MANFRED SOMMER
BERNHARD STREIKO
MANFRED FRANKE

CPU-SIMULATION

WIE ARBEITET EIN COMPUTER?

CBT
COMPUTER BASED
TRAINING

GABLER

Sommer/Streiko/Franke
CPU-Simulation — Wie arbeitet ein Computer?

Deutscher Hochschul-Software-Preis:

Die in diesem Programmpaket enthaltene Software „CPUSIM“ erhielt 1990 den vom Bundesminister für Bildung und Wissenschaft verliehenen „Deutschen Hochschul-Software-Preis“.

Prof. Dr. Manfred Sommer
Bernhard Streiko
Manfred Franke

CPU-Simulation

Wie arbeitet ein Computer?

GABLER

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Sommer, Manfred:
CPU-Simulation – wie arbeitet ein Computer? / Manfred Sommer ; Bernhard Streiko ; Manfred Franke. – Wiesbaden : Gabler
(Computer based training)
NE: Streiko, Bernhard; Franke, Manfred:
1-Platz-Version. – 1991
ISBN 3-409-19268-9
10-Platz-Version. – 1991
ISBN 3-409-19267-0

Der Gabler Verlag ist ein Unternehmen der Verlagsgruppe Bertelsmann International.

Lektorat: Hans-Ulrich Bauer

Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Verarbeitung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säurefreiem und chlorarm gebleichtem Papier gedruckt. Die Einschweißfolie besteht aus Polyäthylen und damit aus organischen Grundstoffen, die weder bei der Herstellung noch bei der Verbrennung Schadstoffe freisetzen.

Satz: Buchdruckerei Loibl, Neuburg a. d. Donau
Druck und Bindung: Wilhelm & Adam, Heusenstamm
Printed in Germany

ISBN 3-409-19268-9 (1-Platz-Version)
ISBN 3-409-19267-0 (10-Platz-Version)

Vorwort

Zielsetzung

Die Arbeit mit Computern im Berufsleben ist mittlerweile zu einer gewohnten Erscheinung geworden. Aus diesem Grunde dringt die Informatik verstärkt in die Hochschulausbildung, berufliche Weiterbildung und Erstausbildung auch derjenigen Fachrichtungen vor, die auf Tätigkeiten in den sog. Misch- und Randberufen der Datenverarbeitung vorbereiten. Die Vermittlung speziell der Vorgänge innerhalb des Rechners gestaltet sich aber äußerst schwierig, da es abgesehen von schwarzen Bauteilen auf verwirrenden Leiterplatinen wenig zu zeigen gibt. Die entscheidenden Aktionen finden aber gerade in dieser „black box" statt.

Das vorliegende Lernprogramm CPUSIM soll diese Lücke bei der Veranschaulichung durch Darstellung einer fiktiven, didaktisch reduzierten CPU (CPU = central processing unit = Zentralprozessor) und Simulation der wichtigsten internen Vorgänge schließen. CPUSIM zeigt in seiner graphischen Darstellung den schematischen Aufbau eines Zentralprozessors. Der Benutzer kann anhand von mitgelieferten oder selbsterstellten Beispielprogrammen die Abarbeitung der Maschinenbefehle (Datenfluß, Speicherbelegung) am Bildschirm verfolgen.

CPUSIM richtet sich an alle, die erste Eindrücke vom Innenleben eines Mikroprozessors gewinnen möchten: insbesondere in anwendungsinformatischen Studien- und Ausbildungsgängen, in denen nur begrenzte Zeit für die Vermittlung von Hardware-Kenntnissen zur Verfügung steht. Das Lernprogramm kann vorlesungs- oder unterrichtsbegleitend, in PC-Praktika, in Gruppenarbeit oder im Selbststudium eingesetzt werden.

Begleitheft und Diskette

Zum Lieferumfang des Lernprogrammes CPUSIM gehört neben einer **Originaldiskette** mit sämtlichen Programm- und Textdateien eine oder mehrere **Userdisketten** je nach der Anzahl der Lizenzen, die Sie erworben haben. Dazu erhalten Sie dieses **Begleitheft**, in dem

- die Installation von CPUSIM,
- der Aufbau von CPUSIM,
- die Funktionen von CPUSIM
- das Üben mit CPUSIM

beschrieben werden. Das Begleitheft verwendet – wo immer möglich – dieselben Texte wie die kontextsensitive Hilfe von CPUSIM auf dem Bildschirm. Ob Sie also lieber mit einer gründlichen Lektüre dieses Begleitheftes oder mit einer ersten Simulation eines mitgelieferten Beispielprogrammes in CPUSIM „einsteigen" wollen, ist weitgehend Ihrem persönlichen Lernstil überlassen. Auf jeden Fall sollten Sie als nächstes die Installationshinweise im 1. Kapitel lesen.

Beim Gabler-Verlag dürfen wir uns für die hervorragende Kooperation und die wertvollen Anregungen vor allem für die Gestaltung dieses Begleitheftes bedanken. Verbesserungen der Lernsoftware selbst resultieren aus ihrem praktischen Einsatz in meiner Lehrveranstaltung „Wirtschaftsinformatik I" an der Hochschule für Wirtschaft und Politik Hamburg. Den daran in den letzten Semestern beteiligten Studentinnen, Studenten und Tutoren gilt unser besonderer Dank. Schließlich sollte die Akademische Software Kooperation nicht vergessen werden, die mit dem Deutschen Hochschul-Software-Preis einen attraktiven Anreiz zur Herstellung und Verbreitung von Teachware ins Leben gerufen hat.

Hamburg und Bielefeld, im April 1991

Manfred Sommer, Bernhard Streiko, Manfred Franke

Inhaltsverzeichnis

1. Installation von CPUSIM

1.1 Systemvoraussetzungen

Das Lernprogramm CPUSIM ist auf allen IBM-kompatiblen Personalcomputern (XT, AT, PS/2, PS/1) mit mindestens 256 KB Hauptspeicher (RAM) lauffähig. Entscheidend ist, daß Ihr PC mit einem Mikroprozessor INTEL 8088 oder höher arbeitet. Ein mathematischer Koprozessor ist nicht erforderlich und wird von CPUSIM nicht genützt. Als Betriebssystem wird MS-DOS bzw. PC-DOS in der Version 2.0 oder höher vorausgesetzt. Zur Installation der Software von der beiliegenden Diskette auf die Festplatte Ihres PC's benötigen Sie ein Diskettenlaufwerk (5,25 Zoll, 360 KB oder 3,5 Zoll, 720 KB) und eine Festplatte mit mindestens 250 KB freiem Speicherplatz. Für jedes zusätzlich von Ihnen selbst erstellte Testprogramm brauchen Sie weitere 2 KB. Als Bildschirm reicht ein Monochrom-Monitor mit 25 Zeilen à 80 Zeichen aus. Da CPUSIM wegen der besseren Anschaulichkeit mehrfarbig angelegt wurde, ist ein Farbmonitor (CGA, EGA oder VGA) natürlich empfehlenswert. Einen Drucker benötigen Sie nicht.

Außer den Tasten des Schreibmaschinenblocks, den Cursortasten und den Funktionstasten kommen einige Sondertasten zum Einsatz, die im Begleitheft und im Hilfesystem von CPUSIM wie folgt bezeichnet werden, auf Ihrer Tastatur aber auch anders benannt sein können:

Allgemeine Programmsteuerung:

• F1 ... F10	Funktionstasten
• Enter, CR-Taste, RETURN, <↵	Übernahme der Eingaben, Bestätigung
• ESC-Taste, Eing.Lösch	Eingabe löschen, Funktion beenden/ abbrechen, Ende

Cursorsteuerung:

• Pfeiltasten auf, ab, rechts, links	Cursor wird um eine Stelle auf, ab, nach rechts oder links bewegt
• PgUp, Bild hoch	vorige Seite, zurückblättern
• PgDn, Bild runter	nächste Seite, vorblättern
• Home (Posl) / End	Anfang / Ende

Tastatursteuerung:

• SHIFT, ^	Umschaltung Groß-/Kleinschreibung
• CTRL, STRG	Umschaltung Kontrollzeichen
• BACK, <-	Rückschritt und Löschen des letzten Zeichens
• DEL, Lösch, CTRL G	Löschen des Zeichens, auf dem der Cursor steht

CPUSIM arbeitet weitgehend menüorientiert. Die Steuerung des Lichtbalkens zwischen den Menüpunkten erfolgt durch die Cursortasten und nicht mit einer Maus.

1.2 Programminstallation

Zunächst sollten Sie die anliegende Originaldiskette mit einem Schreibschutz versehen. Benutzen Sie diese Diskette nur zum Installieren und nicht zum Speichern von Testprogrammen. Deshalb benötigen Sie eigentlich auch keine Sicherheitskopie der Originaldiskette. Legen Sie diese Diskette in ein Diskettenlaufwerk ein. Machen Sie dieses Laufwerk nun mit

A: <↵

oder bei Verwendung eines eventuell vorhandenen zweiten Laufwerks mit

B: <↵

zum aktiven Laufwerk. Starten Sie jetzt das Installationsprogramm mit

INST_CPU C, <↵

wenn Sie das Lernprogramm auf Ihrer Festplatte im logischen Laufwerk C installieren wollen. Mit der Angabe von D, E, ... oder K statt C, können Sie die Installation auf einem anderen logischen Festplattenlaufwerk vornehmen; mit A bzw. B installieren Sie das Lernprogramm auf dem ersten bzw. zweiten Diskettenlaufwerk. Wenn Sie nur INST_CPU ohne Laufwerkskennung eingeben, erhalten Sie Hinweise zum Installationsvorgang auf Ihrem Bildschirm. Nach erfolgreicher Installation erscheint auf dem Bildschirm die Meldung

„Installation beendet".

Aktivieren Sie die Festplatte nun wieder mit

C: <↵

und nehmen Sie die Diskette zur sicheren Aufbewahrung aus dem Laufwerk. Vergewissern Sie sich, ob der Bildschirmtreiber ANSI.SYS in Ihrer Konfigurationsdatei CONFIG.SYS enthalten ist. Falls nicht, ergänzen Sie die Datei CONFIG.SYS mit dem DOS-Editor „EDLIN", einem sonstigen Editor oder Ihrem Textverarbeitungsprogramm um die Zeile:

DEVICE = Laufwerk; Pfad; ANSI.SYS

Konsultieren Sie Ihr DOS-Handbuch, wenn Sie mit der Einbindung von Treibern noch nicht vertraut sind.

1.3 Installationskontrolle

Das Installationsprogramm hat auf Ihrer Festplatte ein Unterverzeichnis „CPUSIM" angelegt und alle zum Lernprogramm gehörigen Dateien dorthin kopiert:

- CPUSIM.EXE — lauffähiges Simulationsprogramm
- SIMHILFE.TXT — kontextsensitives Hilfeprogramm
- ADD.CPU etc. — sechs mitgelieferte Testprogramme

Testprogramme erkennen Sie an der Dateinamenserweiterung „CPU". Sie enthalten die eigentlichen Anwendungsprogramme, deren Abarbeitung in unserer fiktiven CPU auf dem Bildschirm simuliert wird. Auch die von Ihnen selbst erstellten Testprogramme werden mit der Endung „CPU" im selben Unterverzeichnis abgelegt. Überzeugen Sie sich mit dem DOS-Befehl „DIR", ob die genannten Dateien im Verzeichnis „CPUSIM" enthalten sind. Richten Sie keine Unterverzeichnisse im Verzeichnis „CPUSIM" ein.

1.4 Programmstart

Wechseln Sie mit dem DOS-Befehl „CD" in das Unterverzeichnis „CPUSIM", z.B.

CD; CPUSIM <↵

Falls Sie über längere Zeit kontinuierlich mit dem Lernprogramm arbeiten wollen, sollten Sie die entsprechende Pfadangabe in Ihre AUTOEXEC.BAT-Datei aufnehmen. Bevor Sie das Lernprogramm mit

CPUSIM <↵

starten, legen Sie bitte die Userdiskette in das Diskettenlaufwerk. Der Programmstart funktioniert nur, wenn Sie – wie im Abschnitt 1.2 beschrieben – durch den Installationsvorgang eine lauffähige Version erstellt haben. Die Datei CPUSIM.EXE auf Ihrer Originaldiskette ist so nicht lauffähig.

Nach dem Start von CPUSIM meldet sich das Lernprogramm mit zwei Eingangsmasken, in denen sich das Programm vorstellt (Abb. 1 und Abb. 2). Jetzt können Sie die Userdiskette wieder aus dem Laufwerk entnehmen, müssen dies aber nicht.

Abb. 1: Erste Eingangsmaske

Auf der zweiten Eingangsmaske sehen Sie eine Kommandozeile mit den zur Steuerung erforderlichen Funktionstasten :

<F1> Aufruf des Hilfemenüs (diverse Informationstexte)

<F2> Aufruf des Laufmodus (Ablauf von Testprogrammen)

<F3> Aufruf des Editmodus (Erstellen von Testprogrammen)

<ESC> Ende der Simulation (falls ein Testprogramm im Arbeitsspeicher steht, hat man noch die Möglichkeit, es zu sichern).

Abb. 2: Zweite Eingangsmaske

Hinweise zur Kommandozeile

Bei jeder Bildschirmmaske erscheint am unteren Rand eine Kommandozeile. Sie enthält eine Liste aller Funktionen mit den dazugehörigen Funktionstasten, die aus dem jeweiligen Bildschirm heraus aufgerufen werden können. Ein gewolltes oder ungewolltes Betätigen anderer als der in der Kommandozeile aufgeführten Steuertasten bleibt ohne Wirkung — gegebenenfalls hören Sie einen Warnton, sofern Sie diesen nicht ausgeschaltet haben (vgl. 3.3). Grundsätzlich bietet Ihnen die Kommandozeile eine Orientierungshilfe für ein einfaches Arbeiten mit CPUSIM ohne häufiges Nachschlagen im Begleitheft.

Starten eines Testprogrammes

Folgende Schritte sind in der angegebenen Reihenfolge auszuführen:

- <F3> Umschalten in den Editmodus
- <F2> Anzeigen der vorhandenen Testprogramme
- ↑↓ Auswahl eines Testprogrammes durch Bewegen des Lichtbalkens mit den Cursortasten

- ↵ Auswahl mit RETURN bestätigen
- <F4> Wechsel in den Laufmodus (wenn Ladevorgang erfolgreich)
- <F6> Starten des Testprogrammes.

1.5 Datensicherung

Die von Ihnen selbst erstellten Testprogramme werden mit dem Befehl „Sichern" (<F3> im Editmodus) im Verzeichnis „CPUSIM" auf der Festplatte abgelegt. Sie können Ihre Testprogramme aber auch auf eine separate formatierte Arbeitsdiskette kopieren, indem Sie vor dem Programmnamen zuerst eine Laufwerksbezeichnung eingeben, z. B.:

A:;programmname

Wenn sich im Laufwerk A die Userdiskette befindet (i. d. R. der Fall), so **muß** sie für die Sicherung gegen die Arbeitsdiskette ausgetauscht werden.

2. Aufbau und Funktionsweise der vom Lernprogramm simulierten CPU

2.1 Motivation und Zweck des Lernprogramms

Die Datenverarbeitung mit Computern aller Größenordnungen und Rechnerarchitekturen folgt grundsätzlich dem sogenannten EVA-Prinzip (Abb. 3):

- Eingabe
- Verarbeitung
- Ausgabe

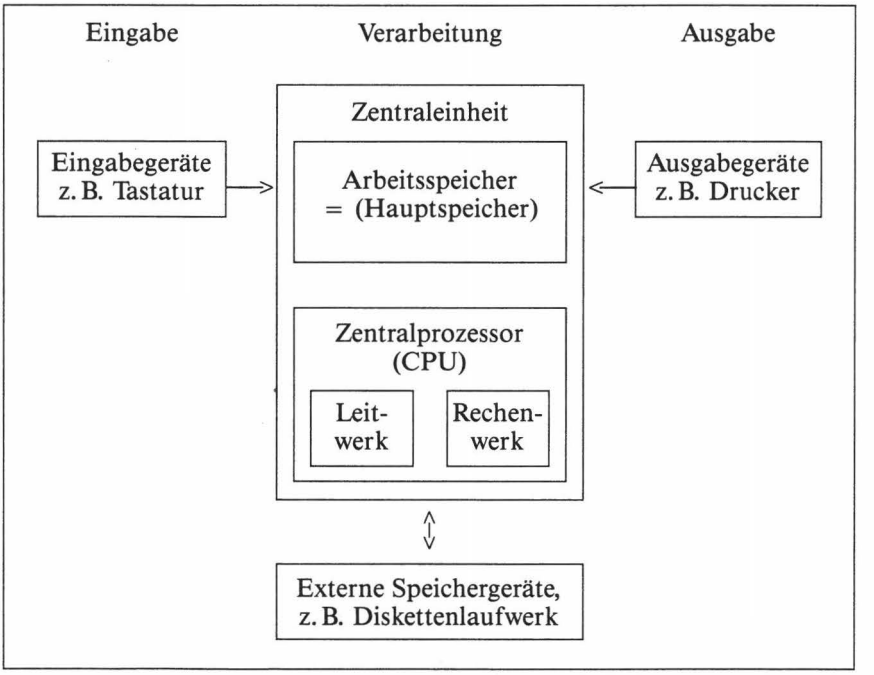


Abb. 3: Das EVA-Prinzip

Während die Eingabe- und Ausgabegeräte, die zusammen mit den externen Speichergeräten (Festplatten-, Disketten-, Magnetbandlaufwerke) die peripheren Geräte bilden, relativ anschaulich sind oder in ihrer Funktionsweise durch Analogien verständlich gemacht werden können, bereitet die Vermitt-

lung des Aufbaus der Zentraleinheit, des Zusammenwirkens ihrer wichtigsten Komponenten sowie der dabei verwendeten elementaren Verarbeitungsregeln erhebliche didaktische Probleme. Durch die Höchstintegration Hunderttausender Schalt- und Speicherelemente ist erstens eine Stufe der Miniaturisierung von Zentraleinheiten erreicht, die nichts mehr sichtbar werden läßt. Zweitens entfernen sich die sprachlichen und graphischen Beschreibungsmittel der Verarbeitungsregeln im Zuge der Endbenutzer-DV immer weiter von der maschinensprachlichen Ebene, auf der sie in der Zentraleinheit nach wie vor ausgeführt werden.

Während sich der interne Aufbau einer Zentraleinheit vereinfacht und schematisch durch Schaubilder, z. B. in Büchern oder auf OHP-Folien, darstellen läßt (statische Struktur), wirken ihre Komponenten bei der internen Verarbeitung in einzelnen Arbeitsphasen (von Neumann-Zyklus) dynamisch zusammen, was drittens nur in einer Bewegtbild-Darstellung demonstriert werden kann. Filme und Videos erfüllen zwar die Anforderung an dynamische Präsentation, sind aber (zu vertretbaren Kosten) nur passiv rezipierbar und gestatten kein interaktives Lernen. Mit einer Bildschirmsimulation auf Basis dieser Lernsoftware ist es hingegen viertens möglich, die Verarbeitungsregeln dieser didaktisch reduzierten CPU zu maschinenorientierten Programmen selbst zusammenzustellen und die Arbeitsweise der so programmierten Zentraleinheit am Bildschirm „in Zeitlupe und mit Stoptaste" zu verfolgen.

Das Lernprogramm CPUSIM konzentriert sich deshalb auf die Vorgänge in der Zentraleinheit. Der Datenfluß von und zu peripheren Einheiten wurde bewußt ausgeblendet. Er könnte Gegenstand eines weiteren Lernprogramms sein, in dem wieder mit dem Medium einer Bildschirmsimulation gezeigt würde, wie diese Aufgaben von einem (Mikrocomputer-)Betriebssystem gelöst werden.

2.2 Bits, Bytes und Dualzahlen

Ein Bit (= binary digit) ist ein Element eines binären Zeichensystems. Ein binäres Zeichensystem besteht nur aus zweiwertigen Zeichen. Beispiele:

- Strom fließt / Strom fließt nicht
- Schalter offen / Schalter geschlossen
- magnetisiert / nicht magnetisiert
- Loch / kein Loch

Diese Beispiele weisen bereits auf mechanische, elektromagnetische, elektrooptische und elektronische Darstellungs- und Verarbeitungsmöglichkei-

ten von Informationen hin. Deshalb basiert die Darstellung von Informationen in der Datenverarbeitung auf zwei Zeichen: der binären Null (0) und der binären Eins (1 oder L). Da die meisten Informationen aber aus mehr als zwei verschiedenen Zeichen bestehen, müssen zur Zeichendarstellung mehrere Bits (jeweils 0 oder 1) verwendet werden. In der Datenverarbeitung haben sich acht Bits (= 1 Byte) als kleinste ansprechbare Informationseinheit durchgesetzt.

Ein binäres Zahlensystem, das zugleich ein Stellensystem ist, heißt duales Zahlensystem. Es besteht aus den dualen Nennwerten 0 und 1. Alle Dezimalzahlen lassen sich als Summe von Zweierpotenzen darstellen. Beispiel:

47 =	1	mal 2 hoch 5	(2*2*2*2*2	= 32)		32
	0	mal 2 hoch 4	(2*2*2*2	= 16)	+	0
	1	mal 2 hoch 3	(2*2*2	= 8)	+	8
	1	mal 2 hoch 2	(2*2	= 4)	+	4
	1	mal 2 hoch 1	(2	= 2)	+	2
	1	mal 2 hoch 0	(	= 1)	+	1 = 47

Dreht man den Kasten mit den Einsen und Nullen in die waagerechte Position, so daß die höchste Stelle links steht, dann kann man die duale Entsprechung der dezimalen Zahl 47 ablesen:

47	=101111
dezimal	dual

Auf diese Weise werden alle Daten im Rechner umgewandelt, damit der Prozessor mit ihnen umgehen kann.

Wenn Sie in CPUSIM-Testprogrammen Zahlenwerte als Daten abspeichern, müssen Sie die internen Grenzen des Arbeitsspeichers beachten. Es stehen maximal 10 Stellen (=10 Bits) zur Verfügung. Da die Zahlen binär gespeichert werden, ergibt sich ein Umfang von $2^{10} = 1024$ darstellbaren Zahlen. Sie können nur ganze Zahlen (Integer-Zahlen) verarbeiten; gebrochene Zahlen (Real-Zahlen) werden nicht dargestellt. Hingegen können positive und negative Zahlen durch das ganz links stehende, höchstwertige Bit (Vorzeichenbit) unterschieden werden:

0 = positiv,
1 = negativ.

Damit gibt es $2^9 = 512-1$ positive Zahlen (also 511) und die Null
und $2^9 = 512$ negative Zahlen.

Der Zahlbereich erstreckt sich also
von −512 (1000000000)
über −1 (1111111111)
und 0 (0000000000)
und +1 (0000000001)
bis +511 (0111111111).

Eine Eingabe von Zahlen außerhalb dieses Bereiches im Editmodus ist nicht möglich. Dezimale Kommastellen sind ebenfalls nicht darstellbar. Achten Sie bei bei der Codierung arithmetischer Befehle darauf, daß das Rechenergebnis diesen Bereich nicht über- bzw. unterschreitet. CPUSIM gibt ggf. eine entsprechende Fehlermeldung aus. Negative ganze Zahlen werden durch sogenannte Zweierkomplemente dargestellt. Sie gehen aus der positiven Zahl gleicher absoluter Wertigkeit dadurch hervor, daß erstens in allen Bits die 0 durch die 1 ersetzt wird (und umgekehrt) und zweitens eine 1 addiert wird, z.B.

```
      0000000001  (= +1)
1.    1111111110
2. + 0000000001
   ------------
   = 1111111111  (=Zweierkomplement von 1 = −1)
```

2.3 Bestandteile der Zentraleinheit

Wie in Abb. 3 bereits gezeigt, besteht die Zentraleinheit aus

- dem Zentralprozessor (CPU) und
- dem Arbeitsspeicher.

Gehen wir zunächst auf den Arbeitsspeicher ein.

Der Arbeitsspeicher

Der Arbeitsspeicher wird auch als Hauptspeicher, Zentralspeicher, interner Speicher oder Primärspeicher bezeichnet. Er stellt dem Zentralprozessor die zur Verarbeitung anstehenden Befehle und Eingabedaten bereit und speichert Zwischenergebnisse sowie schließlich die Ausgabedaten. Für die Speicherung von Daten im Rechner sind zwei wesentliche Komponenten verantwortlich:

- Halbleiterspeicher: Speicherplatz für die Aufnahme der zu speichernden Information,

- Adresse: Benennung des Speicherplatzes, um die Information später wiederzufinden („Hausnummer").

Unter Berücksichtigung der Tatsache, daß die Adreßnummern im Rechner binär dargestellt werden, und wenn der Adreßbus z.B. eine Breite von 20 Bits hat, stehen insgesamt 2^{20} = 1 048 576 Adreßnummern zur Verfügung. Damit lassen sich ebensoviele Speicherplätze von der Größe 1 Byte (= 8 Bits) adressieren; die CPU verfügt damit über eine Speicherkapazität von 1 048 576 Byte (= 1 MByte).

Da die Register des Prozessors eine Größe von 16 Bits haben, muß die Adreßbildung für einen 20-Bit-Adreßbus durch eine Umrechnung mit zwei Teiladressen vorgenommen werden:

- die Basisadresse (Segmentadresse) legt einen Speicherbereich (Segment) innerhalb des 1 MByte-Speicherraumes fest,
- die Offsetadresse (Relativadresse) gibt die genaue Position in diesem Segment an; sie steht im Befehlszähler zur Verfügung.

Der Arbeitsspeicher ist ein elektrischer Halbleiterspeicher, der von einer CPU für den schnellen Datenzugriff gebraucht wird. Er ist gewöhnlich so organisiert, daß immer 8 Speicherstellen gleichzeitig (das sind 8 Bits oder 1 Byte) angesprochen oder adressiert werden können. Im PC-Bereich war viele Jahre eine Speichergröße von 640 KByte (das sind ca. 640 000 Speicherwörter) vorherrschend. Der Trend geht heute zu Arbeitsspeichern von mehreren Mega-Byte Größe.

Der Arbeitsspeicher von CPUSIM verfügt über eine Speicherkapazität von 64 Speicherwörtern, die jeweils eine Größe von 10 Bits aufweisen. Dieses ungewöhnliche Format wurde gewählt, um ein noch überschaubares CPU-Modell zu schaffen, das dennoch die wichtigsten Funktionen einer CPU darstellen kann. Die interne Aufteilung eines solchen Feldes hängt davon ab, ob reine Daten oder Befehle in ein solches Feld abgespeichert werden. Sollen nur Daten gespeichert werden, so stehen alle 10 Bits zur Verfügung. Ein Befehl besteht grundsätzlich aus einem Operationsteil und einem Adreßteil. Mit einer Adreßbusbreite von 6 Bit sind 64 Speicherwörter (2^6) adressierbar, mit den verbleibenden 4 Bits lassen sich 16 verschiedene Befehle codieren. Damit kann ein Speicherwort einen Befehl und eine Adresse aufnehmen. CPUSIM modelliert also eine „Ein-Adreß-Maschine". In der grafischen Darstellung des Arbeitsspeichers erscheint vor dem jeweiligen Speicherfeld zur besseren Orientierung noch einmal die Adresse in dezimaler Form, durchnumeriert von 0 bis 63. Die angesprochene Adresse wird sowohl im Editmodus wie auch im Laufmodus hell unterlegt; die Adresse, die im Befehlszähler steht, hat eine andere Farbe.

Der Prozessor

Die eigentliche Verarbeitung der Eingabedaten nach Maßgabe der Programmbefehle zu Ausgabedaten geschieht im Zentralprozessor, der seinerseits aus dem Rechenwerk als der Befehlsausführungseinheit und dem Steuerwerk (oder Leitwerk) als der Befehlsaufbereitungseinheit besteht (Abb. 3).

Bei größeren EDV-Anlagen werden diese Funktionen auf mehrere, entsprechend spezialisierte Prozessoren verteilt (Zentralprozessor, Ein-, Ausgabeprozessor etc.), die unter Umständen auch noch mehrfach vorhanden sein können. Doch auch schon im PC-Bereich werden zur beschleunigten Abwicklung rechenintensiver Programme sog. arithmetische Co-Prozessoren eingesetzt. Damit kann man den Befehlsdurchsatz erhöhen, d.h.: in einer bestimmten Zeitspanne können mehr Befehle bearbeitet werden.

Besteht die Zentraleinheit selbst aus zwei oder mehr Zentralprozessoren, spricht man von Multiprozessorsystemen, mit denen neuartige Rechnerarchitekturen (Parallelverarbeitende Computer, Neuronale Netze) realisiert werden, die vor allem im Großrechnerbereich von Bedeutung sind, aber inzwischen auch als Zweiprozessormaschinen bereits Einzug in die oberste Leistungsklasse von PC's gehalten haben. In diesem Lernprogramm wird jedoch davon ausgegangen, daß ein einziger Prozessor alle auszuführenden Elementarfunktionen und Steuerungen übernimmt.

Register

Um die enorme Arbeitsgeschwindigkeit des Zentralprozessors nutzen zu können, sind weitere Speicherelemente erforderlich, die eine schnelle Zwischenspeicherung von Daten für die Ausführung von Rechenoperationen und für die Adressierung ermöglichen. Für diese Aufgabe eignen sich elektronische Halbleiterspeicher, die in der Prozessorumgebung Register genannt werden. In CPUSIM können Sie die wichtigsten Register kennenlernen:

- Adreßregister
- Akkumulator (AX-Register)
- Befehlszähler (Instruktionsadreßregister)
- Datenregister
- Instruktionsregister
- Interruptregister

Zur Anordnung aller Komponenten des Zentralprozessors betrachten Sie am besten die Laufmodus-Maske auf dem Bildschirm oder ihre Wiedergabe in der Abb. 6 weiter unten in diesem Begleitheft.

Der Befehlszähler (IAR)

Der Befehlszähler (IAR = Instruktionsadreßregister) ist bei vielen PC-Prozessoren ein 16-Bit-Zählregister, das die Nummer des nächsten auszuführenden Befehls enthält. In Verbindung mit einer Basisadresse wird dieser Wert als Offset zu einer 20-Bit-Adresse umgerechnet, um den ganzen Arbeitsspeicher ansprechen zu können.

Der Befehlszähler von CPUSIM ist 6 Bit groß und enthält direkt die Arbeitsspeicheradressen. Jedes Mal, wenn ein Befehl zur Ausführung aus dem Arbeitsspeicher geholt wird, muß der Befehlszähler um 1 erhöht und somit auf den „aktuellen" Stand gebracht werden. Dies impliziert natürlich, daß die Befehle im Arbeitsspeicher unter aufeinanderfolgenden Adressen gespeichert sind. Eine Ausnahme bilden die Sprungbefehle. Hier wird der Befehlszähler mit der im Sprungbefehl enthaltenen Adresse überschrieben. Beim Aufruf eines Unterprogrammes wird der Befehlszähler noch um 1 erhöht und sein Wert dann in einem dafür vorgesehenen Speicherbereich (Stack) abgelegt. Nach Abarbeitung des Unterprogrammes wird diese Adresse wieder in den Befehlszähler übernommen, um so das Hauptprogramm an der richtigen Stelle fortsetzen zu können.

Das Adreßregister (SAR)

Das Adreßregister ist ein spezieller Speicher (SAR = Speicheradreßregister). Es bildet als Endstation des Adreßbusses das Verbindungsglied zwischen den am Adreßbus angeschlossenen Registern und dem Arbeitsspeicher. Es wertet die eingehenden Adressen aus und organisiert den Zugriff auf die unter den jeweiligen Adressen abgespeicherten Informationen. Während des Programmlaufes enthält das Adreßregister immer die Adresse des Speicherplatzes, der gelesen oder beschrieben werden soll.

Das Adreßregister vieler Prozessoren hat eine Größe von 20 Bit und arbeitet mit einer besonderen Adressierlogik zusammen, die die endgültigen Adressen erst berechnet. Das Adreßregister von CPUSIM kann 6 Bits aufnehmen und somit 64 Speicherplätze verwalten. Da die Adresse in Dualschreibweise vorliegt, steht sie zur besseren Übersicht noch einmal in dezimaler Form im Registersymbol der graphischen Darstellung (Laufmodus).

Das Datenregister (SIR)

Das Datenregister (SIR = Speicherinformationsregister) ist ein spezieller Speicher. Es bildet einen Puffer zwischen dem Datenbus und dem Arbeits-

speicher. Hier werden aus dem Arbeitsspeicher angeforderte Daten zunächst zwischengespeichert, um dann über den Datenbus an das entsprechende Register weitergeleitet zu werden. Andererseits gelangen Daten aus anderen CPU-Registern nur über diesen Zwischenspeicher in den Arbeitsspeicher. Die Größe des Datenregisters entspricht der Größe eines Speicherwortes im Arbeitsspeicher: in vielen Prozessoren hat es eine Größe von 16 oder 32 Bit, in CPUSIM 10 Bit.

Das Instruktionsregister (IR)

Das Instruktionsregister (auch Befehlsregister genannt) hat in PC-Prozessoren in Verbindung mit einer Befehls-Warteschlange (Queue) die Aufgabe, den auszuführenden Befehlscode für die Übergabe an den Decodierer zwischenzuspeichern. Es hat i. d. R. eine Größe von 8 Bit. Der Prozessor ist in der Lage, eine Fülle von Anweisungen in Bruchteilen von Sekunden auszuführen. Um ein rationelles Arbeiten zu gewährleisten, muß er ständig mit neuen Befehlen gefüttert werden.

Da die Befehle aus dem Arbeitsspeicher geholt werden müssen, was (gemessen an der Prozessorgeschwindigkeit) viel Zeit in Anspruch nimmt, ist es sinnvoll, sie für den Prozessor schon vorher in einer Befehls-Warteschlange bereitzustellen, auf die er ohne Zeitverlust zugreifen kann. Sie wird während der Prozessoraktivität ständig mit nachfolgenden Befehlswörtern aufgefüllt. CPUSIM verfügt aus Gründen der Übersichtlichkeit nur über einen Prozessor, der die Befehle sowie Ein- und Ausgaben steuert. Darum wurde auf die Simulation einer Befehls-Warteschlange verzichtet.

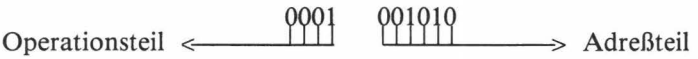
Je nach Aufbau und Organisation des Arbeitsspeichers ist es möglich, daß der übertragene Speicherinhalt nicht den vollständigen Befehlscode enthält (Mehradreßmaschinen). In diesem Fall ist der Befehlscode auf die nachfolgenden Speicherplätze verteilt. Die hier simulierte CPU ist aus Gründen der Übersichtlichkeit als Ein-Adreß-Maschine konzipiert. Der gesamte Befehl läßt sich in einem Speicherplatz unterbringen und kann mit einem Lesezyklus vollständig erfaßt werden.

Mit Hilfe der im Befehlszähler stehenden Adresse wird die Stelle im Arbeitsspeicher ermittelt, an der der nächste auszuführende Befehl steht. Dieser dort stehende Befehlscode wird in das Instruktionsregister übertragen und decodiert. Nach der Decodierung des Befehlscodes wird der Adreßteil über den Adreßbus zum Adreßregister weitergeleitet, von wo aus dann der Zugriff auf den entsprechenden Speicherplatz des Arbeitsspeichers vorgenommen wird. Während des Programmablaufes enthält das Instruktionsregister also immer den Programmbefehl, der gerade ausgeführt wird.

Der Decodierer

Alle Befehle, die die CPU ausführen kann, werden durch einen sog. Befehls- oder Operationscode eindeutig definiert. Die spezifischen Bitfolgen werden vom jeweiligen Prozessorhersteller festgelegt. In der Regel stehen Hilfsprogramme – z. B. Assembler – zur Verfügung, die diese Bitfolgen sehr ähnlich wie in CPUSIM aus mehr oder weniger selbsterklärenden Kurzwörtern (mnemotechnische Befehle wie „MOV", „ADD", „JMP") erzeugen.

Der Befehlscode spaltet sich in einen Operationsteil und einen (oder mehrere) Adreßteil(e) auf. In CPUSIM wird nur eine Adresse verwendet (Ein-Adreß-Maschine). Der Befehlscode hat hier eine Länge von 10 Bit, die sich folgendermaßen aufteilen:



Es lassen sich mit den 4 Operations-Bits also insgesamt 16 verschiedene Befehle verschlüsseln. Der Decodierer entschlüsselt nun den im Instruktionsregister zwischengespeicherten Befehlscode, indem er ihn zunächst in Operations- und Adreßteil aufspaltet. In realen PC-Prozessoren geschieht das durch das Lesen aufeinanderfolgender Bytes. Dann decodiert er den Operationsteil, um festzustellen, welcher Befehl auszuführen ist. Dieser Befehl kann nun in Verbindung mit der zugehörigen Adresse durch das Steuerwerk der CPU abgearbeitet werden. Zur besseren Übersicht wird das Ergebnis der Entschlüsselung im Decodierer-Funktionselement der Laufmodus-Maske in sprachlicher Form angezeigt.

Das Interruptregister

Das Interruptregister (auch UA = Unterbrechungsanforderungsregister) ist ein internes Register für das Steuerwerk. Es ermöglicht dem Prozessor, ein laufendes Programm zu unterbrechen und von dort in ein Unterprogramm zu verzweigen. Mit dieser Programmunterbrechung (Interrupt) kann ein schnell arbeitender Prozessor maximal ausgelastet werden, da er beispielsweise zeitintensive Ein-/Ausgabebefehle an Unterprogramme weiterleitet und kurze arithmetische Befehle selbst ausführt. Ein Interrupt kann sowohl durch ein Programm, z. B. zwecks Aufrufen von Betriebssystemroutinen, ausgelöst werden (in Assembler: „INT") als auch von außen durch ein Peripheriegerät (z.B. Bereit-, Nicht-bereit- oder Fertigmeldungen von Druckern und Diskettenlaufwerken).

Wird ein Interrupt angefordert, so wird ein bestimmtes Bit (Flag) in der Steuereinheit gesetzt, das der Prozessor vor einer jeden Befehlsausführung abfragt. Mit dem Setzen dieses Bits wird gleichzeitig eine Adresse im Interruptregister generiert, die die Anfangsadresse des Interrupt-Unterprogrammes darstellt. Nachdem der Prozessor bestimmte Registerinhalte „gerettet" (im Arbeitsspeicher gesichert) hat, u. a. auch die Rücksprungadresse des soeben unterbrochenen Programmes, verzweigt der Ablauf in das Unterprogramm und führt dieses aus. Danach wird das gesetzte Bit gelöscht, die Registerinhalte des unterbrochenen Programmes werden geladen, so daß es wieder fortgesetzt werden kann.

Viele PC-Prozessoren gehen beim Auffinden der Interruptadressen aufgrund der Adreßlänge den Umweg über eine Vektortabelle. Sie wird in den ersten 1024 Speicherplätzen des Arbeitsspeichers abgelegt. Das Interruptregister von CPUSIM enthält bei einer Interruptanforderung (Interruptbehandlung) eine Adresse, die nicht im Arbeitsspeicher vorhanden ist („1111111" = 127). Es soll auch nur angedeutet werden, welche Mechanismen in Bewegung kommen, wenn ein Interrupt abläuft. Mehr kann dieses Lernprogramm nicht leisten, da es den Einführungscharakter nicht verlieren soll.

Der Stackpointer (SP)

Der Stackpointer (Stapelzeiger) ist ein Zeiger (Adreßspeicherplatz). Er enthält die Adresse des obersten Speicherplatzes für einen Speicherbereich (Stack) im Arbeitsspeicher. Benötigt wird er beim Aufruf eines Unterprogrammes zur Speicherung der Rücksprungadresse des aufrufenden Programmes. Bei vielen PC-Prozessoren ist er als 16-Bit-Register ausgelegt. In der hier simulierten CPU hat er gemäß der Adreßbusbreite eine Größe von 6 Bit. Er ist mit der Adresse des letzten Speicherplatzes („111111" = 63) vorbelegt. Diese wird bei jedem weiteren Unterprogrammaufruf um 1 verkleinert, beim Rücksprung wieder um 1 vergrößert. Die letzten 3 Speicherplätze (61 bis 63) des Arbeitsspeichers sind für den Stack reserviert und können von Ihnen nicht durch eigene Befehle oder Daten belegt werden.

Das Rechenwerk

In einem Rechenwerk (auch ALU = Arithmetic Logic Unit genannt) werden arithmetische und logische Operationen mit binären Daten durchgeführt. Es besteht im Wesentlichen aus Registern und Addierschaltungen, weil die meisten Rechenoperationen auf elementare Additionen zurückgeführt werden können. Das wichtigste Register im Rechenwerk ist der Akkumulator, dane-

ben werden bei vielen PC-Prozessoren noch eine Reihe weiterer Register für zusätzliche Operanden, für den Übertrag, für das stellenweise Verschieben, für die Komplementbildung (duale Subtraktion) usw. verwendet.

Es würde aber den Rahmen dieses Lernprogrammes sprengen, die Gesamtheit dieser Register und deren komplexes Zusammenspiel darstellen zu wollen. Das Rechenwerk von CPUSIM besteht daher nur aus zwei Registern:

- dem Akkumulator, der den ersten Operanden und später das Ergebnis der Rechnung enthält,
- und einem Operandenregister, das den zweiten Operanden aufnimmt.

Die Rechenoperation wird durch das blinkende Rechenzeichen im ALU-Funktionselement angezeigt, den dezimalen Wert des Akkumulators können Sie auf dem Bildschirm unterhalb der Binärdarstellung ablesen.

Der Akkumulator (AX)

Der Akkumulator (oder kurz Akku) ist ein spezielles Speicherregister des Rechenwerks in der CPU. Bei vielen Prozessoren heißt es auch AX-Register; es hat dort eine Größe von 16 Bit. Es läßt sich als Ganzes ansprechen, man kann es aber auch in ein höherwertiges Byte (AH) und ein niederwertiges Byte (AL) zerlegen und ansprechen. In CPUSIM hat der Akkumulator eine Größe von 10 Bit. Er kann damit Zahlen im Wertebereich von +511 bis −512 aufnehmen. Der dezimale Wert des Akkumulatorinhalts wird auf dem Bildschirm unterhalb der Binärdarstellung angezeigt.

Benötigt wird der Akkumulator immer dann, wenn eine der vier in diesem Programm möglichen Grundrechenarten ausgeführt wird oder ein Speicherinhalt des Arbeitsspeichers zu verschieben bzw. neu zu füllen ist (bei echten AX-Registern kommen noch weitere Aufgaben hinzu). Zur Ausführung einer Grundrechenart benötigt man immer 2 Zahlen (Operanden) und eine Rechenanweisung (arithmetischen Operator). Im Akkumulator steht grundsätzlich der 1. Operand oder – anders ausgedrückt – der Operand links vom Operatorenzeichen. Der 2. Operand wird dann zum ersten addiert, vom ersten subtrahiert usw. Das Ergebnis wird wiederum im Akkumulator abgespeichert.

‚Vorbelegt' werden kann der Akkumulator nur durch den CPUSIM-Befehl „Laden". Die Abspeicherung seines Inhaltes erfolgt mit dem Befehl „Speichern". Beim Programmstart befindet er sich also in einem nichtdefinierten Zustand. Wird ohne vorhergehenden Ladebefehl eine Rechenoperation durchgeführt, ist das Ergebnis nicht vorhersagbar. Wie bei anderen Registern auch, wird der Akkumulator nicht gelöscht, sondern immer nur überschrie-

ben. Während des Programmlaufes bleibt der Inhalt des Akkumulators solange bestehen, bis er überschrieben wird. Auch nach dem Abspeichern des Inhaltes bleibt er unverändert im Register stehen.

Der Takt

Das Steuerwerk steuert mit Hilfe von Taktsignalen (Clock) die Folge der Ereignisse, die zur Ausführung der einzelnen Befehle notwendig sind. In Abhängigkeit vom Befehlscode werden über eine Vielzahl von Leitungen Impulse an Geräte und Register ausgegeben, Daten- und Adreßwege geöffnet und geschlossen, Lese- und Schreibvorgänge ausgelöst und vieles mehr. Die Abläufe in der CPU sind zyklisch und wiederholen sich Befehl für Befehl. Die Anzahl der Befehlstakte (Schritte) variiert, so daß längere und kürzere Befehlsfolgen entstehen. Die Taktdauer, die durch einen unabhängigen Taktgenerator (Schwingquarz) festgelegt wird, bewegt sich bei echten Prozessoren im Bereich von Nanosekunden (10^{-9}) und beeinflußt maßgeblich die „Schnelligkeit" eines PC's.

In diesem Lernprogramm lassen sich die Befehlsschritte im Infofeld des Laufmodus verfolgen. Der Takt wird durch das gleichmäßige Knackgeräusch symbolisiert, sofern Sie den Ton nicht ausgeschaltet haben. Beide Maßnahmen bilden aber nicht genau die Realität ab, sondern haben hier mehr Demonstrationscharakter. Aus Gründen der Übersichtlichkeit wurde nämlich auch auf sämtliche Takt- und Steuerleitungen in der grafischen Darstellung verzichtet, um die beiden wichtigsten internen Datenwege von Mikroprozessoren besser zu Geltung zu bringen: den Adreßbus und den Datenbus.

Der Adreßbus

Unter einem Adreßbus versteht man eine Vielzahl von Leitungen, die ausschließlich der Übertragung von Speicheradressen dienen. Er verbindet das Interruptregister, den Stackpointer, den Befehlszähler und das Instruktionsregister mit dem Arbeitsspeicher. Die Auswertung der anliegenden Adressen besorgt das dem Arbeitsspeicher vorgeschaltete Adreßregister. Die zu übertragenden Adressen sind grundsätzlich für das Adreßregister vorgesehen, um einen bestimmten Speicherplatz des Arbeitsspeichers anzuwählen (zu adressieren). Im Lese- bzw. Schreibzyklus wird dann der so adressierte Speicherplatz gelesen bzw. beschrieben. Der Adreßbus vieler Prozessoren hat eine Breite von 20 Bit (20 Datenleitungen); damit kann man ca. 1 Million Speicherplätze adressieren. Der Adreßbus von CPUSIM ist 6 Bits breit, woraus sich eine Arbeitsspeicherkapazität von 64 direkt adressierbaren Speicherplätzen (2^6) ergibt.

Der Datenbus

Unter einem Datenbus versteht man eine Anzahl von Leitungen, die die Übertragung von Informationen (Daten) zwischen Rechenwerk (mit Akkumulator), Instruktionsregister (incl. Decodierer) sowie anderen Registern und dem Arbeitsspeicher ermöglichen. Die Schnittstelle zwischen Arbeitsspeicher und Datenbus bildet das Datenregister. Der Informationstransport vollzieht sich im Gegensatz zum Adreßbus in beiden Richtungen. Es werden sowohl Daten als auch Adressen (wenn sie als Daten behandelt werden) transportiert. Der Datenbus vieler Prozessoren hat eine Breite von 16 oder 32 Bit. Der Datenbus von CPUSIM ist gemäß der Speicherwortgröße des Arbeitsspeichers 10 Bit breit.

2.4 Programmablauf in der Zentraleinheit

Die graphische Realisation des Programmablaufes, so wie sie sich im Laufmodus (siehe 3.3) zeigt, stellt neben verschiedenen Vereinfachungen eine extreme zeitliche Dehnung der tatsächlichen Vorgänge dar. Daten- und Adreßtransporte, die im echten Prozessor innerhalb von Nanosekunden als Spannungszustände anliegen, werden in der Simulation als dynamische Bewegungsabläufe durch fortschreitende Lichtpunkte auf den Daten- und Adreßwegen, begleitet von kurzen „Takt"-geräuschen, veranschaulicht. Dabei erscheinen die Registersymbole der Ausgangs- und Endstation ggf. mit ihren Inhalten ebenfalls in hervorgehobener Farbe. Es wurde zugunsten der Übersichtlichkeit auf die Darstellung von Registerschiebevorgängen, Komplementbildungen etc. verzichtet.

Schreib- Lesezyklus

Die Datenverbindung vom Prozessor zum Arbeitsspeicher und umgekehrt wird durch den Schreib- bzw. Lesezyklus realisiert. Beim Schreibzyklus steht die abzuspeichernde Information im Datenregister, die Arbeitsspeicheradresse als Zielort steht im Adreßregister. Das Steuerwerk löst nun aufgrund des aktuellen Befehls (der den Schreibvorgang enthält) einen Schreibimpuls aus, der den Speicher aktiviert und die anliegenden Datenbits an der anliegenden Adresse ablegt. Je nach Geschwindigkeit des Speichermediums muß ein echter Prozessor (oder der Speicher) Wartezustände (wait states) zur Synchronisation durchlaufen.

Beim Lesezyklus steht die Leseadresse im Adreßregister. Durch einen Leseimpuls des Prozessors gelangt der Inhalt der angegebenen Adresse in das Datenregister und steht dem Prozessor zur Verfügung.

In CPUSIM wird neben der farblichen Gestaltung die Suchadresse hell unterlegt und der Schreib- bzw. Leseimpuls durch ein einprägsames Geräusch untermalt. Die blinkenden Pfeile in der Mitte zeigen die Richtung des Adreß- und Datenflusses an.

Operationszyklus

Unter dem Operationszyklus (oder Befehlszyklus) versteht man die Abfolge der Prozessorzustände, die bis zum Abruf des nächsten Befehls aus dem Arbeitsspeicher durchlaufen werden. Im ersten Schritt wird ein Befehl (oder auch ein Teil davon) aus der Befehlsschlange in das Instruktionsregister geladen. Dabei wird in echten Prozessoren vorher abgefragt, ob ein Interruptbit gesetzt ist und ggf. ein Interrupt auszuführen ist. In der Zwischenzeit wird die Befehlsschlange durch eine andere Systemkomponente wieder mit Befehlen aufgefüllt. In CPUSIM besteht dieser erste Schritt aus drei Phasen:

- Bereitstellen der Befehlsadresse aus dem Befehlszähler im Adreßregister und Erhöhen des Befehlszählers um 1
- Inhalt der Befehlsadresse im Datenregister bereitstellen (Lesezyklus)
- Befehlstransport in das Instruktionsregister und Decodieren

Im zweiten Schritt (auch Executionszyklus genannt) wird der geladene Befehl vom Steuerwerk ausgeführt. Dabei hängt die Anzahl der Teiloperationen vom jeweiligen Befehl ab. Es werden Ein-, Ausgaben, Lese- oder Schreibvorgänge, interne Rechenoperation und Datenübertragungen durchgeführt. In CPUSIM werden die Teilschritte je nach Befehlsart unterschiedlich ausgeführt. In vielen Fällen werden mit dem Adreßteil des Befehls Schreib- oder Lesevorgänge ausgeführt, Rechenoperationen oder Datentransporte schließen sich an. Jeder Einzelschritt wird im Infofeld dokumentiert. Ein Befehlszyklus endet mit der vollständigen Abarbeitung aller Teilschritte des aktuellen Befehls.

3. Funktionsbeschreibung von CPUSIM

3.1 Struktur des Lernprogramms CPUSIM

Ihre Manövriermöglichkeiten im Lernprogramm sehen Sie am besten in der Abb. 4 auf einen Blick. Nach dem Programmstart (siehe 1.4) rufen Sie sinnvollerweise zunächst mit <F2> den Editmodus auf. Sie können zwar auch unmittelbar mit <F4> in den Laufmodus wechseln, finden dann aber natürlich noch kein simulierbares Testprogramm im Arbeitsspeicher vor. Ferner können Sie auch gleich den Hilfemodus aufrufen, wenn Sie das „elektronische Lexikon" statt des Begleitheftes für Auskünfte nutzen wollen. Der Hilfemodus ist auch von allen anderen Punkten des Lernprogramms aus stets erreichbar.

Nach Aufruf der Editmodus-Maske stehen Sie vor der Alternative, entweder ein vorhandenes Testprogramm – von uns mitgeliefert oder von Ihnen in früheren CPUSIM-Sitzungen bereits erstellt – mit <F2> zu laden oder ein neues Testprogramm zu editieren. Natürlich können Sie auch ein vorhandenes Testprogramm laden und dann verändern. Ob alt, völlig neu oder nur modifiziert: sobald Sie glauben, Ihr Testprogramm sei „simulationsreif", wechseln Sie mit <F4> in den Laufmodus. Mit <F6> wird die Simulation von dort aus angestoßen. Was Sie im Edit- und Laufmodus sonst noch tun können, erfahren Sie in den beiden folgenden Abschnitten. Danach geben wir Ihnen einige Hinweise zur kontextsensitiven Online-Hilfe und zu den einzelnen Befehlen von CPUSIM, die Sie spätestens brauchen, wenn Sie Ihr erstes Testprogramm produzieren wollen.

3.2 Der Editmodus

Der Editmodus dient zum Laden und Modifizieren vorhandener bzw. zum Erstellen neuer Testprogramme, die im Arbeitsspeicher abgelegt werden. Auf der linken Seite der Editmodus-Maske (Abb. 5) sind alle CPUSIM-Befehle und die dazugehörigen Tastenkombinationen aufgeführt, mit denen Sie Ihre Testprogramme editieren können.

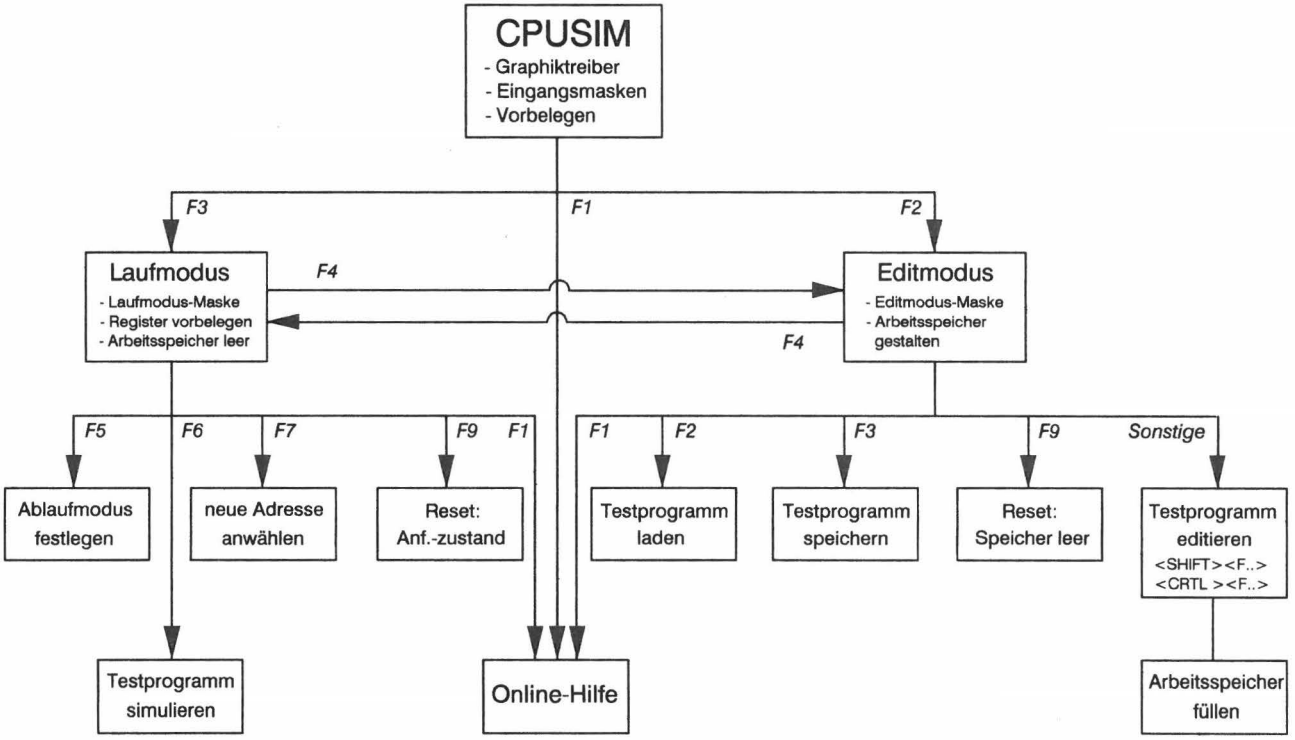


Abb. 4: Steuerung des Lernprogramms

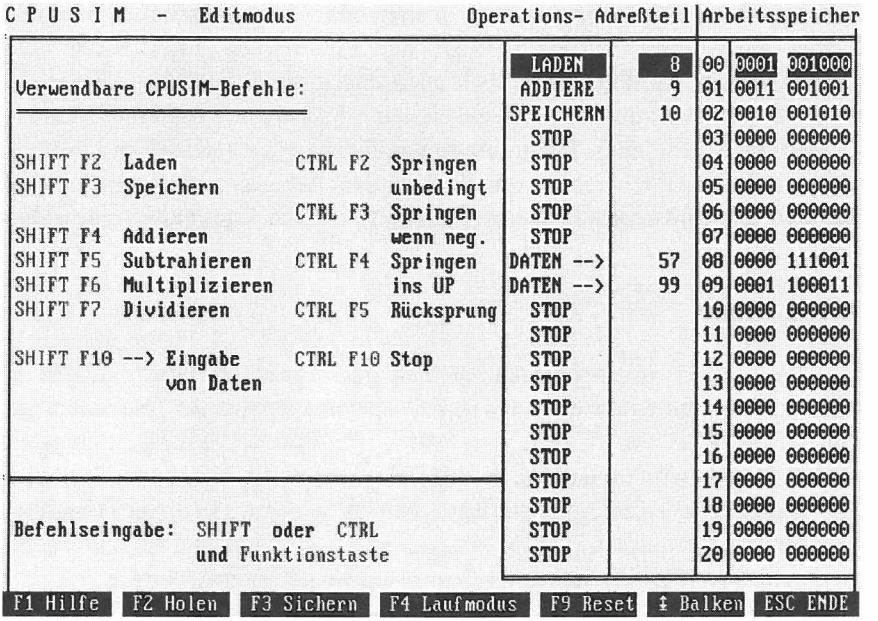


Abb. 5: Editmodusmaske

Bei den meisten Befehlen werden Sie noch zur Eingabe einer Adresse bzw. eines Datenwertes aufgefordert. Hierfür ist ein Eingabefeld unterhalb der Befehlsliste vorgesehen. Mit <RETURN> (↵) wird der Befehl bzw. der Datenwert in den Arbeitsspeicher von CPUSIM auf der rechten Bildschirmseite übernommen. Der Arbeitsspeicher wird durch vier Spalten symbolisiert:

- die erste Spalte links nimmt bei Befehlen den Operationsteil (als Text), bei Datenfeldern den Hinweis ‚Daten' auf,
- die zweite Spalte enthält bei Befehlen den Adreßteil bzw. bei Datenfeldern den Datenwert in dezimaler Schreibweise,
- darauf folgt in der dritten Spalte die aktuelle dezimale Arbeitsspeicheradresse,
- rechts davon steht dann in der letzten Spalte bei Befehlen die binäre Verschlüsselung des Operationsteils (die vier linken Stellen) sowie sein binär verschlüsselter Adreßteil (die rechten sechs Stellen) oder bei Datenfeldern eine zehnstellige Dualzahl.

Ein Lichtbalken kennzeichnet den aktuellen Arbeitsspeicherplatz, auf den man jeweils zugreifen kann. Er steht zu Beginn immer auf dem Speicherplatz

mit der Adresse 00. Ist der jeweilige Speicherplatz mit einem Befehl oder mit Daten beschrieben worden, springt der Lichtbalken automatisch eine Adresse weiter. Er läßt sich jedoch auch durch die Cursortasten bewegen. Man hat hier nicht nur die Möglichkeit, das eingegebene Programm zu überprüfen, man kann auch Testprogrammergebnisse einsehen, die im Laufmodus durch Speicherbefehle zusätzlich in den Arbeitsspeicher geschrieben werden. Die Bildschirmmaske wird unten durch die Kommandozeile abgeschlossen.

Eingabefeld

Bei der Eingabe von Programmbefehlen oder Daten im Editmodus ist es erforderlich, Zahlenwerte für den Arbeitsspeicher einzugeben. Nachdem Sie mit <SHIFT> bzw. <CTRL> + <Funktionstaste> die Eingabe eingeleitet haben, erscheint im Bereich des Infofeldes rechts ein blinkender Pfeil und im Anschluß daran ein hell unterlegtes Feld mit einer ‚0'. In diesem Eingabefeld blinkt der Cursor. CPUSIM erwartet jetzt von Ihnen eine Zahleneingabe in einem Wertebereich, der unter dem Eingabefeld beschrieben ist:

- Eingabe eines Datenwertes (<SHIFT F10>): −512 bis +511
- Eingabe einer Adresse (bei Programmbefehlen): 0 bis + 60

Es werden nur Ziffern (bzw. bei Daten auch ein Vorzeichen am Anfang) akzeptiert, in der Kommandozeile werden drei weitere erlaubte Tasten benannt:

- Mit <BACK> kann eine falsch eingegebene Ziffer gelöscht werden.
- Die Eingabe ist grundsätzlich mit <RETURN> (↵) abzuschließen.
- Mit <ESC> wird die bisherige Eingabe storniert, sie kann wiederholt werden.

Bei allen anderen Eingaben ertönt ein Warnton. Bereichsüberschreitungen führen wie <ESC> zu einer Neueingabe. Sollte dennoch ein Speicherplatz einen falschen Wert, einen falschen Befehl oder eine falsche Adresse enthalten, können Sie ihn jederzeit mit einem anderen Wert überschreiben. Die Eingabe von Befehlen und Daten unterscheidet sich wie folgend beschrieben.

Befehlseingabe

Drückt man die Tastenkombination eines zu codierenden Befehls (z.B. für LADEN: <SHIFT>-Taste mit Funktionstaste <F2>), erscheint im Eingabefeld neben einer Eingabeaufforderung ein blinkender Pfeil. Hier soll die zum Befehl gehörende Adresse (im angegebenen Adreßbereich) eingetragen

werden. Die eingegebene Adresse wird mit <RETURN> bestätigt. Entspricht sie nicht dem zulässigen Adreßbereich, wird sie gelöscht und eine neue Eingabe erwartet. Ist die Eingabe korrekt, wird Befehl und Adresse binär verschlüsselt und in das aktuelle Arbeitsspeicherfeld geschrieben.

Dateneingabe

Sollen für ein Testprogramm Werte (Daten) eingegeben werden, gelangt man mit <SHIFT> und <F10> in das oben beschriebene Eingabefeld und kann nun Werte im unten angegebenen Bereich eintragen. Alle Speicherplätze sind mit dem Stopbefehl vorbelegt. Daher ist es nicht notwendig, am Ende eines Programmes noch einmal den Stopbefehl zu codieren. Es ist jederzeit möglich, bereits gefüllte Speicherplätze zu überschreiben.

Testprogramm laden

Im Editmodus haben Sie auch die Möglichkeit, abgespeicherte Testprogramme aus dem Unterverzeichnis CPUSIM in den Arbeitsspeicher zu laden. Es ist nicht möglich, die Testprogramme direkt von der Arbeitsdiskette zu laden. Sie müssen vor dem Aufruf von CPUSIM auf der DOS-Ebene mit dem COPY-Befehl von der Diskette auf die Festplatte kopiert werden, i. d. R. mit

COPY A: *CPU C:;CPUSIM

Mit <F2> „Holen" wird ein Testprogrammladefenster eröffnet, das Ihnen maximal zehn Testprogrammnamen auf einmal anzeigt. Mit den Cursortasten können Sie nun den Lichtbalken, der zunächst auf dem obersten Eintrag steht, auf das von Ihnen gewünschte Testprogramm bewegen. Wählen Sie das Programm einfach durch RETURN (↵). Enthält das Verzeichnis mehr als zehn Testprogramme, können Sie mit den Cursortasten die Einträge im Bildschirmfenster rollen, so daß sie alle Testprogramme angezeigt bekommen. Um schnell an den ersten bzw. letzten Eintrag zu gelangen, verwenden Sie die Funktionstasten <HOME> (‚Pos1') bzw. <END> (‚Ende'). Mit <ESC> können Sie das Testprogrammladefenster jederzeit schließen. Nachdem Sie ein Testprogramm ausgewählt haben, finden Sie den Programminhalt im Arbeitsspeicher wieder und können damit arbeiten. Das Testprogrammladefenster schließt sich.

Testprogramm speichern

Im Editmodus können Sie ein einmal erstelltes Testprogramm abspeichern. Mit <F3> „Sichern" wird ein Testprogrammspeicherfenster eröffnet, das

zur Eingabe des Namens, unter dem das Testprogramm abgelegt werden soll, auffordert. Die Extension (CPU) wird von CPUSIM automatisch angefügt, die Eingabe ist mit <RETURN> abzuschließen. Ist unter dem eingegebenen Namen bereits eine Datei abgespeichert, erscheint die Meldung: „Datei vorhanden! Überschreiben (J/N)?". Mit der Eingabe ‚N' wird die Funktion ohne weitere Auswirkungen verlassen, mit ‚J' wird das Testprogramm gesichert. Das bisher unter diesem Dateinamen gespeicherte Testprogramm geht dann verloren! Alle Testprogramme werden im Unterverzeichnis CPUSIM mit der Extension „CPU" abgespeichert. Wenn Sie dem Dateinamen die Diskettenlaufwerkskennung voranstellen (z. B. A: Dateiname), wird das Testprogramm direkt auf der Arbeitsdiskette archiviert. **Versuchen Sie auf keinen Fall, Testprogramme auf der Userdiskette abzuspeichern.** Testprogramme dürfen nicht mit anderen Editoren als dem von CPUSIM erstellt oder verändert werden!

Arbeitsspeicher löschen

Die Taste <F9> „Reset" bewirkt im Editmodus ein vollständiges Löschen des gesamten Arbeitsspeichers mit allen dort abgelegten Programmbefehlen und Daten. Um einem versehentlichen Löschen vorzubeugen, erscheint nach Betätigen von <F9> — wenn der Arbeitsspeicher nicht leer ist — im Eingabefeld eine zusätzliche Abfrage, ob der Arbeitsspeicher auch wirklich gelöscht werden soll. Mit der Tastenkombination <CTRL> und J wird dann die Resetfunktion ausgeführt.

Online Hilfe

Mit <F1> kann jederzeit die Online-Hilfe aufgerufen werden (siehe 3.4). Nach dem Anzeigen der Hilfe kehrt das Lernprogramm in den Editmodus zurück.

Fehlermeldungen

Konnte CPUSIM die Funktion nicht erfolgreich durchführen, erscheint eine entsprechende Fehlermeldung. Soll dann der Versuch wiederholt werden, muß erneut die Taste <F2> bzw. <F3> betätigt werden.

Programmabbruch

Mit <ESC> kehren Sie zurück zu den Eingangsmasken.

3.3 Der Laufmodus

Im Laufmodus werden die typischen Vorgänge in einer CPU bei der Abarbeitung von Befehlen bzw. Testprogrammen veranschaulicht. Die wichtigsten Bestandteile einer CPU werden in der Laufmodus-Maske (Abb. 6) durch umrandete Fehler dargestellt und mit den entsprechenden Bezeichnungen versehen. Die Lage der einzelnen Bestandteile und ihrer Verbindungswege hat mit der tatsächlichen Anordnung in einer echten CPU wenig zu tun und wurde nach didaktischen und optischen Gesichtspunkten ausgewählt. Außerdem wurden aus Platzgründen und zur besseren Übersicht einige Vorgänge vereinfacht dargestellt. Hierzu zählen insbesondere Abläufe im Rechenwerk und der Zugriff auf den Arbeitsspeicher.

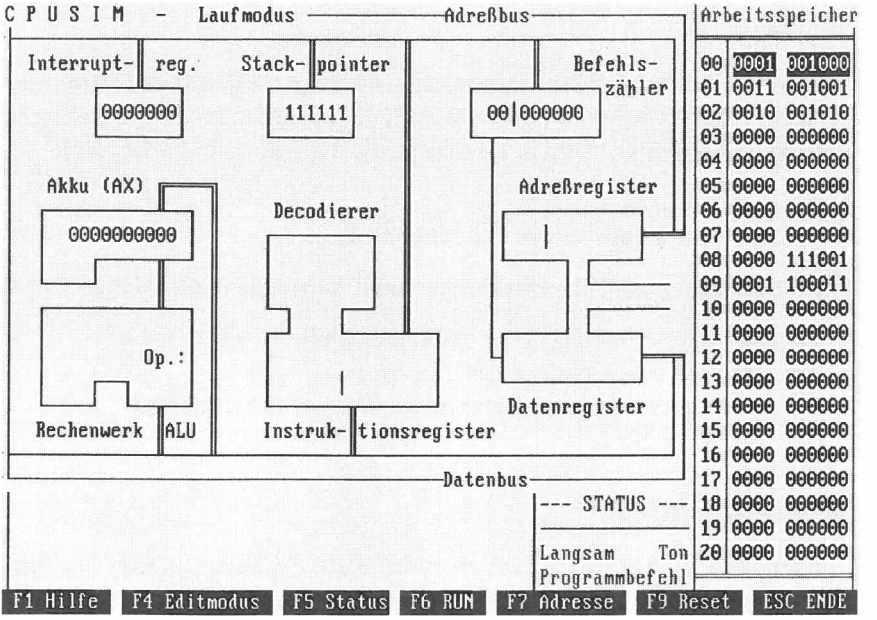


Abb. 6: Laufmodus-Maske

Arbeitsspeicher im Laufmodus

Die rechte Seite des Bildschirms wird von der graphischen Darstellung des Arbeitsspeichers eingenommen, wie Sie Ihnen aus dem Editmodus vertraut ist. Die beiden im Editmodus links stehenden Spalten, die nur zur besseren Lesbarkeit der Befehle hinzugefügt wurden, werden hier im Laufmodus aus Platzgründen überblendet. Wechseln Sie zur Probe einmal durch mehrfaches

Drücken von <F4> zwischen den beiden Modi hin und her! In den verbleibenden Spalten stehen:

- links die zweistelligen dezimalen Arbeitsspeicheradressen (Speichernummer oder Platznummer) und
- rechts die binäre Verschlüsselung des Befehls, der sich aus dem Operationsteil (vier Stellen links) und dem Adreßteil (sechs Stellen rechts) zusammensetzt, bzw. eine zehnstellige Dualzahl.

Im unteren Bildschirmteil wurde links Platz für ein Infofeld gelassen, rechts davon kann man den Status des Programmablaufs ablesen. Die Bildschirmmaske wird ganz unten durch die Kommandozeile abgeschlossen.

Infofeld

Dort, wo im Editmodus das Eingabefeld steht, zeigt CPUSIM im Laufmodus in einem Infofeld während und auch nach dem Programmlauf die einzelnen Befehle und Befehlsschritte in Kurzform an, wie z.B. beim Ladebefehl:

- Befehl: LADEN
- Schritt: Datentransport in den Akkumulator

Hier erscheinen auch Fehlermeldungen und Warnungen, z.B. wenn

- bei einer Rechenoperation der Zahlbereich überschritten wird,
- eine Division durch Null aufgetreten ist oder
- ein nicht interpretierbarer Befehlscode festgestellt wurde etc.

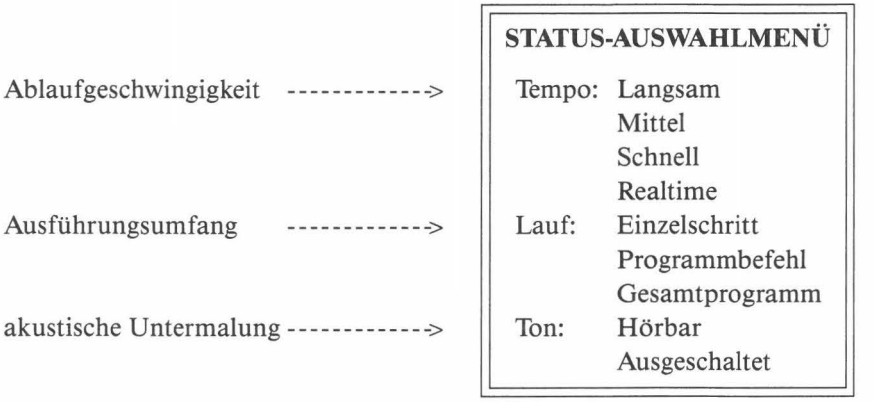
Ablaufmodus festlegen

Mit der Taste <F5 Status> kann man den Status festlegen, in dem das Testprogramm ablaufen soll. Im Status-Auswahlmenü erscheinen folgende Wahlmöglichkeiten:

- Tempo: die Simulationsgeschwindigkeit des Testprogrammes kann zwischen Zeitlupentempo und Quasi-Echtzeit vierstufig variiert werden.
- Lauf: die Simulationsabschnitte des Testprogrammes lassen sich von Einzelschritten eines Befehls über einen kompletten Befehlszyklus bis hin zum Gesamtprogramm erweitern. Im Modus „Einzelschritt“ und „Programmbefehl“ wird immer nur eine Befehlsphase bzw. ein ganzer Befehl ausgeführt. Im Modus „Ge-

samtprogramm“ läuft das komplette Testprogramm ab, es sei denn, man unterbricht den Programmlauf mit der nur hier möglichen Taste <F10> (für Interrupt).

- Ton: die akustische Untermalung, die die Simulation bei den Programmabläufen mit Tönen und Geräuschen unterstreicht, ist abschaltbar.



Die drei Wahlbereiche sind gleichzeitig einstellbar. Die Geschwindigkeitsstufen sind so eingerichtet, daß die Stufe „Langsam“ das Kennenlernen der Abläufe ermöglicht und vertieft, während sich „Schnell“ für das Austesten von Programmen anbietet. „Realtime“ soll Ihnen einen Eindruck von den wahren Prozessorgeschwindigkeiten vermitteln, die in Wirklichkeit noch weitaus höher liegen.

Die Statusauswahl wird mit einem Lichtbalkenmenü gesteuert, im Statusfeld links neben dem Arbeitsspeicher werden alle Änderungen protokolliert. Zu Beginn des Lernprogrammes sind Geschwindigkeit und Laufmodus mit „Langsam“ und „Einzelschritt“ vorbelegt, der Ton ist eingeschaltet.

Testprogramm simulieren

Enthält der Arbeitsspeicher ein Testprogramm, kann man dieses durch die Taste <F6> „RUN“ in dem gewählten Ablaufmodus starten. Der ausgeführte Befehl mit dem aktuell durchlaufenen Befehlsschritt wird im Infofeld mit einem Kurztext benannt, so daß man in jeder Phase der Simulation über den Prozessorzustand informiert ist. Dazu zeigt die Kommandozeile dem Benutzer an, wie er die Simulation steuern kann. Im „RUN“-Modus sind

keine Eingriffe möglich, bis auf die Interruptbehandlung mit <F10> im Status „Gesamtprogramm" (siehe unten).

Der „RUN"-Modus endet im Status „Einzelschritt" mit der Abarbeitung eines einzelnen Befehlsschrittes. Im Modus „Programmbefehl" wird der aktuelle Befehl mit allen Schritten ausgeführt, im Modus „Gesamtprogramm" wird das komplette Testprogramm aus dem Arbeitsspeicher durchlaufen. Das Ende des „RUN"-Modus erkennen Sie daran, daß die Aktivitäten des Bildschirms ruhen und die untere Bildschirmzeile wieder die Kommandozeile des Laufmodus anzeigt.

Interruptbehandlung

Läuft das Testprogramm im Modus „Gesamtprogramm", können Sie es durch die Taste <F10> unterbrechen. Der Interrupt als Unterbrechung eines laufenden Programmes kann in CPUSIM nur symbolisch als Hardwareinterrupt von außen ausgelöst werden. Dabei wird im Interruptregister eine Adresse erzeugt, die nicht im Arbeitsspeicher liegt. Hier soll angedeutet werden, daß eine im Rechner fest implementierte Interruptroutine angefordert wird. Die Simulation zeigt noch den Adreßtransport zum Adreßregister und weist durch ein blinkendes „HALT" im Decodierer auf die Unterbrechung hin. Im Infofeld erscheint der erläuternde Text:

Die Adresse des Interruptregisters verzweigt den Programmablauf zu einem Festwertspeicher, der eine Interruptbehandlungsroutine enthält.

Das Drücken einer beliebigen Taste führt in den Laufmodus zurück, Sie können Ihr Testprogramm dann mit <F6> „RUN" fortsetzen. Da CPUSIM keine Bildschirmein- und -ausgaben, Diskettenspeicherzugriffe und höhere Verwaltungsfunktionen (Betriebsystemaufgaben) simuliert, die auf Interruptaufrufen basieren, ist eine ausführlichere Interruptbehandlung hier nicht notwendig.

Neue Adresse anwählen

Beim erstmaligen Start eines Testprogrammes beginnt die Abarbeitung der Befehle bei der Adresse 00. Mit der Taste <F7> „Adresse" ist es jederzeit möglich, die Adresse des nächsten auszuführenden Befehls neu zu setzen. Im Modus „Einzelschritt" ist darauf zu achten, daß <F7> nur zu Beginn bzw. am Ende eines Befehls ausgeführt werden kann.

Zentralprozessor initialisieren (Reset im Laufmodus)

Die Taste <F9> „Reset" finden Sie sowohl im Editmodus wie im Laufmodus mit analogen Auswirkungen. Im Laufmodus bewirkt <F9> eine Initialisierung der simulierten CPU, wobei das Testprogramm im Arbeitsspeicher erhalten bleibt. Der Befehlszähler wird auf Null gesetzt, d. h., der Programmablauf beginnt, wenn er mit <F6> „RUN" erneut gestartet wird, wieder mit der Anfangsadresse 00. Die CPU-Graphik wird gleichfalls in den Anfangszustand zurückversetzt. Der mit <F5> gewählte Status bleibt jedoch erhalten.

Online-Hilfe

Auch aus dem Laufmodus heraus kann die kontextsensitive Online-Hilfe aufgerufen werden (siehe 3.4). Nach dem Anzeigen der Hilfe kehrt das Lernprogramm in den Laufmodus zurück.

Fehlermeldungen

Treten Fehler während des Programmlaufes auf (z. B. Bereichüberschreitungen), so werden sie in Form von Fehlermeldungen besonders hervorgehoben im Infofeld angezeigt. Der „RUN"-Modus endet zwar an dieser Stelle, kann aber erneut mit <F6> gestartet werden. Man muß sich jedoch darüber im klaren sein, daß Registerinhalte fehlerbedingt beliebige Werte annehmen können und das Testprogramm sein Ziel nicht erreicht!

3.4 Online-Hilfe

Das kontextsensitive Hilfemenü ist so aufgebaut, daß Sie es von jeder Programmebene aus durch <F1> „Hilfe" aufrufen können. Auf dem Bildschirm präsentiert sich dann eine Online-Hilfe-Maske (Abb. 7) zu dem zuletzt durchlaufenen Programmteil. Umfaßt der Hilfetext mehrere Bildschirmseiten, können Sie mit den Tasten <PGDn> und <PGUp> den gesamten Hilfetext zum aktuellen Stichwort durchblättern. Die Seitenanzeige im unteren rechten Hilferahmen gibt Auskunft über die aktuelle Position; „Seite 03/05" bedeutet: die Hilfe umfaßt 5 Bildschirmseiten, von denen jetzt Seite 3 angezeigt wird.

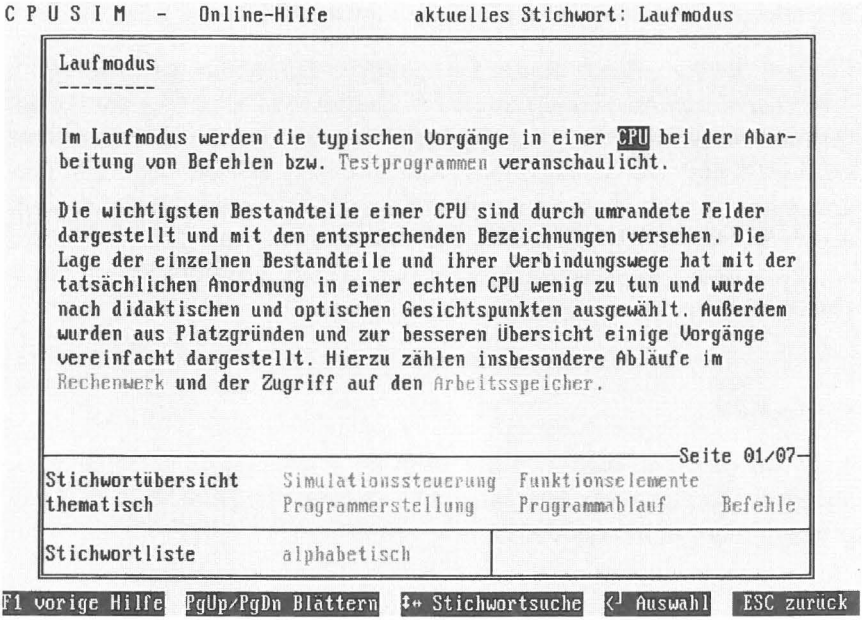


Abb. 7: Online-Hilfe-Maske

Wenn Sie weitere Erläuterungen zu dem angezeigten Themenkreis suchen oder Informationen zu anderen Bereichen benötigen, bietet Ihnen das Hilfemenü drei verschiedene Möglichkeiten, die gewünschten Erklärungen zu finden:

1. Querverweise

Enthält ein Hilfetext Begriffe, zu denen es weitere Hilfen gibt, so sind diese Begriffe (als Querverweise) optisch hervorgehoben. Der aktuelle Querverweis (zu Beginn immer der erste) ist hell unterlegt, die Hilfe dazu können Sie direkt durch <RETURN> anwählen. Alle Querverweise sind durch die Cursortasten erreichbar.

2. Thematische Übersichten

Unter dem Hilfetext stehen Ihnen sechs thematisch strukturierte Stichwortübersichten zur Verfügung über

- die Simulationssteuerung in CPUSIM,
- die Aufgaben der Funktionselemente im Laufmodus,

- das Erstellen von Testprogrammen,
- den Ablauf von Testprogrammen und
- die vorhandenen CPUSIM-Befehle.

3. Alphabetische Übersicht über alle Stichworte

Von hier aus gelangen Sie direkt zu den Detailinformationen.

Die sechs Übersichts-Stichworte sind auf jeder Hilfeseite verfügbar, so daß Sie jederzeit globale und detaillierte Hilfestellungen zu allen Gebieten bequem erreichen können.

Rückwärtsmanövrieren mit <F1> „Vorige Hilfe"

Wenn Sie die Funktionstaste <F1> drücken, erscheint auf dem Bildschirm die Hilfe zu dem Stichwort, das Sie unmittelbar vorher angewählt hatten. Ein weiterer Tastendruck auf <F1> zeigt Ihnen die davor gewählte Hilfe usw. Sie haben damit die Möglichkeit, nach dem Anwählen verschiedener Querverweise wieder an den Ausgangspunkt der Suche zurückzukehren. Ist die erste Hilfe erreicht, erscheint eine entsprechende Meldung im Feld unten rechts. Das Hilfemenü wird mit der <ESC>-Taste verlassen. Sie gelangen dann wieder dorthin, von wo Sie das Hilfemenü aufgerufen haben.

Übersicht Simulationssteuerung

Das Lernprogramm CPUSIM enthält folgende Strukturelemente:

- Eingangsmaske
- Editmodus (Erstellen/Editieren von Testprogrammen)
- Laufmodus (Austesten von erstellten Programmen)
- Hilfemenü
- Kommandozeile
- Fehlermeldungen
- Begleitheft

Übersicht Funktionselemente

Folgende Bestandteile eines Mikroprozessors und Elemente seiner unmittelbaren Umgebung werden in CPUSIM graphisch dargestellt:

- Adreßbus
- Adreßregister (SAR)
- Decodierer
- Instruktionsregister

- Akkumulator
- Arbeitsspeicher
- Befehlszähler
- Datenbus
- Datenregister
- Interruptregister
- Prozessor (CPU)
- Rechenwerk (ALU)
- Stackpointer (SP)
- Steuerwerk und Takt

Übersicht Programmerstellung

Folgende Gesichtspunkte zur Erstellung von Testprogrammen mit CPUSIM sind näher ausgeführt:

- Programmrahmen (äußere Bedingungen und Formalismen)
- Daten- Programmbereich (wie werden Daten und Programme im Speicher abgelegt)
- Realisierung von Schleifen
- Unterprogramme

Übersicht Programmablauf

Zur Abarbeitung von Befehlen in CPUSIM können Sie sich informieren über:

- Beschreibung zum Ablauf von Testprogrammen
- Operationszyklus (oder Befehlszyklus)
- Schreibzyklus – Lesezyklus
- Befehlsschlange (Queue)
- Adressierung
- Interrupt

Übersicht CPUSIM-Befehle

Folgende Anweisungen kann CPUSIM ausführen:

- Laden
- Speichern
- Addieren
- Subtrahieren
- Multiplizieren
- Dividieren
- Adressierung
- Springen
- Springen-unbedingt
- Springen-wenn-neg
- Springen-ins-UP
- Rücksprung
- Stop

3.5 Erstellen eigener Testprogramme

3.5.1 Allgemeine Hinweise

CPUSIM ermöglicht es Ihnen, Testprogramme selbst zu erstellen und ihren Ablauf zu beobachten. Die Anweisungen dieser Testprogramme sind maschinenorientiert und ähneln den Befehlen in Assembler. Ein CPU-Befehl besteht grundsätzlich aus einem Operations- und einem Adreßteil. Der Operationsteil enthält den binär verschlüsselten Code des auszuführenden Befehls, der Adreßteil die Adresse des Speicherplatzes im Arbeitsspeicher, auf dessen Inhalt sich der Befehl bezieht. Beispiel „Laden“:

eine Zahl (z. B. 105) wird aus dem Arbeitsspeicher in den Akkumulator geholt:

- Operationsteil: anlagenabhängiger Maschinencode für den Ladebefehl; in CPUSIM: 0001
- Adreßteil: es folgt nicht die zu ladende Zahl, sondern die Adresse des Speicherfeldes, in dem diese Zahl abgelegt ist (z.B. 10); in CPUSIM: 001010

Die Codierung für diesen Befehl sieht nun so aus:

Der Anwender hat dafür zu sorgen, daß die zu ladende Zahl unter der Adresse 10 abgelegt ist. Die Binärverschlüsselung des Ladebefehls besorgt der Editor. Ob CPUSIM den unter einer bestimmten Adresse vorgefundenen Inhalt nun als Daten oder als neuen Befehl zu interpretieren hat, hängt vom Zusammenhang ab. Wenn Daten eingegeben werden, wo ein Befehl erwartet wird, erfolgt eine Fehlermeldung. Der Editor von CPUSIM erleichtert Ihnen die Programmierung durch einfache Tastenkombinationen ganz erheblich. Folgende Hinweise sollten Sie jedoch beachten:

- Das Programm sollte grundsätzlich in der Adresse 00 (Anfang) des Arbeitsspeichers beginnen, da der Befehlszähler mit 0 vorbelegt ist; man könnte auch im Laufmodus mit <F7 Adresse> einen anderen Startpunkt wählen, was aber aus praktischen Gründen nicht ratsam ist.
- Der erste Programmbefehl ist in der Regel „Laden“, um Daten in den Akkumulator zu bringen.
- Der letzte Programmbefehl (im Hauptprogramm) muß der „Stop“-Befehl sein, im Unterprogramm „Rücksprung“.

- Die unteren Adressen des Arbeitsspeichers sind frei zu halten, da sie als Stack für die Unterprogramm-Rücksprungadressen benutzt werden.
- Der Anweisungsteil eines Programmes (in Assembler: Code-Segment), in dem alle Befehle abgelegt sind, sollte im Arbeitsspeicher einen zusammenhängenden Bereich einnehmen.
- Der Datenbereich enthält alle für das Programm erforderlichen Werte, die als Zahlen auf den Speicherplätzen eingetragen sind. Für die Programmausgaben ist hier ebenfalls Speicherplatz vorzusehen, damit die Programmergebnisse nicht verlorengehen.
- Der Wertebereich der Ein- und Ausgabedaten umfaßt nur ganze Zahlen in dem Intervall von —512 bis +511. Die Eingabe von Daten außerhalb dieses Bereiches ist nicht möglich. Kommt es bei Rechenoperationen zu Ergebnissen, die nicht in diesem Intervall liegen, wird das Ergebnis im Akkumulator nicht mehr vorhersagbar (und deshalb in jedem Falle falsch). Das Programm stürzt nicht ab, es wird nur eine entsprechende Fehlermeldung angezeigt.
- Zur besseren Übersicht (und zur Fehlervermeidung) sind Programmbereich und Datenbereich voneinander zu trennen.

3.5.2 Die CPUSIM-Befehle: Erläuterungen und mitgelieferte Testprogramme

Da bei diesem Lernprogramm für den Operatorteil 4 Bits zur Verfügung stehen, könnten maximal 16 verschiedene Befehle codiert werden; 11 Befehle sind zur Zeit in CPUSIM implementiert. Im folgenden werden diese Befehle erläutert. Anhand der mitgelieferten Testprogramme können Sie zusätzlich ihre Funktionsweise im Laufmodus dynamisch nachvollziehen.

3.5.2.1 Laden <SHIFT> + <F2>

Der Befehl „Laden" bewirkt, daß der Arbeitsspeicher unter der angegebenen Adresse gelesen und der darin stehende Speicherinhalt in den Akkumulator gebracht wird. Der Speicherinhalt selbst bleibt hiervon unberührt und besteht unverändert weiter. Ist dies nicht der erste Programmbefehl, so ist darauf zu achten, ob im Akkumulator ein Wert steht, z. B. ein Ergebnis aus einer vorherigen Rechenoperation, der noch nicht gesichert ist. Durch den Ladebefehl wird nämlich der alte Inhalt des Akkumulators überschrieben und steht nicht mehr zur Verfügung. In einem solchen Fall müßte der alte Akkuinhalt vor dem Ladebefehl mit dem Befehl „Speichern" gesichert werden.

Benötigt wird der Ladebefehl für alle hier durchführbaren Rechenoperationen, um die erste Zahl (das ist der 1. Operand) in das Rechenwerk (Akkumulator) zu transportieren. Die 2. Zahl, die an der Rechnung beteiligt ist (das ist der 2. Operand) wird dem Rechenwerk automatisch durch den Rechenbefehl zugeführt. Den Ladebefehl kann man auch dazu benutzen, den Inhalt eines Speicherplatzes in einen anderen Speicherplatz zu schreiben.

3.5.2.2 Speichern <SHIFT> + <F3>

Der Befehl „Speichern" ist das Gegenstück zum „Laden". Er bringt den jeweiligen Inhalt des Akkumulators an die Stelle im Arbeitsspeicher, die durch die Adresse im Speicherbefehl angegeben wurde. Er dient somit zur Sicherung von Rechenergebnissen, die nach ausgeführter Rechenoperation im Akkumulator stehen. Der Inhalt des Akkumulators bleibt auch nach dem Speicherbefehl unverändert dort stehen, bis er durch einen Lade- oder Rechenbefehl überschrieben wird.

3.5.2.3 Addieren <SHIFT> + <F4>

Der Befehl „Addieren" lädt den Speicherinhalt unter der im Adreßteil angegebenen Adresse in den Speicherplatz des Rechenwerks (Operandenregister), addiert ihn dann zum Inhalt des Akkumulators und speichert das Ergebnis wieder im Akkumulator ab. Beide an der Rechnung beteiligten Zahlen (Operanden) stehen nach Ausführung des Additionsbefehls dem Rechenwerk nicht mehr zur Verfügung. Der erste Operand wird im Akkumulator vom Ergebnis überschrieben. Der zweite Operand im Operandenregister des Rechenwerks ist ohnehin nicht direkt ansprechbar und wird beim nächsten Rechenbefehl überschrieben. Wenn das Ergebnis in einem unzulässigen Datenbereich liegt, also z. B. größer als +511 ist, wird eine Fehlermeldung ausgegeben, der Akkumulator enthält einen undefinierten Wert.

Testprogramm „ADD"

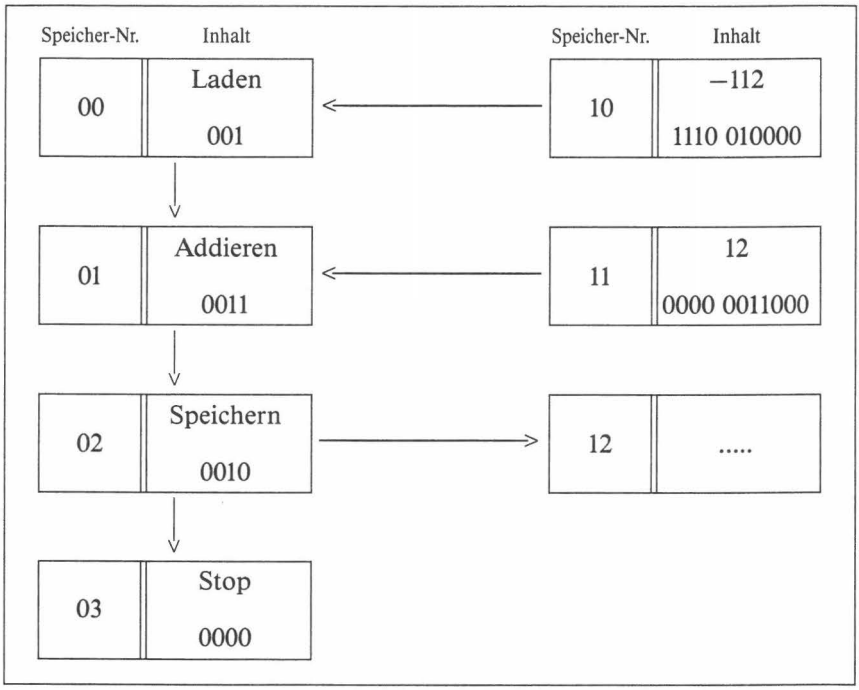
Zielsetzung:

Das Beispielprogramm „ADD" verdeutlicht die Wirkungsweise des Additionsbefehls im Zusammenspiel mit Laden und Speichern von Daten aus dem Arbeitsspeicher. Die Abläufe beim Lese- und Schreibzugriff sind klar erkennbar. Mit veränderten Eingabewerten läßt sich der Additionsbefehl (auch im Hinblick auf negative Zahlen) ganz ausloten.

Ablauf:

- Inhalt von Speicherplatz 10 in den Akku laden
- Inhalt von Speicherplatz 11 zum Akku addieren
- Inhalt des Akkus auf Speicherplatz 12 ablegen
- Stop

Struktur:



3.5.2.4 Subtrahieren <SHIFT> + <F5>

Der Befehl „Subtrahieren" lädt den Speicherinhalt unter der im Adreßteil angegebenen Adresse in den Speicherplatz des Rechenwerks (Operandenregister), subtrahiert ihn dann vom Inhalt des Akkumulators und speichert das Ergebnis wieder im Akkumulator ab. Beide an der Rechnung beteiligten Zahlen (Operanden) stehen nach Ausführung des Subtraktionsbefehls dem Rechenwerk nicht mehr zur Verfügung. Im Gegensatz zur Addition ist hier darauf zu achten, welchen Operanden man in den Akkumulator und welchen man durch den Subtraktionsbefehl in das Operandenregister des Rechenwerks lädt, weil die Subtraktion nicht kommutativ (umkehrbar) ist. Wenn das Ergebnis in einem unzulässigen Datenbereich liegt, also z. B. kleiner als −512 ist, wird eine Fehlermeldung ausgegeben, der Akkumulator enthält einen undefinierten Wert.

Testprogramm „SUB“

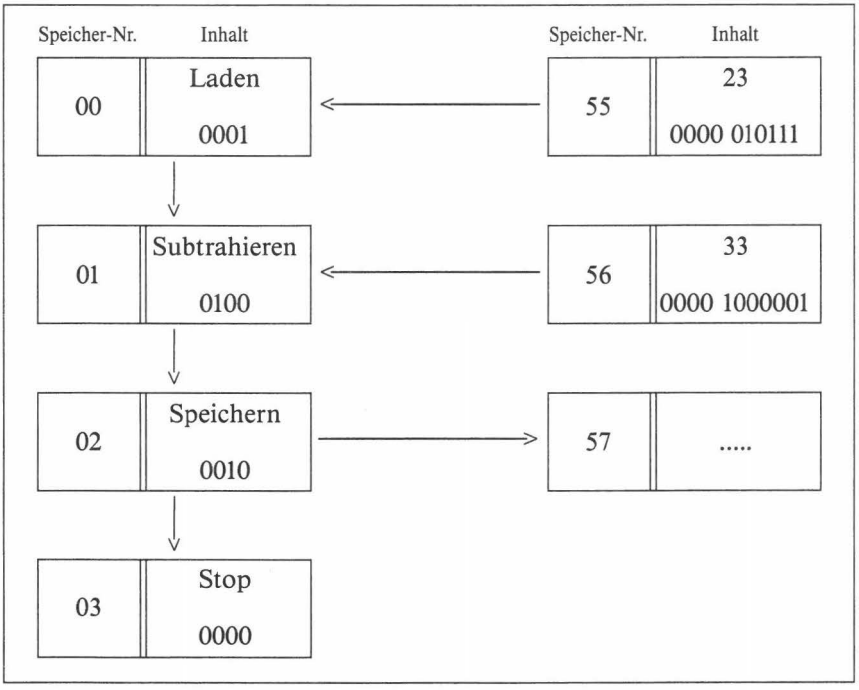
Zielsetzung:

Das Beispielprogramm „SUB“ zeigt den Subtraktionsbefehl in einer einfachen Anwendung im Zusammenspiel mit Laden und Speichern von Daten aus dem Arbeitsspeicher. Die Abläufe beim Lese- und Schreibzugriff sind klar erkennbar. Dabei ist es unerheblich, wo im Arbeitsspeicher die Daten gespeichert sind (hier: fast am Ende des Arbeitsspeichers). Das Verändern der Eingabewerte (z. B. das Subtrahieren von negativen Zahlen) gibt weiteren Aufschluß über die Funktionsweise des Subtraktionsbefehls.

Ablauf:

- Inhalt von Speicherplatz 55 in den Akku laden
- Inhalt von Speicherplatz 56 vom Akku subtrahieren
- Inhalt des Akkus auf Speicherplatz 57 ablegen
- Stop

Struktur:



3.5.2.5 Multiplizieren <SHIFT> + <F6>

Der Befehl „Multiplizieren" lädt den Speicherinhalt unter der im Adreßteil angegebenen Adresse in den Speicherplatz des Rechenwerks (Operandenregister), multipliziert ihn dann mit dem Inhalt des Akkumulators und legt das Ergebnis wieder im Akkumulator ab. Beide an der Rechnung beteiligten Zahlen (Operanden) stehen nach Ausführung des Multiplikationsbefehls dem Rechenwerk nicht mehr zur Verfügung. Wenn das Ergebnis in einem unzulässigen Datenbereich liegt, also kleiner als −512 oder größer als +511 ist, wird eine Fehlermeldung ausgegeben, der Akkumulator enthält einen undefinierten Wert.

Testprogramm „MUL"

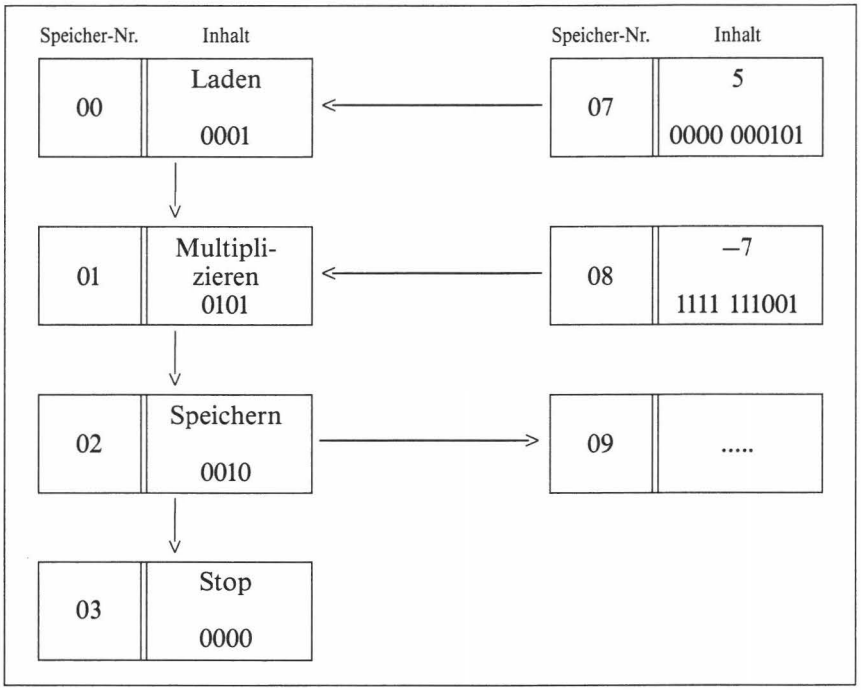
Zielsetzung:

Das Beispielprogramm „MUL" demonstriert den Multiplikationsbefehl an einem einfachen Zahlenbeispiel. Dabei finden die Regeln für den Umgang mit negativen Zahlen Anwendung. Lohnenswert sind auch Variationen der Eingabedaten, die die Grenzen des verfügbaren Zahlenbereichs berühren und der Umgang mit der Null. Quadratzahlen können durch Zugriff auf einen einzigen Speicherplatz realisiert werden.

Ablauf:

- Inhalt von Speicherplatz 07 in den Akku laden
- Inhalt von Speicherplatz 08 mit dem Akku multiplizieren
- Inhalt des Akkus auf Speicherplatz 09 ablegen
- Stop

Struktur:



3.5.2.6 Dividieren <SHIFT> + <F7>

Der Befehl „Dividieren" lädt den Speicherinhalt unter der im Adreßteil angegebenen Adresse in den Speicherplatz des Rechenwerks (Operandenregister), dividiert den Inhalt des Akkumulators durch diesen Speicherinhalt und legt den ganzzahligen Teil des Ergebnisses im Akkumulator ab. Divisionsreste können nicht gespeichert werden. Beide an der Rechnung beteiligten Zahlen (Operanden) stehen nach Ausführung des Divisionsbefehls dem Rechenwerk nicht mehr zur Verfügung. Im Gegensatz zur Multiplikation ist hier darauf zu achten, welchen Operanden man in den Akkumulator und welchen man durch den Divisionsbefehl in das Operandenregister des Rechenwerks lädt, weil die Division nicht kommutativ (umkehrbar) ist. Eine Division durch Null wird vor ihrer Ausführung erkannt und durch eine entsprechende Fehlermeldung angezeigt. Der Akkumulatorinhalt ändert sich nicht.

Testprogramm „DIV“

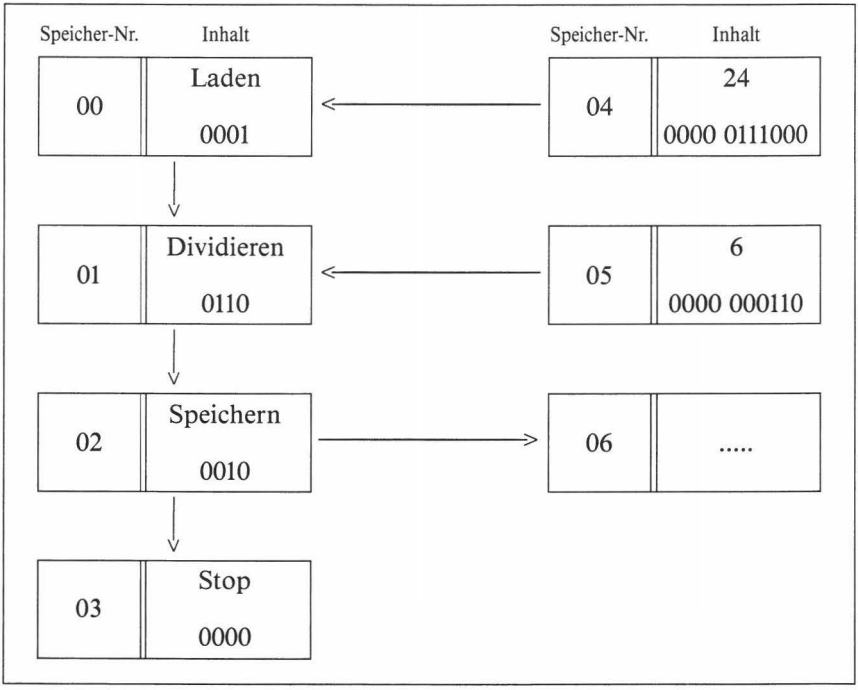
Zielsetzung:

Der Divisionsbefehl führt eine Ganzzahldivision von Akkumulator und Speicher durch, ein eventuell auftretender Divisionsrest wird nicht berücksichtigt. Interessante Aspekte ergeben sich mit veränderten Eingabewerten, besonders im Hinblick auf den Wert Null.

Ablauf:

- Inhalt von Speicherplatz 04 in den Akku laden
- Inhalt des Akkus durch Inhalt des Speicherplatzes 05 dividieren
- Inhalt des Akkus auf Speicherplatz 06 ablegen.
- Stop

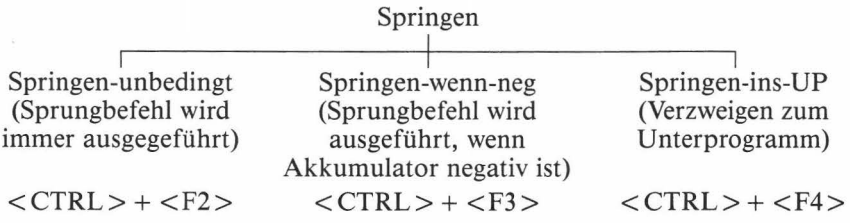
Struktur:



3.5.2.7 Sprungbefehle und Schleifen

Die Gruppe der Sprungbefehle nimmt einen wichtigen Stellenwert im Bereich der Programmanweisungen ein, da sie die Möglichkeit bieten, den linearen Programmverlauf zu unterbrechen und Kontrollstrukturen (Verzweigungen

und Schleifen) sowie Unterprogramme zu formulieren. In CPUSIM stehen drei verschiedene Sprungbefehle zur Verfügung:



Springen, unbedingt <CTRL> + <F2>

Der unbedingte Sprungbefehl „Springen-unbedingt" bewirkt, daß der Prozessor als nächsten Befehl den ausführt, den er unter der im Sprungbefehl angegebenen Adresse findet. Dazu wird der Befehlszähler mit dieser Adresse überschrieben. Nach dem Sprungbefehl erfolgt keine Rückkehr zur ursprünglichen Position, wie es beim „Springen-ins-UP" (Verzweigen zum Unterprogramm) geschieht.

Springen, wenn negativ <CTRL> + <F3>

Der bedingte Befehl „Springen-wenn-neg" wird nur unter der Voraussetzung ausgeführt, daß der aktuelle Wert, der im Akkumulator steht, negativ ist. Ist er positiv oder Null, wird der Sprungbefehl ignoriert und mit dem Befehl fortgefahren, der in der nächsten Speicherstelle steht.

Springen ins Unterprogramm <CTRL> + <F4>

Der Befehl „Springen-ins-UP" veranlaßt einen Sprung zu der im Befehl angegebenen Adresse. Zuvor wird jedoch die Arbeitsspeicheradresse, die im Befehlszähler steht, in der letzten, noch nicht belegten Speicherstelle des Arbeitsspeichers gesichert, um nach Abarbeitung des Unterprogrammes mit dem Rücksprungbefehl zu der Ausgangsposition im Hauptprogramm zurückzukehren. Im Unterprogramm selbst können beliebig viele Befehle codiert werden (auch weitere Unterprogrammaufrufe). Damit der Prozessor das Ende des Unterprogrammes erkennen kann, muß das Unterprogramm immer mit einem Rücksprungbefehl abgeschlossen werden.

Rücksprung <CTRL> + <F5>

Der Befehl „Rücksprung" beendet grundsätzlich ein Unterprogramm und muß in jedem Falle dort (und nur dort!) codiert sein, andernfalls erscheint eine Fehlermeldung. Der Rücksprungbefehl zeigt dem Prozessor an, daß ein Unterprogramm, das durch den Befehl „Springen-ins-UP" aufgerufen wurde, jetzt abgearbeitet ist und die Rückkehr ins Hauptprogramm erfolgen

soll. Die dazu erforderliche Rücksprungadresse wurde beim Aufruf des Unterprogrammes auf einem speziellen Speicherplatz im Arbeitsspeicher hinterlegt, auf den der Stackpointer zeigt. Über diesen Stackpointer kann sie jetzt wieder in den Befehlszähler zurückgebracht werden. Damit kann das aufrufende Programm hinter dem Unterprogrammaufruf fortgesetzt werden. Da die Rücksprungadresse aus dem Arbeitsspeicher geholt wird, ist bei diesem Befehl ein Adreßteil nicht erforderlich.

Programmierung von Schleifen

Wiederholungsschleifen können – wie in Assembler – durch Sprungbefehle realisiert werden. Die von den höheren Programmiersprachen her bekannten Schleifenarten (Zählschleifen mit „for", kopfgesteuerte „while"- und fußgesteuerte „repeat"-Schleifen usw.) sind dabei auf eine sehr einfache Struktur zu reduzieren, in der die bedingte Sprunganweisung „Springen-wenn-neg" eingesetzt wird.

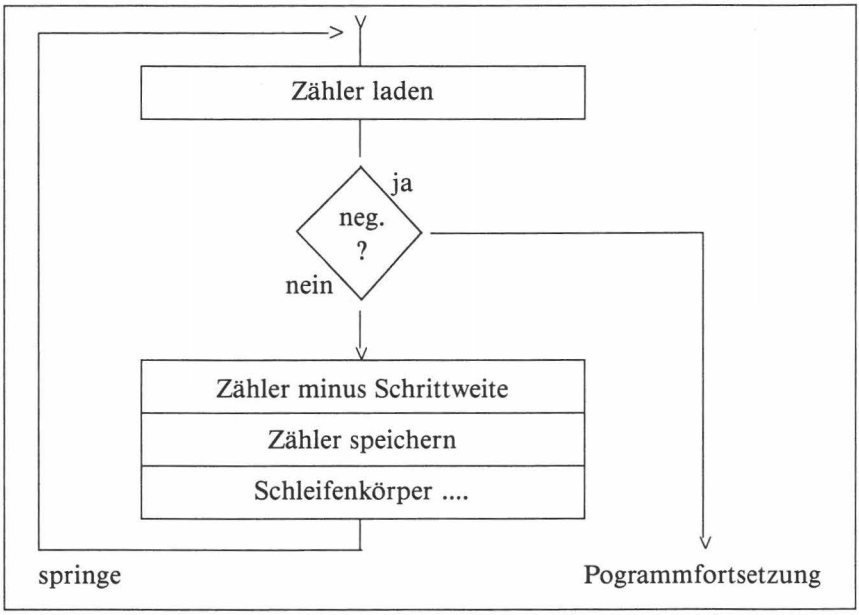
Für eine einfache Zählschleife („for"-Schleife) ist ein Speicherplatz des Arbeitsspeichers mit der Anzahl der Schleifendurchläufe minus 1 zu besetzen, ein anderer mit der Schrittweite (in der Regel mit 1). Am Schleifenanfang ist dieser Wert in den Akkumulator zu laden. Anschließend wird der zuvor genannte Sprungbefehl durchlaufen.

Die entscheidende Bedeutung kommt der bedingten Verzweigung zu:

- ist der Akkuinhalt negativ, verzweigt das Programm zu einer Adresse außerhalb der Schleife (Abbruchbedingung);
- ist der Akkuinhalt positiv oder Null, fährt das Programm mit den nächsten Befehlen (Subtraktion der Schrittweite, Abspeichern der Zählvariablen, Befehle des Schleifenkörpers) fort.

Ist der letzte Befehl des Schleifenkörpers durchgeführt, verzweigt das Programm durch einen unbedingten Sprung zum Schleifenanfang, wo erneut der Zähler in den Akkumulator geladen wird. Der letzte Schleifendurchgang erfolgt dann bereits mit einem negativen Zählerstand (in der Regel –1). Danach wird die Schleife dann verlassen. So wird deutlich erkennbar, wie umständlich ein Prozessor mit den „schönen" Schleifenkonstrukten der höheren Programmiersprachen umgehen muß und wieviel Aufwand hinter ihrer Ausführung steckt.

Das hier dargestellte Beispiel stellt eine von vielen Möglichkeiten dar, wie man eine Wiederholungschleife realisieren kann und soll als Anregung für andere Konstruktionen aufgefaßt werden. Schauen Sie sich doch einmal das mitgelieferte Testprogramm „SCHLEIFE" an!



Testprogramm „SCHLEIFE"

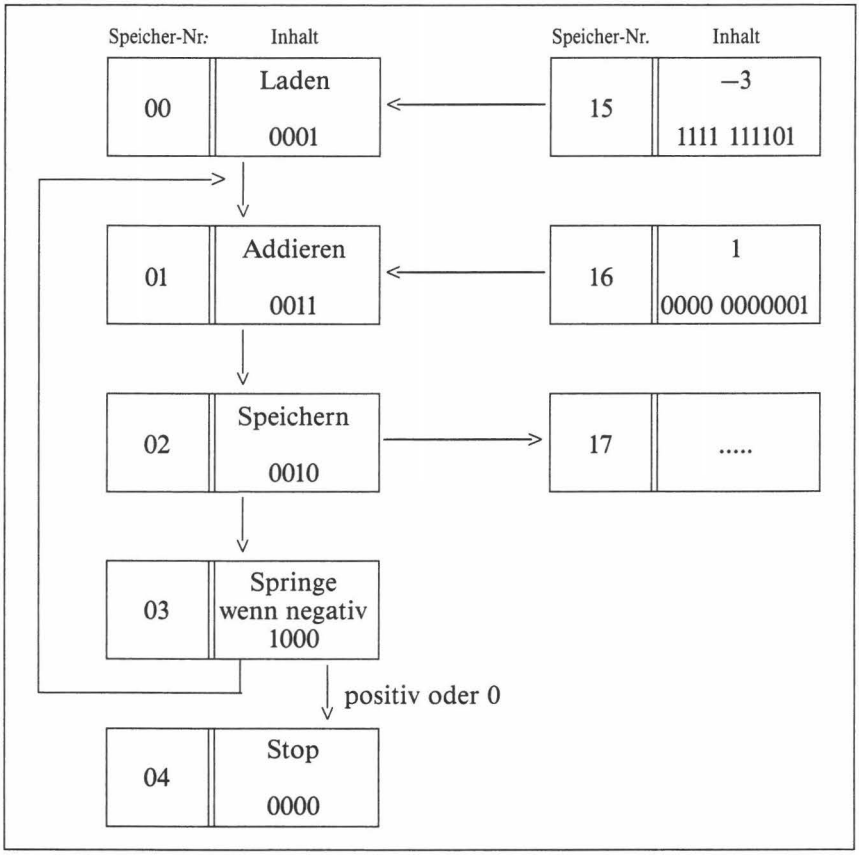
Zielsetzung:

Das Beispielprogramm „SCHLEIFE" demonstriert die Wirkungsweise des bedingten Sprungbefehls am Beispiel einer fußgesteuerten Schleife: nach dem Durchlaufen des Schleifenkörpers (Addieren und Abspeichern) wird der Akkuinhalt geprüft, ob er negativ ist oder nicht. Ist er negativ, springt das Programm an den Anfang der Schleife und durchläuft sie erneut. Ist er positiv oder Null, wird der nächstfolgende Befehl (hier: Stop) ausgeführt. Mit diesem Beispiel ist eine Zählschleife realisiert, die den Akkuinhalt von einem vorgegebenen negativen Wert (Speicherplatz 15, Startwert) um einen anderen Wert hochzählt (hier: 1; Inhalt Speicherplatz 16, Schrittweite), bis der Akkumulator positiv (bzw. 0) wird.

Ablauf:

- Inhalt von Speicherplatz 15 in den Akku laden
- Inhalt von Speicherplatz 16 zum Akku addieren
- Inhalt des Akkus auf Speicherplatz 17 ablegen
- Falls Akku negativ: Addieren von Speicherplatz 16, Speichern, Prüfen, sonst Stop

Struktur:



3.5.2.8 Unterprogrammtechnik

Wenn in einem Programm eine bestimmte Befehlsfolge an verschiedenen Stellen auftritt, so bietet es sich an, diese Befehlsfolge ein einziges Mal in einem Unterprogramm zu codieren. Mit Hilfe des Befehls „Springen-ins-UP" wird diese Befehlsfolge aufgerufen, wenn sie gebraucht wird. Es ist im Lernprogramm die Adresse des Speicherplatzes anzugeben, in dem der erste Befehl des Unterprogrammes steht. Der Unterprogrammaufruf bewirkt zunächst eine Abspeicherung der nächsten Befehlsadresse des aufrufenden Programmes in einem speziellen Bereich des Arbeitsspeichers (Stack), um die Rückkehr in dieses Programm zu gewährleisten. Das Programm verzweigt dann direkt zum ersten Befehl des Unterprogrammes. Von einem Unterpro-

gramm aus kann in derselben Form wieder ein Unterprogramm aufgerufen werden.

Damit der Prozessor erkennt, wann das Unterprogramm beendet ist, muß als letzter Befehl eines jeden Unterprogrammes der Befehls „Rücksprung" stehen. Eine Adresse ist hierbei nicht anzugeben, da der Prozessor die Rücksprungadresse, die er zuvor im Stack gespeichert hat, verwendet. Nach Ausführung des Rücksprungbefehls wird das aufrufende Programm fortgesetzt, der Speicherplatz auf dem Stack wird freigegeben.

Testprogramm „SUBPROG"

Zielsetzung:

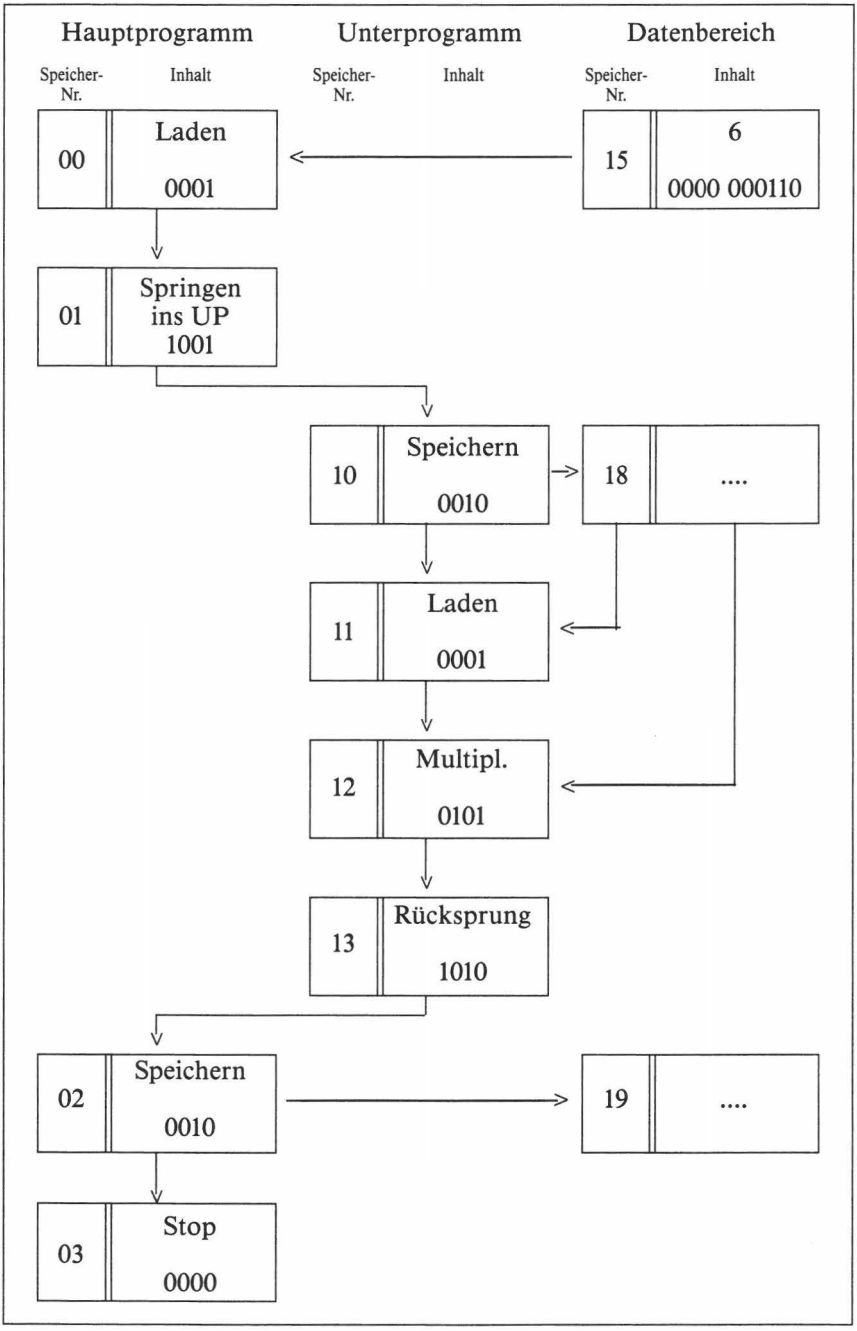
Das Beispielprogramm „SUBPROG" zeigt eine einfache Anwendung zur Unterprogrammtechnik. Hierbei kommen die Befehle „Springen-ins-UP" als Aufruf des Unterprogramms sowie „Rücksprung" als Endemarkierung des Unterprogramms zur Anwendung. Der Unterprogrammaufruf wird technisch durch eine Manipulation des Befehlszählers realisiert, der die Programmbefehle in der eingegebenen Reihenfolge aufruft und ablaufen läßt. Dazu ist der Befehlszählerstand vor dem Unterprogrammaufruf zu sichern, damit beim Rücksprung vom Unterprogramm das Hauptprogramm wieder an der richtigen Position fortgesetzt werden kann.

Das Programmbeispiel errechnet das Quadrat einer Zahl. Im Hauptprogramm wird eine Speicherzahl gelesen, die die Eingabe enthält. Das Unterprogramm errechnet das Quadrat. Um das Unterprogramm für beliebige Eingaben nutzen zu können (Modulcharakter), wird der Eingabewert über den Akkumulator übergeben und dann auf eine weitere Speicherstelle gerettet, um die Multiplikation mit sich selbst durchführen zu können. Eine interessante Erweiterung wäre es, das Programm für die Addition zweier Quadratzahlen umzuschreiben (z.B. Pythagoras: $a^2+b^2=c^2$).

Ablauf:

- Inhalt von Speicherplatz 15 in den Akku laden
- Aufruf des Unterprogramms ab Speicherstelle 10
- UP: Inhalt des Akkus auf Speicherplatz 18 ablegen
- UP: Inhalt von Speicherplatz 18 in den Akku laden
- UP: Inhalt von Speicherplatz 18 zum Akkuinhalt multiplizieren
- Rücksprung zum Hauptprogramm an Speicherstelle 02
- Inhalt des Akkus auf Speicherplatz 19 ablegen
- Stop

Struktur:


Hauptprogramm
Unterprogramm
Datenbereich
Speicher-Nr.
Inhalt
Speicher-Nr.
Inhalt
Speicher-Nr.
Inhalt
00
Laden
0001
15
6
0000 000110
01
Springen ins UP
1001
10
Speichern
0010
18
....
11
Laden
0001
12
Multipl.
0101
13
Rücksprung
1010
02
Speichern
0010
19
....
03
Stop
0000

3.5.2.9 Stop <CTRL> + <F10>

Der Befehl „Stop“ muß am Schluß eines jeden Programms stehen, damit der Prozessor das Bearbeitungsende erkennen kann und keine weiteren Versuche unternimmt, Befehle zu lesen. Der Arbeitsspeicher ist aus diesem Grunde zu Beginn mit Stopbefehlen vorbelegt. Der Stopbefehl darf niemals ein Unterprogramm beenden, da der Prozessor sonst den Programmablauf abbrechen würde und nicht mehr in das Hauptprogramm zurückkehren könnte. Der Stopbefehl benötigt keinen Adreßteil, da keine weiteren Daten für den Ablauf erforderlich sind.

4. Üben mit CPUSIM

Aufgabe 1

Schreiben Sie ein CPUSIM-Programm, daß die quadratische Funktion

$y = a*x^2 + b*x + c$

berechnet; verwenden Sie Parameter $a = 5$, $b = 4$ und $c = 18$. Testen Sie das Programm mit Berechnung des Funktionswertes y für $x = 3$.

Aufgabe 2

Schreiben Sie ein CPUSIM-Programm, das die Gültigkeit der Formel

$(a+b)*(a-b) = a^2-b^2$

demonstriert. Speichern Sie das Ergebnis der linken Seite (der rechten Seite) im vorletzten (letzten) Speicherplatz. Testen Sie das Programm für $a = 8$ und $b = 5$. In den beiden letzten von Ihnen verwendeten Speicherstellen müßte dann jeweils 39 stehen.

Aufgabe 3

Schreiben Sie ein Programm für die Formel $y = a^b$ so allgemein, daß Sie es für verschiedene Werte von a und b verwenden können.

Hinweis: a wird mit b potenziert, indem es (b–1)-fach mit sich selbst multipliziert wird. Den Multiplikationsvorgang können Sie mit einem Unterprogramm ähnlich „SUBPROG", die Anzahl der Multiplikationsvorgänge mit einer Konstruktion wie in „SCHLEIFE" realisieren.

5. Fehlermeldungen und Warnungen

Eitmodus

*** Dateiname angeben ***

Inhalt: beim Eingeben von Pfad und Dateinamen (<F3> „Sichern" von Testprogrammen) wurde <RETURN> gedrückt vor dem Eintrag eines Dateinamens;

Reaktion: Meldung verschwindet nach kurzer Zeit, danach Dateinamen eintragen oder Option mit <ESC> verlassen.

Datei nicht im angegebenen Directory!

Inhalt: beim Suchen von Testprogrammdateien trat ein Lesefehler auf

Reaktion: Option noch einmal mit richtigem Namen anwählen.

Datei konnte nicht gelesen werden!

Inhalt: beim Lesen einer Testprogrammdatei traten Fehler auf, wahrscheinlich hat die Datei kein Testprogrammformat, vielleicht liegt ein Diskettenfehler vor;

Reaktion: Option mit einer anderen Datei wiederholen.

Datei vorhanden! Überschreiben (J/N)?

Inhalt: beim Sichern einer Testprogrammdatei wurde eine Datei unter dem eingegebenen Namen gefunden;

Reaktion: Überschreiben (J), wenn das alte Programm nicht mehr gebraucht wird, oder (N): Option mit neuem Namen starten.

Datei konnte nicht geöffnet werden!

Inhalt: beim Sichern einer Testprogrammdatei traten Fehler auf, z.B. Diskette voll, schreibgeschützt, defekt;

Reaktion: Option mit einer anderen Diskette wiederholen.

Kein Testprogramm in diesem Directory!

Inhalt: in dem Verzeichnis „CPUSIM" ist kein Testprogramm mit der Extension „CPU" abgespeichert;

Reaktion: Testprogramme im angegebenen Directory abspeichern oder dorthin kopieren und Option wiederholen.

Achtung!! Keine CPUSIM-Testdatei!

Inhalt: die gewählte Datei mit der Endung „.CPU" ist kein von CPUSIM erstelltes Testprogramm;

Reaktion: andere Datei wählen, falsches Testprogramm später umbenennen.

Lesefehler in der Testdatei!

Inhalt: das gewählte Testprogramm ist fehlerhaft und entspricht nicht dem CPUSIM-Format;

Reaktion: Testprogramm löschen und ggf. neu erstellen.

** RESET löscht den Arbeitsspeicher! **

Inhalt: bei <F9> „RESET" wird nachgefragt, ob wirklich das im Speicher befindliche Programm gelöscht werden kann;

Reaktion: mit <CTRL> + <J> wird gelöscht, Inhalt unwichtig; jede andere Taste storniert die Option.

ACHTUNG STACKBEREICH!

Inhalt: beim Editieren wurde versucht, einen Speicherplatz > 60 anzuwählen, der für den Stack reserviert ist;

Reaktion: Editieren eines Speicherplatzes mit kleinerer Nummer.

Laufmodus

** PROGRAMMFEHLER Befehl unbekannt **

Inhalt: der Decodierer hat im Befehlscode einen Wert gefunden, für den es keinen Befehl gibt, es wurde vermutlich ein Datenfeld gelesen;

Reaktion: Programm im Editmodus an dieser Stelle ändern.

** PROGRAMMFEHLER Adresse falsch **

Inhalt: der Decodierer hat im Befehlscode eine Adresse gefunden, die nicht im Arbeitsspeicher existiert, es wurde vermutlich ein Datenfeld gelesen;

Reaktion: Programm im Editmodus an dieser Stelle ändern.

** ERROR Zahlbereich überschritten **

Inhalt: im Akkumulator hat sich bei der letzten Rechenoperation ein Wert ergeben, der kleiner als −512 oder größer als +511 ist, Akkumulator nimmt nun einen zufälligen Wert an;

Reaktion: Datenbereich im Editmodus an den entsprechenden Stellen anpassen.

** ERROR Division durch Null **

Inhalt: der Akkumulator soll durch Null geteilt werden, die im Divisionsoperanden steht, Akkuinhalt bleibt erhalten;

Reaktion: Datenbereich oder Programm ändern.

** ERROR Rücksprung ohne UP-Aufruf **

Inhalt: es wurde ein Rücksprungbefehl programmiert, ohne vorher ein Unterprogramm aufzurufen;

Reaktion: Programm im Editmodus an dieser Stelle ändern.

** —> Arbeitsspeicherende erreicht **

Inhalt: der Befehlszähler überschreitet beim Hochzählen die Adresse Nr. 60 und wird auf 0 gesetzt;

Reaktion: überprüfen, ob Programmierung an dieser Stelle nicht verbessert werden kann

** CPU-Befehl noch nicht beendet! **

Inhalt: die Option <F7> „Adresse“ zum Vorgeben einer neuen Adresse arbeitet nur, wenn der letzte Befehl vollständig ausgeführt ist;

Reaktion: mit dem Status „Einzelschritt“ oder „Programmbefehl“ zum Befehlsende vorarbeiten und Option wiederholen.

HILFETEXT NICHT IM AKTUELLEN VERZEICHNIS!!

Inhalt: die Datei „SIMHILFE.TXT“ befindet sich nicht im aktuellen Directory (Unterverzeichnis „CPUSIM“);

Reaktion: Simulation beenden und CPUSIM neu installieren.

LESEFEHLER: KEINE HILFE GEFUNDEN!!

Inhalt: die Hilfetextdatei „SIMHILFE.TXT“ ist verändert oder beschädigt, es liegt ein Lesefehler vor;

Reaktion: Hilfetextdatei überprüfen, ansonsten CPUSIM neu installieren, ggf. Laufwerk überprüfen.

Folgende Meldungen können im Hilfetextfeld unten rechts auftreten:

- Ende der Hilfe erreicht (bei <PGDn>)
- Anfang der Hilfe erreicht (bei <PGUp>)
- keine Hilfe zum Stichwort (gewählter Querverweis kann durch einen Lesefehler nicht ermittelt werden —> s. oben)
- keine vorige Hilfe vorhanden (mit <F1> an den Anfang gekommen)

Abkürzungen

Folgende Abkürzungen werden in CPUSIM benutzt:

Akku = AX	Akkumulator
ALU	Arithmetic Logic Unit (Rechenwerk)
CPU	Central Processing Unit (Prozessor)
IAR = IP	Instruction Pointer (Befehlszähler)
IR	Instruktionsregister
SAR	Speicheradreßregister (Adreßregister)
SIR	Speicherinformationsregister (Datenregister)
SP	Stackpointer
UA	Unterbrechungsanforderungsregister (Interrupt)
UP	Unterprogramm

Stichwortverzeichnis