



FACHHOCHSCHULE DARMSTADT

FACHBEREICH INFORMATIK

MIKROPROZESSORPRAKTIKUM

WS2005/06

Termin 1

C-Programmierung für eingebettete Systeme

Lernziele:

Mit den folgenden Versuchen sollen Sie die Sprache “C” einmal aus einer anderen Sicht kennenlernen. An ganz einfachen Programmen sollen Sie ermitteln, welchen Code ein Compiler erzeugt, wo welche Variablen abgelegt werden, wie ein *call by value / reference* in ARM Assembler umgesetzt wird.

Arbeitsverzeichnis:

Sie arbeiten in dem Verzeichnis `/home/milabuser/mi2/Termin1`. Dort stehen die Dateien *Termin1Aufgabe1.c* als Programmgerüstbeispiel und *makefile* zur Verfügung.

```
// Termin1Aufgabe1.c
//*****
// Lösung zu Termin1
// Aufgabe 1
// Namen: _____; _____
// Matr.: _____; _____
// vom: _____
//....
int main (void)
{
// ....
return (0);
}
```

Aufgabe 1:

Legen Sie im C Programm zwei lokale integer Variablen an. Weisen Sie diesen im Code Werte zu, z.B. 0x1 und 0x2. Übersetzen Sie Ihr Programm und schauen Sie sich den erzeugten Code an. Führen Sie das Programm im Simulator aus. Dokumentieren Sie den erzeugten Code.

Aufgabe 2:

Verwenden Sie nun statt der lokalen Variablen globale Variablen. Was ändert sich? Warum?

```
# makefile
#*****
# Quellendatei
FILE = Termin1Aufgabe1
# Optimierungsstufe
OPTI = 1

all:
# Erzeugen des Assemblercode aus der C-Datei
arm-elf-gcc -S -O$(OPTI) $(FILE).c
# Übersetzen und binden der Quelldatei
arm-elf-gcc -g -O$(OPTI) $(FILE).s -o $(FILE).elf
```

Aufgabe 3:

Legen Sie nun zwei globale und zwei lokale integer Variablen an. Weisen Sie den Variablen Werte zu. Machen Sie nun Zuweisungen der Form “global=lokal“. Was stellen Sie fest? Ändern Sie im Makefile die Optimierung und beobachten die Auswirkungen.

Aufgabe 4:

Vereinbaren Sie einen Funktionsprototypen für eine Funktion “addition“, die 3 Integer Parameter erwartet. Rufen Sie diese Funktion im Anschluss an die Zuweisung nach Aufgabe 3 mit den beiden lokalen und einer globalen Variablen auf.

Aufgabe 5:

Ändern Sie den Funktionsprototypen so ab, dass er statt der lokalen Variablen Pointer auf diese Variablen erwartet und rufen Sie die Funktion entsprechend auf. Was ändert sich und warum? Wo liegen die lokalen Variablen nun?