Projektbericht

Tennis-Scoreboard

Für die Vorlesung
Systemnahe Programmierung I

Studiengang Informatik
Studienrichtung Angewandte Informatik
Duale Hochschule Baden-Württemberg Karlsruhe

Von
Fabian Blatz,
Sven Baumann,
Marcel Borrmann,
Christian Gutermann

Abgabedatum:	3. Oktober 2017
Kurs:	TINF15B4
Dozent:	Prof. Dr. Ralph Lausen

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Motivation dieses Projektes entstand durch die Vorlesung Systemnahe Programmierung im 4. Semester des Studiengangs zum Bachelor of Science - Applied Computer Science. Im Rahmen der Vorlesung wurden Inhalte zur Programmiersprache Assembler, sowie Kenntnisse über den 8051 Mikrocomputer und der Entwicklungsumgebung MCU-8051 IDE vermittelt. Um dieses neu erlangte Wissen zu vertiefen sollte in Gruppen von bis zu drei Personen eine Anwendung mit den gelehrten Mitteln realisiert werden.

1.2 Aufgabenstellung

Die vorgegebene Aufgabenstellung war recht trivial formuliert. In Gruppen sollte sich eine Anwendung überlegt werden, welche mithilfe der MCU-8051 IDE realisiert werden kann. Einschränkungen bezüglich des Umfangs der Anwendung gab es keine.

In Folge dessen haben wir uns dafür entschieden eine Anzeigetafel zu entwickeln, welche es ermöglicht zweistellige Punktzahlen anzuzeigen. Ebenso soll es möglich sein diese Punktzahlen manuell in einzelnen Schritten hochzuzählen. Eine Begrenzung der Punktzahlen auf ein bestimmtes Limit wird softwareseitig nicht vorgenommen und ist somit nur durch die Hardware auf 99 je Seite beschränkt.

2 Grundlagen

2.1 Assembler

Ein Assembler ist eine Übersetzungssoftware für Assemblersprache in Maschinensprache bzw. Binärcode. Häufig wird jedoch auch die Assemblersprache an sich als Assembler bezeichnet.

Assemblersprachen sind hardwarenahe Programmiersprachen. Sie werden auch als Programmiersprachen der zweiten Generation bezeichnet, da sie die Nachfolger der direkten Programmierung mit Zahlencodes sind. Assemblerbefehle sind mnemonische Symbole in Textform und werden mithilfe eines Assemblers direkt in Maschinenbefehle übersetzt. Hierbei repräsentiert ein Assemblerbefehl genau einen Maschinenbefehl und dient somit nur der Verständlichkeit für den Menschen. Programmiersprachen der dritten Genration bzw. Hochsprachen hingegen benötigen einen Compiler welcher die komplexen Programmanweisungen in mehrere Maschinenbefehle übersetzt.

Da sich die Befehlssätze von unterschiedlichen Prozessoren, Mikrocontrollern oder auch digitalen Signalprozessoren unterscheiden gibt es für verschiedene Computertypen auch darauf zugeschnittene Assemblersprachen.

Assemblersprachen werden heute jedoch nur noch selten eingesetzt. Früher war die Programmierung in Assembler notwendig, um die knappen Ressourcen der Mikrocontroller optimal auszunutzen. Durch technische Fortschritte steigt die Leistungsfähigkeit der Chips jedoch immer weiter an, wodurch mittlerweile immer mehr C-Compiler auch in diesem Bereich die Assembler ablösen. [1]

2.2 Der 8051 Mikrocomputer

Der 8051 Mikrocontroller gehört zur Familie der MCS-51 8-Bit- Mikrocontroller von Intel, welche 1980 vorgestellt wurden. Der original 8051 ist zwar mittlerweile veraltet, jedoch gibt es hunderte Varianten (Derivate) davon, die teilweise durchaus auf dem aktuellen Stand der Technik sind. [2]

Der allererste Mikroprozessor i4004 wurde 1970 von Intel gebaut. Auch die Mikroprozessoren wie der 8051 und 8086 wurden von der Intel Corporation entwickelt. Der Originale 8051 ist ein maskenprogrammierter Mikrocontroller und benötigt für einen Befehl mindestens 12 Takte. Befehls- und Datenspeicher sind logisch getrennt, auch wenn dieser über einen einzigen gemultiplexten externen Bus adressiert werden. Allerdings ist es umstritten ob es sich dabei um eine Harvard-Architektur oder eine Von Neumann-Architektur handelt. Mit steigender Beliebtheit des 8051 hat Intel den MCS-51- CPU-Kern an viele Halbleiterhersteller lizensiert um damit eine Basis für einen herstellerübergreifenden Industriestandard zu schaffen. Dies führte zu verschiedenen Versionen mit unterschiedlichen Geschwindigkeiten und on-chip RAM, auf denen jedoch auch 8051 Programme anderer Varianten laufen. [2] [3]

Kerndaten des Original 8051 [4]:

- jeweils bis zu 64 kB externer Daten- und Programmspeicher adressierbar
- 128 Byte internes Ram (8052: 256 Byte)
- 2 Timer/Counter (8052: 3 Timer/Counter)
- 2 externe Interrupts
- 4 8-bit I/O Ports, zwei davon für den Zugriff auf externen Speicher
- Hardware UART

2.3 Entwicklungsumgebung MCU-8051 IDE

Die MCU-8051 DIE ist eine freie integrierte Entwicklungsumgebung für auf dem 8051 basierende Mikrocontroller. Sie unterstützt zwei Programmiersprachen: C und Assemblersprache. Ebenso verfügt sie über einen eigenen Simulator und Assembler. [5]

Abbildung 1: MCU-8051 IDE

In der Bildschirmmitte (Abb. 1) befindet sich der Editor, in dem der auszuführende Code bearbeitet werden kann. Im unteren Teil der DIE werden Arbeitsspeicher, Register, Timer, sowie die Ports des simulierten Mikrocontrollers angezeigt. In dem kleinen Fenster wird eine LED Display simuliert, auf der die Anwendung dargestellt wird.

3 Konzept

Die Nachfolgende Betrachtung dient der Analyse der benoetigten Module die zur korrekten Implementierung und Ausfuerhung des geplanten Scoreboards benoetigt werden.

3.1 Analyse

Zur vollstaendigen und korrekten Implementierung des geplanten Funktionsumpfangs sind folgende Komponenten von Noeten:

- Anzeigemoeglichkeiten
- Speichermoeglichkeiten fuer den momentanen Spielstand
- Taster, um den Speicherstand zu inkrementieren, dekrementieren oder zurueckzusetzen
- Ein Konzept, um die Tasterbedienung moeglichst bedienerfreundlich zu gestalten

3.1.1 Anzeige

Fuer die Anzeige wurde ein Multiplex-LED-Display verwendet, welches folgende Eigenschaften aufweist:

- 4x7 LED-Felder
- Gesteuert ueber 12-Pins an 2 Ports, in diesem Fall P2 und P3

3.1.2 Speicherstruktur

Hier einfuegen wie Spielstaende gespeichert werden, bitte, danke

3.1.3 Taster

Um das Scoreboard anzusteuern stehen dem Nutzer 3 Taster zur verfuegung, welche einem Punkt fuer den linken Spieler, dem rechten Spieler und einer Reset-Taste entsprechen. Da in der Simulationsumgebung leider keine Druckknoepfe sondern nur Schalter gibt, muss der Nutzer hier den Schalter eigenstaendig Ein- und wieder Ausschalten.

3.1.4 Tasterroutine

Da es nicht vom Nutzer erwartet werde kann, dass er es schafft, Taktgenau zwischen einzelnen Ueberpruefungen des Tasterstatus die Taste zu betaetigen, wurde eine Routine implementiert, um bei laengerer Tasterbetaetigung keine Mehrfachzaehlungen zu beguenstigen.

3.2 Programmentwurf

4 Implementation

Die Implementierung des Projekts findet in einer einzelnen Assemblerdatei statt, welche in verschiedene SSubroutinen"gegliedert ist.
Die wichtigsten Subroutinen hierbei sind:

- checkports
- buttonpressed
- Sowie die Routinen zur Anzeige des Punktestands

4.0.1 Checkports

Checkports ist, wie der Name schon sagt, die Routine zur Ueberpruefung des Portstatus der einzelnen Taster.

```
checkPorts:
acall show
JB F0, buttonpressed
MOV C,P0.0
JC increaseleft
MOV C,P0.1
JC increaseright
MOV C, P0.2
JC resetgame
sjmp checkPorts
```

Da es sich hierbei ebenfalls um die "Hauptroutine"des Programms handelt, welche in regelmaessigen Abstaenden zur Laufzeit des Scoreboards ausgefuehrt wird, wird zu beginn der Punktestand ueber den Befehl ächall showäusgefuehrt.
Daraufhin wird der momentane Status der einzelnen Taster-Port-Bits in das Carry-Bit

geschrieben, welches daraufhin ueber "JCäuf Ein- bzw. Ausgeschaltet (Also 1 oder 0) ueberprueft wird. Sollte eine 1 in C stehen, wird zur entsprechenden Inkrementierungsroutine des jeweiligen Spielers gesprungen.
Sollte kein Taster gedrueckt worden sein, beginnt die Routine erneut.

4.0.2 Buttonpressed

Buttonpressed ist eine Behilfsroutine, welche durchlaufen wird, falls ein Taster gedrueckt, jedoch noch nicht losgelassen wurde.

```
buttonpressed:
 acall show
MOV C, P0.0
JC buttonpressed
MOV C, P0.1
JC buttonpressed
MOV C, P0.2
JC buttonpressed
CLR F0
JMP checkports
```

Auch hier, da es eine Routine ist die potentiell fuer eine Laengere Zeit durchlaufen wird, wird zu beginn der momentane Punktestand ausgegeben. Daraufhin wird ueberprueft, ob einer der drei Taster noch immer gedrueckt, also auf 1 gesetzt, ist. Sollte dies der Fall sein, beginnt die Routine erneut.
Werden alle Faelle durchlaufen, wird der Wert des Flag-0 (F0), welcher in dieser Implementierung dem Taster-Status entspricht, zurueck auf 0 gesetzt und das Programm springt zurueck in die Hauptroutine, "checkports", um ein weiterfuehren der Funktionalitaeten zu ermoeglichen.

4.0.3 Show und Increase

5 Zusammenfassung

Zusammenfassend wurde eine sehr grundlegende Implementierung eines Tennis-Scoreboards erreicht. Der Weg zur Implementierung war sehr lehrreich und zeigte die Limitierungen der Hardware deutlich auf.
So fehlte es zur Implementierung der Satzanzeigen an freien Ports, dies deutete sowohl auf die angesprochene Hardwarelimitierung als auch die aeusserst wichtige Konzeptionsphase hin. Hier haette ein Konsens zur Implementierung unter Benutzung einer anderen Display-Art erreicht werden koennen.
Die begangenen Fehler und gezogenen Schluesse werden sicher bei der Konzeption und Implementierung folgender Mikrocontrollerprojekte hilfreich sein.

Literaturverzeichnis

[1] Assemblersprache. https://de.wikipedia.org/wiki/Assemblersprache. Datum: 20.06.2017.

[2] Intel MCS-51. https://de.wikipedia.org/wiki/Intel_MCS-51. Datum: 20.06.2017.

[3] Mikroprozessor. http://www.weller.to/his/h10-70-74-erste-mikros.htm. Datum: 20.06.2017.

[4] Microcontroller-8051. https://www.mikrocontroller.net/articles/8051. Datum: 20.06.2017.

[5] MCU-8051-IDE. https://en.wikipedia.org/wiki/MCU_8051_IDE. Datum: 20.06.2017.

Abbildungsverzeichnis

Abkürzungsverzeichnis

BSP Beispiel