Pong 8051 TINF16B4

Bericht Systemnahe Programmierung

Maurice Heumann, Mirko Müller, Alexander Rengers

Inhaltsverzeichnis

[Einleitung 2](#_Toc517257899)

[Motivation 2](#_Toc517257900)

[Aufgabenstellung 2](#_Toc517257901)

[Grundlagen 2](#_Toc517257902)

[Assembler 2](#_Toc517257903)

[Der 8051 Mikrocomputer 2](#_Toc517257904)

[Entwicklungsumgebung MCU-8051 IDE 2](#_Toc517257905)

[Konzept 3](#_Toc517257906)

[Analyse 3](#_Toc517257907)

[Programmentwurf 3](#_Toc517257908)

[Implementation 5](#_Toc517257909)

[Zusammenfassung 5](#_Toc517257910)

# Einleitung

## Motivation

Im Zuge der Vorlesung „Systemnahe Programmierung“ konnten wir verschiedene Grundlagen in Assemblerprogrammierung erlernen. Dieses Projekt soll nun eine eigene Anwendung des Erlernten darstellen, indem wir unser neues Wissen in eine funktionierende Anwendung umsetzen.

## Aufgabenstellung

Die Aufgabenstellung war es, eine eigene Anwendung mit der Assembler-Programmiersprache zu erstellen. Diese Anwendung soll auf der 8051-Prozessorarchitektur basieren und mit Hilfe der MCU-8051-IDE erstellt werden.

# Grundlagen

## Assembler

Eine Assemblersprache, kurz auch Assembler genannt, ist eine Programmiersprache, die auf den Befehlsvorrat eines bestimmten Computertyps (d. h. dessen Prozessorarchitektur, hier Intel-8051) ausgerichtet ist.

Assemblersprachen bezeichnet man deshalb als maschinenorientierte Programmiersprachen und weiterhin als Programmiersprachen der zweiten Generation: Anstelle eines Binärcodes der Maschinensprache können Befehle und deren Operanden durch leichter verständliche mnemonische Symbole in Textform (z. B. „MOVE“), Operanden z. T. als symbolische Adresse (z. B. „PLZ“), notiert und dargestellt werden.

Der Quelltext eines Assemblerprogramms wird mit Hilfe einer Übersetzungssoftware (Assembler) in Maschinencode übersetzt. Dagegen übersetzt in höheren Programmiersprachen (Hochsprachen, dritte Generation) ein sogenannter Compiler abstraktere (komplexere, nicht auf den Prozessor-Befehlssatz begrenzte) Befehle in den Maschinencode der gegebenen Zielarchitektur – oder in eine Zwischensprache.

## Der 8051 Mikrocomputer

MCS-51 ist die Bezeichnung einer 1980 von Intel vorgestellten Familie von 8-Bit-Mikrocontrollern. Bei einem Mikrocontroller sind im Optimalfall alle Teile eines Computersystems (Prozessor, Programmspeicher, Datenspeicher und Ein-/Ausgabeeinheiten) in einem einzigen Baustein zusammengefasst. Zu Beginn hatte sie nur drei Mitglieder mit den Bezeichnungen 8051, 8031 und 8751. Beim 8031 befindet sich das ROM in einem externen Baustein, wohingegen es sich beim 8051 und 8751 im Baustein selbst befindet – entweder in einem maskenprogrammierten ROM (8051) oder in einem EPROM (8751). Die Familie wurde zunächst in NMOS-Technologie, nach einigen Jahren dann auch in der heute üblichen CMOS-Technologie hergestellt.

## Entwicklungsumgebung MCU-8051 IDE

Die MCU-8051-IDE (Integrated Development Environment

) ist eine frei verfügbare Entwicklungsumgebung für verschiedene Mikrocontroller, die auf der 8051 Architektur basieren. Die Software bietet weiterhin einen eigenen Simulator und Assembler. Weiterhin werden zwei Programmiersprachen unterstützt: Die C- und die Assembler-Programmiersprache. Wir setzten diese IDE ein für die Entwicklung und Simulation unseres Programmes. Dabei war die Hardwaresimulation essentiell um unser Programm zu visualisieren.

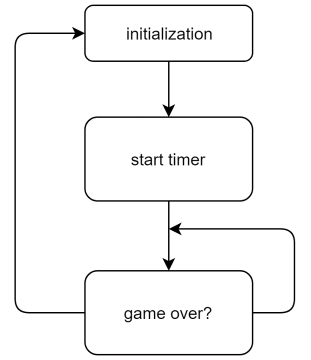
# Konzept

Im Folgenden stellen wir unseren Programmentwurf vor mit der graphischen Darstellung des Programmablaufes. Die Idee unseres Programmes ist es, das bekannte Spiel Pong auf der 8051-Architektur zu realisieren. Dafür verwenden wir eine 8x8-LED-Matrix, zur Darstellung des Balles und der beiden Schläger, sowie eines einfachen Keypads um Eingaben des Spielers entgegenzunehmen.

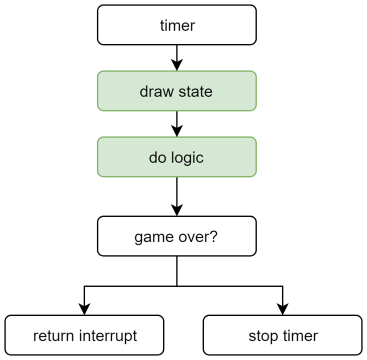
## Analyse

## Programmentwurf

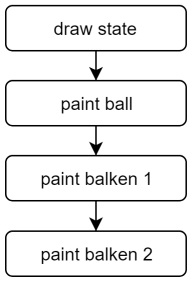
### Hauptschleife

In der Hauptschleife wird ein Timer gestartet, der Parallel für die Ausführung der Logik zuständig ist. Anschließend wird in einer Schleife geprüft, ob das aktuelle Spiel vorbei ist und falls ja zurück zum Initialisierungspunkt gesprungen, um das Spiel neu zu starten.

### Timer

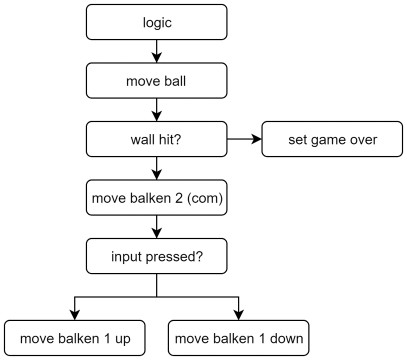
Um das Zeichnen der Objekte mit der Aktualisierungsfrequenz der LED-Matrix zu synchronisieren wird ein Timer benötigt. Zusätzlich zur Zeichenlogik soll auch die Spiellogik verarbeitet werden.

### Zeichnen

Die Zeichenlogik unterteilt sich in 3 Schritte:

* Das Zeichnen des Balls als einzelnen Punkt
* Das Zeichnen des linken und
* des rechten Balkens als vertikale Folge von 3 Punkten.

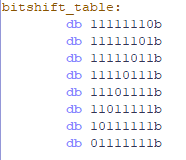
### Logik

Die Spiellogik berechnet zunächst die Folgepostition des Balles. Wird dabei die linke Wand berührt, ist das Spiel vorbei und das „game over“-Flag wird gesetzt.

Ansonsten wird der rechte Balken analog zur Position des Balles verschoben.

Der linke Spielerbalken wird entweder nach Oben oder nach Unten verschoben, abhängig davon, ob der Schalter am Keypad gedrückt ist.

### Zeichenlogik

Um zeichnen zu können müssen die entsprechenden Bits der Ports der LED-Matrix auf 0 gesetzt werden.

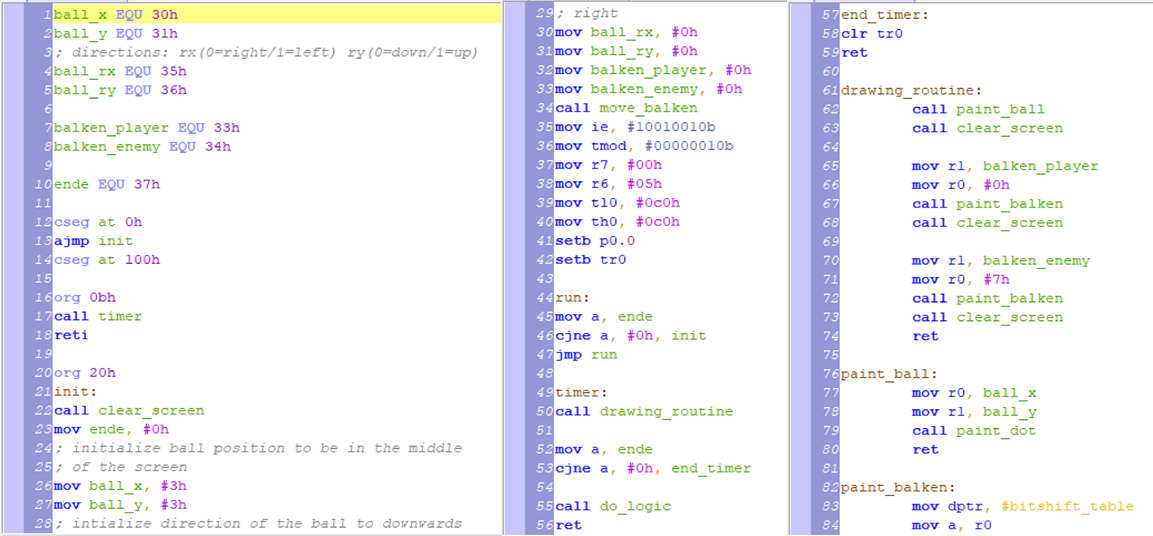
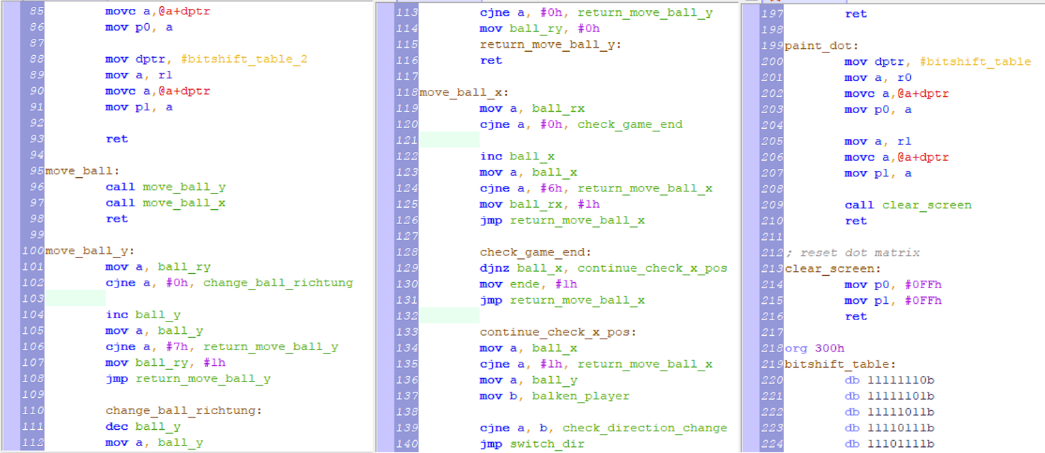
Hierfür müssen die Zielkoordinaten in eine Bitfolge übersetzt werden.

Bei den Koordinaten (3,3) muss beispielsweise das 3. Bit 0 sein und die restlichen 1.

Das entspricht dem Komplement von 23. Normalerweise würde man den Exponenten über Bitshifting lösen, jedoch erlaubt der 8051 nicht über mehrere Bits auf einmal zu shiften.

Der effizientere Ansatz ist daher das Auflösen der Werte über eine Tabelle deren Einträge die Ergebnisse der Exponenzierung beinhaltet.

# Implementation

Hier ist unser Programmcode dargestellt, auf dem das Programm basiert.

# 

# Zusammenfassung

In diesem Projekt konnten wir lernen, wie man die Grundlagen der Assemblerprogrammierung in einer Anwendung umsetzt. Weiterhin lernten wir die differenzierenden Merkmale, die die systemnahe Programmierung ausmachen, kennen und konnten dieses Wissen in diesem Projekt anwenden. Verschiedene Schwierigkeiten wie die Eingabe konnten von uns gelöst werden und die Anwendung Pong konnte zu unserer Zufriedenheit erstellt werden. Sicherlich kann man noch weitere Funktionalitäten implementieren, dies sprengte jedoch den Rahmen dieser Vorlesung.