4-Gewinnt auf einem Mikrocomputer der 8051-Famile

von

Max Heidinger, Sophia Matthis, Pascal Riesinger, Martin Stephan

Kurs TINF17B1

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Mit dem 8051 ist ein relativ einfacher Einstieg in die Assemblerprogrammierung möglich. Assembler ist sehr maschinennah und bietet aus diesem Grund nicht die Annehmlichkeiten, die beispielsweise objektorienterte Programmiersprachen wie Java mit sich bringen.
An einem Simulator lässt sich gefahrlos erproben, wie mit Pointern und sehr begrenztem Speicherplatz umzugehen ist.
Durch die andere Herangehensweise an ansonsten vertraute Programmierstrukturen wie zum Beispiel Vergleichen, die nun mit Jump-Befehlen und negativen Vergleichen implementiert werden müssen, wird das logische Denkvermögen geschult.

1.2 Aufgabenstellung

Es soll ein Spiel nach dem bekannten Spielkonzept von „4-Gewinnt“ in Assembler auf dem 8051 entwickelt und implementiert werden.
Dafür wird für die Visualisierung der Spielfläche eine entsprechende Hardware gewählt, auf der die Spielstände der beiden Spieler angezeigt werden (Output). Die Auswahl der Spalte, in der ein „Spielstein“ platziert werden soll, muss ebenfalls über eine entsprechende Hardware gelöst werden (Input).
Die beiden Spieler müssen damit in der Lage sein, abwechselnd sogenannte Spielstei-

ne in selbst ausgewählte Spalten zu werfen, welche dann am oberen Ende des Stapels angefügt werden. Hat ein Spieler es geschafft, 4 seiner Spielsteine hintereinander in eine Reihe oder Spalte zu bringen, hat er gewonnen.
Dabei soll sicher gestellt sein, das nicht mehr Spielsteine in eine Spalte geschmissen werden als Platz ist. Auch wichtig ist, dass die Spielsteine der beiden Spieler klar voneinander zu unterscheiden sein müssen, damit es nicht zu Verwechslungen kommen kann.

2 Grundlagen

2.1 Assembler

Ein Assembler ist ein Übersetzer, der Assemblersprache in Maschinensprache (Binärcode) umwandelt, was bei Assemblersprache ohne irgendwelche Umwege möglich ist.
Dies ist möglich, da die Assemblersprache eine sehr maschinennahe Sprache ist und dabei zwischen der Maschinensprache und den sogenannten höheren Programmiersprachen angesiedelt ist. Dabei sind die Progaramm-Instruktionen durch symbolhafte Abkürzungen repräsentiert und nutzen den vollen Befehlssatz eines Prozessors, im Gegensatz zu höheren Programmiersprachen, die sich auf eine Auswahl des Befehlssatzes beschränken.
Die Assemblersprache arbeitet sehr nahe an der Prozessarchitektur, was sich auf eine sehr gute Effizienz auswirkt. Der große Nachteil dabei ist aber, dass jeder Prozessor eine eigene Architektur hat, was einen eigenen Befehlssatz mit sich zieht. Somit muss zum Teil ein Programm ganz umgeschrieben werden, um auf einem neuen Prozessor laufen zu können. Bei Intel sind alle Prozessoren abwärts miteinander kompatibel.

2.2 Der 8051 Mikrocomputer

Die ersten Mikroprozessoren der 8051-Reihe wurden im Jahr 1980 von Intel entwickelt. Es handelt sich dabei um einen direkten Nachfolger der 8048-Familie. Die 8051-Familie erfreute sich extrem großer Beliebtheit. So wurden über 250 Familienmitglieder von verschiedensten Herstellern wie Philips, Siemens, AMD, OKI und weiteren gebaut und produziert. Der Höhepunkt der Beliebtheit des 8051 war das Jahr 1995, in welchem diese Mikroprozessorfamilie einen Marktanteil von bis zu 30 Prozent erreichte und täglich mehr als eine Million Prozessoren hergestellt wurden.

Auch technisch war der 8051 zu damaligen Zeiten hochmodern, was man folgenden Eckdaten entnehmen kann:

- $1,2$ - $18MHz$ Taktrate, oft werden $12MHz$ verwendet
- 4 kByte ROM
- 128 Byte RAM
- 4 8-Bit Eingabe- und Ausgabeports
- 2 16-Bit Zähler beziehungsweise Zeitgeber
- Eine USART-Schnittstelle
- 5 Interruptquellen
- Bei einer Taktrate von $12MHz$ laufen
  - 58% der Befehle in $1\mu s$
  - 40% der Befehle in $2\mu s$

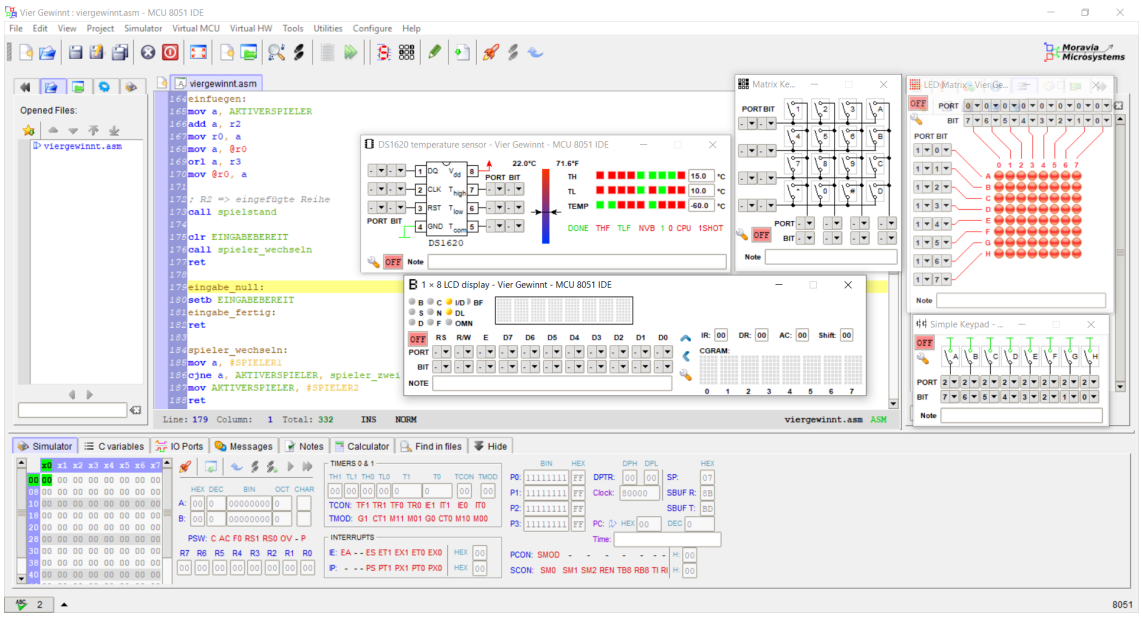

Abbildung 2.1: MCU-8051 IDE

– 2% der Befehle in $4\mu s$

ab. Die langsamsten Befehle sind beispielsweise Multiplikation und Division.

2.3 Entwicklungsumgebung MCU-8051 IDE

In diesem Kapitel geht es um die in dem Projekt zum Einsatz kommende IDE (Integrated Development Evironment). Dabei handelt es sich um einen Simulator zur Entwicklung verschiedener Mikrocontroller unter optionaler Verwendung von emulierten elektronischen Peripheriegeräten. Dabei handelt es sich beispielsweise um eine 8-Segement-Anzeige, LEDs, LED-Displays, LED-Matrizen, eine einfache Eingabe oder Matrizen-Eingabe, LCD-Anzeigen oder Temperatur-Sensoren. Dies können vom Anwender frei hinzufügt und nach ordnungsgemäßer Konfiguration eingesetzt werden. Die Software wurde sowohl für Linux, als auch Windows Betriebssysteme entwickelt.

Unterteilt wird die IDE in vier verschiedene Bereiche. Mittig befindet sich der eingebaute Texteditor mit zusätzlichem Syntaxhighlighting, -validierung und -autocompleting - In diesem kann in C und Assembler programmiert werden. Links und rechts daneben befindet sich die Projektübersicht und einem Panel zum Debuggen. Darunter die Anzeige für den Speicher, Akkumulator und dem Register des Prozessors. Diese beinhaltet darüber hinaus noch den Timer und die Ports für eine externe Kommunikation.

3 Konzept

Entwickelt werden soll ein 4-Gewinnt Spiel für zwei Spieler. Die Ausgabe soll auf einer $8 \cdot 8$ LED-Matrix erfolgen. Die Eingabe wird durch 8 Schalter ermöglicht.

3.1 Analyse

Da die Entwicklungsumgebung keine mehrfarbige LED-Matrix bereitstellt, werden die Steine des zweiten Spielers durch Blinken von den Steinen des ersten Spielers unterschieden.

3.2 Programmentwurf

Der Programmfluss, wie er in Abbildung 3.1 dargestellt ist, beschreibt den Ablauf des Programms.
Nach dem Start des Simulators beginnt die Initialisierung des Programms - der benötigte Speicher wird zurückgesetzt, Spieler 1 als aktiver Spieler ausgewählt und der Timer gestartet. Danach beginnt die Hauptschleife.
In ihr wird wiederholt abgefragt, ob aktuell eine Eingabe getätigt werden kann – also, ob das Keypad vollständig auf false gestellt wurde. Ist dies möglich prüft eine Schleife auf eine Eingabe durch den aktiven Spieler. Wurde diese ausgeführt, wird anhand des Zuges überprüft, ob der Spieler es geschafft hat 4 Steine nebeneinander zu set-

zen. Ist dies der Fall, hat er gewonnen - andernfalls kommt es zum Spielerwechsel und die Hauptschleife beginnt von Neuem. Sobald ein Spieler gewonnen hat, wird die Spielerzahl des Siegers angezeigt. Es besteht die Möglichkeit, ein neues Spiel zu beginnen, indem auf dem Keypad der erste und letzte Taster betätigt werden. Dann wird der Spieler gewechselt – also fängt der Verlierer an – und das Spiel startet erneut.

Zusätzlich zu dieser Hauptschleife wird über ein Timerinterrupt regelmäßig eine Routine ausgeführt, welche das aktuelle Spielbrett ausgibt. Dabei werden die vom ersten Spieler gesetzten Steine auf der LED-Matrix durchgängig beleuchtet, während die Steine des zweiten Spielers blinken.

> **Hinweis:** Die Wiederholrate, mit welcher die LED-Matrix aktualisiert wird ist zu groß, um den Blinkeffekt sehen zu können. Sie musste allerdings für die Entwicklung im Simulator so stark erhöht werden, da der Simulator deutlich langsamer als die tatsächliche Hardware ist.

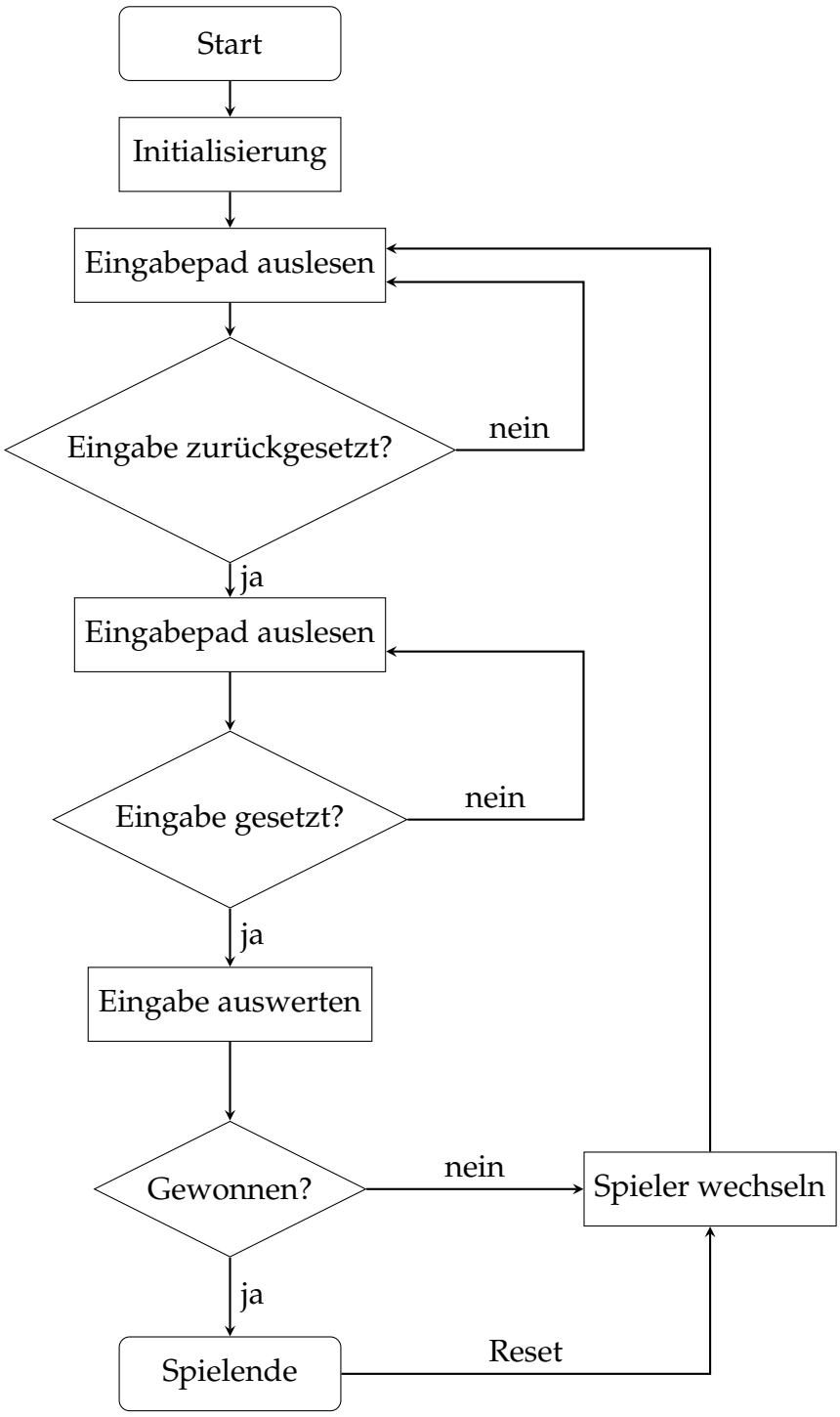

Abbildung 3.1: Programmfluss des Spieles

4 Implementation

In Assembler

5 Zusammenfassung

War gut, nochmal machen!