Schnell-Einstieg in den Mikrocodesimulator MikroSim2000

Die folgenden Seiten sollen dem Neueinsteiger einen schnellen Überblick über die Tragweite der Mikrocodesimulation mit Hilfe des Mikrocodesimulators MikroSim2000 geben. An einem ausgewählten Beispiel, welches das Register A (R1) des Mikrocodesimulators schrittweise um Eins erhöhen soll wird im Folgenden in die Funktionsweise des Simulators eingegangen. Zunächst wird im „Erkundungsmodus“ der 49-Bit lange Mikrocode untersucht, der bereits für sich alleine die Inkrementierung bewerkstelligen kann. Anschließend wird in den Mikrocodesimulationsmodus übergewechselt, der auf 3 Mikrocodesimulator-Dateien aufsetzt, die dem Programm als Beispieldateien beigefügt sind. Die Inkrementierung des Registers A kann entweder in einer Folge von Mikrocoden geschehen (Beispiel1), in einer Mikrocode-Programmschleife (Beispiel2) oder in einer Sequenz Mikrocode-programmierten Maschinensprachebefehle.

Installation des Mikrocodesimulators MikroSim2000

Der Mikrocodesimulator MikroSim2000 wird als Programm-Paket mit einem beigelegten Setup-Programm installiert. Beim Setup-Start kann als Installationssprache zwischen Deutsch und Englisch gewählt werden. Die Setup-Sprache ist gleichzeitig die Startsprache für den zweisprachig ausgelegten Mikrocodesimulator. Mit der Kommandozeilen-Anweisung „MikroSim.exe **0**“ bzw. „MikroSim.exe **1**“ kann zwischen der englischen (Standard) und der deutschen Sprache gewählt werden. Ohne Komandozeilen-Parameter startet der Mikrocodesimulator standardmäßig in der englischen Version. Im Mikrocodeprogramm selber kann man aber immer noch bei jeder Simulation zwischen der deutschen und englischen Version wählen.

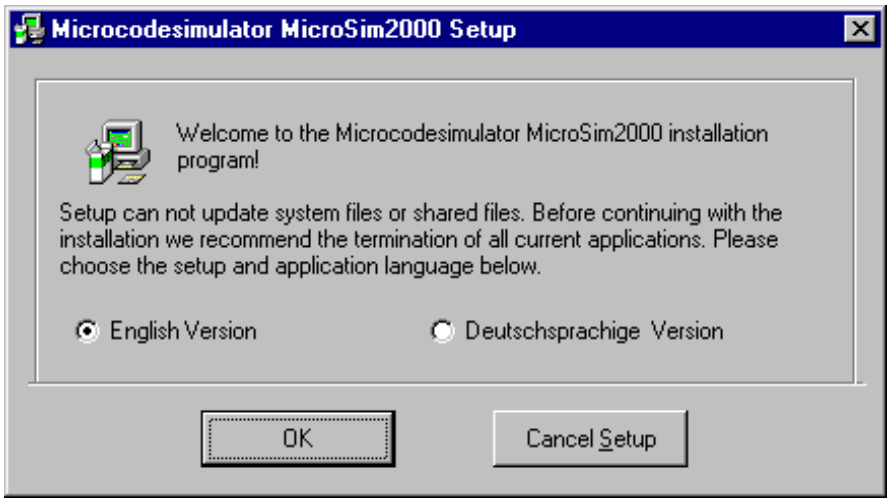

Abb. 1: Setup-Programm mit Sprachauswahl.

Das Setup-Programm fragt bei der Installation nach dem Verzeichnis, in dem das Programm eingerichtet werden soll. Nach dem Bestätigen des Zielverzeichnisses oder der Angabe eines neu zu erstellenden Zielverzeichnises läuft die Installation von alleine ab. Mit dem Setup wird eine neue

Programmgruppe in die Startleiste eingerichtet, in der der Mikrocodesimulator und das entsprechend sprachige Hilfe-Datei als Programm-Symbol abgelegt werden.

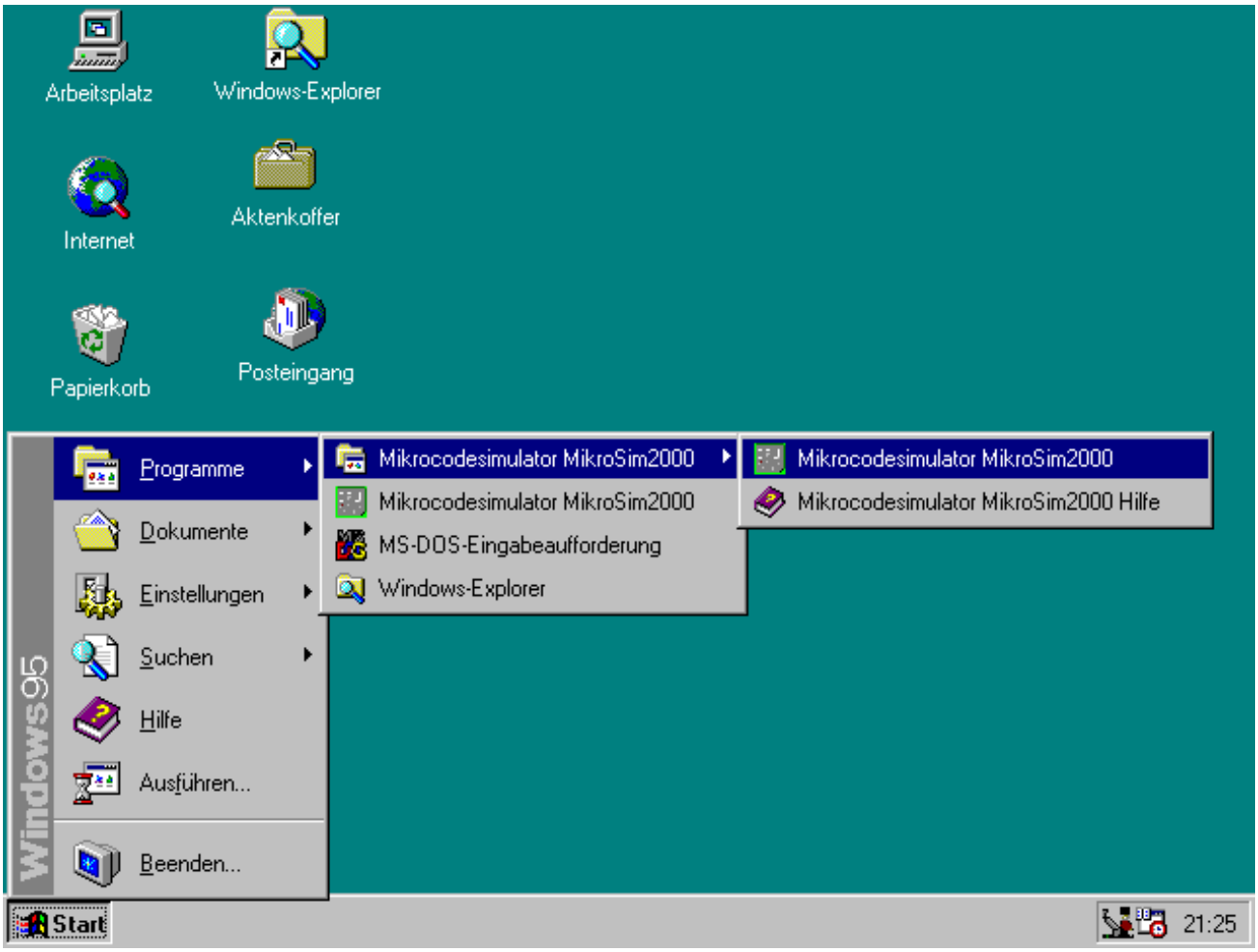

Abb. 2: Bei der Programm-Installation wird der Mikrocodesimulator in einer Programm-Gruppe und als Applikation in das Startmenü unter MS-Windows95/98/NT aufgenommen.

Der Mikrocodesimulator wird als 32-Bit-Applikation in die Registry von Windows eingerichtet und kann auf Wunsch deinstalliert werden. Sowohl Programm-Installation als auch -Deinstallation sollten über die Systemsteuerung erfolgen.

Liegt keine Installationdiskettensatz zu diesem Programm vor, kann der Mikrocodesimulator auch ohne große Mühe zum Laufen gebracht werden, wenn die EXE-Datei „**MIKROSIM.EXE**“ die Dateien „**VB40032.DLL**“, „**CTL3D32.DLL**“ und „**COMDLG32.OCX**“ in dem Systemverzeichnis von Windows vorfindet. Diese Dateien sind häufig von anderen Programm-Installationen dort bereits abgelegt worden.

Starten des Simulators im Erkundungsmodus

Der Mikrocodesimulator MikroSim2000 wird durch Klicken auf ein grün-graue CPU-Programm-Symbol gestartet. Der Mikrocodesimulator öffnet ein anfangs grünes, leeres Fenster. Der Benutzer befindet sich in dem sogenannten *„Erkundungsmodus“*. In diesem Modus kann die CPU schrittweise aufgebaut werden, indem unter zuhilfenahme des Menüpunkts *„Optionen/Simulationsstufe“* nacheinander CPU-Bausteine wie Register, Recheneinheit (ALU), exterens RAM, Mikrocode-ROM und der Mikrocode-Adressrechner dem Simulator hinzugefügt werden. Die Register tragen vor der Initialisierung mit dem Reset-Knopf der CPU eine Bezeichnung, die auf die Funktion der Bausteine und deren Register hindeuten sollen. Mit dem Hinzufügen des letzten Bausteins, dem "Mikrocode-Adressrechner" ist die CPU komplett (Abb. 4).

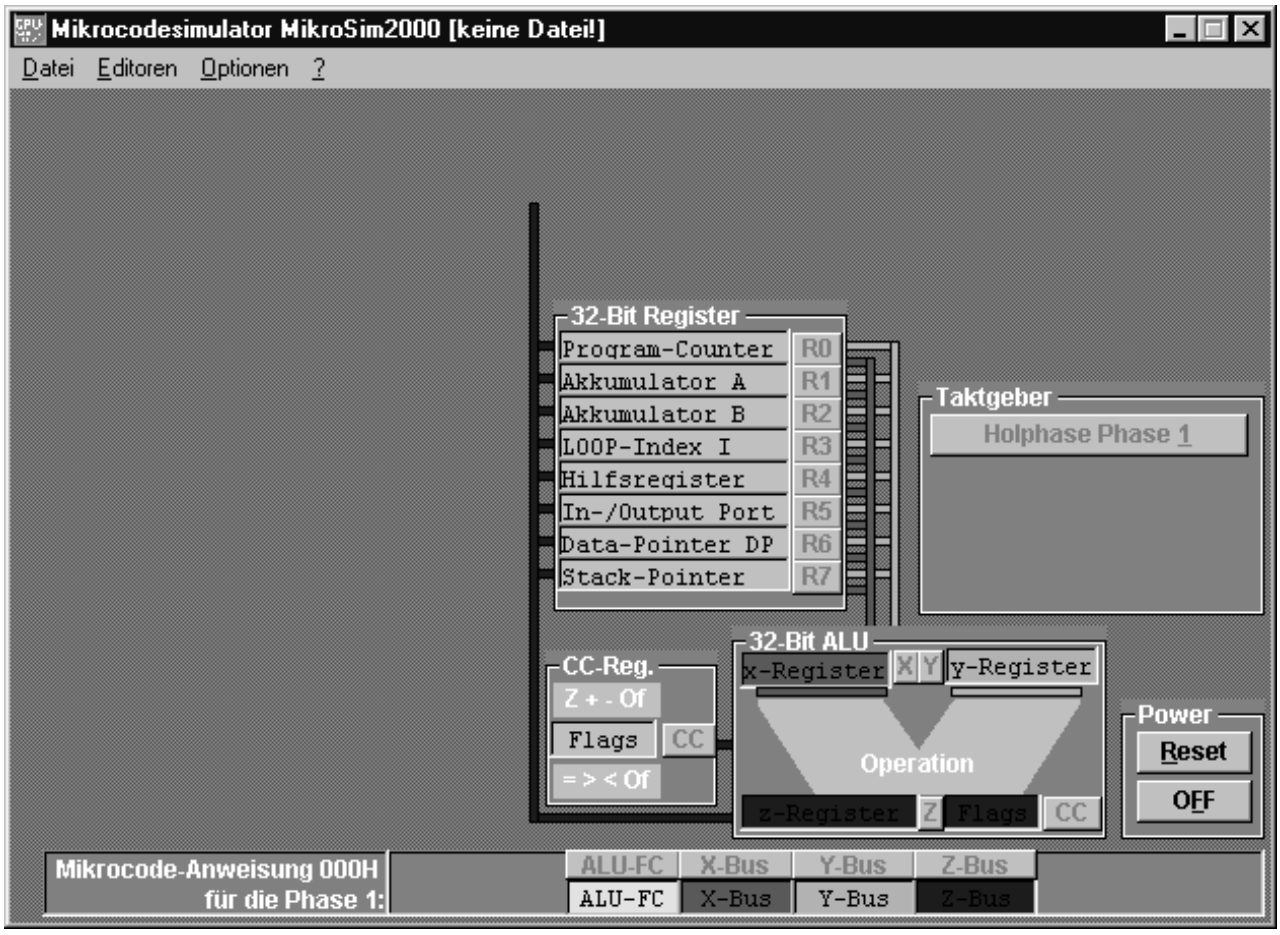

Abb. 3: Schrittweiser Aufbau des Mikrocodesimulators im Erkundungsmodus.

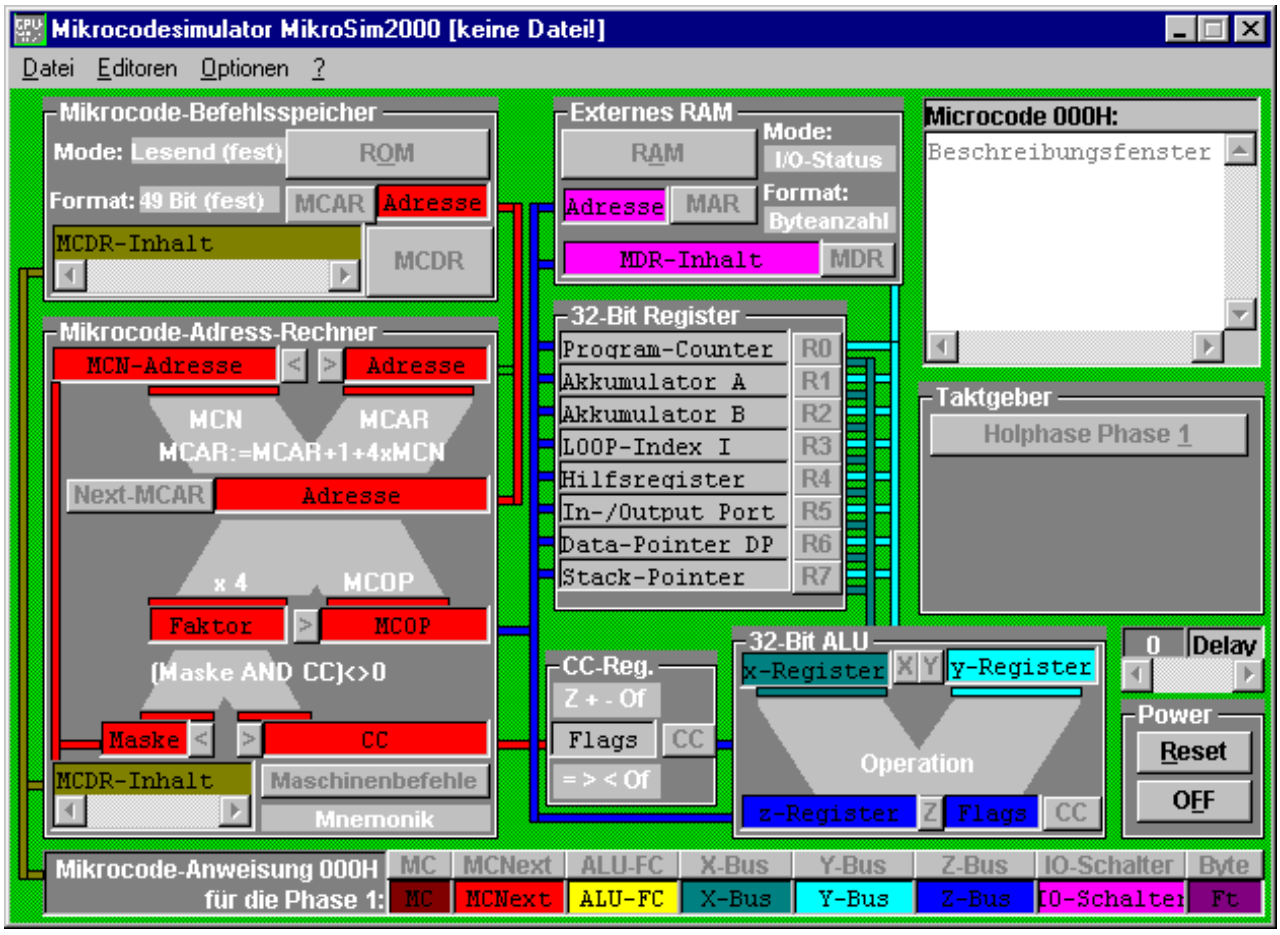

Abb. 4: Vollständig aufgebauter Mikrocodesimulator im Erkundungsmodus.

Dort, wo zusätzliche Informationen den Verlauf von Datenleitungen und die Funktionsweise von Register (farbige Textfelder auf grauem Hintergrund) näher erklären, wechselt der Standard-Mauszeiger in einen Mauszeiger mit Fragezeichen . Mit einem Mausklick auf einen solch aktivierbaren Fensterbereich öffnet sich ein Fenster mit weiteren nützlichen Informationen (Abb. 5).

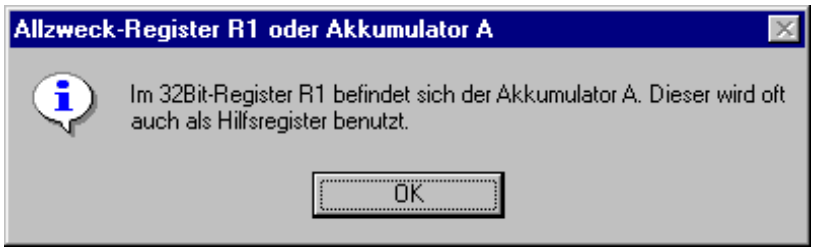

Abb. 5: Informationen zum Register R1.

Nachdem der Benutzer nun mit den Bezeichnungen der CPU vertraut ist, kann mit der Erkundung der Wirkungsweise der einzelnen Bits des am unteren Bildschirmrandes dargestellten Mikrocodes begonnen werden. Zwecks Übersichtlichkeit klicke man alle Bausteine der CPU bis auf den Register-Baustein und der ALU mit dem Menüpunkt „*Optionen/Simulationsstufe/... und RAM*“ in den Hintergrund (Abb. 6).

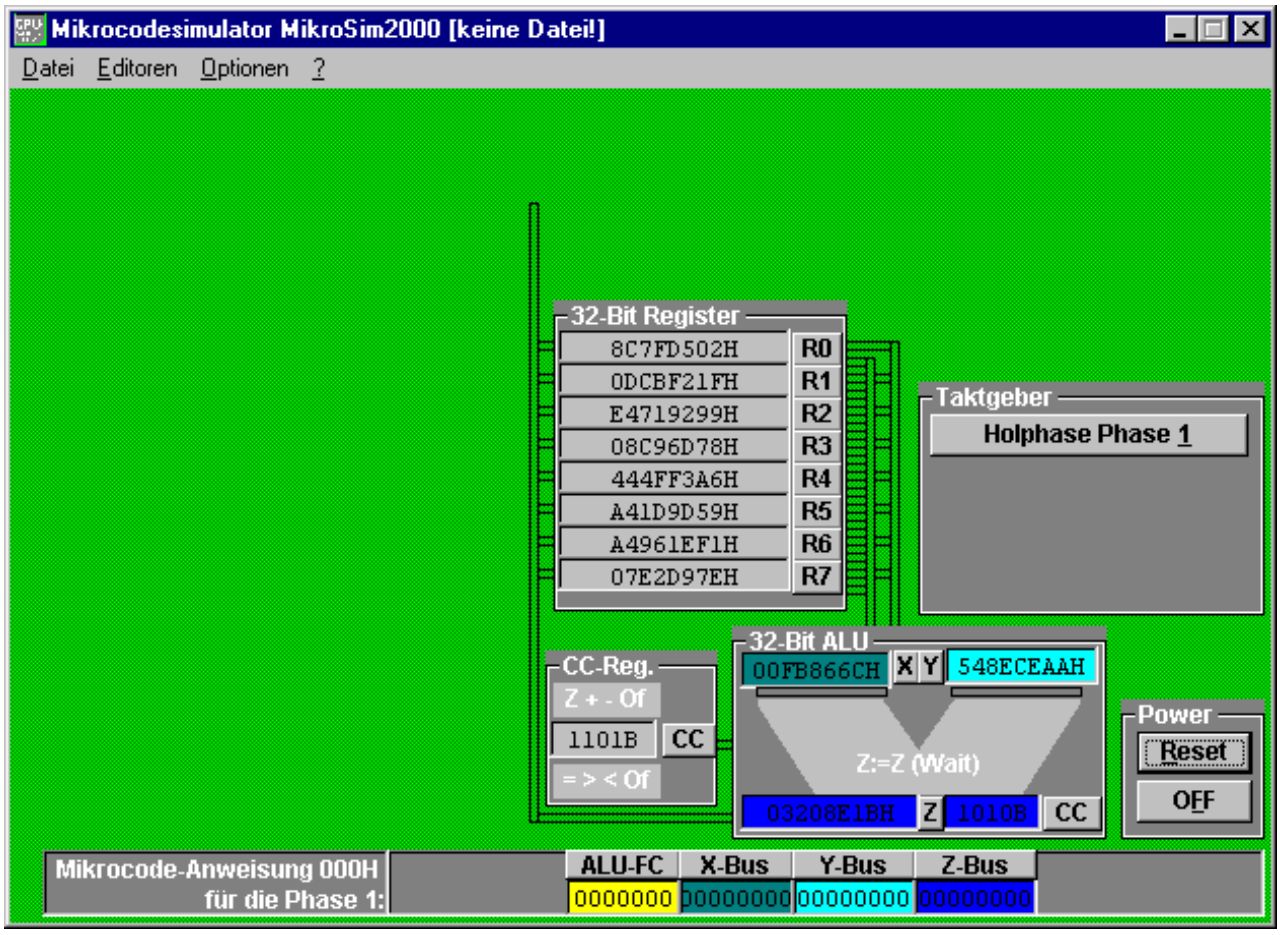

Abb. 6: Initialisierter Mikrocodesimulator im Erkundungsmodus durch Drücken auf den “Reset-Button”.

Um die einzelnen Bits der vier ausgewählten Bitgruppen des 49-Bit langen Mikrocodes auf ihre Wirkung hin zu untersuchen, wir der Simulator zunächst mit dem Reset-Knopf initialisiert. Wie in Abb. 6 abgebildet tragen nun alle Register anstelle der erklärende Beschriftung einen Registerinhalt.

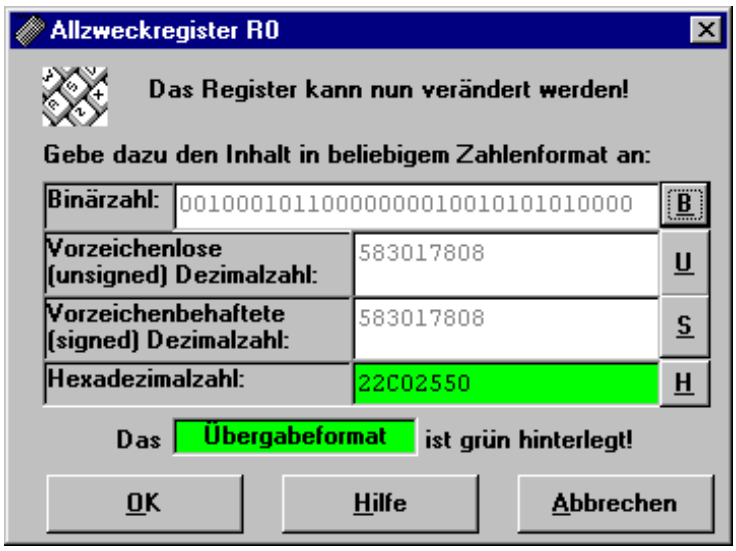

Abb. 7: Im Systemzahlen-Editor können die Registerwerte des Mikrcodesimulators verändert werden.

Registerinhalte können mit Hilfe des Programm eigenen Systemzahl-Editor verändert auf vier Arten, der binären, dezimal-vorzeichenlosen, dezimal-vorzeichenbehafteten und hexadezimalen Schreibweise dargestellt werden (Abb. 7). Jede Zahleneingabe in diesem Editor wird sofort in die übrigen Formate umgerechnet. Mit dem Verlassen des Editors mit „OK“ wird das Zahlenformat in das zu beschreibende Register übergeben, welches im Systemzahl-Editor grün hinterlegt ist. Je nach Verwendung des Mikrocodesimulator-Register als Schalter- bzw. Flag-Register oder als Zahlenregister kann dies entweder das Binär- oder das Hexadezimalformat sein.

Der Mikrocodesimulator kann im Erkundungsmodus auf diese Weise durch Austesten von der Schalterstellungen und Registerinhalte erkundet werden. Schreibt man z.B. in das X-Bus- und Z-Bus-Register des aktuellen Mirkocode Wert “01000000“ hinein (Abb. 8), so kann man durch Drücken des Taktgeber-Schalters erkennen, das sich die Busleitungen in der sogenannten Holphase vom Register R0 zum X-Bus öffnen und ein X-Wert in das Register der Arithmetisch-Logischen Recheneinheit ALU geholt werden. In der Bringphase Phase3 wird über den Z-Bus ein Rechenergebnis der Arithmetisch-Logischen Recheneinheit ALU in das Register R0 zurückgebracht. Setzt man zusätzlich das 7-Bit-Register des ALU-Funktionscodes im Mikrocodesimulator auf den Wert 9, so kann man nachverfolgen, wie durch Drücken des Taktgebers das Register R0 schrittweise nach jedem Hol-Rechen-Bring-Zyklus um Eins erhöht wird, da nun in der Rechenphase Phase 2 das X-Register der ALU um Eins inkrementiert wird.

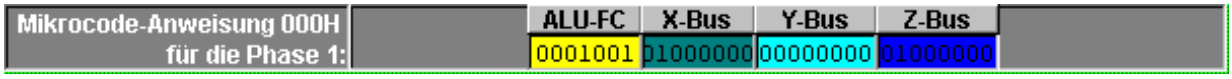

Abb. 8: Diese Mikrocode-Befehlszeile im Erkundungsmodus inkrementeiert das Register R1.

Auf ähnliche Weise können alle Bits der 49-Bit langen Mikrocodeanweisung untersucht werden. Über den Befehlsumfang und die Abarbeitungsreihenfolge der 49-Bit langen Mikrocode geben die Hilfe-Dateien zum Simulator näher Auskunft.

Um den Umgang mit dem Mikrocodesimulator zu üben, werden vorbereitete und dem Simulator beigelegte Beispiel-Dateien besprochen. Eine Datei des Mikrocodesimulators öffnet man mit Hilfe des Menüpunktes „*Datei/Öffnen* ...“.

Beispielprogramm Beispiel #1:

Im Erkundungsmodus wird genau ein Mikrocodebefehl bearbeit, nämlich der, der am unteren Bildschirmrand nach Bitgruppen aufgegliederte. Der nächste Schritt wäre, zu erkunden, wie eine Folge von 49-Bitcodes diese Inkrementierung bewerkstelligen könnten. Mit dem Laden oder neu Anlegen einer Mikrocodeprogramm-Umgebung wird der Erkundungsmodus verlassen. Mit dem Menüpunkt „*Datei/Öffnen...*“ werden neben der Simulationsdatei "*Beispiel1.ROM*" alle anderen zu dem Projekt dazugehörigen Dateien geladen. Mit dem Laden wird gleichzeitig die komplette CPU aufgebaut. Der Erkundgsmodus wird durch den Simulationsmodus abgelöst. Es kann aber immer noch interaktiv in den Programmablauf eingegriffen werden.

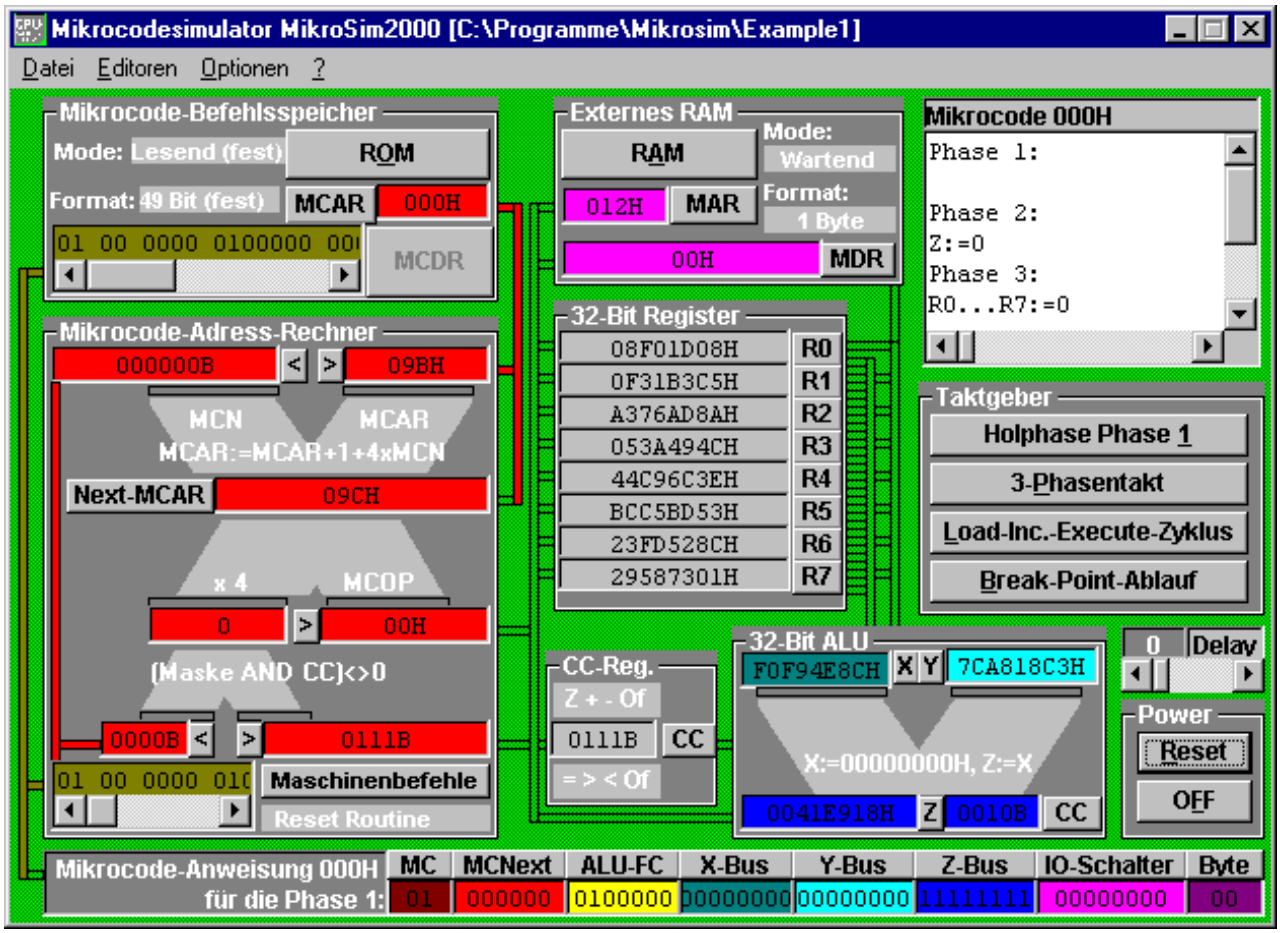

Abb. 9: Initialisierter Mikrocodesimulator mit geladener Beispieldatei "EXAMPLE1".

Man lade das „*Beispiel1.ROM*“ in den Simulator. Durch Drücken auf den Button mit der Beschriftung "Reset" (Abb. 9) wird die CPU für den Simulationsmodus mit der Beispieldatei "*Beispiel1.ROM*" initialisiert. Sämtliche Buttons der Oberfläche sind nun aktivierbar. So auch der ROM-Button oben links im Mikrocodebefehlsspeicher des Mikrocodesimulators. Mit ihm kann man sich den Inhalt der Mikrocode-ROM-Beispieldatei "*Beispiel1.ROM*" anschauen. Nur einige der ersten acht Zeilen sind mit 49-Bit-Mikrocoden belegt (Abb. 10).

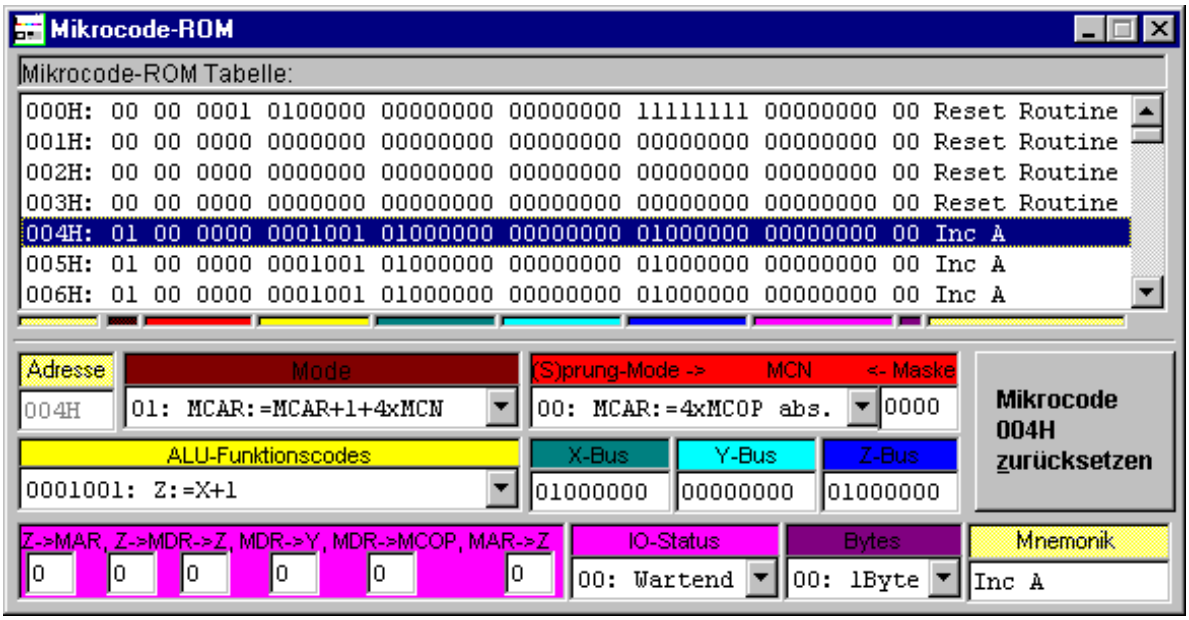

Abb. 10: Mikrocodebefehlsspeicher-Inhalt. Die Mikrocodes können mit dem Editor bitgruppenweise bearbeitet werden.

Der erste abzuarbeitende Mikrocode ist der mit der Adresse 000H. Nach einem Reset startet der Simulator stets bei dieser Anfangsadresse. Hier nutzt man gleich die Gelegenheit, den Simualator zu initialisieren und die Registerwerte auf Null zu setzen. Dies geschieht, indem in der ALU in der Rechenphase die Konstate 0 erzeugt wird und in der Bringphase in alle Register geschrieben wird. Der nächste auszuführende Mikrocodebefehl steht an der Adresse 004H. Dorthin springt der Simulator, da in der Rechenphase die nächste Mikrocodeadresse die Adresse 4 ist. Ab dort wird nun schrittweise ein Dreiphasentakt abgearbeitet, der nach jeder Ausführung eines Mikrocodebefehls das Register A um Eins erhöht hat und den Mikrcodeprogrammzeiger auf den nächsten Befehl gesetzt hat. Bei der Mikrocode-Adresse 007H generiert der Mikrcode-Adressrechner als neues Sprungziel die Adresse 004H, womit sich der Simulator in einer Endlosschleife befindet. Das geladene Mikrocode-Programm kann entweder schrittweise in den drei Einzeltakten untersucht sucht werden oder Dreiphasentaktweise abgearbeitet werden.

Beispielprogramm Beispiel #2:

Im Beispielprogramm "EXAMPLE2" wird der Endlosaddition eine ausführlichere Initialisierung beim "Booten" angefügt. Der Stackpointer R7 wird z.B. auf den Hexadezimalwert 800H gesetzt. Außerdem beginnt die Endlosaddition auf dem Segment 02H. Dies kann man z.B. durch Drücken des "Maschinensprachebefehle"-Buttons (im Simulator unten links) anzeigen lassen (Abb. 10). Nur das Segment 00H und 02H sind belegt und mit einem roten anstelle eines grünen Punktes in der Übersichtstafel markiert. Die Segmente tragen die Bezeichnung "Reset Routine" und "INC A" (inkrementiere Register A).

Beispielprogramm Beispiel #3:

Im Beispielprogramm "EXAMPLE3" wird die Endlosaddition durch ein Maschinenspracheprogramm im RAM erledigt. Das RAM kann man sich durch Drücken auf den RAM-Button (im Externen RAM) anzeigen lassen (Abb. 11).

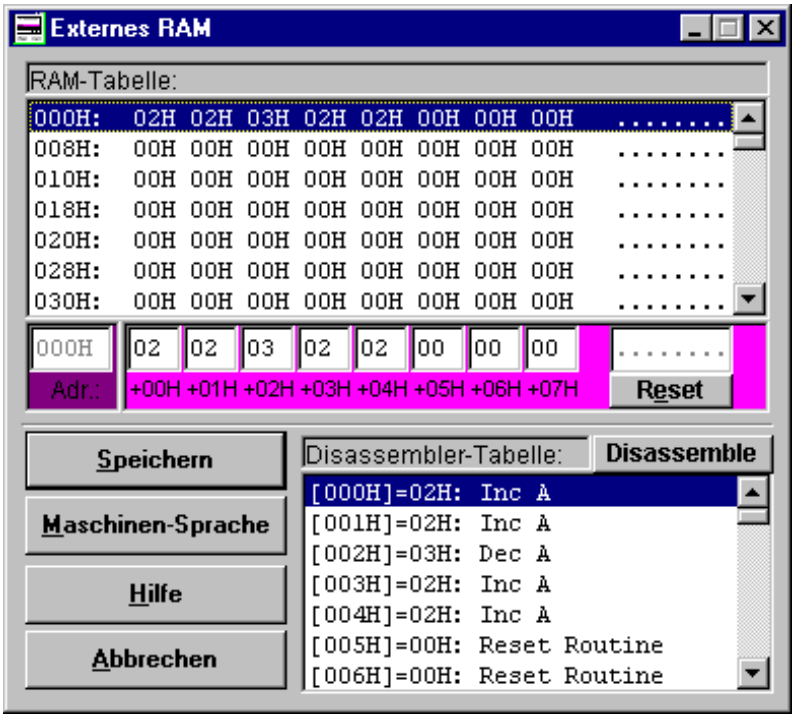

Abb. 11: Externes RAM des Beispielprogramms "EXAMPLE3". Die Maschinensprachebefehl-Sequenz 02H-02H-03H-02H-02H bewirkt eine Inkrementierung und Dekrementierung des Registers R1 (Akkumulator A).

Die Befehlssequenz 02H-02H-03H-02H-02H auf den ersten Speicherstellen des 2048 Byte großen CPU-RAMs bewirken mit 02H eine Inkrementierung des Registers R1 und mit 03H eine Dekrementierung. Dies kann leicht in dem Maschinensprachebefehls-Editor erkannt werden, der über den RAM-Editor mit dem Button "Maschinen-Sprache" erreicht werden kann (Abb. 12).

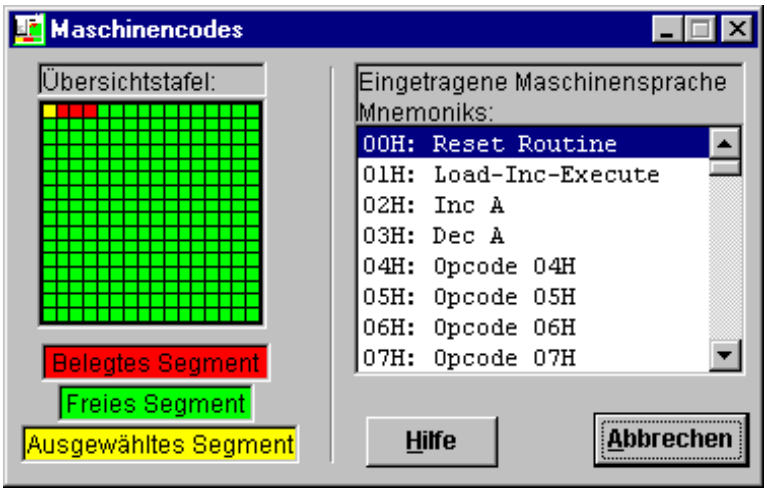

Abb. 12: Der Maschinensprachebefehls-Editor zeigt die Mikrocode-Segmente des Mikrocode-ROMs an. Die Maschienensprachebefehle entspringen der mnemonischen Bezeichnung der ROM-Segmente.

Das Beispielprogramm enthält auf dem Segment 01H einen Mikrocodesequenz, welche schrittweise

1. die RAM-Werte einliest und als Adressen im Mikrocode-ROM interpretiert (Load-),
2. den Programm-Counter, der auf den nächsten Maschinensprachebefehl im RAM zeigt, um eins erhöht (Inkrement-) und
3. zur Ausführung an die Mikrocode-ROM-Adresse springt (Execute-). Dort wird im Segment 02H letzten Endes der Maschinenbefehl INC A ausgeführt, sowie im Segment 03H der Befehl DEC A.

Nach der Ausführung der Schritte 1-2-3 wird der Zyklus mit dem nächsten Maschinensprachebefehl wiederholt durchgeführt. Die maschinenbefehlsweise Abarbeitung des Mikrocodes kann mit dem Load-Inkrement-Execute-Zyklus-Buttons durchgeführt werden. Welcher Maschinensprachebefehl gerade abgearbeitet wird, kann im Mikrocodeadressrechner unten rechts unterhalb des Maschinensprachebefehle-Buttons eingesehen werden.

Beispielprogramm Beispiel #4:

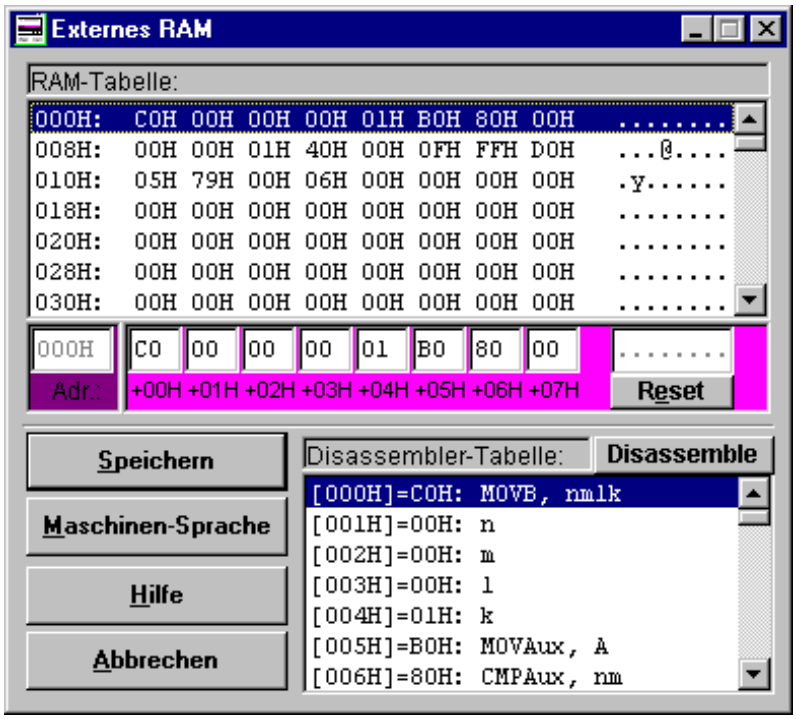

Abb. 13: Externes RAM mit der Maschinensprachebefehlssequenz, die die Fakultät der ins Register R1 geladenen Zahl berechnet.

Das Beispielprogramm "EXAMPLE4" bildet den krönenden Abschluß einer Mikrocodesimulation. Hier wird die Fakultät der Zahl 4 berechnet, die bei der Initialisierung durch das im RAM abgelegte Maschinenprogramm automatisch in das Register R1 geschrieben wird. Initialisieren Sie wie gewohnt den Simulator mit RESET und drücken Sie den Taktgeber-Button Break-Point. Lassen Sie sich nun von den flackernden Signalleitungen beeindrucken bis, im Register R2 der Hexadezimalwert 18H für die Zahl 4!=1*2*3*4=24 steht. Das Programm hält automatisch am Break-Point-Befehl FFH an. Sie können aber auch die Geschwindigkeit des Signalflußes mit einer durch die Wahl von Taktsequenzen und der Schrittverzögerung (siehe Abb. 9) selber bestimmen. Für Interessierte ist hier das aus Abb. 13 kommentierte Assemblerprogramm zur Fakultätsberechnung:

[000H]=C0H: MOVB, nmlk
[001H]=00H: n
[002H]=00H: m
[003H]=00H: l
[004H]=01H: k
[005H]=B0H: MOVAux, A
[006H]=80H: CMPAux, nm
[007H]=00H: n
[008H]=00H: m
[009H]=00H: Reset
[00AH]=01H: Load-Inc-Execute
[00BH]=40H: JG nm (Jump)
[00CH]=00H: n
[00DH]=0FH: m
[00EH]=FFH: Break-Point, Halt
[00FH]=D0H: MULB, Aux
[010H]=05H: DEC Aux
[011H]=79H: JP nm
[012H]=00H: n
[013H]=06H: m