As time Goes By ...

Die IBM 1401

Warum war sie so erfolgreich?

Wolf-Rainer Novender*

3. Februar 2025

User:Swtpc6800 Michael Holley - Own work

Letzte Änderung am 3. Februar 2025, dsn=1401.tex
Wolf-Rainer.Novender@ieee.tum.de

Thanks to Van Snyder supporting me with the Fortran-version of his AUTOCODER assembler.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

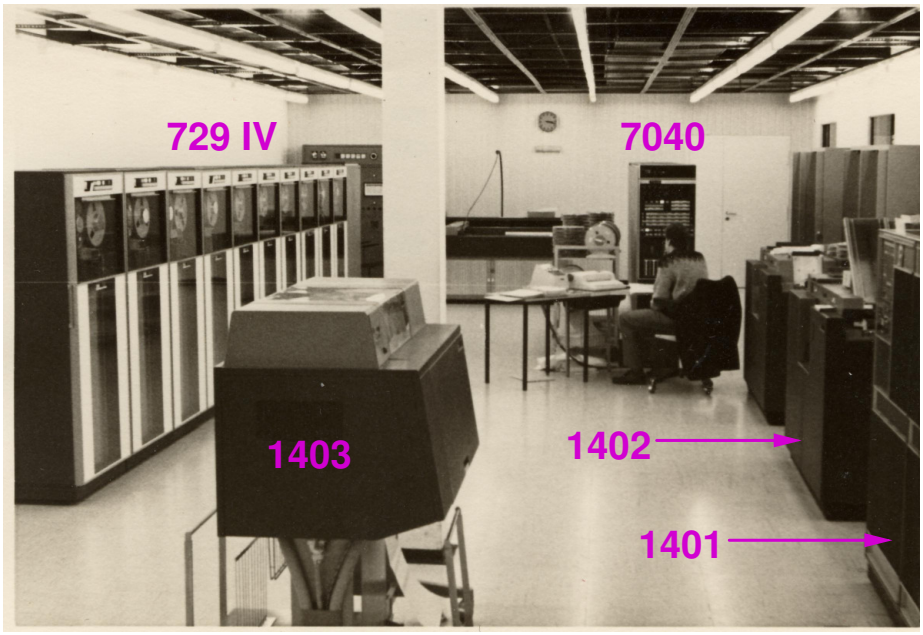


Abb. 1: Rechenzentrum der Technischen Hochschule Darmstadt (ca. 1970, Quelle: NN)

Vorwort

Dieser Bericht beruht auf den Erfahrungen des Autors am Hochschulrechenzentrum (HRZ) der Technischen Hochschule in Darmstadt (THD)[1] in der Zeit von 1969 bis zum Abbau der Rechenanlage 1974/75 [1].

Dieser Rechner (»die 1401«) wurde im HRZ als *Front-End* für den Großrechner IBM 7040 eingesetzt (Abb. 1). Der Datenaustausch zwischen den beiden Rechnern erfolgte über Magnetbänder. Die 1401 hatte 4 k Speicherplätze, an ihr waren 4 umschaltbare Bandeinheiten (729 IV), ein Schnelldrucker 1403 und eine Kartenlese-/stanzeinheit 1402 angeschlossen.

IBM kündigte die 1401 im Oktober 1959 an. Die Resonanz, vor allem für kleinere und mittlere Unternehmen, war groß. Zitat aus [2]: »*In all, by the mid-1960s nearly half of all computer systems in the world were 1401-type systems.*«. Dazu beigetragen hatten derzeit u. a. die innovative Technik, Verarbeitungsgeschwindigkeit und universellen Einsatzmöglichkeiten dank vielseitiger Peripheriegeräte, der Wirtschaftsaufschwung in den Nachkriegsjahren, in den USA als auch in Europa und nicht zuletzt der günstige Mietpreis. In Deutschland wurden vom IBM Werk Sindelfingen bis 1962 über 1000 1401-Systeme ausgeliefert [3]. Zu den ersten Großkunden gehörten Anfang der 60-er vor allem Kreditinstitute und Versicherungen.

Neben der Hardware wurden auch kostenlos Dienstprogramme und deren Dokumentationen angeboten. Programmiert wurde vorwiegend auf Maschinenbefehlsebene und mit symbolischen Assembler (SPS, Autocoder). Für den kaufmännischen Bereich standen unter anderem ein COBOL-Compiler und RPG (*Report Program Generator*) zur Verfügung. Zur Materialverwaltung gehörte auch der Stücklistenprozessor BOMP (*Bill Of Material Processor*), eines der ersten Datanbanksysteme. Obwohl der Schwerpunkt der Anwendungen im kommerziellen Bereich lag, entwickelte IBM für den wissenschaftlichen Bereich einen FORTRAN-Compiler.

[1] heute TU Darmstadt

Beeindruckend sind die vielen Hardware-Optionen für den Kunden zur Leistungssteigerung [4], was sich naturgemäß auf den Kauf-/Mietpreis niederschlägt. Eine Übersicht programmtechnischer Daten findet man auf der Webseite von Van Snyder [5].

Viele Programmierer, Techniker und Zeitzeugen aus dieser 1401-Ära haben sich in den USA zusammengeschlossen, restaurieren und führen funktionstüchtige 1401-Rechner mit ihrer Peripherie vor [6].

Da es noch funktionierende Maschinen gibt, wenn auch nur im Museum, hat der Autor beschlossen, im Folgenden die Besonderheiten dieses Rechners in der Gegenwartsform zu beschreiben.

Hinweis zur Darstellung im Text: Instruktionen und deren Bedeutung (in Englisch) werden durch das Zeichen ≡ getrennt.

Beispiel: `/ I99` ≡ *Clear Storage*

1 Grundlegendes

Quelle: https://en.wikipedia.org/wiki/IBM_1401
Referenz: [7]

Die 1401 ist eine der ersten Rechner von IBM, der mit Transistoren und einem echten Magnet-Kernspeicher bestückt ist . Die Schaltkreise sind einheitlich auf Steckkarten vorwiegend in der Größe von etwa 11.5 cm × 6.5 cm auf einen Rahmen mit Stecksockeln untergebracht (SMS, *Standard Modular System*). Die Verdrahtung der Schaltkreise erfolgt auf der Sockelseite mittels *Wire-Wrap*-Verbindungen. Die Rahmen sind schwenkbar und können mit einem Handgriff heraus geklappt werden. Dadurch sind die Steckkarten von beiden Seiten zugänglich.

Alle Rechner der 1400-Reihe basieren auf dem Dezimalsystem. Der Adressraum der 1401 reicht, je nach Modell, von 0 bis 1399 (1.4 k), 1999 (2 k), 3999 (4 k), 7999 (8 k), 11999 (12 k) und 15999 (16 k). Die Instruktionen (das Programm) und Daten werden wie Text abgespeichert, jedes Zeichen im BCD-Format (*Binary Coded Decimal*) in eine Speicherzelle. Numerische Daten werden wie Text gespeichert, jede Ziffer ist adressierbar.

Ein Speicherplatz besteht aus 8 Bits (kein Byte) : zwei Zonenbits B und A, vier Bits mit der Wertigkeit 8, 4, 2 und 1, Bit C zur Paritätsprüfung (auf ungerade). Eine Besonderheit stellt das 8. Bit dar: die Wortmarke. Sie dient zum Markieren von Instruktionen und Datenfelder und ermöglicht es, Felder unterschiedlicher Länge zu definieren und zu verarbeiten (variable Wortlänge).

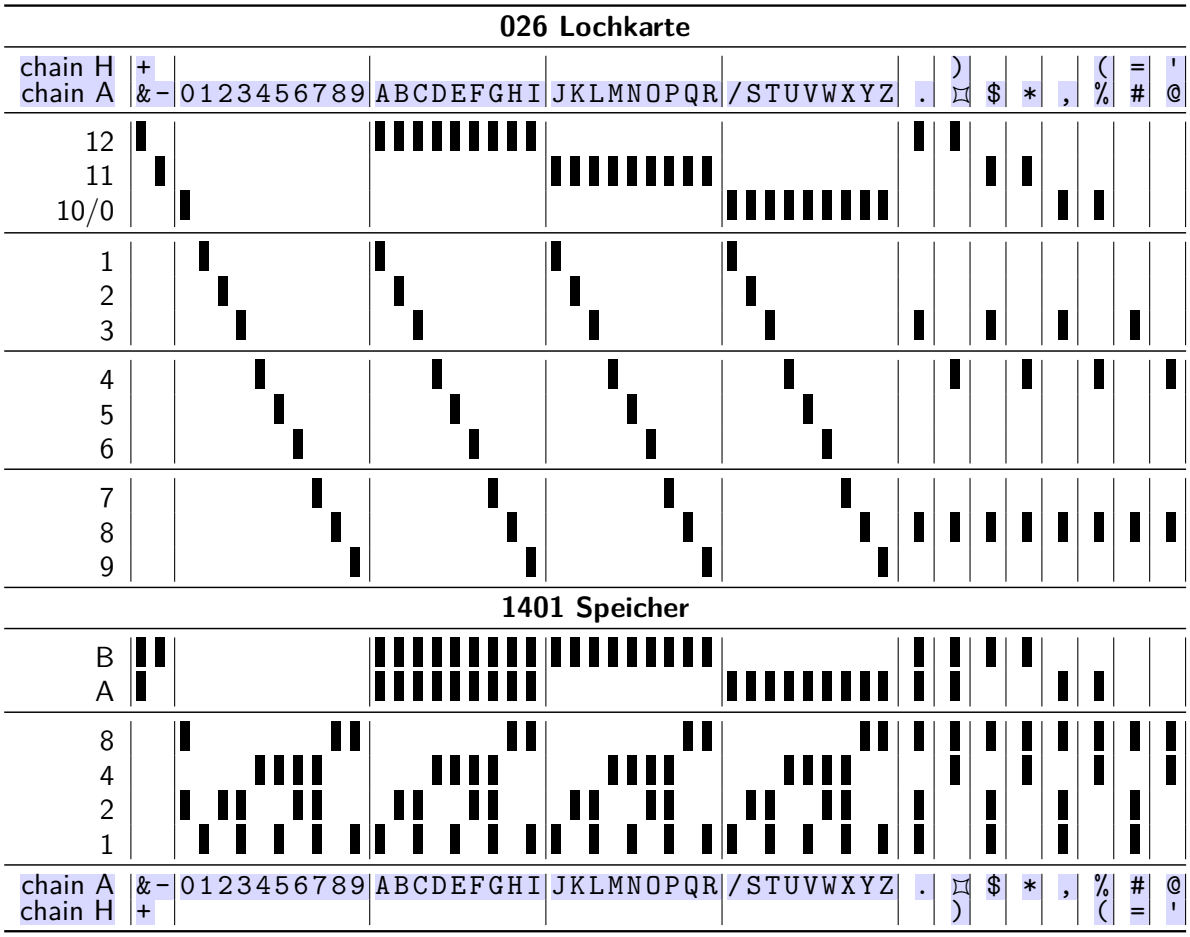
Die Instruktionen werden direkt in den Datenfeldern ausgeführt (*memory-to-memory*). Es gibt keinen Akkumulator, keinen Stack. Unterprogrammtechnik ist möglich.

Die Zykluszeit beträgt 11.5 µs, das entspricht einer Taktfrequenz von ≈ 87 kHz.

Abmessungen der Zentraleinheit 1401-D mit 4k (H ×B ×T in cm)[2]: ca. 150 ×150 ×80
Elektr. Anschlusswerte: 380/220 V, 50 Hz, ca. 4–5 kVA (1401, 1402, 1403, ohne Bandeinheiten)
Gewicht der Zentraleinheit: mindestens 500 kg

[2] *Double Cube*

Tab. 1: BCD Codierung



2 Codierung

Die Datenerfassung erfolgt im Allg. durch einen Schreiblocher[3] (Typ 026) auf einer Lochkarte mit 80 Spalten und 12 Zeilen (Patent IBM, 1928). Die Nummerierung/Wertigkeit der Zeilen von oben nach unten lautet: $12 - 11 - \overset{10}{0} - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9$. Die Zeilen 12 und 11 bezeichnet man als Zone oder Zonung (z. B. 11-er-Zonung). Mit den Zeilen 0 bis 9 lassen sich alle Dezimalzahlen ohne Vorzeichen mit einer Lochung darstellen (siehe Tabelle 1). Werden Buchstaben hinzugenommen, kombiniert man Nummernlochungen mit einer Zonenlochung (zwei Lochungen/Spalte). Da zwei Zonen für das vollständige Alphabet nicht genügen, benutzt man auch die Zeile 0 als Zonung und bezeichnet sie als 10-er Zonung.

Wie man aus der Codetabelle 1 erkennt, können nur maximal 39 Zeichen damit dargestellt werden: 0–9, A–Z $\equiv$ 36 plus die drei Sonderzeichen '`&-/`'. Beim Verarbeiten der Lochkarten wird jedes Zeichen auf die 6 Bits BA8421 der 1401 umcodiert (übersetzt nach BCD), und zwar wie folgt: Zonungen 12 ➞ BA, 11 ➞ B, 10 ➞ A, Nummern 1–9 ➞ binäre Darstellungen mit den Bits 8-4-2-1.

Die Codetabelle 1 soll im Folgenden zum Verständnis bei der Adressierung, Indexierung und dem vorzeichenbehafteten Rechnen beitragen.

[3] Der Name deutet an, dass der Locher die Lochkarte auch gleichzeitig spaltengerecht beschriftet.

Ein Leerzeichen (*Blank, Space*) auf der Lochkarte setzt kein Bit. Die Null wird aus den beiden Bits 8-2 gebildet. Dabei spielt es keine Rolle, ob sie zum Rechnen oder als Text verwendet wird. Eine negative Zahl wird in der *Einerstelle* mit dem B-Bit markiert, z. B. wird die Zahl −7 zum Buchstaben P. Alle anderen Zonungen oder keine in der Einerstelle definieren eine positive Zahl. Somit lässt sich z. B. die Zahl +7 als 4-2-1 ≡ 7, A-4-2-1 ≡ X oder B-A-4-2-1 ≡ G codieren.

Die 1401 verfügt zusätzliche über Rechner-spezifische Sonderzeichen, die zur internen Steuerung benötigt werden (Auswahl):

Name	Zeichen	Lochkarte	Speicher
record mark	ǂ	0-2-8	A82
mode change	Δ	11-7-8	B8421
group mark	ǂ	12-7-8	BA8421
lozenge	⌑	12-4-8	BA84
tape mark	√	7-8	8421

Einige Lochkombinationen lassen sich nicht mit dem Schreiblocher 026 produzieren und müssen durch das ausführende Programm im Speicher generiert werden.

Bei der Kombination 'Zonung mit Nummern' bleibt nach Codierung des englischen Alphabets (26 Buchstaben) eine Kombination übrig, die mit der Lochung 0-1 ≡ A-1 dem Sonderzeichen `/` zugeordnet wird.

> Die 1401 wurde hauptsächlich für kommerzielle Anwendiungen eingesetzt. Weiterhin diente sie auch mit ihrer schnellen Peripherie als *Front-End*-Rechner in Verbindung mit Großrechnern im wissenschaftlichen Bereich.

Der kaufmännische Bereich verwendet andere Sonderzeichen als die Programmentwickler. Um die Anzahl der benötigten Sonderzeichen codieren zu können geht man zu einer Dreier-Lochung über und verwendet Kombinationen aus der Lochung 8 mit Nummern zwischen 3 und 7 und den Zonenlochungen.

Da der für *beide* Anwendungen benötigte Sonderzeichensatz *gleichzeitig* nicht verfügbar ist, verwendet man für die 1401 je nach Anwendung *unterschiedliche Zeichendarstellung bei gleicher Codierung* (Lochkarte, Rechner). In der Praxis sieht das so aus, dass man je nach Einsatz unterschiedliche Druckerketten (*printer chain*) im Schnelldrucker 1403 einsetzt (siehe Kapitel Der Schnelldrucker 1403). Für kommerzielle Anwendung verwendet man die Zeichenkette A mit den Sonderzeichen `&-.⌑$*,%#@` (*chain A*). Für FORTRAN Programme (für technisch/wissenschaftliche Anwendungen) und Programmierung in Autocoder[4] benutzt man die Zeichenkette H, die *statt dessen* folgende Sonderzeichen enthält: `+-.)$*,(='`. Die beiden Druckerketten unterscheiden sich im Prinzip nur in den 5 Druckzeichen `+)(='` (siehe Tabelle 1: chain H, A).

> Um beide Zeichensätze gleichzeitig benutzen zu können, führte IBM den Schreiblocher 029 ein und erweiterte gleichzeitig den Zeichensatz mit einigen Sonderzeichen zum EBCD (*extended BCD*). Zu diesem Zeitpunkt enthielt die BCD-Codierung nur Großbuchstaben, was erklärt, warum zu der damaligen Zeit von Computern generierte Schreiben/Rechnungen keine Kleinbuchstaben und Umlaute enthielten. Im nächsten Schritt zum System S/360 musste auch der Zeichenvorrat eines Bytes (8-Bit) auf Lochkarte verfügbar gemacht werden, was zur EBCDIC (*Extended Binary Coded Decimal Interchange Code*) Codierung führte.

[4] ein symbolischer Assembler für 1401-Programmentwicklung

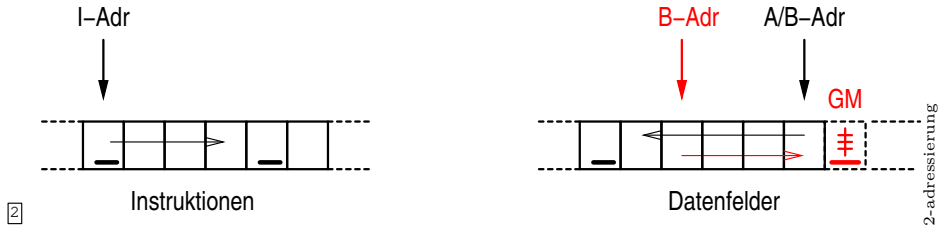


Abb. 2: Adressierung von Instruktionen und Datenfeldern. Es bedeuten: I-Addr Instruktionsadresse, B-Adr B-Adresse, A/B A- oder B-Adresse, GM Gruppenmarke; Erläuterung siehe Kapitel 3.2

3 Adressierung

Daten und Instruktionen können jede Speicherzelle der 1401 belegen. Eine Trennung zwischen Daten und Instruktionen gibt es nicht. Es gibt jedoch Speicherbereiche, die für bestimmte Operationen reserviert oder Ein-/Ausgabeoperationen zugeordnet sind.

3.1 Datenfelder und Instruktionen – die Wortmarke

Zur Beschreibung werden folgende Begriffe benutzt:

Der Adressraum beginnt bei 1 (links, *left most*) und endet mit der höchsten Adresse (rechts, *right most*), je nach Modell bei 1399 ,1999, 3999, 7999, 11999, 15999.

Eine Instruktion wird von links nach rechts abgearbeitet. Die Instruktionsadresse ist die Speicheradresse, bei der die Instruktion (*op-code*) beginnt.

Datenfelder werden von rechts nach links verarbeitet, also von der niederwertigen Zeichenposition (rechts, Einerstelle, *low order, junior position*) zur höherwertigen Position (links, *high order*). Die Adresse eines Datenfeldes bezieht sich auf die niederwertigste Position, bei numerischen Feldern ist dies die Einerstelle, bei Textdaten ist es das Textende[5].

Die Übertragung von Datenfeldern auf/vom Magnetband läuft umgekehrt ab, von links nach rechts, vom Textanfang zum Textende.

Wie erkennt die 1401 den Beginn einer Instruktion bzw. das Ende eines Datenfeldes?

Dazu dient das zusätzliche Bit, die *Wortmarke*, symbolisch dargestellt durch einen Unterstrich _. Das Schema in Abbildung 2 verdeutlicht den Sachverhalt.

Die Wortmarken bestimmen die Längen der Datenfelder und Instruktionen (*variable Wortlänge*). Instruktionen beginnen mit einer Wortmarke und enden bei der nächsten Wortmarke (es gibt einige Ausnahmen). Die Verarbeitung von Datenfeldern beginnt bei der angegebenen Adresse (A/B-Adr)[6], bewegt sich von der *low order* Position zur *high order* Position (nach links) und endet mit der nächsten Wortmarke.

Bei einer Bandoperation wird der Bereich von B-Adr bis zur Gruppenmarke (GM) verarbeitet (nach rechts), d. h. beim Schreiben auf ein Magnetband wird beim Erreichen der Gruppenmarke mit Wortmarke der Schreibvorgang abgeschlossen, beim Lesen vom Magnetband werden die

[5] Der Begriff 'höher-/niederwertige' Stelle bezieht sich eigentlich auf ein numerisches Feld. Da die 1401 numerische Daten und Textdaten gleichermaßen abspeichert ist bei Text die niederwertigste Position das Textende, also rechts.

[6] muss nicht unbedingt der Beginn eines Datenfeldes sein

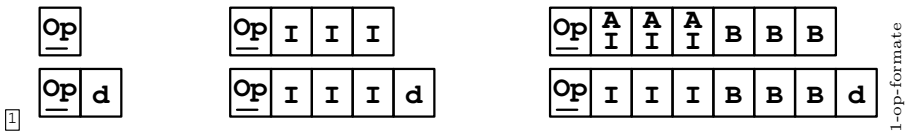


Abb. 3: 1401 Instruktionsformat

Daten ab B-Adr abgespeichert bis entweder der Bandsatz zu Ende ist (*inter-record gap*) oder eine Gruppenmarke mit Wortmarke im Speicher erreicht wird.

3.2 Instruktionen und ihre Formate

Instruktionen können zwischen einer bis acht Speicherzellen lang sein (Abbildung 3). Instruktionslängen von 3, 6 oder >8 sind ungültig. Es bedeuten:

`Op` Instruktionscode (*operation code*), einstellig, muss eine Wortmarke haben

`d` Instruktionserweiterung, Operationsergänzung (*d-character, d-modifier*), einstellig

`AAA` A-Adresse, dreistellig

`BBB` B-Adresse, dreistellig

`III` Instruktions-Adresse, dreistellig

Die Adresse einer Instruktion wird im I-Adress-Register (dreistellig) gespeichert (siehe auch Abb. 2).

3.3 Adressbereiche

Die A- und B-Adressen[7] in den Instruktionen sind immer dreistellig und enthalten, mit wenigen Ausnahmen, Speicheradressen von Instruktionen und Datenfeldern.

Die Adressen bis 999 sind direkt darstellbar. Ab 1000 erhält die Hunderterstelle (*high order position*) zusätzlich Bits in den Positionen A und B nach folgendem Schema:

- von 1000–1999 ➙ A-Bit
- von 2000–2999 ➙ B-Bit
- von 3000–3999 ➙ BA-Bit

Damit wird aus der Hunderterstelle ein Buchstabe bzw. ein Sonderzeichen. Mit Hilfe der Codierungstabelle 1 lassen sich daraus die 3-stelligen Adressen bis 3999 erzeugen (Tabelle 2).

BeispieL: Adresse 3250 ➙ 3 2 50 ➙ $\overset{BA}{2}$ 50 ➙ B50 .

Bei den Adressen mit einer Null in der Hunderterstelle (10XX, 20XX, 30XX) entstehen die Sonderzeichen A82 ≡ ≠, B82 ≡ ! und BA82 ≡ ? . Für Adressen $>$ 3999 siehe Abschnitt Adressen größer als 3999.

[7] Die Bezeichnungen haben nichts mit den A- und B-Bits zu tun.

Tab. 2: Codierung der Adressen von 0 bis 3999 als dreistellige Adresse

Adressbereich	Adresse	Adressbereich	Adresse	Adressbereich	Adresse
0 - 999	000 - 999				
1000 - 1099	ǂ01 - ǂ99	2000 - 2099	!00 - !99	3000 - 3099	?00 - ?99
1100 - 1199	/00 - /99	2100 - 2199	J00 - J99	3100 - 3199	A00 - A99
1200 - 1299	S00 - S99	2200 - 2299	K00 - K99	3200 - 3299	B00 - B99
1300 - 1399	T00 - T99	2300 - 2399	L00 - L99	3300 - 3399	C00 - C99
1400 - 1499	U00 - U99	2400 - 2499	M00 - M99	3400 - 3499	D00 - D99
1500 - 1599	V00 - V99	2500 - 2599	N00 - N99	3500 - 3599	E00 - E99
1600 - 1699	W00 - W99	2600 - 2699	O00 - O99	3600 - 3699	F00 - F99
1700 - 1799	X00 - X99	2700 - 2799	P00 - P99	3700 - 3799	G00 - G99
1800 - 1899	Y00 - Y99	2800 - 2899	Q00 - Q99	3800 - 3899	H00 - H99
1900 - 1999	Z00 - Z99	2900 - 2999	R00 - R99	3900 - 3999	I00 - I99

3.4 Spezielle Adressbereiche

Bestimme Adressbereiche sind für Ein- und Ausgabeoperationen reserviert.

- Die Adressen 1 bis 80 erhalten den Inhalt einer Lochkarte nach einer Leseoperation.
- Eine Stanzoperation stanzt den Inhalt der Speicheradressen von 101 bis 180 in eine Lochkarte.
- Eine Druckoperation druckt den Inhalt der Speicherzellen 201 bis 332[8] als eine Zeile auf dem Schnelldrucker aus.
- Falls die 1401 mit der Option *Column Binary* ausgerüstet ist, wird der Inhalt einer Lochkarte auf zwei zusätzliche Bereiche aufgeteilt, und zwar überträgt man die obere Hälfte (Zeilen 12 bis 3) in den Bereich 401 bis 480, die untere Hälfte (Zeilen 4 bis 9) in den Bereich 501 bis 580. Das Kapitel Binäre Verarbeitung befasst sich mit dieser Option.

Drei weitere Bereiche sind reserviert für Indexregister (siehe Kapitel Indexierung).

Benutzt man diese reservierten Speicherbereiche nicht, kann man sie für Daten und Instruktionen verwenden.

3.5 Indexierung

Beim Durchsuchen von Tabellen und/oder Schreiben/Lesen von geblockten Datensätzen führt man gleiche Instruktionen durch, jeweils mit modifizierten A-/B-Adressen. Da die Adressen im Prinzip numerische Datenfelder darstellen, kann man mit ihnen auch rechnen und z. B. vor jeder Ausführung eine Konstante zu einer Adresse addieren oder subtrahieren. Eine bequemere Art der Adressen-Modifikation bieten die Indexregister, die drei feste, dreistellige Speicher belegen, falls man sie benötigt: X1 von 87–89, X2 von 92–94 und X3 von 97–99. Dazu werden die A-/B-Adressen der Instruktion in den Zehnerstellen mit zusätzlichen Bits markiert: X1 ➞ A-Bit, X2 ➞ B-Bit, X3 ➞ BA-Bits. Vor jeder Ausführung der Instruktion wird, entsprechend der Zonenbits in der Zehnerstelle der jeweiligen Adresse, der Inhalt des entsprechenden Indexregister automatisch zu der Adresse addiert. Danach ändert man nur noch die Werte der Indexregister, die Instruktion bleibt unverändert.

[8] Je nach Druckermodell sind andere Obergrenzen möglich

Beispiel: die Adresse (A und/oder B) einer Instruktion hat den Wert 3250 (B50) und soll mit dem Indexregister 2 modifiziert werden: ➙ B $\overset{B}{5}$ 0 ➙ BN0 .

3.6 Adressen größer als 3999

Mit der bisherigen Zahlencodierung nach Tabelle 2 lassen sich dreistellige Adressen von 0 bis 3999 erzeugen. Für Adressen ab 4000 ist der Bit-Vorrat der Hunderterstelle erschöpft. Teilt man den maximal möglichen Speicher von 16 k in 4 Bereiche je 4 k auf und weist jedem dieser Blöcke ein Zonung in der *Einerstelle* zu, gewinnt man den gewünschten Adressraum. Dazu erhält die Einerstelle zusätzlich zu ihrem numerischen Wert folgende Zonung: 0–3999 ➙ keine, 4000–7999 ➙ A-Bit, 8000–11999 ➙ B-bit, 12000–15999 ➙ BA-Bits.

Beispiele:
$$\begin{aligned}
3250 &= 0 + 3250 \rightarrow \text{kein Bit} + 3250 \rightarrow 32\ 5\ 0 \rightarrow \texttt{B50}\\
4896 &= 4000 + 0896 \rightarrow \text{A-Bit} + 0896 \rightarrow 08\ 9\ \overset{A}{6} \rightarrow \texttt{89W}\\
11345 &= 8000 + 3345 \rightarrow \text{B-Bit} + 3345 \rightarrow 33\ 4\ \overset{B}{5} \rightarrow \texttt{C4N}\\
15009 &= 12000 + 3009 \rightarrow \text{BA-Bits} + 3009 \rightarrow 30\ 0\ \overset{BA}{9} \rightarrow \texttt{?0I}
\end{aligned}$$

Vorgehensweise: Man zieht von der Adresse solange 4 k-Blöcke ab, bis eine Adresse < 4 k übrig bleibt. Diese Adresse kann man z. B. mit Hilfe der Tabelle 2 umwandeln. Je nach Adressbereich der Originaladresse versieht man die Einerstelle mit den entsprechenden Zonungs-Bits und erhält ein neues Zeichen gemäß Tabelle 1 für die Einerstelle.

Mathematisch ausgedrückt heißt das, man zerlegt die Originaladresse a

$$a = \underbrace{\text{int}(a/4000)}_{b} * 4000 + \underbrace{\text{mod}(a, 4000)}_{c}$$

wobei das Resultat der Ganzzahldivision b die Anzahl der 4 k-Blöcke angibt und c eine gültige Adresse < 4 k darstellt. Die binäre Darstellung von b stellt die Zonenbits der Einerstelle dar.

Sind die Adressen indiziert trägt die Zehnerstelle auch ein Zonen-Bit. Nimmt man die Original-Adressen von dem vorherigen Beispiel, ergeben sich mit Indizierung folgende neue Adressen:

Beispiele:
$$\begin{aligned}
3250 \rightarrow \texttt{B50} \text{ indiziert mit } \texttt{X1} \rightarrow\ & B\ \overset{A}{5}\ 0 \rightarrow \texttt{BV0}\\
4896 \rightarrow \texttt{89W} \text{ indiziert mit } \texttt{X1} \rightarrow\ & 8\ \overset{A}{9}\ W \rightarrow \texttt{8ZW}\\
11345 \rightarrow \texttt{C4N} \text{ indiziert mit } \texttt{X3} \rightarrow\ & C\ \overset{BA}{4}\ N \rightarrow \texttt{CDN}\\
15009 \rightarrow \texttt{?0I} \text{ indiziert mit } \texttt{X2} \rightarrow\ & ?\ \overset{B}{0}\ I \rightarrow \texttt{?!I}
\end{aligned}$$

Die etwas kompliziert anmutende Adressenumsetzung wird erst ab einer Speichergröße > 4 k notwendig. Zur Erleichterung gibt es eine zusätzlich Instruktion (*modify address*), die diese Adressenberechnung (Addition, Subtraktion) und Umwandlung in eine gültige, dreistellige Adresse vornimmt.

4 Binäre Verarbeitung

> Wie eingangs schon erwähnt benutzte man die 1401 wegen ihrer schnellen Peripherie zur Ein- und Ausgabe in Verbindung mit der 7040 (Abb. 1). Wenn in diesem Kapitel von 'der 1401' gesprochen wird, handelt es sich genau genommen um ein Dienst-Programm, welches auf der 1401 zum Datenaustausch zwischen der 1401 und der 7040 ausgeführt wird.

An der 1401 werden Lochkarten-Jobs mit dem Kartenleser 1402 auf ein Magnetband geschrieben (*batch jobs*). Die 7040 verarbeitet dieses Magnetband und schreibt die Ergebnisse auf ein zweites Magnetband, welche anschließend wieder an der 1401 mit dem Schnelldrucker 1403 ausgedruckt und mit dem Stanzer 1402[9] gestanzt werden. Die Datenein- und -ausgabe mit der 1401 und die Verarbeitung mit der 7040 laufen unabhängig voneinander ab [1].

Die 7040 ist eine Wortmaschine mit 36 Bits. Die Codierung der Zeichen entspricht der BCD-Codierung der 1401 mit jeweils 6 Bits. Dabei werden jeweils $36/6 = 6$ Zeichen in einem Wort abgespeichert.

Speziell bei der Programmentwicklung an der 7040 ist es wünschenswert, getestete (Unter-)Programme in übersetzter Form aufzuheben und diese, um Übersetzungszeit zu sparen, als *Binärdeck* an Stelle der Quellprogramme zu verwenden. Dieses Binärdeck enthält das übersetzte Programm im Maschinencode mit relativen Adressen (*relocatable*). Da die 7040 keinen Plattenspeicher besitzt kann der Benutzer seine (Unter-)Programmbibliotheken nur auf Magnetbänder oder in Form von Lochkarten aufbewahren. Das Arbeiten mit Magnetbändern ist sehr aufwändig. Somit bleibt nur die Möglichkeit, die Binärdaten im Lochkartenformat aufzuheben.

Die Binärdaten/Binärdecks werden wortweise mit jeweils 36 Bits auf $36/12 = 3$ Spalten einer Lochkarte abgebildet, insgesamt 24 Worte. Die verbleibenden Spalten $73-80$ erhalten eine Identifikation und Nummerierung. Theoretisch könnte eine binäre Lochkarte bis zu 12 Lochungen in einer Spalte haben[10].

Beim Einlesen einer Lochkarte in die Speicherplätze 001 bis 080 wird jede Spalte auf eine gültige BCD-Lochung geprüft. Mit der Option *Column Binary* kann diese Gültigkeitsprüfung aufgehoben werden[11]. Ein spezieller Lesebefehl des Kartenlesers (*read column binary*) überträgt die oberen 6 Zeilen $(12-3)$ der Lochkarte bit-identisch in die Speicheradressen $401-480$, die unteren Zeilen $4-9$ in die Adressen $501-580$.

Umgekehrt erlaubt diese Option, diese beiden Bereiche mit einem Stanz-Befehl (*punch column binary*) zusammenzuführen und in eine Lochkarte zu stanzen (s. u.).

Bei den Magnetbandeinheiten 729 IV handelt es sich um schnelle 7-Spur Bandeinheiten. Die 6 Spuren entsprechen genau dem Abbild der Daten im Kernspeicher BA8421. Im Kernspeicher wird das Prüfbit automatisch so gesetzt, dass die Anzahl der Bits eines Zeichens einschließlich der Wortmarke ungerade ist (*odd parity*). Bei der Übertragung des Speicherinhalts von *BCD*-Zeichen auf Magnetband wird ein eigenes Prüfbit mit gerader Parität (*even parity*) in die 7. Spur des Magnetbandes gesetzt. Beim Schreiben von *Binärdaten* mit gerader Parität gibt es ein Problem: wenn das Zeichen im Kernspeicher nur ein A-Bit enthält (Leerzeichen, *Blank*) geht beim Lesen das A-Bit verloren. Damit das nicht passiert codiert man auf dem Magnetband Binärdaten grundsätzlich mit ungerader Parität.

[9] Die 1402 ist eine kombinierte Lochkartenlese- und -stanzeinheit.
[10] Das Duplizieren von Binärkarten mit einem normalen Schreiblocher ruiniert den Locher.
[11] Dank dieser Option ist es möglich, auch fremde Lochkarten-Kodierungen zu verarbeiten.

Beim Einlesen von Lochkarten-Jobs werden alle BCD-Karten geblockt mit gerader Parität auf das Magnetband geschrieben. Steuerkarten, das sind Karten mit einem $-Zeichen in Spalte 1, werden nicht geblockt. Die 1401-Instruktion zum Übertragen eines Speicherbereichs auf Magnetband enthält in der A-Adresse die Nummer der Bandeinheit mit einer Kennung der Kodierung (gerade oder ungerade), die B-Adresse gibt die Anfangsadresse des Speicherbereiches an, von dem von links nach rechts die Daten auf das Magnetband übertragen werden sollen. Das Ende des Schreibvorgangs wird durch eine Gruppenmarke am Ende des Schreibpuffers angezeigt (siehe Abb. 2).

Die 1401 erkennt anhand von Steuerkarten (`$IBLDR, $CDICT, $TEXT, $DKEND`), ob die *nächste* Lochkarte BCD- oder binäre Daten enthält (wichtig zur Umschaltung der Codierung, s. u.). Binärdaten werden ebenfalls geblockt, jedoch mit ungerader Parität auf Magnetband geschrieben. Zuvor müssen sie aber aus den beiden Teilpuffern (s. o.) zusammengeführt und in den Schreibpuffer kopiert werden, und zwar in der Reihenfolge 401-501-402-502-…(*Move and Binary Decode*). Der Speicherplatzbedarf ist doppelt so groß wie bei BCD-Karten.

Das Datenformat auf dem Magnetband ist geblockt mit fester Satz- und variabler Blocklänge. Die aktuellen Block-/Satzlängen und die Information der Codierung des *nächsten Blocks* (gerade/ungerade) stehen am Anfang jeden Datensatzes. Damit weiß der Rechner, der das Band einliest, in diesem Fall die 7040, mit welcher Codierung der *nächste* Block eingelesen werden muss. Stimmt die Codierung beim Lesebefehl nicht mit der Codierung auf dem Band überein wird eine Fehlerbehandlung veranlasst, die Zeit kostet: Bandsatz zurücksetzen (*Backspace*), Bandsatz erneut lesen mit geänderter Codierung.

Umgekehrt läuft der Datentransfer genauso ab. Die 7040 schreibt die Ergebnisse in gleicher Form auf Magnetband, das Datenformat ist jetzt geblockt mit *variabler* Satzlänge (wegen der Druckzeilen). Die Kodierung auf dem Magnetband erfolgt in gleicher Weise wie bei der 1401. Binäre Daten werden dekodiert, in die Speicherpositionen 401 - 480 bzw. 501 - 580 kopiert (*Move Binary code*) und binär gestanzt (*punch column binary*). BCD Daten werden in den Druckerbereich kopiert (201 - 332) und gedruckt, das *erste Zeichen einer Zeile wird unterdrückt* und als Vorschubzeichen interpretiert.

5 Programmierung

5.1 Maschinencode

Da viele 1401-Instruktionen sehr kurz sind und keine Adressen benötigen ist es möglich, kleine Programmsequenzen manuell an der Hilfskonsole einzugeben (siehe Titelbild). Dazu gehören unter anderem I/O-Operationen für den Kartenleser/-stanzer und Drucker.

Beispiel: der Lesebefehl einer Lochkarte besteht aus dem Zeichen `1` mit Wortmarke. Gibt man das Bitmuster `C1M` (M steht für Wortmarke, C ist das Prüfbit auf ungerade) mittels der Kippschalter an der Hilfskonsole ein (8 Kippschalter) – die Adresse ist unwichtig, da man mit einem weiteren Schalter diese Bitkombination auf *alle* Adressen verteilen kann – und startet den Rechner, liest der Kartenleser mit maximaler Geschwindigkeit (800 Karten/min) solange Lochkarten ein, bis die letzte Instruktion am Speicherende erreicht ist. Diese Maßnahme ermöglicht es, z. B. die Mechanik des Kartenlesers zu überprüfen (Servicetechniker).

In gleicher Weise lässt sich die Funktionsweise von Kartenleser, -stanzer und Drucker überprüfen. Druckertest: `2`, Bitkombination `C2M`; druckt Zeilen mit '2' in allen Stellen (600 Zeilen/min).[12] Stanzertest: `4`, Bitkombination `C4M` stanzt in alle Spalten eine '4' (250 Lochkarten/min).

Es gibt auch kombinierte Instruktionen:
`3`: lesen und drucken, Bits `21M`
`5`: stanzen und lesen, Bits `41M`
`6`: stanzen und drucken, Bits `42M`
`7`: lesen, stanzen und drucken, Bits `C421M`.

Kombinierte Instruktionen erhöhen die Verarbeitungsgeschwindigkeit, da sie Ein-/Ausgabeoperationen *gleichzeitig* auslösen. Die Lese-, Stanz- und Druckgeschwindigkeiten selber ändern sich *nicht*.

5.2 SPS und Autocoder

Beim Programmieren im Maschinencode ist neben der Kenntnis der Instruktionen auch das Berechnen der Adressen notwendig. Zur Unterstützung stellt IBM das *Symbolic Programming System* (SPS) zur Verfügung, ein einfaches Lochkartenpaket, in das man sein eigenes Programm in das Kartenpaket einfügt und lädt[13]. Statt der einstelligen, teilweise schwer zu merkenden Maschinenbefehle werden einfach zu merkende Kurzbefehle verwendet (*mnemonic op codes*). Die Programmanweisungen müssen spaltengenau gelocht werden, eine Karte für jede Anweisung. SPS läuft auf einer Minimalkonfiguration der 1401 ohne Bandeinheiten und stanzt nach erfolgreicher Umsetzung der SPS-Befehle ein *Object*-Deck, welches man anschließend ausführen (laden) kann.

Eine wesentliche Erleichterung bietet der Assembler *Autocoder* [8], dessen Eingabe wesentlich flexibler ist. Dazu gehören u. a. Macro-Bibliotheken wie IOCS (*Input-Output Control System*), die Standardroutinen enthalten und das Programmieren weiterhin vereinfachen[14]. Die 1401 muss mit vier Bandeinheiten[15] und einem Kernspeicher $\geq 4\,\mathrm{k}$ ausgerüstet sein.

[12] Die Vorgehensweise setzt die '2' in alle Speicherbereiche, also auch in den Druckbereich 201–332.
[13] Der Unterschied zwischen den Begriffen 'Laden' und 'Starten' wird weiter unten erläutert.
[14] Die 1401-FORTRAN- und COBOL-Compiler sind vollständig in Autocoder geschrieben.
[15] Eine spätere Version kommt mit nur 2 Bandeinheiten aus.

> Für technisch/wissenschaftliche Anwendungen entwickelte IBM in den 1950-er Jahren die Programmmiersprache FORTRAN (*FORmula TRANslation*) und stellte einen Compiler für ihre Großrechnerserie 70XX zur Verfügung. In der gleichen Dekade entstand für kaufmännische Aufgaben die Programmiersprache COBOL (*Common Business Oriented Language*).

Für beide Programmmiersprachen existieren für die 1401 Compiler [9, 10].

Es ist möglich, Programme für andere 1401-Rechner zu übersetzen. Eine Kontrollkarte (*CTL*) im Quellprogramm steuert die Übersetzungsoptionen und beschreibt die Konfiguration des Rechners, auf dem das Programm übersetzt wird und die des Zielrechners. Das ausführbare Programm kann sowohl als Kartendeck ausgestanzt als auch auf ein Magnetband geschrieben werden. Näheres zum Booten/Laden von Programmen siehe Abschnitt Der Boot-Vorgang – *Clear-Storage, Bootstrap*.

5.3 Verarbeitungsgeschwindigkeit – *Timing*

Die 1401 bietet viele Möglichkeiten, die Verarbeitungsgeschwindigkeit durch Hardware-Zusätze bzw. -Modifikationen und durch geschicktes Programmieren zu erhöhen. Eine einfach Möglichkeit bietet die Option *Overlap*. Dank eines zusätzlichen Adress-Registers (*O-register*), können Ein- und Ausgabeoperationen und interne Operationen überlappt ablaufen. Damit verringert sich die Sperrzeit der Zentraleinheit während einer Datenein-/ausgabe.

> Anwendung fand diese Option vor allem bei langsameren Geräten wie Lochstreifenleser (IBM 1011) und Dokumentenleser (IBM 1419).

Das Thema *Timing* spielt eine wichtige Rolle. Das Handbuch der 1401 [7] enthält exakte Angaben zur Berechnung der Rechenzeit, die jede Instruktion in Anspruch nimmt. Bei Ein- und Ausgabeoperationen über Lochkarte, Drucker oder Magnetband, bei denen Mechanik eine Rolle spielt, fallen zu der Übertragungszeit der Daten mechanische Verzögerungszeiten an. In dieser Zeit wartet die Zentraleinheit auf die Beendigung eines mechanischen Zyklus. Diese Zeit kann genutzt werden, um andere Operationen durchzuführen oder vorzubereiten. Die Wartezeiten lassen sich mit einer bedingten Sprunganweisung überprüfen. Ein Beispiel enthält das Kapitel Der Schnelldrucker 1403.

Der Kartenleser benötigt zum Einlesen einer Lochkarte 75 ms (Lese-Zyklus). Unter der Voraussetzung, dass der Lesebefehl `1` bereits im vorigen Zyklus gegeben wurde, entfallen 21 ms auf die mechanische Startzeit des Lesers (Transport der Lochkarte vom Einleseschacht zu den Abfragebürsten) und 44 ms zur Übertragung der Daten. Es verbleiben 10 ms Verarbeitungszeit, in der man z. B. den Karteninhalt bearbeiten kann. Wenn der nächste Lesebefehl nicht innerhalb dieser 10 ms erfolgt, rastet die mechanische Kupplung aus und der Lesebefehl kann erst zum nächsten Zyklus (75 ms) ausgeführt werden. Dadurch sinkt die Lesegeschwindigkeit von 800 cpm (*cards per min*) auf 400 cpm.

Beim Stanzen (`4`) sieht es ähnlich aus: unter den gleichen Voraussetzung wie beim Lesen dauert ein Stanzzyklus $4 \times 60\,\text{ms} = 240\,\text{ms}$, davon entfallen 37 ms auf die Startzeit, 181 ms auf die Datenübertragung und 22 ms Verarbeitungszeit. Das ergibt eine maximale Stanzgeschwindigkeit von 250 cpm. Verpasst der neue Stanzzbefehl den Zyklus, beträgt die Wartezeit max. 60 ms.

Mit der Option *Read Release and Punch Release Feature* kann man einen Lese- oder Stanzbefehl vorzeitig auslösen. Damit fallen die Startzeiten weg. Beim Kartenleser gewinnt man 21 ms, beim Stanzer sind es 37 ms zusätzliche Verarbeitungszeit. Wichtig ist, dass der Befehl *Start Read Feed*

Tab. 3: d-Operationsergänzung für die bedingte Verzweigungsinstruktion (kleiner Auszug)

d-Op.	Indikator
9	Lochung im Kanal 9 des Vorschubbandes? (siehe Der Schnelldrucker 1403)
A	letzte Karte im Kartenleser ?
B-G	Konsolschalter X ein? (X=B–G)
H	Kartenleser in Arbeit?
I	Kartenstanzer in Arbeit?
K	Ende des Magnetbandes erreicht?
L	Übertragungsfehler beim Magnetband ?
P	Drucker in Arbeit?
/	Vergleichsoperation zwischen den Datenfeldern der A- und B-Adresse, ungleich?
R	Drucker, Vorschub in Arbeit?
S	Vergleichsoperation, gleich?

$\equiv$ 8 rechtzeitig von einem *Read a Card* $\equiv$ 1 gefolgt wird. Zum Stanzen lautet der Befehl *Start Punch Feed* $\equiv$ 9, dem der eigentliche Stanzbefehl *Punch a Card* $\equiv$ 4 rechtzeitig folgen muss. Durch diese Option erhöht sich weder die Lese- noch Stanzgeschwindigkeit. Aber man gewinnt dadurch Verarbeitungszeit *zwischen* der Ein- und Ausgabe.

Multiplikationen und Divisionen müssen per Programm gemacht werden. Das Manual [7] beschreibt zwei Routinen, mit denen man multiplizieren und dividieren kann unter Benutzung der standardmäßig vorhanden Instruktionen (addieren, subtrahieren). Mit der Option *Multiply-Divide Feature* lassen sich diese Operationen mit einer Hardware-Instruktion durchführen. Der Speicherbedarf für die Programmversionen (ohne Arbeitsbereiche) beträgt bei der Multiplikation etwa 75 Speicherplätze, für die Division ca. 250 Speicherplätze. Die benötigte Rechenzeit beider Methoden hängt stark von der Stellenzahl, bei den Programmversionen auch von den Zahlenwerten der Operanden ab. Bei Versuchen mit 1- bis 10-stelligen Operanden schwankt der Zeitgewinn zwischen 5-fach (10-stellig) bis 40-fach (1-stellig) (siehe [7, *Processing Time*]).

Die Verwendung der Indexregister (siehe Indexierung) ist eine zusätzliche Option. Sie macht die Programmierung einfacher, benötigt etwas weniger Instruktionen und erhöht die Rechengeschwindigkeit. Mit dieser Option erhält man auch zwei Registerinstruktionen: *Store A-Address Register* und *Store B-Address Register*, die es ermöglichen, die A- und B-Adressen *nach* einer beliebigen Instruktion zu speichern. Das vereinfacht z. B. das Verarbeiten von Datensätzen mit variabler Länge und ermöglicht, die Rücksprungadrese für Unterprogrammaufrufe zu speichern.

5.3.1 Bedingungen abfragen – *Branch*

Eine einfache Verzweigungsinstruktion ist vierstellig, z. B. `B III` mit `III` als neue Instruktionsadresse (Abb. 3 auf Seite 7).

Während der Programmausführung können Ereignisse auftreten, die einen Indikator setzen, z. B. Überlauf, Lesefehler, letzte Karte, etc. Die Instruktion `B III d` (*Branch if Indicator On*) fragt diese Indikatoren ab, und zwar spezifiziert die Operationsergänzung `d`, welcher Indikator abgefragt werden soll. Die Tabelle 3 zeigt einen Ausschnitt, welche Indikatoren abgefragt werden können.

Je nach angeschlossenen Peripheriegeräten wie Lochstreifenleser/-stanzer, Belegschriftleser, Datenfernübertragung, usw. existieren zusätzlich Instruktion, mit denen der Gerätestatus abgefragt werden kann (siehe [5, *D-modifier indicators for five-character branch*]).

5.3.2 Kombinierte Instruktionen

Zu den kombinierten Instruktionen gehören die bereits beschriebenen Ein- und Ausgabeinstruktionen (siehe Maschinencode).

Einige Instruktionen kombinieren die Ausführung mit einer unbedingten Verzweigung, z. B. *Read and Branch* ≡ `1 III`. Die angehängte Verzweigungsadresse '`III`' ist bei fast allen Instruktionen möglich, die keine A-Adresse zur Ausführung benötigen. Dazu eignen sich alle ein- und zweistelligen Instruktionen. Beispiel für eine zweistellige Instruktion: *Control Carriage and Branch* ≡ `F III 1`, die vor der Verzweigung einen Seitenvorschub am Drucker auslöst. Eine Ausnahme ist die Instruktion *Clear Storage and Branch* ≡ `/ III BBB`, die Speicherplätze ab der B-Adresse löscht und anschließend die Instruktion mit der Adresse '`III`' ausführt.

Der Zeit- und Speichergewinn ist gering. Eine einstellige Operation benötigt 2 Maschinenzyklen (ohne I/O), die unbedingte Verzweigung ist vierstellig und braucht 5 Zyklen. Die Kombination der beiden Instruktion zu einer ist ebenfalls vierstellig. Der Zeitgewinn (Prozessor) beträgt 2 Zyklen, man gewinnt einen Speicherplatz.

Den Vorteil der kombinierten Verzweigungsinstruktion erkennt man z. B. beim Laden von Programmen mit der letzten Instruktion auf einer Karte: `1 026` (siehe Programmladen von Lochkarten und Tabelle 1, letzte Anweisung).

5.3.3 Ketten – *Chaining*

Um das Ketten von Instruktionen besser zu verstehen muss man sich die Arbeitsweise der beiden Adress-Register A und B anschauen. Eine häufig verwendeten Instruktion *Move Character to A or B Word Mark* kopiert Datenfelder von einem Speicherbereich (A-Adresse) zu einem Bereich mit der B-Adresse: `M AAA BBB`. Nach der Ausführung enthalten die A- und B-Register die alten Adressen minus der Anzahl von Zeichen, die übertragen wurden. Wieviel Zeichen übertragen werden hängt von der Größe der Felder ab. Grundsätzlich beendet das kürzere Feld die Übertragung. Der Einfachheit wegen sollen beide Felder gleich groß sein. Dies bedeutet, dass nach der Übertragung die beiden Register die Anfangsadressen der benachbarten Felder enthalten. Um diese Felder ebenfalls zu übertragen braucht man keine Adressen mehr anzugeben, die einstellige Instruktion `M` veranlasst automatisch, die A- und B-Adressen von der vorherigen Operation zu übernehmen.

Allgemein bedeutet dies, dass man keine Adressen angeben muss, wenn die Ausführung der vorherigen Instruktion die gewünschten Adressen hinterlassen hat. Falls nur die Zielfelder nebeneinander liegen genügt es, die A-Adressen anzugeben: `M AAA`. Durch geschicktes Anlegen der Datenfelder und Programmierung kann man auf diese Weise Speicherplatz effizienter nutzen und Rechenzeit einsparen. Beispiel: Falls ein Feld mit der Adresse `XYZ` ein bestimmtes Zeichen `d` enthält soll zu der Adresse `III` gesprungen werden: *Branch if Character Equal* ≡ `B III XYZ d`. Wenn das zu prüfende Datenfeld z. B. fünfstellig ist, lautet die Instruktionsfolge: `B III XYZ d B B B B`.

Tab. 4: Instruktionssatz der 1401 (Ausschnitt)

Op	d?	Description	Op	d?	Description
1		read a card	D		move numeric
2		write a line	Y		move zone
4		punch a card	L	✘	load character to A word mark
A		add	M	✘	move character to A or B word mark
S		subtract	C		compare
?		zero and add	F	✘	control carriage
!		zero and subtract	K	✘	select stacker
,		set workmark	B	✘	branch
)		clear word mark	N		no operation
/		clear storage	.		halt

5.3.4 Der Instruktionssatz

Der Umfang an *Grundinstruktionen* ist gering (ca. 30) und reicht für viele kommerzielle Anwendungen aus (Tabelle 4).

Ein ✘ in Spalte 'd?' bedeutet, dass diese Instruktion einen zusätzlichen Operanden haben kann, der die Instruktion modifiziert. Beispiel: `K 2` ≡ *select stacker 2* (1402). Der d-Operand bei der bedingten Verzweigungsinstruktion `B III d` beschreibt die Bedingung. Eine ausführliche Übersicht aller Instruktionen enthält die Webseite [5, *Instructions*]. Instruktionen, die im Zusammenhang einer Hardware-Option ausgeführt werden können, sind nicht in der Tabelle 4 aufgeführt, z. B. `H` ≡ *Store B-Address Register*.

5.4 Der Boot-Vorgang – *Clear-Storage, Bootstrap*

Der Magnetkern-Speicher behält auch nach dem Ausschalten des Rechners seine Informationen[16]. Falls man den Speicherinhalt nicht analysieren will (*Debugging*) wird man den Speicher vor der Benutzung restlos löschen. Es gibt eine Löschinstruktion *Clear Storage* ≡ `/ III`, die ab der Adresse '`III`' den Speicher abwärts bis zur nächsten Hunderter-Adresse mit Blanks löscht, einschließlich aller Wortmarken.
Beispiel: Löschen des gesamten Druckbereichs 201 – 332 ≡ `/ 332 /`. Diese Instruktionsfolge nutzt das Ketten aus. Die erste Instruktion löscht den Speicherbereich 332 ➞ 300, danach enthält das B-Register die Hunderter-Adresse 300 – 1 = 299, die mit der nächsten Instruktion ohne Adresse, einstellig, den Speicher von 299 bis 200 löscht.

Das funktioniert auch an der untersten Speichergrenze, d. h. ein Löschbefehl für den Kartenlesebereich 001 – 080 liefert 000 – 1= –1. Diese Adresse existiert nicht und wird durch die höchste vorhandene Adresse ersetzt (Modulo max. Kernspeicheradresse). Bei einem 4 k Rechner enthält das B-Register die Adresse 3999 ≡ `I99`. Man erhält also mit diesem Löschbefehl gleichzeitig die maximale Speichergröße des Rechners.

Der gesamte Speicher der 1401 wird per Software gelöscht, standardmäßig untergebracht auf zwei Lochkarten, gefolgt von einer 3. Karte, die ein kleines Programm enthält, den sogenannten *Boot-Strap*. Diese Kartenfolge wird normalerweise von den Assemblern oder Compilern vor dem eigentlichen Programm generiert.

[16] Nach Einschätzung des Autors sollte man sich nicht darauf verlassen.

Zum *Laden* und Ausführen eines neuen Programms mit dem Kartenleser werden die Programmkarten in den Kartenleser gelegt und anschließend die Taste LADEN am Kartenleser gedrückt[17].

Diese Taste bewirkt folgendes:

- alle Wortmarken im Speicherbereich 002 bis 080 werden gelöscht;
- in Speicherplatz 001 wird eine Wortmarke gesetzt;
- das Instruktions-Register wird auf Adresse 001 gesetzt und
- der Kartenleser wird gestartet.

Sobald der Inhalt der ersten Karte in die Speicherbereiche 001 bis 080 übertragen worden ist führt der Rechner den Befehl ab der Instruktionsadresse 001 aus.

```
1 ,008015,022026,030034,041,045,053,0570731026
2 L072116,105106,110117B101/I99,027A075029)027B0010270B026/0991,001/001117I00
```

Betrachtet man den Inhalt der ersten Karte passiert folgendes: , 008 015 ≡ *set word mark* in Adresse 008 und 015, gerade rechtzeitig, um die nächste Instruktion , 022 026 ausführen zu können. Wir erinnern uns, dass eine gültige Instruktion immer eine Wortmarke haben muss. Die erste Wortmarke wurde durch die Hardware der LADEN-Taste gesetzt.

Da die Instruktion , ≡ *set word mark* zwei Adressen haben kann, kann immer eine Wortmarke mehr gesetzt werden als für die nächste Instruktion notwendig ist. Mit diesen zusätzlichen Wortmarken werden die Instruktionen vorbereitet, die ausgeführt werden sollen, in diesem Fall die Kombi-Instruktion ab Spalte 41: 1 026 ≡ *read a card, branch to 026*. Damit wird die zweite Karte eingelesen, die Wortmarken bleiben. Die Instruktionen auf der zweiten Karte müssen so angeordnet sein, dass sie in das vorher gesetzte Wortmarkenmuster passen.

Eine genaue Analyse der *Clear-Storage*- und *Bootstrap*-Karte erfolgt im Anhang *Clear-Storage* und *Bootstrap*.

5.5 Bibliotheken

Um das Programmieren zu vereinfachen enthält der Assembler AUTOCODER eine Macrobibliothek mit einigen Standardroutinen. Diese Bibliothek kann der Benutzer mit eigenen Routinen (Macros) erweitern. Es gibt verschiedene Möglichkeiten, wie Macros in ein Programm eingebunden werden können:

- als feste Sequenz, d.h. die Instruktionen werden unverändert 1:1 an der Stelle des Aufrufs eingefügt;
- als Modell mit Parameterersetzung; es gibt Pflicht- und optionale Parameter. Die symbolischen Parameter im Modell werden durch aktuelle Argumente im Aufruf ersetzt. Fehlen optionale Argumente im Aufruf werden die dazugehörigen Anweisungen *nicht* eingefügt.

Zu den Standardmacros gehören

CALL fügt ein Macro ein, Parameterersetzung ist möglich; der Macroaufruf wird durch eine unbedingte Verzweigungsinstruktion ersetzt;

[17] Im Unterschied dazu *startet* die START-Taste ein bereits geladenes Programm, z. B. nach einem programmierten oder manuellen Stopp oder im Fehlerfall, nach Beseitigung des Fehlers.

INCLD im Prinzip wie **CALL**, ohne Verzweigungsinstruktiuon;

CHAIN n wiederholt die vorangegangene Instruktion ohne A-/B-Adressen n-mal;

OVLAY leitet eine *Overlay*-Operation ein; diese Programmsequenz ermöglicht das Nachladen von Programmteilen (Lochkartenversion);

TOVLY wie **OVLAY**, zum Nachladen vom Magnetband;

MA erlaubt Adressenberechnung $\geq$ 4k (siehe Adressen größer als 3999).

Zu den immer wiederkehrenden Aufgaben gehört das Programmieren der Ein- und Ausgaberoutinen. Besondere Anforderungen muss an die Verwaltung von Magnetbändern gestellt werden. Dazu verwendet man sogenannte *Label*, die am Anfang und am Ende eines Bandes geschrieben werden und helfen sollen, dass Bänder nicht unbeabsichtigt überschrieben und gelöscht werden können und die richtigen Bänder verarbeitet werden. Sie helfen auch, die Datensicherheit zu erhöhen durch geeignete Fehlererkennung und -behandlung.

IBM bietet dazu die Macrobibliothek IOCS (*Input/Output Control System*) an, die die Programmierung der Ein-/Ausgaben unterstützt [11].

Diese Macros sind Modelle, die mittels Parameterersetzung an die Problemstellung angepasst werden können. Sie werden zu den Standardmacros (siehe oben) in die Macrobibliothek des Assemblers eingefügt.

Die Konfiguration der Macros geschieht am Anfang des Anwendungsprogramms. Dazu stehen zwei Eingaben zur Verfügung[18]:

DIOCS *Descriptive IOCs*; beschreibt allgemein Rechnerkonfiguration und Ein-/Ausgabedateien, die benutzt werden sollen;

DTF *Define The File*; beschreibt detailliert jede Ein-Ausgabedatei.

Mit diesen Vorgaben generiert der Assembler zusammen mit den IOCS-Macros die Instruktionen zum Lesen, Schreiben, zur Verarbeitung der Labels und Fehlerbehandlung der Dateien. Statt der Instruktionen *Read a Card, Punch a Card, Write a Line, Read Tape, Write Tape* verwendet man die nachstehenden Macros zusammen mit dem Dateinamen und eventuellen Parametern.

OPEN aktiviert das Gerät oder die Datei, überprüft bei Magnetbändern das korrekte *Header label*;

CLOSE schreibt bei Magnetbändern das *Trailer label* und deaktiviert die Datei oder das Gerät;

GET stellt einen Eingabesatz zur Verfügung;

PUT erzeugt einen Ausgabesatz.

Die Macros **OPEN** und **CLOSE** sorgen für ein sicheres Verarbeiten von Banddateien (*Label handling*)/.

Die Einbindung der IOCS-Macros empfiehlt sich vor allem bei *Overlap*-Operationen (siehe Verarbeitungsgeschwindigkeit –*Timing*), da eine effiziente Anwendung von Overlap-Instruktionen sehr gute Kenntnisse und Erfahrungen des Programmierers voraussetzt.

[18] Diese Vorgehensweise ähnelt der Beschreibung in COBOL: `ENVIRONMENT DIVISION / INPUT-OUTPUT SECTION / FILE-CONTROL`.

5.6 Selbst modifizierende Programmierung

Daten und Instruktionen im Kernspeicher der 1401 unterscheiden sich im Prinzip nicht. Daten können als Instruktionen verwendet werden, wenn sie ein gültiges Instruktionsformat und einen gültigen Instruktionscode (*Op Code*) *mit* Wortmarke besitzen. Umgekehrt stellen Instruktionen Datenfelder variabler Länge dar. Es gibt keinen Speicherschutz für Programme.

Das ermöglicht laufenden Programmen in Abhängigkeit von Entscheidungen, ihre eigenen Adressen und/oder Instruktionen selber zu modifizieren.

Beispiel: Die Instruktion `N xyz _` ≡ `No Operation`, dient dazu, die Daten `xyz` zu überspringen und mit der nächsten gültigen Instruktion fortzufahren. Ändert ein Programm die Instruktion `N` z. B. in ein `B` um, verzweigt das Programm zur Adresse `xyz`. Programmierer machen von dieser Technik Gebrauch, um Anweisungen zu modifizieren bzw. zu (de)-aktivieren. Diese Vorgehensweise wendet man auch bei großen Programmen an, um sie auszutesten ohne sie erneut übersetzen zu müssen.

5.7 *Self-Checking Features*

Referenz: https://ibm-1401.info/1401GuidePosterV9.html

The IBM 1401 was with much self-checking in hardware, in order to improve the reliability of the input, output, and CPU processing. These features included:

1401 CPU self-checks

- Memory parity

1402 self-checks

- Automatic hole count check between read stations
- Automatic Hollerith validity check before storage entry
- Automatic hole count check between punch station and punch check station

1403 self checks

- Automatic parity check of character to be printed
- Automatic validity check of character to be printed
- Hammer action to be performed/performed

729 Magnetic Tape self checks

- Vertical parity check by character
- Horizontal check by record
- Two-gap head verifies validity at the time data is written
- Dual-level sensing provides additional checking of tape read and write operations and error-free operation on valid data.
- IBM phase encoding provides highly reliable method of bit encoding on hypertape so all single-bit and most double-bit errors can be corrected.

6 Die Kartenlese- und Stanzeinheit 1402

Die 1402 vereinigt einen Kartenleser und -stanzer als eine Einheit und gehört zur Grundausstattung einer 1401. Damit lässt sich der Rechner programmieren ohne Bandeinheit (siehe auch SPS und Autocoder). Selbst der FORTRAN-Compiler existiert in einer Lochkarten-Version, bestehend aus mehr als 60 Phasen (Overlay). Über einige technische Details wurde bereits in den Kapiteln Binäre Verarbeitung und Verarbeitungsgeschwindigkeit – *Timing* geschrieben.

Die 1402 Einheit besitzt 5 Ausgabefächer für die gelesenen und/oder gestanzten Lochkarten, die programmgesteuert angesprochen werden können: der Kartenleser die ersten drei von rechts, der Stanzer die ersten drei von links, d. h. das mittlere Ausgabefach kann sowohl Karten vom Leser als auch vom Stanzer aufnehmen. Nach einer Lese- und/oder Stanzoperation kann mit der Instruktion *Select Stacker* ≡ `K d` das gewünschte Fach mit der Operationsergänzung `d` angewählt werden. Da das mittlere Fach vom Leser und Stanzer erreicht werden kann lassen sich auf bequeme Weise gestanzte Karten in einen gelesenen Kartenstapel einfügen.

Die Hardware-Option *Punch Feed Read Feature* erlaubt es, Karten auf der Stanzerseite einzulesen und *zusätzliche* Informationen in die Karten zu stanzen. Dazu wird die Stanzeinheit mit einem zusätzlichen Bürstensatz *vor* dem Stanzwerk versehen[19], der den Inhalte der Karte in den Kartenlesebereich überträgt zwecks Verarbeitung. Die neu zu stanzenden Informationen müssen sich im Stanzbereich befinden. Stellen alle Zeichen vom Lesebereich mit den zusätzlichen Zeichen vom Stanzbereich gültige Lochkombinationen dar, werden die Zeichen des Stanzbereiches in die Lochkarte gestanzt.

Der Lochkartenleser stoppt nach Einlesen der letzten Karte. Das erlaubt dem Operateur, Karten nachzulegen. Eine besondere Bedeutung kommt dem Konsolschalter A zu, der standardmäßig an jeder 1401 vorhanden ist[20]. Wenn Konsolschalter A eingeschaltet ist, kann der Programmierer die Bedingung *Last Card* mit einer bedingten Verzweigungsanweisung abfragen: `B III A` ≡ *Branch Last Card*, d. h. das Ende des Kartenstapels wird erkannt und man kann die entsprechenden Abschlussroutinen ausführen. Der Operateur muss dazu erneut den Kartenleser starten, damit diese Sprunganweisung ausgeführt wird. Vielfach ist es üblich, das Ende einer Lochkartenfolge mit einer speziellen Kennung zu beenden, z. B. '`99999`' oder '`END`', die im Programm abgefragt wird. Um in dem letzteren Fall das erneute Starten zu umgehen, legt man eine Leerkarte hinter diese Kennungskarte.

Beim Stapelbetrieb (*batch processing*), wenn viele Benutzerjobs mit Programmen und Daten auf ein Magnetband geschrieben werden, hat der Benutzer keine Möglichkeit, den Kartenleser oder Schnelldrucker abzufragen (*Off-line*-Betrieb). In diesem Fall signalisiert das Betriebssystem der 7040 das Ende eines Datenstroms durch eine Steuerkarte, z. B. `$IBSYS` oder `$JOB`-Karte oder wenn das Ende eines Magnetbandes erreicht wird. Umgekehrt weiß das Benutzerprogramm bei der Ausgabe vom Magnetband nicht, auf welcher Zeile der Drucker zu Beginn seiner Druckausgabe steht (siehe dazu Kapitel Der Schnelldrucker 1403).

Die Option *Column Binary Feature* wird ausführlich im Abschnitt Binäre Verarbeitung behandelt. Diese Option ermöglicht es auch, Lochkarten mit fremden Kodierungen einzulesen und zu verarbeiten.

[19] Lochkarten werden mechanische abgetastet durch sogenannte Bürsten.
[20] Die Schalter B bis G sind eine Hardware-Option.

Typ	Anwendung	Zeichen	Wieder-holung	max. Druckleistung in Zeilen/min
A	kommerziell	48	5	600
H	wissenschaftlich	48	5	600
N	rein numerisch	16	15	1285
T	Text Groß-/Kleinschreibung	120	2	ca. 200 ‡

‡ keine verlässliche Angabe

Tab. 5: 1403-Druckerketten

7 Der Schnelldrucker 1403

Referenz: https://en.wikipedia.org/wiki/IBM_1403

Der IBM 1403 Schnelldrucker ist von seiner Wirkungsweise her ein Kettendrucker, der zeilenweise eine Endlos-Papierbahn beschreibt (Zeilendrucker). Das Papier wird ruckweise vorgeschoben. Eine umlaufende Kette enthält mehrfach die Drucktypen (siehe Tabelle 5). Durch elektromagnetisch betätigte Hämmer wird die Papierbahn und das Farbband gegen das gewünschte Zeichen der Kette gedrückt. Die Anzahl der Hämmer entspricht der Anzahl der Schreibstellen, z. B. 132. Der Anschlag eines Hammers muss so kurz sein, dass kein verschmiertes Schriftbild entsteht. Die Anschläge sind zeitlich so versetzt, dass nie zwei Hämmer gleichzeitig betätigt werden, da sonst die Kette unter Umständen zu stark abgebremst würde (siehe auch »*Animation of the print train on the IBM 1403 N1 printer* https://righto.com/ibm1401/printchain-n1.html).

Das Funktionsprinzip erzeugt ein Druckbild, bei dem die Druckzeichen einer Zeile immer waagerecht exakt ausgerichtet sind. Eventuelle Ungenauigkeiten äußern sich höchstens in minimalen horizontalen Abstandsabweichungen, die vom Leser nicht so störend wahrgenommen werden wie bei vertikalen Abweichungen, wie sie bei Walzen-/Trommeldruckern etc. auftreten können.

Der Druckvorgang ist mechanisch und elektrisch trickreich gelöst [12].

Die Kette – ähnlich wie die Glieder eine Fahrradkette – nimmt 240 Zeichen auf. Je nach Anwendung gibt es unterschiedliche Ausführungen (Tabelle 5). Standardmäßig enthält sie die 26 Zeichen des englischen Alphabets, 10 numerische Zeichen und 12 Sonderzeichen. Diese 48 Zeichen passen 5-mal auf die Kette. Damit ist der Drucker in der Lage, eine Zeile innerhalb von 100 ms zu drucken, unabhängig vom Zeichen selbst und von der Anzahl der Zeichen in der Zeile. Das entspricht einer permanenten Druckleistung von 600 Zeilen pro Minute.

Wie bereits im Kapitel Codierung erwähnt kommen für kommerzielle und wissenschaftliche Anwendungen zwei verschiedene Zeichensätze zum Einsatz (*A/H chain*). Aber auch die Verwendung anderer Zeichenketten, sogar mit kundenspezifischen Symbolen ist möglich. Je nach Größe und daraus folgend der Anzahl der Wiederholungen des Zeichensatzes auf der Kette variiert die Druckleistung (Tabelle 5). Das Austauschen einer Druckerkette ist relativ einfach, da die Kette in einer Kasette untergebracht ist, die man mit wenigen Handgriffen im Drucker wechseln kann.

Der Drucker ist so konzipiert, dass er zwei oder mehr schmale Formulare (z. B. Adressetikettten) nebeneinander gleichzeitig drucken kann. Auch mehrfach Endlosformulare mit Durchschlag sind möglich.

Eine weitere Vorrichtung, die Vorschubsteuerung (*tape controlled carriage*), vereinfacht vor allem im kommerziellen Bereich das Ausfüllen von vorgedruckten Formularen, z. B. Schecks, Rechnungen, Adressetiketten etc., bei denen nur in bestimmte Zeilen gedruckt wird. Die Steuerung besteht aus einem Endlos-Lochband[21] mit 12 Kanälen/Spuren (*channel*) und genauso vielen Spalten wie Zeilen auf eine Seite des Endlosformulars gedruckt werden können (z. B. 72 Zeilen pro Seite). Dieses Endlos-Lochband ist mit dem Zeilenvorschub des Druckers über eine manuell bedienbare Kupplung verbunden. Durch Lochungen in den 12 Kanäle markiert man Zeilen, die man direkt ansteuern will oder die auch abgefragt werden können. Dies lässt sich vorteilhaft beim Ausfüllen von Formularen einsetzen. Im Prinzip können alle 12 Kanäle benutzt werden, aber grundsätzlich ist Kanal 1 reserviert für eine Lochung, die den Anfang einer Seite darstellt[22]. Eine Lochung in Kanal 9 signalisiert die letzte bedruckbare Zeile einer Seite. Ein 1401-Programmierer fragt nach jedem Druckbefehl ab, ob Kanal 9 eine Lochung anzeigt. Wenn ja, wird das Formular vorgeschoben bis zum Kanal 1. Damit wird vermieden, dass über den Falz gedruckt wird.

Hinweis: Wortmarken im Druckbereich haben keine Auswirkung auf den Druck, sie werden ignoriert. Es gibt eine Instruktion (*write word marks*), die den Inhalt des Druckpuffers ausdruckt und anschließend eine Zeile druckt, die in den Druckpositionen, die eine Wortmarke enthalten, eine '1' druckt (hilfreich bei der Entwicklung von 1401-Programmen).

Neben dem bequemen Austausch von Druckerzeichen/-ketten gibt es viele weitere Optionen, mit denen die Verarbeitung der Druckdaten verbessert werden kann. Beispiel:

Der Druckvorgang einer Zeile dauert 100 ms. Davon ist die 1401 für 84 ms gesperrt, d. h. sie kann in dieser Zeit keine weiteren Instruktionen ausführen. Mit dem optionalen Druckpuffer beträgt diese Sperrzeit nur 2 ms, bis zum Drucken der nächsten Zeile verbleiben 98 ms. Mit der Instruktion *branch if printer busy* kann man die Sperrzeit abfragen und mit *branch if printer carriage busy* auch den Vorschub, um die freie Zeit für andere Aufgaben zu benutzen, z. B. den Lesevorgang einer Lochkarte zu initiieren (siehe auch Programmierung).

> Der IBM 1403 Schnelldrucker hatte eine sehr lange Betriebszeit. 1959 kam er mit der 1401 auf den Markt und wurde, mit Modifikationen, noch bis in die 1990-er Jahre in Verbindung mit anderen IBM-Systeme, z. B. S/370, eingesetzt.

[21] Das Lochband wird zu einer Schlaufe zusammen geklebt.

[22] In FORTRAN wird das erste Zeichen einer Druckzeile nicht gedruckt sondern als Vorschubzeichen interpretiert. Eine '1' bedeutet sofortiger Vorschub nach Kanal 1, eine '0' eine Leerzeile vor dem Druck.
Der Kanal 1 wird auch angesteuert durch die Taste GRUNDSTELLUNG/CARRIAGE RESTORE KEY am Drucker. Fehlt zu einem Vorschubzeichen eine entsprechende Lochung im Vorschubband, schiebt der Drucker im Schnellvorschub (fast 2 m/s) das Papier vor, bis …… der Papiervorrat leer ist oder der Maschinenbediener den Drucker stoppt. Damit das nicht passieren kann stanzt man ein Loch in alle 12 Kanäle, jeweils um eine Zeile versetzt.

8 Resümee

Der vorliegende Bericht stellt nur einen kleinen Ausschnitt aus dem Anwendungsgebiet der 1401 dar. Er basiert auf den Erfahrungen in einem Rechenzentrum, welches einen Großrechner für technisch-/wissenschaftliche Anwendungen betrieb [1]. Dort wurde die 1401 fast ausschließlich zur Durchführung des Stapelbetriebs mit einer 7040 eingesetzt. Deshalb fehlen Beschreibungen von anderen Anwendungsgebieten und Peripheriegeräten. Auf die Programmierung wurde nur dann eingegangen, wenn sie nach Meinung des Autors zu den Besonderheiten der 1401 gehören.

8.1 Worin liegt das Erfolgskonzept der 1401?

Es gibt *kein* Betriebssystem. Der Instruktionssatz ist überschaubar (Tabelle 4, Seite 16), die Speicherkapazität bzw. der Adressraum ist beschränkt.

Alle entwickelten Anwendungen sind im Prinzip *Stand-Alone*-Programme. Die Programmierung und Ausführung dieser Programme ist direkt und hängt im Prinzip nur von der vorhandenen Hardware ab. Es sind keine Betriebssystemkenntnisse erforderlich. Mit geringem Aufwand lassen sich einfache Programme erstellen, wie z. B. Lochkarten auflisten (tabellieren) und/oder duplizieren, Gruppensummen bilden, formatierte Ausgabelisten, Lieferscheine und Rechnungen erstellen, Schecks ausfüllen. Im Gegensatz dazu war das Stecken von Schalttafeln für die gleichen Aufgaben mit Tabelliermaschinen und Kartendopplern aufwändiger. Historisch gesehen hat IBM mit dem System 1400 die elektro-mechanische Datenverarbeitung abgelöst.

Will man verschiedene Anwendungsprogramme mit der 1401 ausführen, ist das wiederholte Laden mit Lochkarten lästig. Statt dessen lassen sich Programme auch vom Band laden (siehe Programmladen vom Magnetband). Es liegt alleine in der Hand des Programmierers, mit den vorhandenen Programmiermöglichkeiten, z. B. Overlay-Technik, eine Programmbibliothek auf Magnetband zu organisieren. Damit kann man sich sein 'eigenes Betriebssystem' bauen.

Zu der weiten Verbreitung der 1401-Rechner-Familie hatte auch der Umstand beigetragen, dass 1401 Programme und Dokumentationen von IBM kostenlos zur Verfügung gestellt wurden (EPL, *European Program Library*, Paris). Im Prinzip entsprach dieses Vorgehen von IBM dem heutigen *Open-Source*-Gedanken.

Eingangs wurde schon erwähnt, dass durch die große Skalierbarkeit der 1400-Serie die Hardware- und Softwareausstattung an die jeweiligen Anforderungen angepasst und praktisch ohne Investitionsverluste auch erweitert werden konnte, und das zu sehr günstigen Kosten. Vermutlich wird der Kostenfaktor der entscheidende Faktor bei der Auswahl des Systems gewesen sein.

8.2 Ausblick

Im Kapitel Codierung wurde gezeigt, welche Probleme bei der Kodierung und Darstellung von Zeichen aufgetreten waren. Dieses galt nicht nur für die Darstellung auf Lochkarten sondern umfasste auch die Kodierung bei Magnetbändern, Lochstreifenleser, Datenfernübertragung und anderen Rechnerfamilien wie die 7000-Großrechner und viele mehr. Diese unterschiedlichen Kodierungen erschwerten den Datenaustausch, auch zwischen anderen IBM-Geräten und IBM-Rechnern.

Mit Einführung der S/360-Architektur im Jahre 1964 legte IBM sich auf ein einheitliches Konzept fest für das Daten- und Speicherformat, den Instruktionssatz und die Schnittstellen zu anderen

IBM-Geräten. Mit dem *Byte* als kleinste adressierbare Speichereinheit und Vereinheitlichung anderer Systemkomponenten war das Ende der 1401-Ära vorhersehbar. IBM bot auf den S/360-Rechnern 1401-Emulationsprogramme an, um den Umstieg auf die neue Rechnerarchitektur zu erleichtern.

Die Rechenanlage 7040/1401 der TH Darmstadt wurde 1975 stillgelegt und abgebaut.

Interessant ist das Softwarepaket SIMH [13], das Emulationen für viele historische Rechnersysteme enthält. Dieses Paket stellt einen 1401-Simulator zur Verfügung (I1401), der Kommandozeilen orientiert arbeitet. Ronald Mak entwickelte eine grafische Oberfläche ROPE [14] für den Assembler AUTOCODER, mit der man interaktiv 1401-Programme schreiben, testen und ausführen kann (siehe Anhang *Ron's Own Programming Environment – ROPE*).

Abb. 4: Speicherlöschroutine – die ersten zwei Lochkarten

```
№  1       6       11      16      21      26      31      36      41      46      51      56      61      66      71      76
1. ,008015,022026,030034,041,045,053,0570731026
2. L072116,105106,110117B101/I99,027A075029)027B0010270B026/0991,001/001117I00
```

A *Clear-Storage* und *Bootstrap*

A.1 Programmladen von Lochkarten

Um ein neues Programm in den Kernspeicher zu laden, muss der bisherige Inhalt gelöscht (*clear storage*) und der Ladevorgang initialisiert werden (*Bootstrap*).

Die im Folgenden beschriebene Löschroutine entstammt dem IBM-1401-Handbuch [7]. Sie kommt mit dem einfachsten Befehlssatz ohne optionale Instruktionen aus.

Die Routine wird im Maschinencode auf zwei Lochkarten codiert. Abbildung 4 zeigt den Inhalt der Lochkarte, einschließlich der Spaltenposition. Lädt man diese Karten (Taste LADEN, siehe Der Boot-Vorgang – *Clear-Storage, Bootstrap*), steht das Abbild der Lochkarte in den Kernspeicherzellen 001 bis 080.

Listing 1: Inhalt der 1. Speicherlöschkarte

```
1  LABEL   OP     OPERANDS         CT   LOCN   INSTRUCTION TYPE   CARD A-ADDR   B-ADDR
2
3          ORG    1                     0001
4          SW     8,15              7   0001   , 008 015            4   008      015
5          SW     22,26             7   0008   , 022 026            4   022      026
6          SW     30,34             7   0015   , 030 034            4   030      034
7          SW     41                4   0022   , 041                4   041
8          SW     45                4   0026   , 045                4   045
9          SW     53                4   0030   , 053                4   053
10         SW     57,73             7   0034   , 057 073            5   057      073
11         R      26                4   0041   1 026                5   026
```

Listing 1 zeigt den dazugehörigen Assemblercode. In der Spalte INSTRUCTIONS erkennt man den Maschinencode der Lochkarte, die Spalten LABEL, OP und OPERANDS enthalten die symbolischen Autocoder-Anweisungen (*Mnemonic Op Code*), CT und LOCN die Instruktionslänge und -adresse. Im Prinzip setzt diese Karte nur Wortmarken für sich selbst (8, 15, 22, 26, 30, 34, 41, 45) und für Instruktionen der zweiten Karte (53, 57, 73). Die Instruktionen und Positionen der Wortmarken sind so ausgeklügelt, dass sie auch auf die Instruktionen der zweiten Karte passen. Diese wird mit der letzten Instruktion `1 026 _` eingelesen. Hier erkennt man einen Vorteil der kombinierten Instruktion. Würde man die letzte Instruktion in zwei zerlegen, `1 B 026 _`, könnte die Sprunganweisung nicht mehr ausgeführt werden, da sie durch die neue Karte überschrieben worden wäre.

Die letzte Instruktion der ersten Karte springt auf die Adresse 026 der zweiten Karte: `/ I99` ≡ *Clear Storage* ab Adresse `I99` ≡ 3999[23] (siehe Listing 2). Sie löscht den Bereich von 3999 bis 3900. Damit der nächste Hunderter-Bereich gelöscht wird, muss die A-Adresse um 100 vermindert werden. Da es sich um dreistellige Adressen handelt (keine Datenfelder), funktioniert

[23] Als Adresse ist die höchste Adresse des Modells einzutragen, hier für eine 4 k-Maschine.

eine Subtraktion mit 100 nicht. Statt dessen addiert man die konstante Komplementäradresse 'I00' ≡ 3900 vom Speicherplatz 073–075 zu der A-Adresse in der Instruktion A 075 029. Damit die Instruktion / nicht mit in die Addition einbezogen wird, grenzt man das Adressfeld von der Instruktion temporär ab, in dem man vorher eine Wortmarke in 027 setzt, addiert und dann die Wortmarke wieder entfernt.

Die nächste Anweisung B 001 027 0 fragt die Hunderter-Stelle 027 der *Clear Storage*-Instruktion ab. Wenn sie eine 0 enthält sind alle Speicherplätze von 100 bis 3999 gelöscht. In diesem Fall werden die Instruktionen ab Speicherplatz 001 ausgeführt.

Listing 2: Inhalt der 2. Speicherlöschkarte

```
1  LABEL   OP    OPERANDS             CT   LOCN   INSTRUCTION TYPE   CARD A-ADDR   B-ADDR
2
3          ORG   1                         0001
4          LCA   072,116               7   0001   L 072 116            4   072      116
5          SW    105,106               7   0008   , 105 106            4   105      106
6          SW    110,117               7   0015   , 110 117            4   110      117
7          B     101                   4   0022   B 101                4   101
8  CSI99   CS    3999                  4   0026   / I99                4  3999
9          SW    027                   4   0030   , 027                4   027
10         A     AD3900,CSI99+3        7   0034   A 075 029            5   075      029
11         CW    027                   4   0041   ) 027                5   027
12         BCE   001,CSI99+1,0         8   0045   B 001 027 0          5   001      027
13         B     026                   4   0053   B 026                5   026
14         DCW   '/0991,001/001117'   16   0072                        5
15 AD3900  DCW   'I00'                 3   0075                        6
```

Die Instruktion L ≡ *Load Character to A Word Mark* überträgt ein Feld von der A-Adresse zur B-Adresse, wobei die Länge des zu übertragenden Feldes die Wortmarke im A-Feld bestimmt. Das zu übertragende Feld wird von Position 72 der Lochkarte von rechts nach links in die Speicheradresse 116 übertragen. Die Übertragung endet mit der ersten Wortmarke in Speicherplatz 53 (gesetzt in der 1. Karte). Dann steht ab der Adresse 101 der Text '/0991,001/001117' (Listing 3). Mit den Wortmarken in 105, 106, 110 und 117 entstehen daraus ausführbare Instruktionen, außerhalb des Kartenlesespeichers, die durch den anschließenden Sprungbefehl nach Adresse 101 ausgeführt werden.

Listing 3: Inhalt der des Speicherbereichs 101–116

```
12 LABEL   OP    OPERANDS             CT   LOCN   INSTRUCTION TYPE   CARD A-ADDR   B-ADDR
13
14         ORG   101                       0101
15         CS    099                   4   0101   / 099                7   099
16         R                           1   0105   1                    7
17         SW    001                   4   0106   , 001                7   001
18         CS    001,117               7   0110   / 001 117            7   001      117
19         DCW   ' '                   1   0117                        7
```

Die Sequenz ab 101 'reinigt das Feld': der Kartenlesebereich wird gelöscht (/ 099), eine neue Lochkarte eingelesen, eine Wortmarke in 001 gesetzt und sich mit Hilfe einer Kombi-Instruktion selber gelöscht (/ 001 117). Das Instruktionsregister erwartet ab Spalte 1 eine ausführbare Instruktion.

Das ist die 3. Karte, die *Bootstrap*-Karte,[24] die die Vorbereitungen trifft, dass sich das Programm selber laden kann. Die im Folgenden beschriebene Methode ist eine von vielen Möglichkeiten,

[24] nach einem alten, englischen Sprichwort „*he pulls himself up by his bootstraps*“

Abb. 5: Bootstrapkarte mit Beispiel von Listing 4

```
№ 1    6    11   16   21   26   31   36   41   46   51   56   61   66   71   76
3. ,008015,022029,036040,047054,061068,072/061039              ,0010011040
   M3632702F1.340GOOD MORNING 1401       L031363,340341,343347,0400401040
                                         /333080
```

ein Programm zu laden.

Um diesen Ladevorgang besser verdeutlichen zu können betrachten wir ein kleines Beispiel (Listing 4). Das generierte Kartendeck ohne die ersten beiden Karten, die den Speicher löschen, sind in Abbildung 5 dargestellt.

Listing 4: Kleines Beispiel-Programm

```
20 LABEL  OP    OPERANDS                CT  LOCN  INSTRUCTION TYPE
21
22 START  MCW   'GOOD MORNING 1401',270  7  0333  M 363 270
23 AGAIN  W                              1  0340  2
24        CC    1                        2  0341  F 1
25        H     AGAIN                    4  0343  . 340
26        DCW   @GOOD MORNING 1401@     17  0363              LIT
27        END   START                                 / 333 080
```

Die Bootstrap-Karte 'pflastert' den Weg für die nachfolgenden Programm-Karten:

1. in dem Speicherbereich 061 bis 072 werden die Instruktionen `, 001 001 1 040 _` mit den Wortmarken in 061, 068 und 072 ausführbar gemacht[25]. Weiter geht es mit Schritt 3.

2. Die Instruktionen von 061 bis 072 setzen eine Wortmarken in 001, lesen eine neue Karte ein und springen zur Adresse 040 auf der neuen Karte. Weiter geht es mit Schritt 4.

3. Anschließend löscht die Kombi-Instruktion `/ 061 039 _` alles hinter sich (039 bis 001) und springt zur Adresse 061. Weiter geht es mit Schritt 2.

4. Jede weitere Programmkarte muss einen bestimmten Aufbau haben:

   - Bereich 001 bis 039: Instruktionen und Daten
   - Bereich 040 bis 060: Laden dieser Daten in die korrekte Speicheradresse und Setzen der benötigten Wortmarken. Dabei ist es wichtig, dass diese Instruktionen in die vom Schritt 1 gesetzten Wortmarken (040, 047, 054 und 061) passen, da der Lochkartenlesebereich mit jeder neuen Programmkarte überschrieben wird, aber die Wortmarken bestehen bleiben.
   - Bereich 061 bis 072: Wiederholung der Sequenz von Schritt 2.

5. Die letzte Karte enthält nur die Kombi-Instruktion `/ xyz 080`, die den Kartenlesebereich löscht und zu der ersten ausführbaren Adresse des geladenen Programms '`xyz`' springt.

[25] mit einer Wortmarke in der Instruktionsadresse und einer abschließenden Wortmarke, falls keine weitere Instruktion folgt

Die Spalte INSTRUCTION des Listings 4 enthält den Maschinencode des Beispielprogramms. Der Textstring '`GOOD MORNING 1401`' wird in den Druckbereich übertragen, die Zeile gedruckt und ein Papiervorschub zum Kanal 1 (neue Seite) veranlasst. Danach hält das Programm an und springt nach erneutem Starten zur Adresse 0340 (`. 340`).

Dieser Code erscheint auf der zweiten Karte in Abbildung 5 ab Spalte 001. Den String '`GOOD MORNING 1401`' wandelt der Assembler in ein Datenfeld (*Literal*) um und speichert ihn hinter die letzte Programmanweisung (hier Adresse 0348 bis 0363, 17 Zeichen). Zur Erinnerung: Instruktionen werden von links nach rechts, Datenfelder von rechts nach links verarbeitet und adressiert (außer Lese-/Schreibbefehle für Magnetbänder und ähnliche Geräte). Die Spalten 040 bis 060 enthalten die Anweisungen zum Setzten der korrekten Wortmarken.

Die hier beschriebene Methode ist eine von vielen und zeigt nur das Grundprinzip eines *selbstladenden* Programms.

A.2 Programmladen vom Magnetband

Statt Programme von Lochkarten zu laden lässt sich ein übersetztes Programm einschließlich der *Clear Storage, Bootstrap*-Routinen auch von einem Magnetband laden. Dazu befindet sich an der Zentraleinheit der 1401 eine Taste TAPE LOAD, die automatisch von Bandeinheit 1 einen Bandsatz in den Speicherbereich 001 einliest und die Instruktion ab Speicherplatz 001 ausführt.

Anders als bei Lochkarten kann das Magnetband auch die Wortmarken übertragen, und zwar durch ein vorangestelltes *word-separator*-Zeichen (A841), welches beim Zurücklesen in den Kernspeicher automatisch eine Wortmarke dem folgenden Zeichen hinzufügt. In diesem Fall entfallen die Instruktionen zum Setzen der Wortmarken, das Format zum Laden von Konstanten und Instruktionen vereinfacht sich dadurch. Außerdem wird pro Instruktion und/oder Konstante jeweils ein Bandsatz geschrieben, was bei Programmen mit vielen Instruktionen auch zu vielen ungeblockten Bandsätzen führt.

Bei der häufigen Ausführung verschiedener Anwendungen kann es vorteilhaft sein, alle Programme auf *einem* Magnetband zu speichern (Programmbibliothek), vorausgesetzt, dass eine Bandeinheit permanent verfügbar ist. Die Auswahl, welche Anwendung geladen wird, trifft man am einfachsten mit den freien Konsolschaltern B – G[26]. Da die 1401 kein Betriebssystem hat, muss sich der Anwender selber eine Methode ausdenken, wie er das gewünschte Programm vom Magnetband selektieren kann. Es gibt viele Möglichkeiten.

Beispiel: Jedes Programm enthält eine Programm-ID als Konstante, z. B. eine Zahl zwischen 1 und 63 und eine kleine Routine, die die Konsolschalter abfragt und daraus ebenfalls eine Zahl bildet, die mit der Programm-ID vergleicht und entweder die Ausführung beginnt oder das nächste Programm nachlädt. Da der Platzbedarf auf dem Magnetband wegen des ungeblockten Bandformats sehr groß wird, können solche Suchoperationen ziemlich lange dauern.

Eleganter und wesentlich schneller geht es, wenn man das Programm z. B. mit Lochkarten normal lädt und dann

- eine Gruppenmarke mit Wortmarke hinter das letzte Datenfeld bzw. die letzte Instruktion setzt und anschließend

[26] Mit der Option *Console Inquiry Station 1407* besteht die Möglichkeit, Informationen mittels einer Konsolschreibmaschine zwischen Programm und Benutzer auszutauschen. Diese Option wurde oft in Zusammenhang mit einem Plattenspeicher IBM 1405 (*Disk Storage Unit - RAMAC*) für Direktabfragen verwendet.

- den ganzen Speicherbereich mit der Instruktion *Write Tape with Word Marks* als ein Satz auf Band schreibt.

Achtung: eine Gruppenmarke mit Wortmarke beendet die Datenübertragung auf ein Magnetband. Deshalb müssen alle im Programmen vorhandenen Gruppenmarken mit Wortmarken vor der Übertragung 'entschärft' (z.B. mit *Clear Word Mark*) und nach der Übertragung wieder gesetzt werden.

Das Laden und Selektieren ist Aufgabe des Programmierers, es gibt kein Betriebssystem.

B *Ron's Own Programming Environment – ROPE*

Die folgenden Abbildungen zeigen die Arbeitsumgebung des 1401-Simulators ROPE.

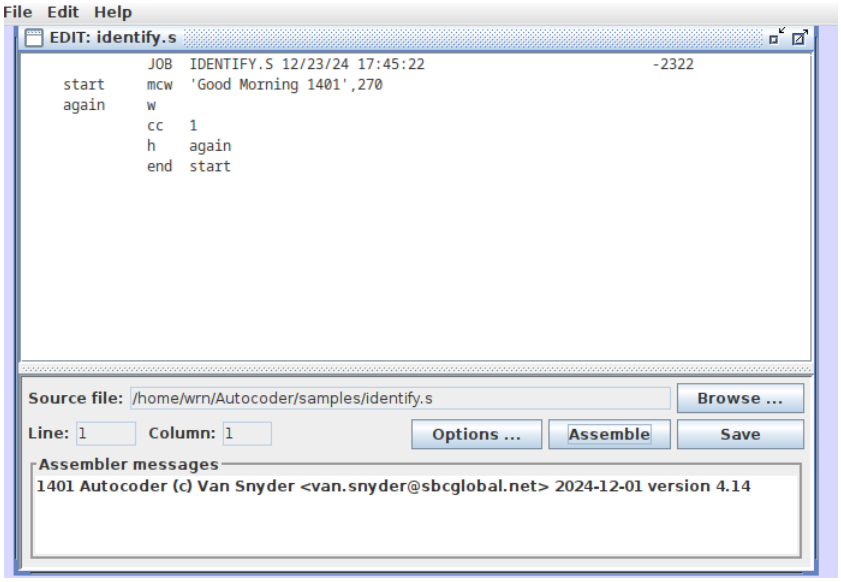


Abb. 6: ROPE-Editor für Autocoder-Programme

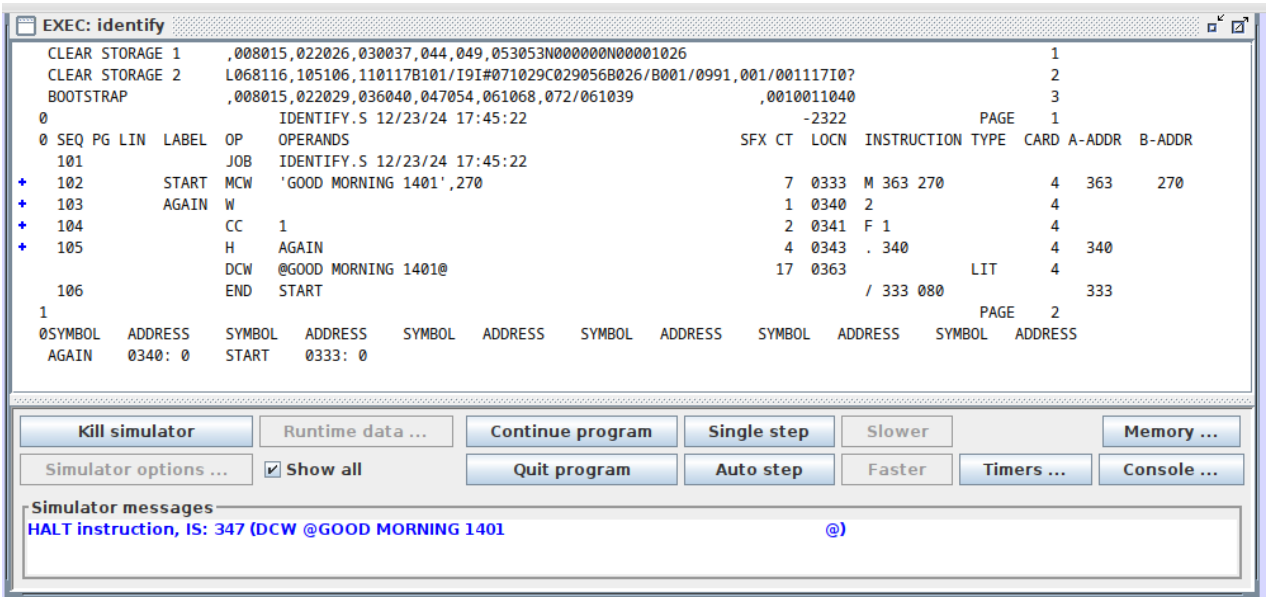


Abb. 7: Assembler-Listing und Schaltflächen für die Programmausführung

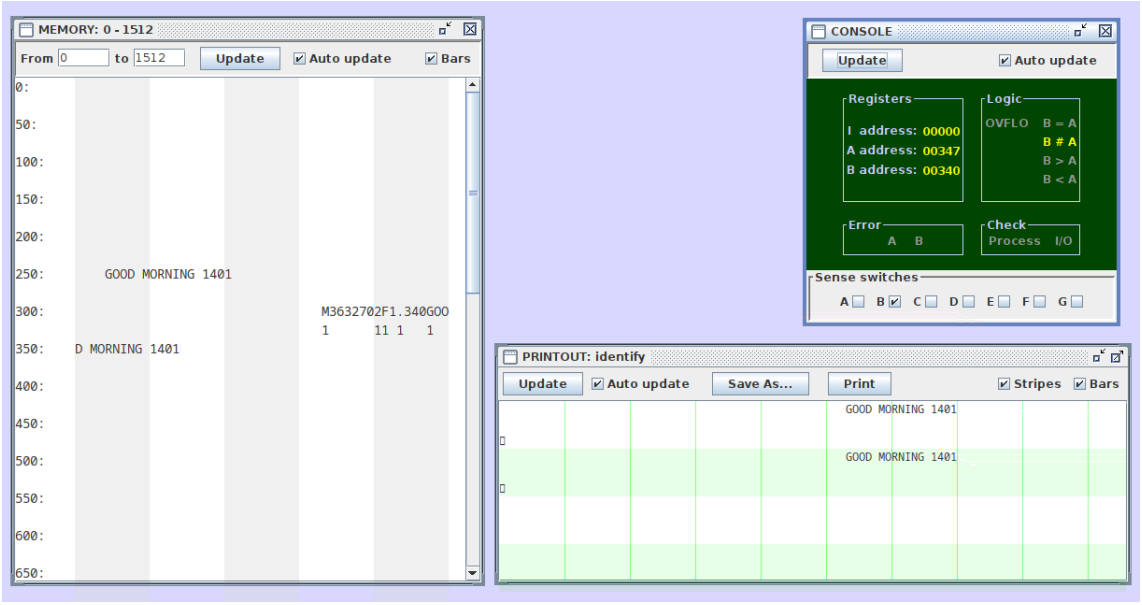


Abb. 8: links: Kernspeicherauszug; die Wortmarken werden durch eine '1' dargestellt; rechts oben: Konsolen-Anzeige mit Adressregister, Logik- und Fehlerindikatoren sowie Status der Konsolenschalter; rechts unten: Schnelldrucker-Ausgabe

Abb. 9: Konfiguration des 1401-Simulators SIMH

Literatur

[1] Wolf-Rainer Novender. *As time goes by … Der Hochschulrechner IBM 7040 der TH Darmstadt*. Okt. 2022. URL: www.thm.de/iem/wolf-rainer-novender.

[2] Dag Spicer. *Back to Life. The story behind CHM's IBM 1401 restauration*. 2009. URL: https://computerhistory.org/core-magazine/.

[3] Günther Sandner und Hans Spengler. *Die Entwicklung der Datenverarbeitung von Hollerith Lochkartenmaschinen zu IBM Enterprise-Servern*. Eigenverlag G. Sandner und H. Spengler Böblingen, 2006. ISBN: 978-3-00-019690-4.

[4] *Custom Features for IBM 1401, 1440, and 1460 Data Processing Systems*. IBM System Reference Libraries A24-3313.

[5] Van Snyder. *IBM 1401 Programming Hints*. 27. Jan. 2015. URL: https://ibm-1401.info/VansOverview.html.

[6] *The IBM 1401 Demo Lab and Restoration Project. Computer History Museum*. URL: www.ibm-1401.info/index.html.

[7] *IBM 1401 Data Processing System. Reference Manual*. A24-1403-5. Apr. 1962.

[8] *Autocoder (on Tape) Language. Specifications and Operating Procedures IBM 1401 and 1460*. IBM Systems Reference Library C24-3319. Nov. 1964.

[9] *COBOL (on Tape) Specifications. IBM 1401*. IBM Systems Reference Library C24-1492.

[10] *Fortran Specifications and Operating Procedures. IBM 1401*. IBM Systems Reference Library C24-1455. 27. Apr. 1965.

[11] *Input/Output Control System (on Tape). Specifications and Operating Procedures IBM 1401 and 1460*. IBM Systems Reference Library C24-1462. 23. Dez. 1964.

[12] *IBM 1403 Printer Component Description*. IBM System Reference Libraries GA24-3073.

[13] Robert Supnik. *Computer Simulation and History*. 16. Juli 2024. URL: https://simh.trailing-edge.com/.

[14] Ron Mak. *1401 Software Development. available for use on Win7, Mac and Linux*. 7. Jan. 2024. URL: https://ibm-1401.info/1401SoftwDevel.html.