UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Übungsblatt 9

Unterprogramme, Adressierung RAM und ROM

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 steht im Moodle für die Übung entsprechendes **Template** 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 Sortieralgorithmus auf dem ATmega16 - BubbleSort

Auf einem ATmega16 soll der einfache Sortieralgorithmus *Bubble-Sort* implementiert werden. Dabei wird im RAM ein unsortiertes Bytearray abgelegt und in-place sortiert, d.h. ohne dass weiterer Speicher im RAM reserviert werden muss.

Der Sortieralgorithmus *Bubble-Sort* durchläuft ein Zahlenarray und vergleicht dabei benachbarte Zahlenwerte. Je nach verwendeter Vergleichsoperation werden gegebenenfalls die benachbarten Zahlen getauscht, sodass eine aufsteigende oder absteigende Sortierung erreicht wird.
Im folgenden Pseudocode ist der Algorithmus für eine absteigende Sortierung gegeben.

```
void BUBBLESORT(word Startaddress, byte Arraylength)
{
  For (Ocount = Arraylength - 1; Ocount >= 1; Ocount--)
  {
    For (Icount = 0; Icount < Ocount; Icount++}
    {
      If{Array[Icount] < Array[Icount + 1]}
      {
        Temp = Array[Icount];
        Array[Icount] = Array[Icount + 1];
        Array[Icount + 1] = Temp;
      }
    }
  }
}
```

Der erste Parameter enthält die 16-Bit breite Startadresse des Arrays im Arbeitsspeicher und der zweite Parameter gibt die Länge des zu sortierenden Arrays an.

(a) Anlegen eines Projektes und Initialisierungen

Beginnen Sie mit einem neuen Assembler-Projekt für den ATmega16, indem Sie dafür sorgen, dass Ihnen die entsprechenden Namensdefinitionen bereitgestellt werden..

Definieren Sie sich gegebenenfalls verwendete Konstanten oder Namen einzelner Register. Reservieren Sie weiterhin im RAM-Speicher unter dem Namen `ARRAY` ab Adresse `$0060` 10 Bytes. An dieser Stelle wird später das Datenarray abgelegt und sortiert. Legen Sie im ROM-Speicher hinter Ihrem Assembler-Code aufeinanderfolgend 10 beliebige, unsortierte 8-Bit Zahlen unter dem Namen `NUMBERS` ab. Diese Zahlen bilden den Anfangszustand des zu sortierenden Arrays und sollten daher noch nicht absteigend sortiert sein.

(b) Kopieren des Zahlenarrays

Beginnen Sie nun die eigentliche Programmierung. Lesen Sie in einer Schleife nacheinander die 10 Zahlen aus dem ROM-Speicher aus und legen Sie diese zur späteren Sortierung sequentiell im RAM-Speicher an den reservierten Stellen (`ARRAY`) ab. Rufen Sie anschließend das nachfolgend zu implementierende Unterprogramm `BUBBLESORT` auf. Dieses soll im `Y`-Register die Anfangsadresse und in einem zweiten Register die Länge des zu sortierenden Arrays übergeben bekommen. Nach der Sortierung soll der ATmega in einer Endlosschleife verharren.

(c) Implementierung des Algorithmus

Implementieren Sie nun im Unterprogramm `BUBBLESORT` den Algorithmus. Dieser sortiert die Elemente des Arrays, dessen Startadresse er im `Y`-Register erwartet und dessen Länge in einem weiteren Register an das Unterprogramm übergeben werden soll.

Simulieren Sie das Verhalten vollständig im **ITImega16-Simulator** und verfolgen Sie die einzelnen Schritte im Speicher.