UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Übungsblatt 7

Assembler - I/O

Vorlesung *Technische Grundlagen der Informatik 1*, Sommersemester 2020
Erstellt von Dr.-Ing. Kristian Ehlers

Verwenden Sie für Ihre Implementierungen einen beliebigen Editor und simulieren Sie ihren Assembler-Code anschließend im **ITImega16-Simulator**. Ihnen stehen im Moodle für beide Übungen entsprechende **Templates** mit den Assembler-Dateien und gegebenenfalls Layout-Dateien für die Konfiguration der Peripherie zur Verfügung.

Es seien hiermit noch folgende Anmerkung zur Nutzung des **ITImega16-Simulators** gegeben:

- Um die Namensdefinitionen wie z.B. DDRA, PORTA, PINA usw. nutzen zu können, müssen diese mithilfe des Includes `.include "m16def.inc"` bereitgestellt werden.
- Die Konfigurationsdateien der Peripherie müssen auf `layout.js` enden.
- Die Direktiven `.list` bzw. `.nolist` erlauben es, nachfolgenden Quellcode im Simulator anzuzeigen oder auszublenden.

Aufgabe 1 Iterative Berechnung der Fibonacci-Zahlen in Assembler

In dieser Aufgabe soll ein iterativer Algorithmus zur Berechnung der Fibonacci-Zahlen in Assembler implementiert werden. Fibonacci-Zahlen bilden eine unendliche Folge, für die gilt:

$$F(1) = 1,\ F(2) = 1,\ F(n) = F(n-1) + F(n-2) \quad \forall n \in \mathbb{N} \wedge n \geq 3$$

Schreiben Sie eine Abfolge von Assemblerdirektiven, die die iterative Berechnung der Fibonacci-Zahl $F(n)$ entsprechend der gegebenen Bildungsvorschrift berechnen.

Aufgabe 2 Parallele Ein- und Ausgabe auf dem ATmega16 - Hammingdistanz

Mithilfe eines ATmega16 Mikrocontrollers soll die Hammingdistanz zweier 8-Bit Zahlen berechnet werden. Unter der Hammingdistanz versteht man die Anzahl an Bitstellen zweier Zahlen, die sich bei einem bitweisen Vergleich unterscheiden. Die Hammingdistanz der Zahlen 1_{10} und 2_{10} beträgt 2 und die der Zahlen 15_{10} und 1_{10} beträgt 3. Die beiden Zahlen liegen als Binärzahlen als Eingaben an den Ports A und B des ATmega16 vor. Es sollen in einer Endlosschleife beide Zahlen eingelesen, deren Hammingdistanz berechnet und anschließend kodiert auf dem Port C ausgegeben werden. An jedem Pin von Port C ist je eine LED angeschlossen, die leuchtet, wenn ein HIGH-Pegel (1) an diesem Pin ausgegeben wird. Tabelle 1 verdeutlicht das gewünschte Ausgabeverhalten.

Implementieren Sie das Verhalten in Assembler für den ATmega16. Definieren Sie sich gegebenenfalls verwendete Konstanten oder Namensdefinitionen einzelner Register.

Hammingdistanz	Leuchtende LEDs
0	keine LED
1	LED an PINC0
2	LED an PINC0 & PINC1
3	LED an PINC0 & PINC1 & PINC2 ≡ PINC0 - PINC2
4	LED an PINC0 - PINC3
5	LED an PINC0 - PINC4
6	LED an PINC0 - PINC5
7	LED an PINC0 - PINC6
8	alle LEDs

Tabelle 1: Ausgabeverhalten des ATmega in Abhängigkeit von der ermittelten Hammingdistanz.