FH Bielefeld
University of
Applied Sciences

Embedded Systems: Aufgabenblatt 5

Prof. M. König

Praktikum 5 (10 Punkte) - Assembler

Die Abgabe der Lösungen erfolgt im Ilias bis zum angegebenen Abgabezeitpunkt. Die Besprechung und Bewertung der **eingereichten** Lösungen erfolgt am folgenden Praktikumstermin.

Vorbereitung

Infomaterial finden Sie u.a. im Ilias:

- ARMv7-M Architecture Reference Manual, Kapitel A4 („The ARMv7-M Instruction Set“) und A5 („Thumb Instruction Set Encoding“)
- Cortex-M4 Devices Generic User Guide, Kapitel 3 „The Cortex-M4 Instruction Set“

Praktikum

Aufgabe 1 *(4 Punkte)*

Sehen Sie Sich an, wofür die Optionen `-O0, -O2` und `-Os` des Compilers `arm-none-eabi-gcc` verwendet werden, z.B. https://gcc.gnu.org/onlinedocs/gcc-6.4.0/gcc/Optimize-Options.html.

Setzen Sie den Pfad für ausführbare Programme auf die Tools der (TivaC-)Toolchain der Energia-Entwicklungsumgebung und öffnen Sie ein Fenster mit einer Kommandozeile. Ermitteln Sie Ihre Compiler Version (`arm-none-eabi-gcc --version`). Idealerweise haben Sie die Version 6.3.1.

Laden Sie die vier Dateien `blink.c, startup_gcc.c, EK-TM4C123GXL.ccxml` und `blink.ld` aus ILIAS. Für diese Aufgabe betrachten wir hauptsächlich die Datei blink.c (s. Listing).

Übersetzen Sie die Dateien und bauen Sie eine ELF-Datei für das TivaC-Board mit Optimierung `-O0`. Ermitteln Sie die Programmgröße und generieren Sie ein Disassembly. Sie können folgende Befehle in der Kommandozeile ausführen (oder sich ein Makefile schreiben). Für die folgenden Befehle ist der `{PATH_TO_SYSTEM_DIR}` durch den Pfad zum TivaC „system“-Ordner in Ihrer Energia-Installation zu ersetzen, z.B. “C:\Users\User\AppData\Local\Energia15\packages\energia\hardware\tivac\1.0.3\system”:

```
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -I{PATH_TO_SYSTEM_DIR}
-std=gnu11 -c -o blink.o blink.c

arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -I{PATH_TO_SYSTEM_DIR}
-std=gnu11 -c -o startup_gcc.o startup_gcc.c

arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -T blink.ld -Xlinker
--gc-sections -o blink.elf blink.o startup_gcc.o

arm-none-eabi-size blink.elf

arm-none-eabi-objdump -h -S blink.elf > blink0.asm
```

```c
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

volatile uint32_t ui32Loop;

void delay() {
   for(ui32Loop = 0; ui32Loop < 200000; ui32Loop++);
}

int main(void) {
   // Enable the GPIO port.
   SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

   // Do a dummy read to insert a few cycles
   ui32Loop = SYSCTL_RCGC2_R;

   // Set up GPIO port.
   GPIO_PORTF_DIR_R = 0x08;
   GPIO_PORTF_DEN_R = 0x08;

   // Loop forever.
   while(1) {
      // Turn on the LED.
      GPIO_PORTF_DATA_R |= 0x08;

      // Delay for a bit.
      delay();

      // Turn off the LED.
      GPIO_PORTF_DATA_R &= ~(0x08);

      // Delay for a bit.
      delay();
   }
}
```

Listing der Datei blink.c

Sie können das Programm mit den folgenden Schritten auch auf das Board überspielen:
`DSLite load -c EK-TM4C123GXL.ccxml blink.elf`

Sie sollten nun die Größe des Programmcodes ermittelt und u.a. die Datei `blink0.asm` generiert haben. In der Datei `blink0.asm` finden Sie ein Disassembly des Programms `blink.c`.

Gehen Sie zu der 10. Zeile nach der Zeile mit `000029c <main>:` Die Zeile hat die folgende Form (erste drei Zeichen ggf. anders):
`2ae:` `4b0d  ldr  r3, [pc, #52]  ; (2e4 <main+0x48>)`

Erklären Sie, was Ihnen das Disassembly in den Spalten zeigt. Erklären Sie weiterhin genau die Anweisungen der Assemblerbefehle dieser und der folgenden 18 Zeilen bis zur Zeile mit
`2da:  e7ee  b.n  2ba <main+0x1e>`

Aufgabe 2 *(4 Punkte)*

Führen Sie nun die oben dargestellten Schritte mit der Option `-O2` aus. Generieren Sie ein weiteres Disassembly in einer Datei `blink2.asm`.

Vergleichen Sie die beiden Größen der Programmcodes, die mit den unterschiedlichen Optimierungsparametern generiert wurden. Vergleichen Sie auch den Inhalt der Dateien `blink0.asm` *und* `blink2.asm`*. Welche wesentliche Änderung hat der Compiler am auszuführenden Binärcode (siehe die Disassemblies* `.asm`*) durchgeführt?*

Aufgabe 3 *(2 Punkte)*

Ändern Sie nun das Programm, indem Sie das Keyword `volatile` im Quellcode `blink.c` löschen. Führen Sie die oben beschriebenen Schritte mit der Option `-O2` nochmals durch und sehen Sie Sich das neue Disassembly an.

Was und warum hat der Compiler nun etwas Anderes generiert? Wie wirkt sich dies auf die Programmausführung auf dem Board aus?