

ULog - Kurzeinstieg

1. Das Programm wurde bereits zu DOS-Zeiten konzipiert und vielfach im Schulunterricht eingesetzt. Die Windows-typischen Eingabeelemente liefert nicht die Funktionalität wie die zum ursprünglichen Programm gehörigen, wenn man nicht unvertretbaren Aufwand beim Umprogrammieren eingehen will. Das Programm läuft in einem Fenster. Wo es sinnvoll ist, werden die grafischen Möglichkeiten von Windows genutzt.

Eingaben können über Textfelder oder über Menüleisten am unteren Bildschirmrand getätigt werden. Beide Eingabeelemente können **durch Betätigen der ESC-Taste verlassen** werden, eventuell auch über andere Tasten, die hierzu angeboten werden.

In **Textfeldern** kann man sich mit den Cursorsteuertasten oder der Maus bewegen, ENTER schließt eine Zeile ab und springt zur nächsten.

Die **Menüleisten** (dunkelgrau unterlegt) bieten die Auswahl verschiedener Funktionen. Die Funktionen können mit den Cursorsteuertasten mit anschließendem ENTER oder durch Anklicken mit der Maus ausgewählt werden. Eine andere Möglichkeit besteht darin, den Buchstaben zu drücken, den man rechts von der gewünschten Funktion nach einem »-Zeichen findet. ESC verlässt die Menüleiste, ohne eine Funktion auszurufen.

Die Menüleisten reagieren teilweise passend je nach Situation auf Mausclicks in das restliche Programmfenster.

Einige **Programmfunktionen**, laufen zeitlich unbegrenzt. ESC verlässt diese Funktionen zum gewünschten Zeitpunkt. Diese Möglichkeit wird entsprechend angezeigt.

2. Eine Hilfe-Funktion wurde nicht implementiert, da dies nur über ein weiteres Fenster realisiert werden kann, andererseits das Programm so angelegt ist, dass die Bedienung keine Probleme schaffen dürfte. Ausnahmen sind die Mikroprogrammierung des CPU-Modells und die Assembler-Simulation. Zu diesen beiden Themen sind dem Programmpaket entsprechende Textseiten hinzugefügt, die ausgedruckt an Schüler verteilt werden können.

3. Unter **Schaltungen** findet man u.a. die Menüpunkte Gatter, Speicher, Schaltplan und QMC-Verfahren.

Unter dem Menüpunkt „Gatter“ findet man die Grundbausteine NOT-, AND- und OR-Gatter, aufgebaut aus Transistoren, dazu aus ihnen zusammengesetzte MUX- und DMUX-Schaltungen. Unter „Speicher“ werden RS-FlipFlops, EEPROMs und der Aufbau von Speicherchips behandelt.

Selbst entwickelte Schaltungen können unter dem Menüpunkt „Schaltplan“ eingegeben und getestet werden.

Das „QMC-Verfahren“ erstellt aufgrund einer Tabelle von Ein- und Ausgabewerten selbständig eine passende Schaltung. Das automatisierte QMC-Verfahren nutzt nicht alle Möglichkeiten der Optimierung, wie z.B. das Verneinen von Ausgangsspalten.

4. Das **CPU-Modell** gestattet, es in Mikrocode zu programmieren und z.B. einen Interpreter für Assembler-Befehle zu schaffen. Der Beginn des Interpreters ist als Datei 8088.cpu dem Paket beigelegt. Wird die CPU-Simulation gestartet, so werden weiß unterlegt an verschiedenen Stellen Zahlenwerte angezeigt, durch die die Arbeitsweise der CPU verdeutlicht wird.

Der Mikrocode müsste eigentlich komplett durch Dualzahlen beschrieben werden. Dies wurde hier durchbrochen, um unnötige Schwierigkeiten zu vermeiden.

Ein Mikrobefehl setzt sich zusammen aus einer 8-stelligen Dualzahl (B), die als Operand genutzt werden kann, und 4 zweistelligen Dezimalzahlen zur Ansteuerung der MUXe bzw. des DMUX. Hinzu kommt eine dreistellige Dezimalzahl, die bestimmt, wo der nächste Mikrobefehl zu finden ist (ROM-„Ziel“).

Für die MUXe gilt: 01 .. 09 schalten die Register 1 bis 9 durch, 10 Daten vom Datenbus, 11 die Zahl B und 12,13,14 die Ergebnisse von AND, XOR, ADD (letzteres nur für MUX4).

Für das DMUX gilt, 01..09 schaltet den Eingang auf die Register 1 bis 9, 10 den Eingang auf den Datenbus. Hat man zu Beginn „Adressenumschaltung“ gewählt, so bestimmt 11, dass der Eingang als ROM-„Ziel“ interpretiert wird.

Die Zieladresse für das ROM setzt sich aus zwei Teilen zusammen: Im Dualsystem wird als erste Stelle das E-Flag (Gleichheit der Operanden) der ALU, als zweite Stelle das C-Flag (Übertrag) der ALU genommen. Ab der dritten Stelle findet sich das ROM-„Ziel“. Der nächste Mikrobefehl hängt also außer von dem „Ziel“ auch von E- und C-Flag ab!

5. Die **Assembler-Simulation** entstand aufgrund der Eigenschaft moderner Windows-Versionen, den direkten Programm-Zugriff auf die CPU zu blockieren. Die Maschinensprache der Simulation orientiert sich an derjenigen von PC's und bietet einen Grundbestand aus den Befehlen, die am häufigsten benötigt werden. Programme in Maschinencode können auch im CPU-Modell gestartet werden, wenn vorher das Programm 8088.cpu unter „CPU-Modell“ eingelesen wurde. Der Assemblercode ist begrenzt auf 128 Zeilen, der Adressraum auf &H00 - &HFF.

Um die Leistungsfähigkeit von Maschinenprogrammen gegenüber kompilierten Basic-Programmen zu zeigen, werden unter dem Menüpunkt „Vergleich“ die Zeiten gemessen, die entsprechende Programme zum Aufsuchen von Primzahlen bis 100000 benötigen.

6. Der **endliche Automat** kann Texte analysieren. Der Automat wird durch Regeln beschrieben, die links von “->“ eine Zeichenkette mit angehängten (eingebetteten) Zustand ergänzt wird und rechts von “->“ mit einem Zustand endet. Zustände sind dabei Zeichenketten, die in [und] eingeschlossen sind. Die Regeln werden so abgearbeitet, dass der Automat prüft, ob er in einer vorgegebenen Zeichenkette die Zeichenkette links von “->“ findet und bei Erfolg diese Zeichenkette durch den Zustand rechts von “->“ ersetzt. Dies wiederholt sich, bis keine Regel anwendbar ist oder der Endzustand [E] erreicht ist. Ist im Endzustand kein weiteres Zeichen vorhanden, so ist der Ausgangsstring „akzeptiert“, in allen anderen Fällen nicht.

Der Prozess startet mit einem vorgegebenen String, an dem der Rechner den Startzustand [S] anhängt.

7. Unter dem Menüpunkt **Platine** kann man Positionen auf einer Fläche angeben, die miteinander verbunden werden sollen. Die Positionen könnten die Anschlüsse von Bauteilen auf einer Platine sein, die Verbindungen Leiterbahnen. Dargestellt wird, wie der Rechner die Verbindungen automatisch erzeugt, so dass sich diese nicht kreuzen.