Hochschule RheinMain
Fachbereich Design Informatik Medien
Marcus Thoss, M.Sc.

Mikroprozessortechnik
SS 2020
LV 2522

Übungsblatt 1

Wir beschäftigen uns in diesem Praktikum zunächst mit Assemblerprogrammierung auf unterster Ebene, um ein pratisch motiviertes Gefühl für die Arbeitsweise eines Mikroprozessors zu entwickeln. Als einfache Architektur für den Einstieg wird die *ATmega*-Architektur der Firma Microchip (vormals Atmel) verwendet. Um den Einstieg und die Fehlersuche in Programmen weiter zu erleichtern, und damit alle Übungen auch am privaten PC durchgeführt werden können, beginnen wir damit, unsere Programme auf einem *simulierten* Prozessor auszuführen.

Beachten Sie die Hinweise am Ende des Blatts, insbesondere, wenn Sie aus vorherigen Veranstaltungen nicht mit dem Erstellen und Testen von Assemblerprogrammen für Atmega-Prozessoren vertraut sind!

Aufgabe 1.1 (Vorbereitung):

Verwenden Sie für die folgenden Übungen als Vorlage die Datei `avr_template.S`, die Sie auf dem Laborserver unter `https://wwwvs.cs.hs-rm.de/lehre/mt20ss/material.html` finden.

a) Kopieren Sie die Vorlage in eine Datei `mpt1a.S` und erstellen Sie daraus mit `avr-gcc` eine Binärdatei `mpt1a` mit Debug-Informationen (mit welchem Flag des `gcc`?).

b) Starten Sie `simulavr` (mit `simulavr-disp`) in einer Shell sowie `avr-gdb` für `mpt1a` in einer weiteren Shell. Verbinden Sie `gdb` mit dem Simulator und laden Sie das Programm. Setzen Sie einen Breakpoint auf `endloop` und starten Sie.

c) Legen Sie sich ein Makefile und ein Startskript für diese Build- und Testschritte an.

d) Versionieren Sie Ihre Quell-, Skript- und Make-Dateien in GitLab.

Aufgabe 1.2 (Das erste eigene Assemblerprogramm):

a) Bilden Sie den folgenden C-Quelltext auf AVR-Assemblercode ab. Verwenden Sie die Register, die durch die C-Variablennamen benannt sind, für Konstanten die Hexdezimalschreibweise und geben Sie jeweils das Ergebnisregister und dessen Inhalt an. Zur Verfügung stehen Ihnen u.a. folgende Instruktionen:

```
add  <Register>,<Register>        (ADD without Carry)
breq <Adresse bzw. Sprungmarke>   (Branch if Equal)
brne <Adresse bzw. Sprungmarke>   (Branch if Not Equal)
eor  <Register>,<Register>        (Exclusive OR)
neg  <Register>                   (Two's Complement)
ldi  <Register>,<Konstante>       (Load Immediate)
subi <Register, Kostante>         (Subtract Immediate)
```

C-Quellcode:

```
unsigned char r18 = 0;
unsigned char r19;
for(r19 = 3; r19 != 0; r19--) {
  r18 = r18 + r19;
}
```

b) Übersetzen Sie Ihr Programmfragment mit `avr-as` und `avr-ld` und testen Sie dann das erzeugte Binärfile mit `simulavr` und `avr-gdb`.

c) In einem AVR-Prozessor liegt in `r18` der Wert `0x03`. Entwerfen Sie Assemblercode, um mit so wenig Instruktionen wie möglich in `r19` den Wert `0x27` zu erhalten. Sie dürfen beliebig viele Register und beliebige Instruktionen verwenden, aber keine weiteren Zahlenkonstanten! Bestimmen Sie auch die Anzahl der für Ihren Code benötigten Zyklen.

Aufgabe 1.3 (Hinweise / Hilfen):

Wenn Sie mit der AVR-Assemblerprogrammierung nicht (mehr) vertraut sind, wiederholen Sie die Übungsblätter 1 und 2 der LV *Hardwarenahe Programmierung I*, zu finden unter `https://wwwvs.cs.hs-rm.de/lehre/pc19ws/material.html`

Die folgende Literatur sollten Sie sich herunterladen und vor dem Bearbeiten der Übung studieren:

- Woitowitz, R.; Urbanski, K.; Gehrke, W.: Digitaltechnik. 6. Aufl. Springer, 2012, darin Abschnitte 9.1.1, 9.1.4, 9.1.7, 9.2.2, 9.2.4, 9.2.5.2, 9.2.5.3. Aus dem Hochschulnetz (VPN!) zu beziehen unter:
  `https://link.springer.com/book/10.1007/978-3-642-20872-0`
- Eine Übersicht der ATmega-Assemblerinstruktionen und deren Arbeitsweise bietet das *Atmel AVR Instruction Set Manual*
  `http://www.atmel.com/images/Atmel-0856-AVR-Instruction-Set-Manual.pdf`

Um die Übungen durchzuführen, sollten Sie auf einem Linux-PC arbeiten und dort die Pakete *gcc-avr*, *binutils-avr*, *simulavr*, *avr-libc*, *gdb-avr* und *gcc-doc* installieren.

Wir werden natürlich versuchen, im Rahmen des Praktikumsbetriebs online Hilfestellung bei Problemen mit der Tool-Umgebung zu leisten.

Sie könenn sich zum Bearbeiten der Aufgaben auch per `ssh` auf einem der PC-Poolrechner einloggen, vorzugsweise ein Rechner des ITS-Labors, deren Hostnamen nach folgendem Muster gestaltet sind: `itsXX.local.cs.hs-rm.de` haben (mit `XX` von 01..16). Solange das ITS-Labor noch nicht wieder betriebsfähig ist, können Sie auf einen regulären Poolrechner aus `lx1-XX.local.cs.hs-rm.de` ausweichen.