FH Bielefeld
University of
Applied Sciences

SoSe 2015
Prof. M. König

Abgabe: KW 20 2015

Aufgabe 5 (10 Punkte) - Assembler

Praktikum - Vorbereitung

Lesen Sie das Kapitel 13 der ATmega Dokumentation [http://www.atmel.com/Images/doc8161.pdf].

Praktikum

Aufgabe 5.1

Lokalisieren Sie in der Arduino Umgebung die folgenden Dateien:

- `wiring_digital.c`
- `wiring.c`

und vollziehen Sie die Funktionsweise der Funktionen `pinMod`, `digitalWrite` (s. `wiring_digital.c`) und `delay` (s. `wiring.c`) nach.

Frage (*2 Punkt*):

- Erklären Sie, wie in den Funktionen `pinMode, digitalWrite` *und* `delay` *auf die Hardware zugriffen wird.*

Aufgabe 5.2

Übernehmen Sie das Listing 2 (Datei `blink.ino` hinterlegt in ILIAS) und laden Sie es auf das Arduino Uno Board hoch.

```
// Arduino Uno PIN 13 is connected to PORTB5 of ATMega328

void simpleDelay(unsigned int f_iterations) {
    // loop with nop for delay
    for (unsigned long i = 0; i < f_iterations; ++i) {
        for (volatile unsigned long j = 0; j < 1000; ++j) ;
    }
}

void setup() {
    // set direction of DDRB5
    uint8_t *portAdd = (uint8_t*) 0x24; // DDRB address
    *portAdd |= 0x20; // set bit 5
}

void loop() {
    // set and clear PortB5
    uint8_t *pinB = (uint8_t*) 0x25; // PortB address
    *pinB |= 0x20; // set bit 5
    simpleDelay(1000);
    *pinB &= 0xDF; // clear bit 5
    simpleDelay(1000);
}
```

Listing 2: Programm “Blink” mit direktem Hardwarezugriff (blink.ino)

Fragen (*½ + ½ + 1 Punkte*):

- *Erklären Sie, wie man anhand der ATmega Dokumentation (s.o) die Adressen von DDRB und PortB (s. Listing 2) ermittelt.*
- *Woran erkennt man im Schaltbild des Arduino Uno [http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf], dass die LED des Board über den PortB5 angesteuert wird?*
- *Beschreiben Sie nun unter Kenntnis der Aufgaben 5.1 und 5.2, wie eine Hardwareabstraktion mit dem Ansatz der Arduino Software erfolgt.*

Aufgabe 5.3 (*2 Punkte*)

Sehen Sie Sich an, wofür die Option `-O2` des Compilers `avr-gcc` verwendet wird, beispielsweise unter http://gcc.gnu.org/onlinedocs/gcc-4.7.0/gcc/Optimize-Options.html.

Setzen Sie den Pfad für ausführbare Programme auf die Tools der (AVR-)Toolchain der Arduino-Entwicklungsumgebung und öffnen Sie ein Fenster mit einer Kommandozeile. Ermitteln Sie Ihre Compiler Version (`avr-gcc --version`). Idealerweise haben Sie die Version 4.3.2.

```
// Arduino Uno PIN 13 is connected to PortB5 of ATMega328P

typedef unsigned char uint8_t; // set type for unsigned int 8

void simpleDelay(unsigned int f_iterations) {
    // loop with nop for delay
    for (unsigned long i = 0; i < f_iterations; ++i) {
        for (volatile unsigned long j = 0; j < 1000; ++j) ;
    }
}

int main() {
    // set direction of DDRB5
    uint8_t *portAdd = (uint8_t*) 0x24; // DDRB address
    *portAdd |= 0x20; // set bit 5

    for (;;) { // infinite loop
        uint8_t *portB = (uint8_t*) 0x25; // PortB address
        *portB |= 0x20; // set bit 5
        simpleDelay(1000);
        *portB &= 0xDF; // clear bit 5
        simpleDelay(1000);
    }
}
```

Listing 3: Programm “Blink” ohne Arduino (blink.cpp)

Speichern Sie lokal das Listing 3 `blink.cpp` (hinterlegt in ILIAS) ab. Lassen Sie die folgenden Schritte in der Kommandozeile laufen:

```
avr-gcc -g -O0 -mmcu=atmega328p -c blink.cpp
avr-gcc -g -O0 -mmcu=atmega328p -o blink.elf blink.o
avr-objdump -h -S blink.elf > blink1.asm
avr-strip blink.o
```

[Sie können das Programm mit den folgenden Schritten auch auf das Board überspielen, dabei entsprechend *PATH_TO und DEVICE anpassen*:

```
avr-objcopy -j .text -j .data -O ihex blink.elf blink.hex
```

`avrdude -C` *`PATH_TO`*`/avrdude.conf -carduino -patmega328p -P`*`DEVICE`* `-b115200 -D -Uflash:w:blink.hex:i ]`

Sie sollten nun die Dateien `blink.o`, `blink.elf` und `blink1.asm` generiert haben. Notieren Sie Sich die Größe der Datei `blink.o`. In der Datei `blink1.asm` finden Sie ein Disassembly des Programms `blink.cpp`.

Gehen Sie zu der zweiten Zeile nach der Zeile mit `for(;;)`. Die Zeile hat die folgende Form (erste drei Zeichen ggf. anders):

```
172: 85 e2      ldi r24, 0x25   ; 37
```

Erklären Sie, was Ihnen das Disassembly in den Spalten zeigt. Erklären Sie weiterhin genau die Anweisungen der Assemblerbefehle dieser und der folgenden Zeilen bis zur Zeile mit `simpleDelay(1000)`*.*

Welche Bedeutung hat gegen Ende des Textes die Zeile (erste drei Zeichen ggf. anders):

```
19a: e5 cf      rjmp .-42       ; 0x172 <main+0x22>
```

Aufgabe 5.4 (*2 Punkte*)

Führen Sie nun Schritte mit der Option `-O2` aus:

```
avr-gcc -g -O2 -mmcu=atmega328p -c blink.cpp
avr-gcc -g -O2 -mmcu=atmega328p -o blink.elf blink.o
avr-objdump -h -S blink.elf > blink2.asm
avr-strip blink.o
```

Vergleichen Sie die notierte Größe mit der Größe der nun generierten Datei `blink.o`*. Vergleichen Sie den Inhalt der Dateien* `blink1.asm` *und* `blink2.asm`*. Wieso unterscheiden sich die Größe der Objektdateien* `.o`*? Welche wesentliche Änderung hat der Compiler am auszuführenden Binärcode (siehe die Disassemblies* `.asm`*) durchgeführt?*

Aufgabe 5.5 (*2 Punkte*)

Ändern Sie nun das Programm, indem Sie das Keyword `volatile` in der Funktion `simpleDelay` löschen. Führen Sie die Schritte der Aufgabe 5.4 nochmals durch und sehen Sie Sich das neue Disassembly an.

Was und warum hat der Compiler nun etwas anderes generiert?