ARB. DRUCK1
          DVB1
          ED 30
 EPREIS   WT,MRAR                       WARTEN AUF TASTE PFEIL RECHTS
          ACC ZWISP. 2                  EINGABE EINZELPREIS MIT 2 NACHKOMMASTELLEN
```

```
380             MV ZWISP. DRUCK1
390             DVB2
400             ED 2..45
410             MLH ARB ZWISP.        MULTIPLIZIERE MENGE X EINZELPREIS
420             MV ZWISP. DRUCK1
430             DVB3
440   GESPR     ED 2..65              DRUCKEN GESAMTPREIS IN ROT
450             ZEIL1
460             AD ZWISP. GESSUM
470   WARTE     SM1 MRAR
480             BL 1
490             BR1 ARTNR+1           EINGABE NEUER WERTE
500             SM1 MFTS              TASTE STERN
510             BR1 SUMDRU            DRUCKEN DER ENDSUMME
530             BR WARTE
540   SUMDRU    LF LEP UP SAVPL 2
550             MVH GESSUM DRUCK1.
560             DVB4
570             ED 2..65              DRUCKEN GESAMTSUMME
580   ENDE      SM1 MRAR
590             BL 1
600             BR1 ANSCHR-2          NEUE ANSCHRIFT UND EINGABE NEUER WERTE
610             SM1 MLAR
620             BL 1
630             BR1,ANFANG
640             BR /-6
650   UEBERS    *ASSEMBLER/YBLK/PROGRAMMBEISPIEL/YECC*
660   *DVB1     EDF BLACK             BEFEHLS-
670   *DVB2     EDF ZERO BLACK 1      DARSTELLUNG
680   *DVB3     EDF ZERO RED 1        DURCH
690   *DVB4     EDF ZERO RED FTS 1    SYMBOLISCHE
700   *ZEIL1    LF LEP UP SAVPL 1     NAMEN (WERTZUWEISUNGEN)
710   *GESSUM   5                     DEFINIERE DATENSPEICHERWORT ABSOLUT
720   *ARB      GESSUM+5              DEFINIERE DATENSPEICHERWORT RELATIV
730   *ZWISP    ARB+ARB
740   *DRUCK1   1
750   **                              END-ANWEISUNG
```

END

SYMBOL - TABLE

NAMED CONSTANT

GESSUM	0	0	0	0	5
ZFIL1	2	14	2	8	1
ARB	0	0	0	0	10
DRUCK1	0	0	0	0	1
DVB1	3	6	0	0	0
ZWISP	0	0	0	1	4
DVB2	3	6	0	0	1
DVB3	3	6	0	8	1
DVB4	3	6	7	8	1

LABEL

START	0	0	0	0	0
ANFANG	0	1	7	14	12
UEBERS	0	2	0	2	1
ANSCHR	0	1	7	15	2
ARTNR	0	1	7	15	9
MENGE	0	2	0	0	0
EPREIS	0	2	0	0	5
GESPR	0	2	0	0	13
WARTE	0	2	0	1	0
SUMDRU	0	2	0	1	6
ENDE	0	2	0	1	10

END

```
ASS - PASS II

                                                  10                                AUFGABENSTELLUNG: EINFACHE PREIS-/MENGEN-
                                                  20                                                  BERECHNUNG
                                                  30
0 0 0  0  0     0  0   5  2  3                    40  START   NOP 5.2.3             ANFANGSPARAMETER (KA USW.)
                                                  50  *       5                     SETZE BEF.ZAEHLER ABSOLUT AUF 1. BEF.ADRESSE
0 0 0  0  5     2 11   4  0  1                    60          BL 1                  SPRUNG NACH BLOCK 1
0 0 0  0  6     1  0   7 14 12*BL                 70          BR ANFANG
                                                  80
                                                  90  *       START+4076            SETZE BEF.-ZAEHLER (RELATIV)
                                                 100
0 1 7 14 12     2 14   2  8  5                   110  ANFANG  LF,LEP,UP,SAVPL,5
0 1 7 14 13     2 13   0  1  4                   120          TAB 20
0 1 7 14 14     2 11   4  0  2                   130          BL 2
0 1 7 14 15     3  0   0  2  1*BL                140          TT UEBERS
0 1 7 15  0     0 15   0  5  0                   150          CLR GESSUM.
0 1 7 15  1     2 14   2  8  4                   160          LF,LEP,UP,SAVPL,4
0 1 7 15  2     2 10   0  0  5                   170  ANSCHR  CX 5                  SCHREIBMASCHINEN-FREIGABE ZUM DRUCKEN DER
0 1 7 15  3     2 13   0  0 10                   180          TAB 10                ANSCHRIFT (5 ZEILEN).
0 1 7 15  4     2 13   2  2  3                   190          TW 35
0 1 7 15  5     2 14   2  8  1                   200          ZEIL1
0 1 7 15  6     2 10 1 7 15 15                   210          CX,XI,8..-1
0 1 7 15  7     1 10   7 15  3*                  220          BXU,ANSCHR+1
0 1 7 15  8     2 14   2  8  5                   230          2.14.2.8.5            LF LEP UP SAVPL 5
0 1 7 15  9     2 12   0  2  1                   240  ARTNR   WT MRAR               WARTEN AUF EINGABE ARTIKEL-NUMMER
0 1 7 15 10     0  1   0 10  0                   250          ACC ARB.              EINGABE MIT TASTE PFEIL RECHTS
0 1 7 15 11     0  3   0 10  1                   260          MV DRUCK1 ARB.        GANZZAHLIG,UNGLEICH NULL.
0 1 7 15 12     3  6   0  0  0                   270          DVB1                  DRUCKVORBEFEHL
0 1 7 15 13     3  2   0  0 15                   280          ED 15                 DRUCKHAUPTBEFEHL AUS D1
0 1 7 15 14     2 11   4  0  2                   290          BL 2                  ABSOLUTER SPRUNG ZUM ANFANG DES NAECHSTEN
0 1 7 15 15     1  0   0  0  0*BL                300          BR /-/                BLOCK (ADRESSE 0.0.0)
0 2 0  0  0     2 12   0  2  1                   310  MENGE   WT,MRAR               WARTEN AUF TASTE PFEIL RECHTS
0 2 0  0  1     0  1   0 10  0                   320          ACC ARB.              EINGABE MENGE
0 2 0  0  2     0  3   0 10  1                   330          MV ARB. DRUCK1
0 2 0  0  3     3  6   0  0  0                   340          DVB1
0 2 0  0  4     3  2   0  1 14                   350          ED 30
0 2 0  0  5     2 12   0  2  1                   360  EPREIS  WT,MRAR               WARTEN AUF TASTE PFEIL RECHTS
0 2 0  0  6     0  1   1  4  2                   370          ACC ZWISP. 2          EINGABE EINZELPREIS MIT 2 NACHKOMMASTELLEN
```

```
0 2 0 0 7    0  3    1  4  1        380            MV ZWISP. DRUCK1
0 2 0 0 8    3  6    0  0  1        390            DVB2
0 2 0 0 9    3  2    2  2 13        400            ED 2..45
0 2 0 0 10   0  8    1  4 10        410            MLH ARB ZWISP.       MULTIPLIZIERE MENGE X EINZELPREIS
0 2 0 0 11   0  3    1  4  1        420            MV ZWISP. DRUCK1
0 2 0 0 12   3  6    0  8  1        430            DVB3
0 2 0 0 13   3  2    2  4  1        440  GESPR     ED 2..65             DRUCKEN GESAMTPREIS IN ROT
0 2 0 0 14   2 14    2  8  1        450            ZEIL1
0 2 0 0 15   0  5    1  4  5        460            AD ZWISP. GESSUM
0 2 0 1 0    2 12    1  2  1        470  WARTE     SM1 MRAR
0 2 0 1 1    2 11    4  0  1        480            BL 1
0 2 0 1 2    1  1    7 15 10*BL     490            BR1 ARTNR+1          EINGABE NEUER WERTE
0 2 0 1 3    2 12    1  2  5        500            SM1 MFTS             TASTE STERN
0 2 0 1 4    1  1    0  1  6*       510            BR1 SUMDRU           DRUCKEN DER ENDSUMME
0 2 0 1 5    1  0    0  1  0*       530            BR WARTE
0 2 0 1 6    2 14    2  8  2        540  SUMDRU    LF LEP UP SAVPL 2
0 2 0 1 7    0  2    0  1  5        550            MVH GESSUM DRUCK1.
0 2 0 1 8    3  6    7  8  1        560            DVB4
0 2 0 1 9    3  2    2  4  1        570            ED 2..65             DRUCKEN GESAMTSUMME
0 2 0 1 10   2 12    1  2  1        580  ENDE      SM1 MRAR
0 2 0 1 11   2 11    4  0  1        590            BL 1
0 2 0 1 12   1  1    7 15  0*BL     600            BR1 ANSCHR-2         NEUE ANSCHRIFT UND EINGABE NEUER WERTE
0 2 0 1 13   2 12    1  2  2        610            SM1 MLAR
0 2 0 1 14   2 11    4  0  1        620            BL 1
0 2 0 1 15   1  1    7 14 12*BL     630            BR1,ANFANG
0 2 0 2 0    1  0    0  1 10*       640            BR /-6
0 2 0 2 1    1  2 1 1  2  4        650  UEBERS    *ASSEMBLER/YBLK/PROGRAMMBEISPIEL/YECC*
0 2 0 2 2    1  6    7  9  3
0 2 0 2 3    1 13    5 10  3
0 2 0 2 4    0 10 1 0  6  3
0 2 0 2 5    2  0    6  2  3
0 2 0 2 6    1  2    7  9 14
0 2 0 2 7    1  3    5  9 10
0 2 0 2 8    2  4 1 0  5 10
0 2 0 2 9    1  6    7  7 15

                                  660  *DVB1     EDF BLACK            BEFEHLS-
                                  670  *DVB2     EDF ZERO BLACK 1     DARSTELLUNG
```

```
680  *DVB3     EDF ZERO RED 1       DURCH
690  *DVB4     EDF ZERO RED FTS 1   SYMBOLISCHE
700  *ZEIL1    LF LEP UP SAVPL 1    NAMEN (WERTZUWEISUNGEN)
710  *GESSUM   5                    DEFINIERE DATENSPEICHERWORT ABSOLUT
720  *ARB      GESSUM+5             DEFINIERE DATENSPEICHERWORT RELATIV
730  *ZWISP    ARB+ARB
740  *DRUCK1   1
750  **                             END-ANWEISUNG

END
```

06 IN PT
0 0 0 0 0 0 0 0 0 0

ASSEMBLER PROGRAMMBEISPIEL

FIRMA
FRITZ H U B E R
4790 PADERBORN
GRABENGASSE 25
POSTFACH 413

15236	125	2,30	287,50
52369	50	12,50	625,00
65478	15	28,65	429,75
36498	200	0,85	170,00
			1512,25*

Fehlerschlüssel

E 1 Spalte 1 ist ungleich

a) Blank,
b) Stern,
c) Gleichheitszeichen

E 2 Das erste Zeichen eines Namen ist kein Buchstabe

a) im Labelfield
b) bei Namen als Ersetzungszeichen im Alphatext

E 3 Der Name ist größer als 6 Zeichen

E 4 Das Zeichen ist unzulässig, da größer 3.15 (ALC-Code).

E 5 Name in interner Tabelle der festen symbolischen Namen doppelt.
OP-Codes, Ergänzungen usw. dürfen nicht als symbolische Adressen verwendet werden.

E 6 Name war schon als definiert im Adreßbuch (variable Tabelle) abgelegt, daher doppelt.

E 7 Fehler in der Zahl

a) sonstige Zeichen (Buchstabe usw.)
b) kein zulässiges Zeichen
c) zu groß

E 8 Name im zweiten Durchlauf nicht definiert

a) als nicht definiert im Adreßbuch
b) in interner Tabelle nicht vorhanden

E 9 Formaler Fehler

a) In Alphatext am Ende nach einem Schrägstrich kein Zeichen mehr

b) Vor einem Schrägstrich muß ein Blank sein

c) In einem Ausdruck Verknüpfungszeichen vergessen

d) Nach einem Schrägstrich muß Plus, Minus folgen, wenn weitere Angaben vorhanden sind.

e) Mehrere Sonderzeichen hintereinander

f) In Alphatext Ende der Namen nur mit Schrägstrich, Stern

E10 Symbolische Adresse in einer Alphatext-Folgeaussage

E11 Erstes Zeichen nach Trennzeichen ungleich Sonderzeichen

E12 Erstes Zeichen nach Schrägstrich kein Funktionszeichen

E13 Fehler in Alphatext

E14 Sequenzfehler, falsche Lochkartenreihenfolge

E15 Fehler bei Speicherortzuweisung

1. Durchlauf

a) Zeichen im Labelfield

1. und 2. Durchlauf

a) Name vorher nicht definiert
b) Wert zu groß
c) formaler Fehler in der Schreibweise