CH 660 242 A5

(19) SCHWEIZERISCHE EIDGENOSSENSCHAFT
BUNDESAMT FÜR GEISTIGES EIGENTUM

(11) CH 660 242 A5

(51) Int. Cl.[4]: G 06 F 9/44

Erfindungspatent für die Schweiz und Liechtenstein
Schweizerisch-liechtensteinischer Patentschutzvertrag vom 22. Dezember 1978

(12) PATENTSCHRIFT A5

(21) Gesuchsnummer: 4973/83

(22) Anmeldungsdatum: 13.09.1983

(30) Priorität(en): 13.09.1982 US 417014

(24) Patent erteilt: 31.03.1987

(45) Patentschrift veröffentlicht: 31.03.1987

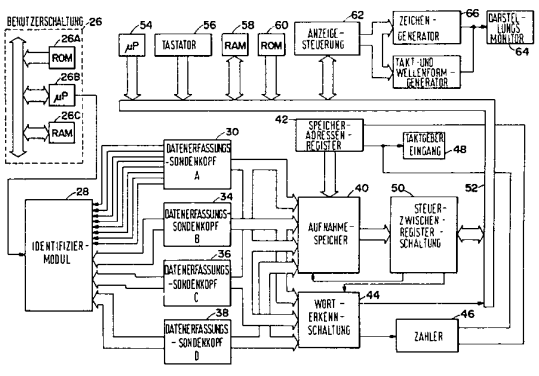
(73) Inhaber:
Tektronix Inc., Beaverton/OR (US)

(72) Erfinder:
Pettet, Mark E., Hillsboro/OR (US)
Hoeren, Gerd H., Lake Oswego/OR (US)

(74) Vertreter:
Dr. Conrad A. Riederer, Bad Ragaz

(54) Anordnung und Verfahren zur inversen Assemblierung.

(57) Das Verfahren zur inversen Assemblierung umfasst den Schritt der Umwandlung einer Adress/-Daten- und Steuerbusabwicklung, die vom Logikanalysator erfasst wird, in einen Satz mnemonischer Angaben in Assemblersprache. Die mnemonischen Angaben entsprechen einem anderen Satz mnemonischer Angaben, die ursprünglich von einem Benutzer als Teil eines Assemblersprachenprogramms in einen Wirtscomputer eingegeben wurden. Die Entscheidungsbaumtabellen teilen dem inversen Assembler die Art mit, in der er das inverse Assemblierverfahren ausführen muss. Der Satz von mnemonischen Angaben in Assemblersprache wird zum Austesten des Assemblersprachprogramms verwendet. Die Anordnung zur inversen Assemblierung besitzt Datenkompressionseigenschaft. Die Speicherung wird gemäss einer Firmware durchgeführt, die in einem Nur-Lese-Speicher (60) gespeichert ist. Das Format der Entscheidungsbaumtabellen entspricht unmittelbar dem Format eines Satzes von Benutzerdokumentationen, die vom Hersteller des Mikrocomputers zur Verfügung gestellt werden. Weiterhin weisen die Adress-/Daten- und Steuerbusabwicklungen Assemblersprachebefehle und Dateninformationen auf. Den Busabwicklungen, die für Assemblersprachebefehle charakteristisch sind, wird ein anderes Kennzeichenbit und den Busabwicklungen, die für Dateninformationen charakteristisch sind, ein wiederum ein anderes Kennzeichenbit zugeordnet. Die Kennzeichenbits versetzen die Anordnung in die Lage, zwischen Befehls- und Dateninformationen während des inversen Assemblerverfahrens zu unterscheiden.



PATENTANSPRÜCHE

1. Anordnung zur inversen Assemblierung, mit der ein Code, der einem gespeicherten Programm zugeordnet ist, in einen entsprechenden Satz mnemonischer Angaben für die Interpretation durch einen Benutzer umgewandelt wird, dadurch gekennzeichnet, dass sie eine Einrichtung (56) für die Eingabe umzuwandelnder Informationen aufweist, die eine Liste von Codeangaben, die dem Code entsprechen, der dem gespeicherten Programm entspricht, und eine Liste der entsprechenden mnemonischen Ausdrücke umfasst, dass eine erste Einrichtung (54, 58, 60) für den Empfang und für die Speicherung der umzuwandelnden Informationen vorgesehen ist, die in einem Speicher (58) in Form eines Satzes von Entscheidungsbaumtabellen abspeicherbar sind, dass eine zweite Einrichtung (30, 34, 36, 38, 40) für die Erfassung und für die Speicherung des dem gespeicherten Programm entsprechenden Code vorgesehen ist, dass die erste Einrichtung (54, 58, 60) den in der zweiten Einrichtung (40) gespeicherten Code mit einem entsprechenden Satz der in der ersten Einrichtung gespeicherten, umzuwandelnden Informationen vergleicht und dass eine Einrichtung (60, 62, 64, 66) zur Erfassung des entsprechenden Satzes umzuwandelnder Informationen von der ersten Einrichtung (54, 58, 60) und zur Darstellung dieses Satzes umzuwandelnder Informationen zur Interpretation für den Benutzer vorgesehen ist.

2. Anordnung nach Anspruch 1, dadurch gekennzeichnet, dass der dem gespeicherten Programm zugeordnete Code Befehle und Dateninformationen enthält und dass eine auf den Code, der dem gespeicherten Programm zugeordnet ist, ansprechende Einrichtung (28) vorhanden ist, die zwischen den Befehlen und den Dateninformationen unterscheidet und die den Befehlen ein erstes Kennzeichen und den Dateninformationen ein zweites Kennzeichen zuordnet, wobei die zweite Einrichtung (30, 34, 36, 38, 40) den Code, der dem gespeicherten Programm zugeordnet ist, und die diesem zugeordneten ersten und zweiten Kennzeichen erfasst und den Code und die diesem entsprechenden ersten und zweiten Kennzeichen abspeichert.

3. Anordnung nach Anspruch 2, dadurch gekennzeichnet, dass das gespeicherte Programm im Speicher (26a) einer Benutzerschaltung (26) enthalten ist, dass der dem Programm zugeordnete Code von einem in der Benutzerschaltung (26) angeordneten Mikroprozessor (26b) erzeugt wird und dass das Format der Tabellen, die die im Speicher (58) der ersten Einrichtung gespeicherten Entscheidungsbaumtabellen bilden, dem Format eines zugeordneten Satzes von Benutzungsdokumenten entspricht, die die Charakteristik des Mikroprozessors beschreiben.

4. Verfahren zum Betrieb der Anordnung nach einem der vorausgehenden Ansprüche, dadurch gekennzeichnet, dass die von einer Benutzerschaltung (26) erzeugbaren binären Codeangaben und die entsprechenden mnemonischen Angaben vorab in Tabellen gespeichert werden, die nach einer Entscheidungsbaumstruktur miteinander verknüpft sind, dass von der Benutzerschaltung (26) während eines Arbeitsablaufs ausgangsseitig erzeugte binäre Codeangaben abgespeichert werden, bis eine vorgebbare binäre Codeangabe festgestellt ist und dass die von der Benutzerschaltung erhaltenen und abgespeicherten binären Codeangaben über die Tabellen mnemonische Angaben in Assemblersprache zurückverwandelt und angezeigt werden.

Die Erfindung bezieht sich auf eine Anordnung und ein Verfahren zur inversen Assemblierung, mit denen ein Code, der einem gespeicherten Programm zugeordnet ist, in einen entsprechenden Satz mnemonischer Angaben für die Interpretation durch einen Benutzer umgewandelt wird. Mit einem inversen Assembler wird der ausführbare Ausgangscode eines Assemblers eines Datenverarbeitungssytems in einen entsprechenden Satz mnemonischer Angaben einer Assemblersprache für das Austesten der Software umgewandelt, die dem ausführbaren Code zugeordnet ist.

Logikanalysatoren führen typischerweise die Funktion eines inversen Assemblers aus. Ein inverser Assembler wird dazu benutzt, einen von einem Assembler eines Datenverarbeitungssystems erzeugten Satz von Angaben eines ausführbaren Code in einen entsprechenden Satz von Angaben einer Assemblersprache zu übersetzen, damit das Softwarepaket, das den ausführbaren Code darstellt, ausgetestet werden kann. Beispielsweise wird, wie in Fig. 1a dargestellt, während der Entwicklung der Software der ausführbare Code entweder von einem Assembler oder einem Compiler erzeugt. Der Assembler gibt einen Satz von Befehlen in der Assemblersprache von einem Benutzer ein und erzeugt hieraus den ausführbaren Code. Der ausführbare Code enthält eine Vielzahl binärer Codeangaben, die den Befehlen der Assemblersprache entsprechen. Benutzt wird der ausführbare Code von einem Zielsystem (das z.B. ein Mikroprozessor sein kann). In Fig. 1b ist eine Softwareanalysiermethode gezeigt, die von einem inversen Assembler für die logische Analyse ausgeführt wird, wobei der vom Zielsystem benutzte ausführbare Code in einen entsprechenden Satz mnemonischer Angaben einer Assemblersprache umgewandelt wird. Der entsprechende Satz mnemonischer Angaben der Assemblersprache wird für das Austesten des Satzes von Assemblersprachebefehlen verwendet. Wenn das Zielsystem, das in Fig. 1a gezeigt ist, die Assemblersprachebefehle gemäss dem entsprechenden Satz mnemonischer Angaben der Assemblersprache richtig ausgeführt hat (indem die vom Benutzer gewünschten Ergebnisse entwickelt werden), dann war der Satz vom Assemblersprachebefehlen (oder einer höheren Sprache, die den Assemblersprachebefehlen entspricht), richtig geschrieben.

Wie oben erwähnt, wandelt der Assembler den Satz der vom Benutzer eingegebenen Assemblersprachebefehle um und erzeugt den ausführbaren Code, der eine Vielzahl binärer Codeangaben enthält. Das Zielsystem ist in der Regel ein Mikroprozessor, jedoch ist das Zielsystem nicht hierauf beschränkt. Der Mikroprozessor führt die vom Assembler erzeugten binären Codeangaben aus. Die Ausführung dieser binären Codeangaben kann durch einen Logikanalysator überwacht werden. Um den Code auszutesten, ist es notwendig, dass die binären Codeangaben (die vom Mikroprozessor ausgeführt werden) in den ursprünglichen Satz von Assemblersprachebefehlen zurückverwandelt werden, d.h. invers in den entsprechenden Satz mnemonischer Angaben der Assemblersprache umgewandelt werden, damit sie vom Benutzer interpretiert werden können.

Um die Vielzahl von binären Codeangaben in die entsprechenden mnemonischen Angaben der Assemblersprache invers zu assemblieren, ist es erforderlich, Informationen, die für den Aufbau einer Tabelle notwendig sind, einem Speicher zuzuführen und sie in diesem zu speichern. Die Tabelle enthält zwei Informationsspalten: Die erste Spalte besteht aus einer Liste von binären Codeangaben, die alle der möglichen Codeangaben aufweist, die von dem Mikroprozessor erzeugt werden können. Die zweite Spalte weist eine entsprechende Liste von mnemonischen Angaben in Assemblersprache (Befehlen) auf, die den Codeangaben zugeordnet sind.

Für jede binäre Codeangabe, die in die Vielzahl von binären Codeangaben einbezogen ist, die vom Mikroprozessor erzeugt werden, bestand der Vorgang der inversen

Assemblierung aus folgenden Schritten: Ablage der binären Codeangabe als Adresse in der ersten Spalte der im Speicher enthaltenen Tabelle und Kennzeichnung der entsprechenden mnemonischen Angabe in Assemblersprache in der zweiten Spalte. Durch die Benutzung dieser Tabelle wird daher die Vielzahl der binären Codeangaben, die vom Mikroprozessor erzeugt werden, in den entsprechenden Satz mnemonischer Angaben in Assemblersprache invers assembliert.

Bei den bisher üblichen inversen Assemblern war die Aufgabe des Aufbaus der Tabelle ziemlich langsam und mühevoll, wenn nicht gar unmöglich. Für die meisten acht Bit Mikroprozessoren wies die erste Spalte der Tabelle eine Liste von 2^8 (d.h. 256) Eingaben auf. Diese Anzahl an Eingaben ist vernünftig. Wenn jedoch ein sechzehn Bit Mikroprozessor verwendet wurde, dann wies die erste Spalte der Tabelle eine Liste von 2^{16} (d.h. 65 536) Eingaben an binären Codeangaben auf. Ein Benutzer muss deshalb 65 536 binäre Codeangaben und die entsprechenden Befehle für den Aufbau der Tabelle eingeben. Wenn eine solche Tabelle aufgebaut wird, dann wird für die Speicherung der Tabelle eine sehr grosse Speicherkapazität benötigt. Als Folge davon hat sich das Verfahren der inversen Assemblierung bei der Benutzung herkömmlicher Assembler (insbesondere in Verbindung mit dem 16-Bit-Prozessor) als unpraktisch, wenn nicht gar unmöglich erwiesen.

Als andere Möglichkeit war der herkömmliche Assembler mit einem Nur-Lese-Speicher ausgestattet, der mit Firmware kodiert war, die für das Austesten des Satzes von Assemblersprachebefehlen massgebend war, die vom Mikroprozessor ausgeführt wurden. Mit dem Nur-Lese-Speicher war es jedoch nur möglich, die Assemblersprachebefehle, die nur von gewissen speziellen Arten von Mikroprozessoren ausgeführt wurden, zu zerlegen. Wenn andere Mikroprozessoren zur Ausführung dieser Befehle verwendet wurden, konnte die Firmware im ursprünglichen Nur-Lese-Speicher die Befehle nicht zergliedern (oder invers assemblieren).

Es war deshalb notwendig, den Nur-Lese-Speicher zu entfernen und ihn durch einen anderen Nur-Lese-Speicher zu ersetzen, in dem Firmware enthalten war, die die Assemblersprachebefehle genau zergliedern konnte. Die Notwendigkeit, den alten Nur-Lese-Speicher zu entfernen und ihn durch einen neuen zu ersetzen, wenn ein unterschiedlicher Mikroprozessor für die Durchführung des Satzes der Assemblersprachebefehle benutz wurde, schränkte den Umfang der den herkömmlichen inversen Assemblern zugeordneten Benutzung erheblich ein.

Der Erfindung liegt die Aufgabe zugrunde, eine Anordnung und ein Verfahren zur inversen Assemblierung zu entwickeln, mit denen es unter Vermeidung der Nachteile herkömmlicher inverser Assembler möglich ist, die Informationen für den Aufbau der Tabelle unter wesentlicher Verminderung der Zahl der Eingaben der binären Codeangaben und der zugehörigen mnemonischen Angaben in Assemblersprache (Befehle) einzugeben, die für die Tabelle benötigt werden, die dem inversen Assemblierverfahren zugeordnet ist.

Die Aufgabe wird erfindungsgemäss durch die im Anspruch 1 beschriebenen Massnahmen gelöst. Die erfindungsgemässe Anordnung erlaubt die Eingabe aller möglichen binären Codeangaben, die vom Assembler erzeugt werden können, und aller entsprechender mnemonischer Angaben in Assemblersprache (Befehle), die den Codeangaben zugeordnet sind, während des Vorgangs des Aufbaus der Tabelle. die binären Codeangaben und die mnemonischen Angaben in Assemblersprache werden in der Tabelle in Form eines Satzes von Entscheidungsbaumtabellen gespeichert, wobei die jedem Zweig des Entscheidungsbaums zugeordneten Elemente in den inversen Assembler der vorliegenden Erfindung in unmittelbarer Übereinstimmung mit dem Format einer Benutzerdokumentation eingegeben werden, die vom Hersteller des jeweiligen Mikroprozessors veröffentlicht wird, und wobei alle Zweige des Entscheidungsbaums in der durch den Entscheidungsbaum gegebenen Form miteinander verbunden sind. Weiterhin ist die Anordnung gemäss der vorliegenden Erfindung in der Lage, den ausführbaren Code in der Weise invers zu assemblieren, dass die darin enthaltenen Befehle und Dateninformationen voneinander unterschieden werden können. Die Befehle sind innerhalb des ausführbaren Codes durch erste Identifizierungsmittel gekennzeichnet. Die Dateninformationen sind innerhalb des ausführbaren Code durch zweite Identifizierungsmittel gekennzeichnet. Die Befehle und die Dateninformationen und die diesen entsprechenden ersten und zweiten Identifizierungsmittel, die als Teil des ausführbaren Code erfasst werden, werden in einem Aufnahmespeicher eines Erfassungsgerätes (z.B. eines Logikanalysators) abgespeichert und vom inversen Assembler verarbeitet.

Mit der Anordnungsgemäss der vorliegenden Erfindung werden die Schwierigkeiten der herkömmlichen inversen Assembler überwunden, indem die Zahl der Tabelleneingaben, die vom Benutzer für den Tabellenaufbau benötigt wird, vermindert wird. Ferner wird das vom Benutzer durchzuführende Verfahren für die Eingabe der binären Codeangaben und der Befehle, die diesen zugeordnet sind, vereinfacht. Weiterhin hat die Erfindung den Vorteil, dass zwischen Befehlen und Dateninformationen, die aus dem ausführbaren Code invers assembliert werden, unterschieden werden kann.

Weitere Bereiche der Anwendbarkeit der vorliegenden Erfindung sind aus der folgenden Beschreibung ersichtlich. Es versteht sich jedoch, dass die Einzelheiten der Beschreibung und der besonderen Ausführungsbeispiele, die sich auf bevorzugte Ausführungsformen der Erfindung beziehen, zur Erläuterung angegeben sind. Verschiedene Änderungen und Modifikationen sind innerhalb des Erfindungsgedankens möglich und werden für den einschlägigen Fachmann aus der detaillierten Beschreibung ersichtlich.

Die Erfindung wird im folgenden anhand eines in einer Zeichnung dargestellten Ausführungsbeispiels näher erläutert, aus dem sich weitere Merkmale sowie Vorteile ergeben. Es zeigen:

Fig. 1a ein Software-Entwicklungsverfahren, bei dem eine höhere Computersache oder eine Assemblersprache zur Erzeugung eines Satzes eines ausführbaren Codes benutzt wird;

Fig. 1b ein Software-Analysierverfahren, mit dem der ausführbare Code in einen Satz mnemonischer Angaben einer Assemblersprache für das Austesten und die Analyse der Assemblersprache (die vom Benutzer erstellte Software) umgewandelt wird;

Fig. 2 ein Konzept eines Entscheidungsbaums, bei dem es sich um ein allgemein für die Speicherung von Tabellen in der Anordnung gemäss der vorliegenden Erfindung benutztes Konzept handelt;

Fig. 3a und 3b gemeinsam ein Paar von Kathodenstrahlröhrendarstellung durch die Anordnung gemäss der vorliegenden Erfindung, bei dem jede der binären Codeangaben, die von einem Mikroprozessor erzeugt werden können und die entsprechenden Befehle im inversen Assembler als Satz von Tabellen gespeichert werden, die in Entscheidungsbaumform abgespeichert werden, wobei die Tabellen für die Vervollständigung eine wesentlich verminderte Zahl von Eingaben benötigen;

Fig. 4 die Vielzahl der von einem Mikroprozessor erzeugten binären Codeangaben, die Adressen-, Steuer- und

Dateninformationen darstellen, die von der Anordnung gemäss der vorliegenden Erfindung erfasst werden, wobei die dargestellten binären Codeangaben Befehle und Dateninformationen beinhalten, die im Aufnahmespeicher des Erfassungsgeräts (dem Logikanalysator) gespeichert und vor dem inversen Assemblierverfahren, das vom inversen Assembler gemäss der vorliegenden Erfindung ausgeführt wird, vorhanden sind;

Fig. 5 die binären Codeangaben gemäss Fig. 4 nach dem von der Anordnung gemäss der vorliegenden Erfindung durchgeführten inversen Assemblierverfahren, wobei die binären Codeangaben gemäss Fig. 5 einen Satz mnemonischer Angaben in Assemblersprache, die den binären Codeangaben entsprechen, einschliessen;

Fig. 6 ein Blockdiagramm der Anordnung gemäss der vorliegenden Erfindung, das an eine Benutzerschaltung angeschlossen ist, wobei die Benutzerschaltung als einen Teil einen Speicher für die Abspeicherung der binären Codeangaben, die den auszutestenden Assemblersprachebefehlen zugeordnet sind, und den für die Ausführung der binären Codeangaben bestimmten Mikroprozessor enthält, der den ausführbaren Code erzeugt, der die Vielzahl binärer Codeangaben umfasst, die sich auf Lesen, Schreiben und in Reaktion hierauf auf Abholinformationen beziehen.

Es wurde oben bereits dargelegt, dass zur inversen Assemblierung des vom Assembler (d.h. dem Mikroprozessor) ausgegebenen ausführbaren Code, d.h. der binären Codeangaben, die Bildung der Tabelle notwendig war. Bei den herkömmlichen inversen Assemblern war der Prozess, der die Bildung der Tabelle umfasst, sehr langsam, mühevoll, wenn nicht gar unmöglich. Es ist deshalb eine Hauptaufgabe der vorliegenden Erfindung, die Anzahl der Eingaben der binären Codeangaben und der entsprechenden Befehle, die für die Bildung der Tabelle notwendig sind, zu vermindern und zugleich deren Eingabe zu vereinfachen. Die zuerst erwähnte Aufgabe, die Verminderung der Anzahl der Eintragungen, wird durch Erkennung des folgenden allgemeinen Prinzips durchgeführt:

$2^n > 2^i + 2^j$, worin $i + j = n$ sind.

Im Kontext der vorliegenden Erfindung ist jede binäre Codeangaben, wenn der Assembler ein Mikroprozessor ist, und wenn ein 16-Bit-Mikroprozessor verwendet wird, jeder Binärcode, d.h. jede Busabwicklung, die vom Mikroprozessor veranlasst wird, eine 16-Bit-Binärzahl. Es gibt jedoch 2^{16} mögliche Binärzahlen, die vom Mikroprozessor erzeugt werden können, deshalb sind 2^{16} mögliche Befehle vorhanden, die invers assembliert werden müssen. Die herkömmlichen inversen Assembler benötigen die Bildung einer Tabelle und deren Speicherung in einem Speicher. Für das oben angegebene Beispiel hat die Tabelle eine erste Spalte, die eine Liste von 2^{16} Binärzahlen (für einen 16-Bit-Mikroprozessor) aufweist, wobei die zweite Spalte eine entsprechende Liste von 2^{16} Befehlen hat. Mit einer so grossen Zahl an Eingaben war die Tabelle schwierig, wenn nicht gar unmöglich, aufzubauen. Als Alternative benötigte der herkömmliche inverse Assembler den Einsatz eines besonderen Nur-Lese-Speichers (ROM) für die inverse Assemblierung durch einen Mikroprozessor in einem Logikanalysator.

Der inverse Assembler der vorliegenden Erfindung benötigt andererseits ebenfalls die Bildung einer Tabelle. Jedoch hat die im Speicher aufzubauende und abzuspeichernde Tabelle die Struktur eines Enscheidungsbaums, bei dem anstelle einer Tabelle, die eine Liste von 2^{16} Eintragungen benötigt, zwei oder mehr Tabellen verwendet werden, die in Form eines Baums verkettet sind. Für das oben beschriebene Beispiel können zwei miteinander verkettete Tabellen benutzt werden, wobei jede Tabelle in ihrer ersten und zweiten Spalte 2^8 Eintragungen hat. Da $2^{16} > 2^8 + 2^8$ werden in den ersten und der zweiten Spalten der beiden Tabellen weniger Eintragungen benötigt als in der einzigen Tabelle nach dem Stand der Technik. In ähnlicher Weise können vier Tabellen verwendet werden, die in Form eines Entscheidungsbaums miteinander verkettet sind, wobei jede Tabelle 2^4 Eintragungen in ihrer ersten und zweiten Spalte benötigt. Da $2^{16} > 2^4 + 2^4 + 2^4 + 2^4$ ist, ist es viel einfacher vier Entscheidungsbaumtabellen statt einer einzigen Tabelle nach dem Stand der Technik zu bilden. Es wird auf die Fig. 2 hingewiesen, in der ein Beispiel des Entscheidungsbaums-Konzept dargestellt ist, das von der vorliegenden Erfindung verwendet wird. In Fig. 2 sind eine Vielzahl von in Entscheidungsbaumform miteinander verketteter Tabellen dargestellt. Die Zusammensetzung jeder dieser Tabellen wird unter Bezug auf die Fig. 3a und 3b beschrieben. Wenn der inverse Assembler der vorliegenden Erfindung zu Beginn auf die Tabelle 1 verweist, um den ausführbaren Code in einen entsprechenden Satz mnemonischer Angaben in Assemblersprache invers zu assemblieren, wird eine Entscheidung getroffen, entweder auf die Tabelle 2 oder die Tabelle 3 zu verweisen. Wenn auf die Tabelle 2 verwiesen wird, wird eine weitere Entscheidung getroffen, entweder auf die Tabelle 4 oder auf die Tabelle 5 zu verweisen. Wenn auf die Tabelle 5 verwiesen wird, wird eine zusätzliche Entscheidung getroffen, entweder auf die Tabelle 10 oder auf die Tabelle 11 zu verweisen. Durch die Bezugnahme auf die Tabellen 1, 2, 5 und 11 in Kombination wird ein binäres Codewort (d.h. das 16-Bit binäre Codewort) vom inversen Assembler der vorliegenden Erfindung erfasst. Die Ausgabe des Mikroprozessors wird invers in entsprechende mnemonische Angaben einer Assemblersprache assembliert. Wenn ein 16-Bit binärer Code invers assembliert werden muss, werden 2^4 Eintragungen in jeder Spalte der Tabellen 1, 2, 5 und 11 benötigt, um den 16-Bit binären Code in entsprechende mnemonische Angaben in Assemblersprache invers zu assemblieren. Es ist sehr viel einfacher 2^4 Eintragungen in jeder Spalte dieser Tabellen vorzusehen, als 2^{16} Eintragungen in jeder Spalte der einzigen Tabelle einzugeben, die dem inversen Assembler des bekannten Stands der Technik entspricht.

In den Fig. 3a und 3b ist eine Kathodenstrahlröhrenanzeige über den inversen Assembler der vorliegenden Erfindung dargestellt. Die Anzeige legt ein spezifischeres Beispiel der Entscheidungsbaumstruktur der Tabelle dar. In Fig. 3a ist die «OPCODE01»-Tabelle dargestellt. Acht (8) Bit binäre Codeangaben erscheinen in der ersten Spalte 10 der «OPCODE01»-Tabelle. Eine zweite Spalte 12 ist eine «Aufruf»-Spalte, um auf eine andere Tabelle Bezug zu nehmen. Eine dritte Spalte 14 ist eine Darstellungsspalte, um einen Teil einer binären Codeangabe auf einen Befehl zu beziehen. Beispielsweise erscheint eine binäre Codeangabe «O1000XXX» in der ersten Spalte 10 in Fig. 3a. Die binären Zahlen 01000 beziehen sich auf einen Increment-Befehl (INC). Die letzten drei Gattungsbits, «XXX» werden, wenn in dieser Tabelle gesucht wird, nicht beachtet und unabhängig von ihrem Wert zu einer Tabelle, die «REG 16» genannt wird, durchgelassen, wie es von der Spalte 12 bestimmt wird.

In Fig. 3b ist die Tabelle «REG 16» dargestellt. Wenn die drei Gattungsbits «XXX» den Wert «000» haben, dann entspricht dieser der Registeridentifierzierung «AX». Durch die Verkettung der «OPODE01»-Tabelle 1 (Fig. 3a) mit der «REG 16»-Tabelle (Fig. 3b) in Entscheidungsbaumform kann festgestellt werden, dass dem binären Code «01000000» ein Increment-Register-Befehl AX («INC AX») entspricht.

Die in den Fig. 3a und 3b dargestellten Tabellen sind in

einmaliger Weise eingeteilt, um die Eintragung binärer Codeangaben (Fig. 3a) und der diesen zugeordneten Befehle zu vereinfachen. Wie oben dargelegt, führt der Mikroprozessor jeden Befehl aus und erzeugt für jeden der Befehle eine binäre Codeangabe. Der Hersteller des Mikroprozessors (der den Satz Assemblersprachebefehle, die in der auszutestenden Software enthalten sind, verarbeitet) stellt eine Benutzerdokumentation zur Verfügung, in der die vom Mikroprozessor erzeugten binären Codeangaben auf die jeweiligen Befehle bezogen sind. Die Einteilung der Tabellen, die in Fig. 3a und 3b dargestellt sind, ist absichtlich in Übereinstimmung mit der Einteilung der Benutzerdokumentation gebracht, die vom Hersteller des Mikroprozessors zur Verfügung gestellt wird. Als Ergebnis dieser einander angepassten Einteilungen ist es leichter, die Vielzahl der binären Codeangaben und die diesen zugeordneten Befehle in den inversen Assembler der vorliegenden Erfindung während des Verfahrens des Aufbaus der Tabelle einzugeben.

In Fig. 4 ist die Vielzahl der binären Codeangaben dargestellt, wie sie in dem Aufnahmespeicher des inversen Assemblers des Logikanalysators gespeichert sind. Diese Codeangaben werden vom Mikroprozessor entsprechend der Vielzahl der Befehle erzeugt und verarbeitet. Die binären Codeangaben werden durch den Logikanalysator (ein Erfassungsgerät) der vorliegenden Erfindung vom Mikroprozessor aufgenommen und im Aufnahmespeicher abgespeichert. Zu beachten ist, dass diese binären Codeangaben nicht dem vom inversen Assembler des Logikanalysators der vorliegenden Erfindung durchgeführten inversen Assemblierverfahren unterworfen worden sind und deshalb keine zugeordneten mnemonischen Angaben in Assemblersprache aufweisen.

Der Mikroprozessor stellt einen Bestandteil einer Benutzerschaltung dar. Die Benutzerschaltung enthält einen ersten Speicher (d.h. einen Nur-Lese-Speicher) für die Speicherung des Satzes von Assemblersprachebefehlen und einen zweiten Speicher (d.h. einen Speicher RAM mit wahlfreiem Zugriff) für die Speicherung von Daten, die vom Mikroprozessor benutzt werden, wenn die im ersten Speicher enthaltenen Befehle ausgeführt werden. Gemäss Fig. 4 enthält eine erste Informationsspalte 16 Adressdaten, die einerseits die jeweilige Adresse im ersten Speicher der Benutzerschaltung, in dem die Assemblersprachebefehle gespeichert sind, und andererseits die jeweilige Adresse im zweiten Speicher der Benutzerschaltung angeben, in dem die Informationsdaten gespeichert sind. Eine zweite Spalte 18 enthält eine Vielzahl von Kennzeichnungsbits, die für die Unterscheidung zwischen den binären Codeangaben, die den vom Mikroprozessor als Teil des ausführbaren Code erzeugten und im ersten Speicher abgespeicherten Assemblersprachebefehlen entsprechen, und den binären Codeangaben verwendet werden, die den vom Mikroprozessor ebenfalls erzeugten und in dem zweiten Speicher abgespeicherten Dateninformationen entsprechen. Eine dritte Spalte 20 enthält in hexadezimaler Form die Assemblersprachebefehle und die Dateninformationen, die den Adressen zugeordnet sind, die in der ersten Spalte 16 erscheinen. Zu beachten ist jedoch, dass die Befehls- und Dateninformationen, die in der dritten Spalte 20 erscheinen, nicht leicht zu entschlüsseln und zu verstehen sind, da die Befehls- und Dateninformationen in Form hexadezimaler Zeichen erscheinen.

Um die im ersten Speicher der Benutzerschaltung abgespeicherten Assemblersprachebefehle zu überprüfen, ist es erforderlich, dass die in Spalte 20 gemäss Fig. 4 erscheinenden Informationen leicht gelesen werden können, d.h. die vom Mikroprozessor wiedergewonnenen und im Aufnahmespeicher des inversen Assemblers des Logikanalysators abgespeicherten Befehls- und Dateninformationen müssen leicht bestimmt und analysiert werden können, um das Ausmass festzustellen, bis zu dem die in letzterem abgespeicherten Befehls- und Dateninformationen den Satz von Assemblersprachebefehlen genau wiedergeben, die im ersten Speicher der Benutzerschaltung abgespeichert sind.

In Fig. 5 ist das Ergebnis, die Vielzahl der binären Codeangaben, wie sie in Fig. 4 gezeigt sind, nach der Verarbeitung im inversen Assemblierverfahren der vorliegenden Erfindung dargestellt. Eine erste Spalte 22 stellt die gleichen Adressdaten dar, die in der ersten Spalte 16 der Fig. 4 gezeigt sind. Eine zweite Spalte 24 enthält jedoch den Satz mnemonischer Angaben in Assemblersprache, der den Assemblersprachebefehlen und Dateninformationen entspricht, die in der dritten, in Fig. 4 dargestellten Spalte erscheinen. Die mnemonischen Angaben in Assemblersprache stellen die im Aufnahmespeicher des inversen Assemblers des Logikanalysators abgespeicherten Befehls- und Dateninformationen dar und werden vom Softwareentwerfer dazu benutzt, den Satz von Assemblersprachebefehlen, die im ersten Speicher enthalten sind, auszutesten.

Wie in den obigen Abschnitten bereits angedeutet, müssen zur Umwandlung der in Fig. 4 dargestellten Informationen in die in Fig. 5 dargestellten Informationen die Informationen in einen Speicher (einen Speicher mit wahlfreiem Zugriff) eingegeben und darin abgespeichert werden, damit die Tabelle aufgebaut werden kann, die in einer ersten Spalte die Vielzahl der binären Codeangaben, die vom Mikrozprozessor erzeugt werden können, und in einer zweiten Spalte die Befehle der mnemonischen Angaben in Assemblersprache, die den Codeangaben entsprechen, enthält. Die Assemblersprachebefehle in binärer oder vorzugsweise in hexadezimaler Form, die in der dritten Spalte der Fig. 4 gezeigt sind, sind als Index für diese Tabelle in deren erster Spalte angeordnet. Die gespeicherten mnemonischen Ausdrücke in Assemblersprache werden zur Bildung der Anzeige auf dem Logikanalysator, wie in Fig. 5 dargestellt, benutzt. Wenn ein 8 Bit- oder ein Mikroprozessor mit grösserer Bitzahl in der Benutzerschaltung in Verbindung mit dem herkömmlichen inversen Assembler eines Logikanalysators verwendet wurde, erwies sich, wie oben dargelegt, die Aufgabe der Bildung der Tabelle als sehr schwierig, wenn nicht gar unmöglich. Die Entscheidungsbaumstruktur der in dem Speicher mit wahlfreiem Zugriff des Logikanalysators gespeicherten Tabelle gemäss der vorliegenden Erfindung wird, wie in den Fig. 2, 3a und 3b gezeigt, dazu benutzt, die Aufgabe der Bildung der Tabelle zu vereinfachen.

In Fig. 6 ist ein Blockdiagramm des Logikanalysators mit inversem Assembler der vorliegenden Erfindung dargestellt.

Eine Benutzerschaltung 26 enthält einen ersten Speicher 26a (in der Regel einen Nur-Lese-Speicher-ROM), der an einen Systembus angeschlossen ist und in dem die Firmware, (Software, die ausgetestet werden soll), gespeichert ist. Sie enthält auch den Mikroprozessor 26b, der mit dem Systembus verbunden ist und der die Firmware/Software ausführt und den ausführbaren Code, d.h. die Vielzahl der binären Codeangaben in Reaktion hierauf, erzeugt. Sie enthält weiterhin einen zweiten Speicher 26c (in der Regel einen Speicher mit wahlfreiem Zugriff - RAM), der mit dem Systembus verbunden ist. Der Mikroprozessor 26b ist weiterhin mit einem Identifiziermodul 28 verbunden. Das Identifiziermodul 28 empfängt die Menge der binären Codeangaben aus dem Mikroprozessor 26b, unterscheidet zwischen den hierbei erhaltenen Assemblersprachebefehlen, die im ersten Speicher 26a abgespeichert sind, und den erhaltenen Dateninformationen, die in dem zweiten Speicher abgespeichert sind, und ordnet ein Kennzeichnungsbit den vom Mikroprozessor 26b erhaltenen Assemblersprachebefehlen und ein zweites Kennzeichnungsbit den vom Mikroprozessor erhaltenen Dateninformationen zu. Ein charakteristisches

Identifiziermodul, das dazu benutzt werden kann, die Funktion des in Fig. 6 dargestellten Identifiziermoduls 28 auszuführen, ist ein von Tektronix, Inc. unter der Produkt-Nr. PM1XX (d.h. ein PM111 für einen Typ 6809 Mikroprozessor) hergestelltes Modul. Die Patentanmeldung Serial-No. 312 466, die am 19. Oktober 1981 hinterlegt wurde, offenbart Einzelheiten des Aufbaus, die dem Identifiziermodul 28 entsprechen. Der zu der Patentanmeldung Serial-No. 312 466 vom 19. Oktober 1981 gehörige Anmeldungstext wird hiermit in die Beschreibung einbezogen. Als Alternative kann ein anderes Identifiziermodul zur Ausübung der Funktion des Identifiziermoduls 28 verwendet werden. Es kann ein von Tektronix, Inc. unter der Standard-Teilenummer PM109-MC 6800 hergestelltes Modul sein.

Ein Datenerfassungssondenkopf A,30 empfängt die Assemblersprachebefehle, die Dateninformationen und die diesen zugeordneten Sondenbits vom Identifiziermodul 28, mit dem der Datenerfassungssondenkopf A,30 verbunden ist. Die andere Seite des Datenerfassungssondenkopfs A,30 ist mit einem Aufnahmespeicher 40 verbunden. Tatsächlich ist eine Vielzahl von Datenerfassungssondenköpfen einschliesslich eines Datenerfassungssondenkopfs B,34 eines Datenerfassungssondenkopfs C,36 und eines Datenerfassungssondenkopfs D,38 über Datenbusse mit dem Aufnahmespeicher 40 verbunden. Jeder der Sondenköpfe 30, 34, 36 und 38 ist an acht Sondenspitzen angeschlossen. Die Sondenspitzen erfassen eine Vielzahl logischer Signale (die charakteristisch für die Menge von binären Codeangaben sind) von den Anschlüssen eines in der Prüfung befindlichen Produkts, z.B. des Identifiziermoduls 28 oder des der Benutzerschaltung 26 zugeordneten Mikroprozessors 26b. Die Sondenköpfe übertragen die Menge von logischen Signalen zum Aufnahmespeicher 40. Der Aufnahmespeicher 40 ist in Abschnitte eingeteilt, die den einzelnen Sondenköpfen 30, 34, 36 und 38 zugeordnet sind. Mit dem Aufnahmespeicher 40 ist auch ein Speicheradressregister 42 verbunden, das bestimmte Stellen im Aufnahmespeicher 40 adressiert und in Übereinstimmung hiermit die Speicherung der logischen Signale aus den Sondenköpfen in den Stellen veranlasst, die im Aufnahmespeicher über das Speicheradressregister 42 adressiert wurden. Die vom Sondenkopf A,30 erfassten logischen Signale werden im Aufnahmespeicher 40 in Stellen gespeichert, die dem für den Sondenkopf A,30 vorbehaltenen Abschnitt entsprechen. In gleicher Weise werden die den Sondenköpfen 34, 36 und 38 zugeordneten logischen Signale im Aufnahmespeicher 40 in Stellen gespeichert, die vom Speicheradressregister 42 bestimmt werden und den Sondenköpfen B, C und D vorbehaltenen Abschnitten entsprechen.

Jeder der Erfassungssondenköpfe ist mit dem Eingang einer Worterkennungsschaltung 44 verbunden. Die Worterkennungsschaltung 44 stellt fest, ob ein gewünschtes Wort aus der Erfassungssonde im Aufnahmespeicher abgespeichert worden ist und beaufschlagt in Reaktion hierauf den Hauptbus des Logikanalysators und inversen Assemblers mit einem Ausgangssignal. Das Ausgangssignal enthält Daten-, Adressen- und Steuerinformationen. Das gewünschte Wort ist ein Wort, das der Benutzer in ein «Menü» auf der Anzeige eingegeben hat, bevor die Datenerfassung begonnen hat. Die Worterkennungsschaltung ist an einen Zähler 46 angeschlossen und aktiviert diesen Zähler 46 in Reaktion auf den Empfang des gewünschten Worts aus dem Erfassungssondenkopf. Der Zähler 46 wird auch von einem Taktgebereingang 48 beaufschlagt, der den Zähler veranlasst, bis zu einer vorgegebenen Grösse zu zählen. Der Zähler 46 ist mit dem Speicheradressregister 42 verbunden und aktiviert dieses Speicheradressregister 42, wenn der Zählstand im Zähler die vorgegebene Grösse erreicht. Wenn das Speicheradressregister 42 vom Zähler 46 aktiviert wird, wird die Adressierfunktion, die das Speicheradressregister ausübt, beendet. Als Ergebnis wird die weitere Speicherung von logischen Signalen aus den Sondenköpfen im Aufnahmespeicher 40 beendet.

Eine Steuerzwischenregisterschaltung 50 ist an den Ausgang des Aufnahmespeichers 40 angeschlossen und empfängt hierdurch im Auslesezustand die im Aufnahmespeicher 40 enthaltenen Daten. Die Steuerzwischenregisterschaltung 50 ist mit dem Hauptbus 52 des inversen Assemblers der vorliegenden Erfindung verbunden und speist hierdurch die gespeicherten Daten in den Hauptbus ein. Darüberhinaus ist die Steuerzwischenregisterschaltung 50 mit dem Speicheradressregister 42, dem Aufnahmespeicher 40 und der Worterkennungsschaltung 44 verbunden. Wenn die Steuerzwischenregisterschaltung 50 vom Hauptbus 52 Steuerdaten empfängt, steuert die Steuerzwischenregisterschaltung in Reaktion hierauf die Speicherauslesebetriebsart des Aufnahmespeichers 40, die Geschwindigkeit, mit der das Speicheradressregister 42 den Aufnahmespeicher 40 adressiert, und das von der Worterkennungsschaltung 44 empfangene Wort.

Ein Mikroprozessor 54, eine Tastatur 56, ein Speicher mit wahlfreiem Zugriff (RAM) 58, ein Nur-Lese-Speicher (ROM) 60 und eine Anzeigesteuerung 62 sind ebenfalls mit dem Hauptbus 52 des Logikanalysators mit inversem Assembler verbunden.

Der Mikroprozessor 54 ist eine zentrale Verarbeitungseinheit. Er verarbeitet die über die Steuerzwischenregisterschaltung 50 aus dem Aufnahmespeicher 40 wiedergewonnene Information in Übereinstimmung mit den von der Tastatur 56 und den im Nur-Lese-Speicher 60 enthaltenen Befehlen und erzeugt ein entsprechendes Steuersignal. Der Speicher mit wahlfreiem Zugriff 58 speichert die Tabellen in Entscheidungsbaumform, wobei die binären Codeangaben und die diesen zugeordneten Befehle über die Tastatur 56 in den Aufnahmespeicher des Logikanalysators mit inversem Assembler eingegeben und darin gespeichert werden. Die im Nur-Lese-Speicher 60 gespeicherte Firmware veranlasst den Mikroprozessor 54 und versetzt ihn in die Lage, die Tabellen in Entscheidungsbaumform für die Speicherung in dem Speicher mit wahlfreiem Zugriff in Reaktion auf den Empfang der über die Tastatur 56 eingegebenen binären Codeangaben und zugeordneten Befehle zu erstellen. Die Anzeigesteuerung 62 arbeitet gemäss Befehlen, die vom Nur-Lese-Speicher 60 empfangen werden, und gemäss Verarbeitungsbefehlen, die vom Mikroprozessor 54 empfangen werden, zur Erzeugung einer Anzeige auf einem Darstellungsmonitor 64 des Logokanalysators mit inversem Assembler. Die Anzeige auf dem Darstellungsmonitor 64 wird mit Hilfe eines Zeichengenerators 66 erzeugt, der zwischen der Anzeigesteuerung 62 auf der einen Seite und dem Darstellungsmonitor 64 auf der anderen Seite angeordnet ist.

Der Darstellungsmonitor 64 erzeugt die in den Fig. 3a und 3b gezeigten Anzeigen. Der Benutzer bzw. Operator des Logikanalysators mit inversem Assembler beobachtet diese Anzeigen auf dem Darstellungsmonitor 64, wenn er über die Tastatur 56 die binären Codeangaben und die diesen zugeordneten Befehle von der Benutzerdokumentation des Mikroprozessors 26 b der Benutzerschaltung in den Logikanalysator mit inversem Assembler überträgt. Diese binären Codeangaben und Befehle stellen alle möglichen binären Codeangaben und die zugeordneten mnemonischen Angaben in Assemblersprache dar, die vom Mikroprozessor 26b innerhalb der Benutzerschaltung 26 erzeugt werden können.

Die Arbeitsweise des in Fig. 6 des Systemblockdiagramms dargestellten Logikanalysators mit inversem Assembler ist in den folgenden Abschnitten in Verbindung mit den Fig. 2, 3a, 3b, 4 und 5 dargelegt.

Der Operator bzw. Benutzer des Logikanalysators mit inversem Assembler geht von dem Satz der Benutzerdokumentation aus, die dem Mikroprozessor 26b innerhalb der Benutzerschaltung 26 zugeordnet ist. Die Benutzerdokumentation wird für die Bildung der Tabelle verwendet, die in Entscheidungsbaumform im Speicher mit wahlfreiem Zugriff 58 abgespeichert wird, wobei die Information über die Tastatur 56 eingegeben wird. Der Operator bzw. Benutzer veranlasst den Logikanalysator mit inversem Assembler der vorliegenden Erfindung auf dem Darstellungsmonitor 64 eine Anzeige ähnlich derjenigen darzustellen, die in Fig. 3a der Zeichnung gezeigt ist. Unter Benutzung des Satzes der Benutzerdokumentation, der Anzeige auf dem Monitor 64 und der Tastatur 56 füllt der Operator bzw. der Benutzer die in Fig. 3a gezeigten Informationen aus und vervollständigt sie, wobei er die binären Codeangaben in der ersten Spalte 10, die Namen der anderen Tabellen, auf die Bezug genommen wird, in der Spalte 12 und die den binären Codeangaben zugeordneten Befehle in der dritten Spalte 12 einschliesst. Der Operator bzw. Benutzer veranlasst den Loganalysator mit inversem Assembler, auf dem Monitor 64 eine Anzeige auszugeben, die ähnlich derjenigen ist, die in Fig. 3b der Zeichnung dargestellt ist, und er vervollständigt die angezeigte Information. Tatsächlich gibt der Operator bzw. Benutzer unter Bezug auf die Benutzerdokumentation, die vom Hersteller des Mikroprozessors 26b innerhalb der Benutzerschaltung 26 ausgegeben ist, die Menge der binären Codeangaben ein, die von dem Mikroprozessor 26b erzeugt werden kann.

Der Mikroprozessor 54 veranlasst in Übereinstimmung mit den Befehlen, die in der Firmware enthalten sind, die im Nur-Lese-Speicher 60 abgespeichert ist, die Speicherung von Tabellen, wie sie in Fig. 3a und 3b erscheinen, im Speicher mit wahlfreiem Zugriff 58. Die Tabellen enthalten die binären Codeangaben und die zugeordneten mnemonischen Angaben, die über die Tastatur 56 eingegeben werden. Der Mikroprozessor 54 speichert die Tabellen in Form eines Entscheidungsbaum, der in Fig. 2 der Zeichnung dargestellt ist, im Speicher mit wahlfreiem Zugriff 58. Als Folge davon wird vom Logikanalysator mit inversem Assembler eine geringere Anzahl von Eingaben für die Bildung der Tabellen benötigt, die im Speicher mit wahlfreiem Zugriff 58 zu speichern sind.

Wenn die Tabellen im Speicher mit wahlfreiem Zugriff 58 in der Form eines Entscheidungsbaums, wie er in Fig. 2 gezeigt ist, abgespeichert sind, ist der inverse Assembler der vorliegenden Erfindung bereit, die Menge der binären Codeangaben vom Mikroprozessor 26b, der der Benutzerschaltung 26 zugeordnet ist, zu erfassen. Die binäre Information, die dem Satz von Assemblersprachebefehlen, die im Nur-Lese-Speicher 26a der Benutzerschaltung gespeichert sind, entspricht, wird der Analyse und dem Austesten unterzogen. Der Mikroprozessor 26b führt die Befehle, die im Nur-Lese-Speicher 26a abgespeichert sind aus und erzeugt in Reaktion hierauf eine Menge von binären Codeangaben. Das Identifiziermodul 28 empfängt die Menge an binären Codeangaben und ordnet den in der Menge der binären Codeangaben enthaltenen Assemblersprachebefehlen ein Kennzeichnungsbit und ein weiteres Kennzeichnungsbit den Dateninformationen zu, die in der Menge der binären Codeangaben enthalten sind und damit empfangen werden. Der Datenerfassungssondenkopf A,30 erfasst die Menge der binären Codeangaben und der zugeordneten Kennzeichenbits und speichert die binären Codeangaben und die Kennzeichenbits im Aufnahmespeicher 40 des Logikanalysators mit inversem Assembler. Das Speicheradressregister 42 adressiert in Reaktion auf die Signale des Taktgebereingangs 48 die Speicherplätze, die im Aufnahmespeicher 40 dem Sondenkopf A,30 zugeordnet sind. Wenn die Worterkennungsschaltung 44 das gewünschte Wort von einer Erfassungssonde empfängt, erzeugt der Zähler 46 ein Ausgangssignal, das das Speicheradressregister 42 aktiviert. Als Folge davon beendet das Speicheradressregister 42 seine Adressierfunktion, wobei es die Speicherung der vom Sondenkopf A,30 empfangenen binären Codeangaben und der Kennzeichnungsbits im Aufnahmespeicher 40 beendet. Der Aufnahmespeicher 40 enthält dann Informationen, in dem in Fig. 4 der Zeichnung dargestellten Format.

Wie bereits erwähnt, enthält der Speicher mit wahlfreiem Zugriff 58 die Tabellen in Form des Entscheidungsbaums, der in Fig. 2 der Zeichnung dargestellt ist. Das besondere Format der Tabellen ist sehr ähnlich demjenigen Format, das in Fig. 3a und 3b der Zeichnung dargestellt ist. Der Mikroprozessor 54 gewinnt die im Aufnahmespeicher 40 enthaltenen Informationen wieder und wandelt unter Benutzung der Tabellen, die im Speicher mit wahlfreiem Zugriff 58 enthalten sind, die Assemblersprachebefehle, die in der dritten Spalte der Fig. 4 gezeigt sind, in entsprechende mnemonische Angaben um, wie sie in der zweiten Spalte 24 der Fig. 5 dargestellt sind, uns speichert die umgewandelten mnemonischen Angaben in Assemblersprache im Speicher mit wahlfreiem Zugriff 58. Die Anzeigesteuerung 62 gewinnt die in dem Speicher mit wahlfreiem Zugriff 58 enthaltenen umgewandelten mnemonischen Ausdrücke in Assemblersprache zurück und zeigt die mnemonischen Angaben auf dem Darstellungsmonitor 64 mit Hilfe des Zeichengenerators 66 in der Form an, wie sie aus Fig. 5 der Zeichnung ersichtlich ist.

Wie oben bereits angegeben, ermöglicht es die im Nur-Lese-Speicher 60 abgespeicherte Firmware dem Mikroprozessor 54, die Tabellen, wie in den Fig. 2, 3a und 3b der Zeichnung dargestellt, in Form eines Entscheidungsbaums abzuspeichern. Nachdem die Tabellen im Speicher mit wahlfreiem Zugriff 58 abgespeichert worden sind, werden die in der Form einer Menge von binären Codeangaben im Aufnahmespeicher 40 gespeicherten Assemblersprachebefehle in einen entsprechenden Satz von mnemonischen Angaben in Assemblersprache mittels der Tabellen umgewandelt und im Speicher mit wahlfreiem Zugriff 58 abgespeichert. Die im Nur-Lese-Speicher 60 kodierte Firmware, die letztlich für diesen Umwandlungsprozess massgebend ist, ist durch den folgenden Algorithmus gekennzeichnet:

Inverser Assembleralgorithmus: Beschreibung des internen Datenformats des Entscheidungsbaums.

Eine Tabelle besteht aus folgendem:

Tabellendefinition: Sie beschreibt die Umgebung für die Tabelle. Sie enthält: Tabellenname, beschreibt, welche Datenkanäle Tabelleneingänge sind und wieviele Eingaben in der Tabelle sind.

0-256 Eintragungen: Jede Eingabe enthält den «WERT». Der «WERT» besteht aus dem binären Wert mit einer «Don't Care»-Maske. Wenn immer der Maskenwert Null ist, dann werden die Daten mit dem «WERT» verglichen. Wenn die Maske eine Eins ist (was ein «X» bezeichnet), dann wird dieses Bit nicht verglichen.

Jede Eingabe enthält dann 0–9 «Actions». Jede «Action» enthält eine wahlweise Datenkette, die darzustellen ist, wahlweise Parametermasken und eine wahlweise Tabelle zum Aufrufen.

Algorithmus für den Aufruf einer Tabelle:

Wie durch die Tabelle definiert, werden die entsprechenden Daten aus dem Aufnahme- (oder Bezugs-)Speicher ausgelesen.

Die erste Eintragung wird dann mit den gelesenen Daten verglichen. Für jedes Datenbit wird folgendes durchgeführt:

Wenn die «Don't Care«-Maske Null ist, dann werden die Datenbits verglichen. Gleichen sie sich, dann wird anschliessend das nächste Bit verglichen. Wenn die «Don't Care»-Maske eine Eins ist, dann ist der Vergleich automatisch positiv. Wenn alle Bits verglichen sind und sich gleichen, dann ist dies die jeweils zu verwendende Eingabe.

Wenn «Ungleich» erscheint, dann wird der Vergleich auf die nächste Eingabe in der Tabelle übernommen. Dies wird fortgesetzt, bis entweder ein positiver Vergleich gefunden oder das Ende der Eingabe erreicht wird. Wenn der letztere Fall eintritt, dann ist diese Tabelle nicht aufgerufen.

Sobald ein positiver Vergleich gefunden worden ist, müssen die «Actions» innerhalb der Parameterangabe interpretiert werden.

Die möglichen «Actions» (Vorgänge) sind:

Darstellung einer Datenkette (z.B. «Verschiebe» -)

Abgreifen von Parameterbits (Bits, die an andere Tabellen übergeben werden sollen)

Aufruf einer anderen Tabelle

Diese Massnahmen werden fortgesetzt, bis auf das Ende der «Actions» (Vorgänge) gestossen wird. Wenn dies eintritt, wird aus der laufenden Tabelle in die aufrufende Tabelle zurückgekehrt. Wenn keine aufrufende Tabelle vorhanden ist, dann wird für diese Zeile der erfassten Daten die inverse Assemblierung durchgeführt.

Es ist einzusehen, dass die Funktion der Zuordnung der Kennzeichenbits zu den Befehls- und Dateninformationen innerhalb des ausführbaren Code durch einen Post-Prozessor-Algorithmus, der im Nur-Lese-Speicher 60 kodiert ist, in Verbindung mit dem Mikroprozessor 54 statt mit dem Identifiziermodul 28 ausgeführt werden kann, wie es oben beschrieben wurde.

Es ist ersichtlich, dass die oben beschriebene Erfindung in verschiedener Weise abgewandelt werden kann. Solche Abwandlungen sind nicht als Abweichungen vom Gedanken und Schutzbereich der Erfindung anzusehen. Alle derartigen Modifikationen, die Fachleuten offensichtlich sind, sollen in den Schutzumfang der Ansprüche einbezogen werden.

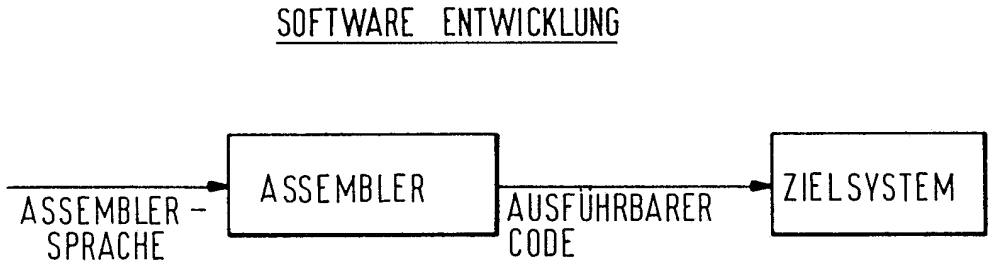


FIG. IA.

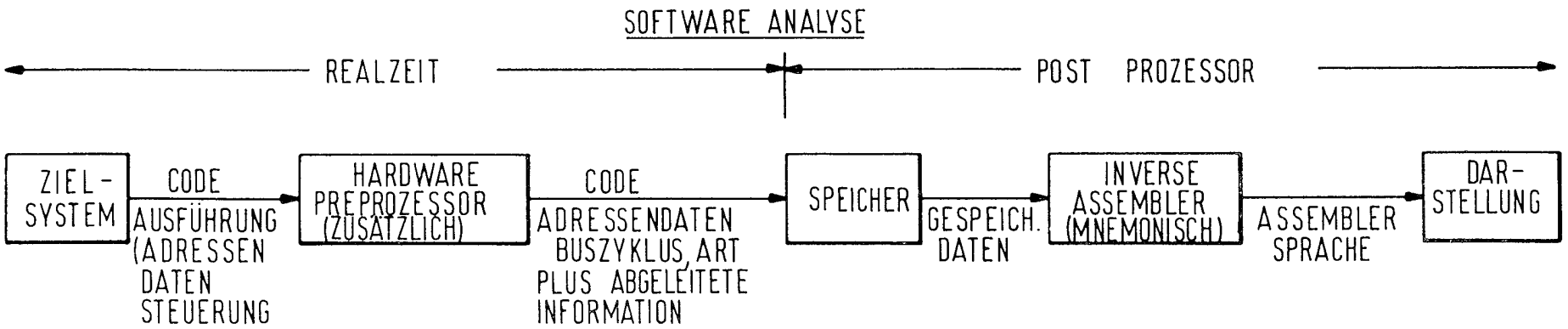


FIG. 1B.

660 242
5 Blätter Nr. 1

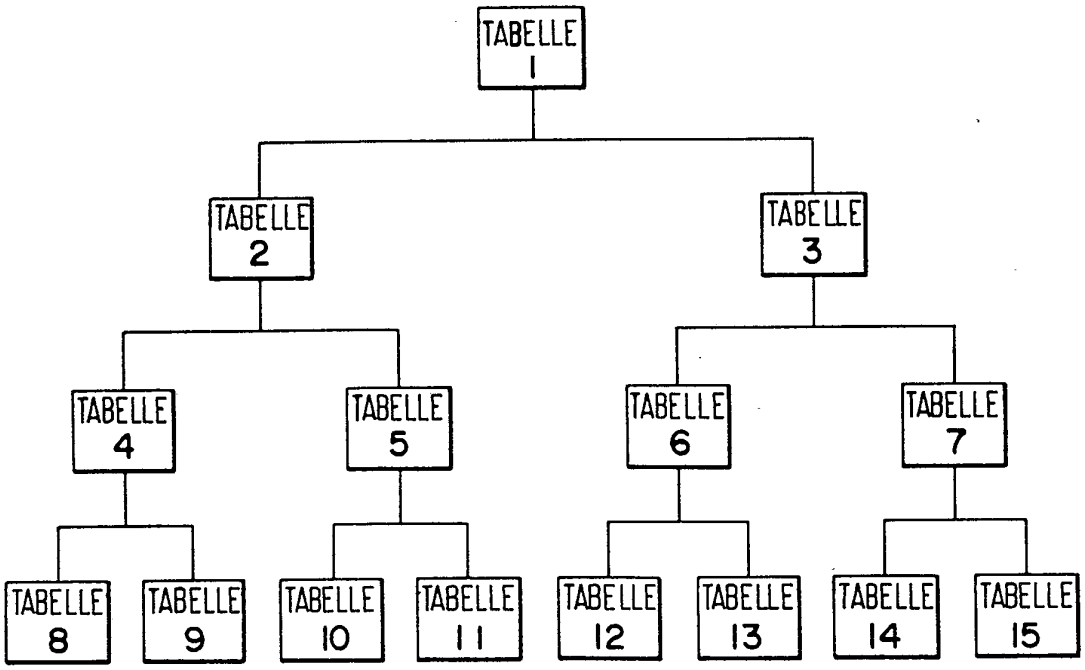


FIG. 2.

ANGABE TABELLENANZEIGE [REFERENZ] VERGLEICHE: START SEQ [0]
STOP SEQ [511]

TRIG =	F7E6D	01011111	1001	FF7D
SRCH =	XXXXX	XXXXXXXX	XXXX	XXXX
MASK =	11111	11111111	1111	1111

SEQ	ADR	CTRL	DATEN
14	FFFF0	0000	00EA
15	FFFF2	0010	00E0
16	FFFF4	0000	00F0
17	FFFF6	0000	0000
18	FE000	0000	7EBC
19	FE002	0010	C704
20	FE004	0001	4206
21	FE006	0000	0204
22	FE008	0000	3300
23	FE00A	0000	B9C0
24	00442	0110	0002
25	FE00C	0001	001A
26	FE00E	0000	02BF
27	FE010	0010	8804
28	FE012	0001	4705
29	FE014	0010	FBE2
	16	18	20

FIG. 4.

DEFINIERE MNEMONISCH TAB.NAME: OPCODE01 MODE: TABELLEN EINGABE

SEQ	P BIN	AUFRUF	ANZEIGE
0	01000XXX		INC
	--------	*TAB	
	-----^^^	REG 16	

1	01001XXX		DEC
	--------	*TAB	
	-----^^^	REG 16	

2	01010XXX		BETÄTIGEN
	--------	*TAB	
	-----^^^	REG 16	
	--------		

10 12 14

FIG. 3A.

DEFINIERE MNEMONISCH TABELLENNAME: REG 16 MODE TABELLEN EINGABE

SEQ	P BIN	ANZEIGE
0	000	AX
1	001	CX
2	010	DX
3	011	BX
4	100	SP
5	101	BP
6	110	SI
7	111	DI
8	XXX	
9	XXX	
10	XXX	
11	XXX	
12	XXX	
13	XXX	
14	XXX	
15	XXX	
16	XXX	
17	XXX	
18	XXX	

FIG. 3B.

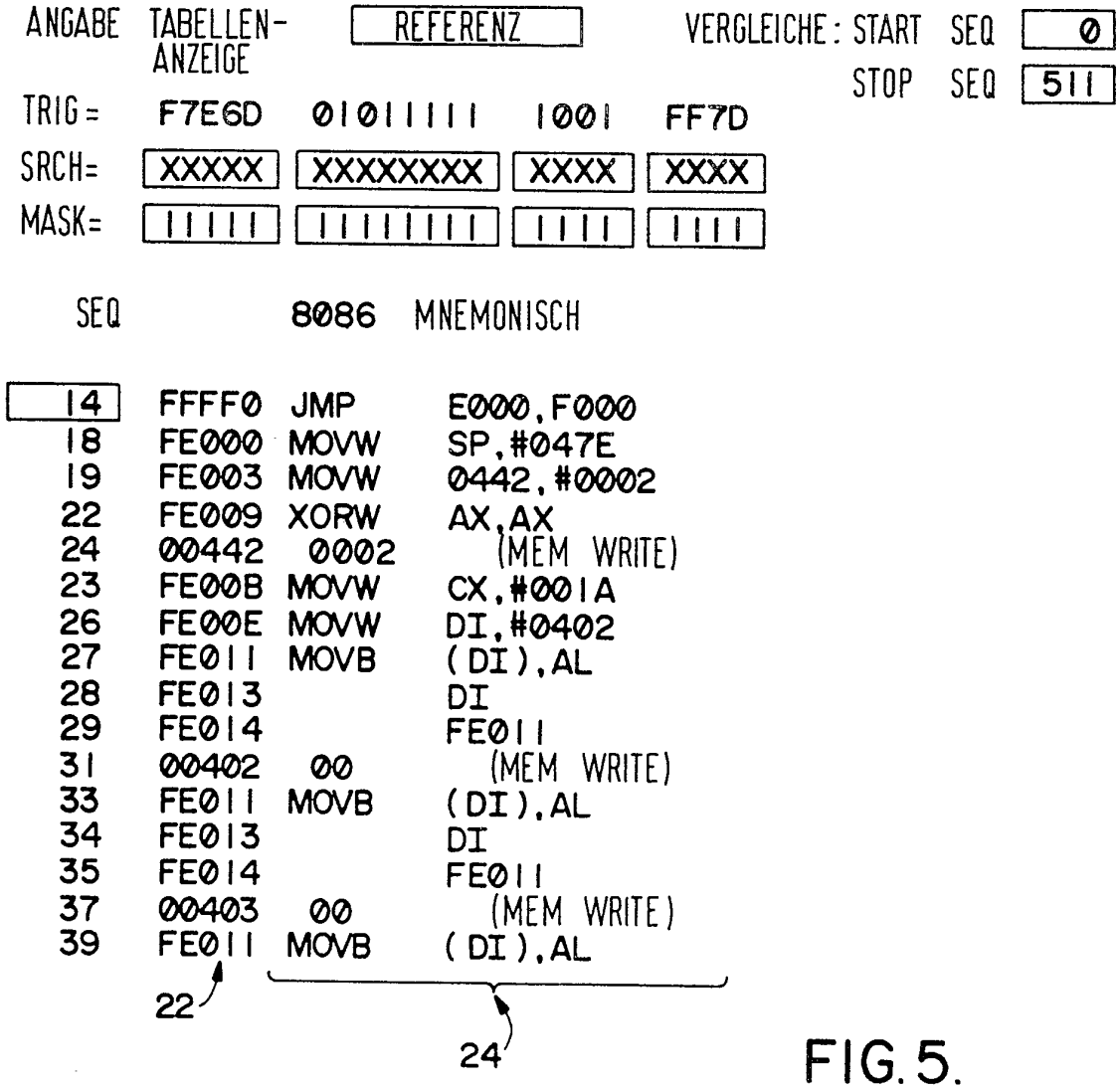

ANGABE TABELLEN-ANZEIGE
REFERENZ
VERGLEICHE: START SEQ 0
STOP SEQ 511
TRIG= F7E6D 01011111 1001 FF7D
SRCH= XXXXX XXXXXXXX XXXX XXXX
MASK= 11111 11111111 1111 1111
SEQ 8086 MNEMONISCH
14 FFFF0 JMP E000,F000
18 FE000 MOVW SP,#047E
19 FE003 MOVW 0442,#0002
22 FE009 XORW AX,AX
24 00442 0002 (MEM WRITE)
23 FE00B MOVW CX,#001A
26 FE00E MOVW DI,#0402
27 FE011 MOVB (DI),AL
28 FE013 DI
29 FE014 FE011
31 00402 00 (MEM WRITE)
33 FE011 MOVB (DI),AL
34 FE013 DI
35 FE014 FE011
37 00403 00 (MEM WRITE)
39 FE011 MOVB (DI),AL
22
24

FIG. 5.

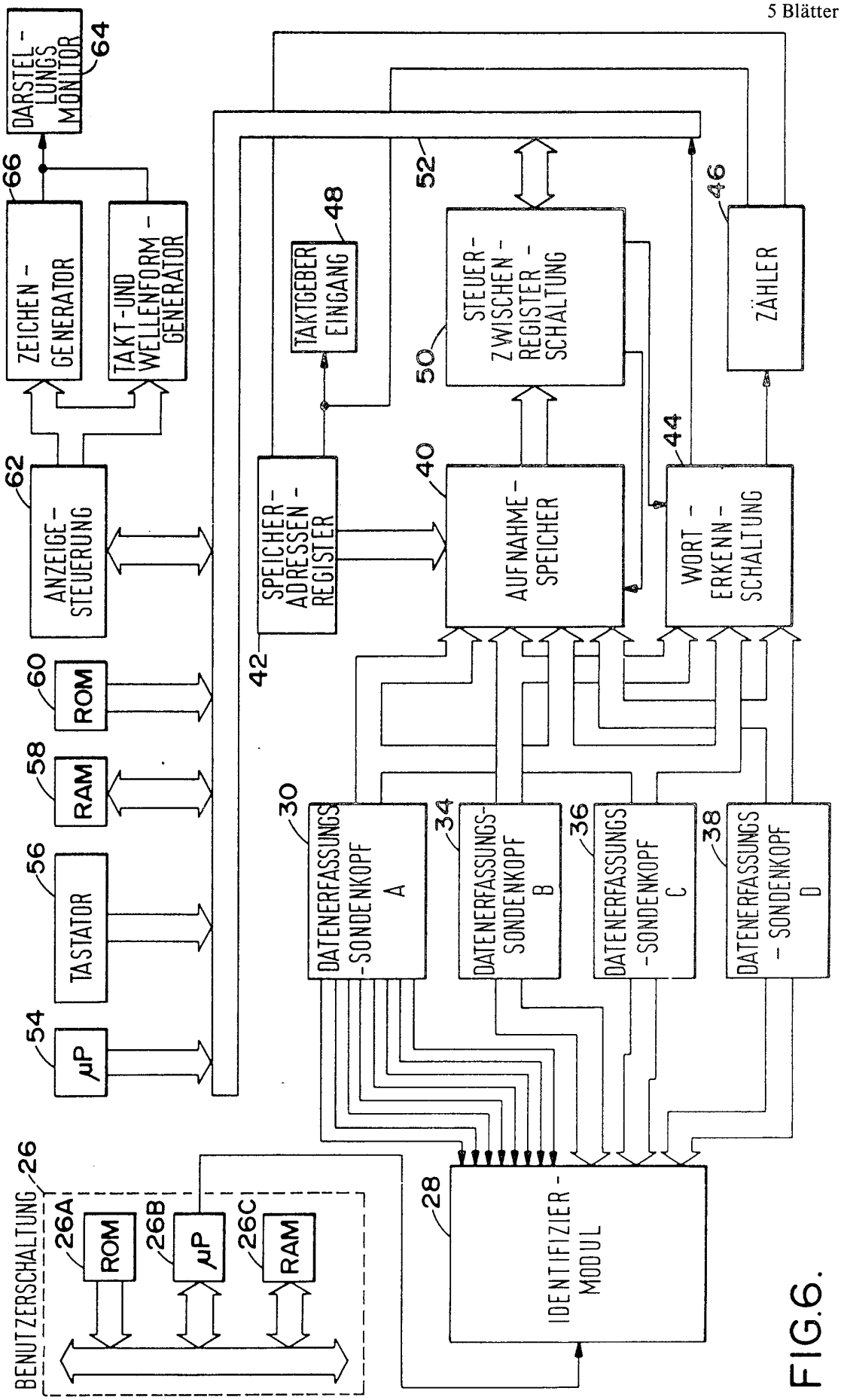

DARSTEL-LUNGS MONITOR
64
66
ZEICHEN-GENERATOR
TAKT-UND WELLENFORM-GENERATOR
62
ANZEIGE-STEUERUNG
60
ROM
58
RAM
56
TASTATOR
54
µP
TAKTGEBER EINGANG
48
SPEICHER-ADRESSEN-REGISTER
42
52
STEUER-ZWISCHEN-REGISTER-SCHALTUNG
50
40
AUFNAHME-SPEICHER
ZÄHLER
46
44
WORT-ERKENN-SCHALTUNG
30
DATENERFASSUNGS-SONDENKOPF A
34
DATENERFASSUNGS-SONDENKOPF B
36
DATENERFASSUNGS-SONDENKOPF C
38
DATENERFASSUNGS-SONDENKOPF D
28
IDENTIFIZIER-MODUL
BENUTZERSCHALTUNG
26
26A
ROM
26B
µP
26C
RAM

FIG.6.