Technische Universität
Clausthal Institut für Informatik
Prof. G. Kemnitz

13. November 2018

Rechnerarchitektur: Laborübung 4 (Float, Kontrollstrukturen)

Hinweise: Schreiben Sie die Lösungen, so weit es möglich ist, auf die Aufgabenblätter. Tragen Sie Namen, Matrikelnummer und Studiengang in die nachfolgende Tabelle ein und schreiben Sie auf jedes zusätzlich abgegebene Blatt ihre Matrikelnummer. Lassen Sie für vorgeführte Experimente vom Betreuer die Punkte auf dem Aufgabenblatt eintragen und geben Sie, wenn Sie fertig sind, alle Blätter ab. Für eine Bescheinigung der erfolgreichen Teilnahme sind in jeder bis auf einer Laborübung mindestens 60% der Punkte zu erreichen.

Name	Matrikelnummer	Studiengang	Punkte von 20	$\geq 60\%$

Aufgabe 4.1: Arbeiten Sie das nachfolgende Programm mit Gleitkommaoperationen im Simulator schrittweise ab:

```
#include <avr/io.h>

float a=41.2513, b=2.573, c=412.96, d, e;

int main(void){
   d=a+b;
   e=c*d;
}
```

a) Ergänzen Sie in der nachfolgenden Tabelle für alle Variablen deren Adressen (hexadezimal), die berechneten Werte gerundet auf zwei Nachkommastellen und die rechnerinterne Darstellung als 4-Byte-Hex.-Zahl[1]: 3P

Variable	Adresse	Soll-Wert	intern (4 Byte)
a		41,25	
b		2,57	
c		412,96	
d			
e			

b) Spalten Sie die 32-Bit-Hex-Werte für die Variablen a und b auf in Vorzeichenbit s, Charakteristik c und Mantisse M. Berechnen Sie daraus nach der Formel in der Vorlesung

$$Z = (-1)^s \cdot (1, M_{-1} \ldots M_{-m}) \cdot 2^{c-0\text{x7F}}$$

die dargestellten Ist-Werte und die Differenz zwischen Soll- und Istwert $Z_{\text{soll}} - Z_{\text{Ist}}$. 2P

Variable	s	c	$1, M$	Z_{Ist}	$Z_{\text{soll}} - Z_{\text{Ist}}$
a					
b					

[1] Die Adressen und Werte lassen sich am einfachsten im Watch-Fenster ablesen (Debug > Windows > Watch > Watch1). Die unter den Adressen der Variablen gespeicherten Byte-Werte sind im Speicher-Fenster ablesbar (Debug > Windows > Memory >Memory1, data IRAM, Adressbereich der Variablen).

c) Probieren Sie aus, welches Ergebnis bei nachfolgender Division entsteht: 1P

```
b = a / 0;
```

Punkte Aufgabe 4.1	

Aufgabe 4.2: Gegeben ist das nachfolgene C-Programm, das die LEDs an Port J im Sekundentakt hochzählen soll:

```
#include <avr/io.h>

int main(void){
 DDRJ = 0xFF;
 uint16_t a;
 uint8_t b;
 while (1){
  PORTJ++;
  for (b=0;b<100;b++){
   // Warteschleife, Sollwartedauer 10ms
   for(a=0;a<10000;a++);
  }
 }
}
```

a) Übersetzen Sie das Programm ohne Compileroptimierung »-O0«. Listen Sie die Maschinenbefehlsfolge auf, die in der innersten Schleife 10.000-mal abgearbeitet wird. Gehen Sie zur Kontrolle im Schrittbetrieb die innerste Schleife durch. (Im Debugger ▶ (Continue, F5), dann ⏸ (Break all, Shift+F5) und dann durchsteppen mit ⬇ (Step Into, F11). Tragen Sie für jeden Befehle der innersten Schleife in die nachfolgende Tabelle ein: Adresse, Befehl und Anzahl der Zyklen[2]: 2P

Adresse (hex)	Maschinenbefehl	Zyklen

b) Die Abarbeitungsdauer für einen Zyklus ist der Kehrwert der Prozessortaktfrequenz

$$t_{\mathrm{Cycle}} = \frac{1}{8\,\mathrm{MHz}} = 125\,\mathrm{ns}$$

Wie lange dauert die Abarbeitung der gesamten Befehlsfolge in der innersten Schleife und auf welchen Wert muss die Anzahl 10.000 der Schleifendurchläufe geändert werden, um die Sollwartezeit von 10 ms möglichst genau anzunähern? Setzen Sie den Werte ein. Laden Sie das Programm auf die Baugruppe und kontrollieren Sie, dass die LEDs an Port J einmal pro Sekunde weiterzählen. 1P

[2]Die Anzahl der Zyklen ist die Anzahl der Maschinentakte, die die Abarbeitung dauert. Sie steht für alle Befehle in der Doku für den AVR-Befehlssatz, zu finden unter http://techwww.in.tu-clausthal.de/site/Dokumentation/ATmega2560/AVR_Instruction.pdf. Die meisten Befehle, z.B. LDD benötigen einen Zyklus, komplexere Befehle wie ADIW benötigen 2 Zyklen. Bedingte Sprünge, z.B. BRCS benötigen bei Sprungausführung zwei und bei Nichtausführung nur einen Zyklus.

c) Übersetzen Sie das Programm als nächstes mit »-O1«. Bestimmen Sie auch hierfür die Befehlsfolge in der innersten Schleife und tragen Sie in die nachfolgende Tabelle für diese Befehle Adresse, Befehl und Zyklusanzahl ein[3]: 2P

Adresse (hex)	Maschinenbefehl	Zyklen

d) Wie lange dauert die Abarbeitung der Befehlsfolge in der innersten Schleife jetzt. Auf welchen Wert muss die Anzahl der Schleifendurchläufe bei Übersetzung mit »-O1« geändert werden, um die Sollwartezeit von 10 ms möglichst genau anzunähern? Setzen Sie den Werte ein und kontrollieren Sie auch hiebei, dass die LEDs an Port J einmal pro Sekunde weiterzählen. 1P

Punkte Aufgabe 4.2	

Aufgabe 4.3: Das nachfolgende Programm zählt für die im Feld »dat« übergebenen Eingabewerte in Zähler »ct1« die Anzahl der Werte kleiner/gleich acht und im Zähler »ct2« die Werte größer 15:

```
#include <avr/io.h>

int8_t dat[6] = {-3, 7, 109, 17, 55, -16};

int main(void){
 register uint8_t idx, ct1=0, ct2=0;
  for (idx=0; idx<6; idx++){
   if (dat[idx]<= 8){
    ct1++;
   }
   else if (dat[idx]>15) {
    ct2++;
   }
  }
}
```

Es soll ohne Compileroptimierung (-O0) übersetzt und im Simulator untersucht werden:

a) Auf welchen Adressen bzw. in welchen Registern werden die vereinbarten Variablen gespeichert? 1P

b) Wie bildet der Compiler die beiden Increment-Operationen »ct1++« und »ct2++« nach. Ergänzen Sie hierzu die Befehlsfolgen in den nachfolgenden Tabellen und überprüfen Sie im Schrittbetrieb, dass tatsächlich die Register mit den Variablen »ct1« und »ct2« um eins erhöht werden. 2P

- »ct1++«:

Adresse	Befehl	ausgeführte Operation

- »ct2++«:

[3] Mit »-O1« wird das Programm nicht mehr anweisungsweise übersetzt. Die innerste Schleife findet man am einfachsten, wenn man im Schrittbetrieb ausprobiert, welche Befehlsfolge immer wieder durchlaufen wird.

Adresse	Befehl	ausgeführte Operation

c) Mit welchen Maschinenbefehlen wird die Schleife

```
for (idx=0; idx<6; idx++){...}
```

realisiert? Tragen Sie die Befehle, deren Adressen und die ausgeführten Operationen (inc. Sprungbedingungen und Sprungziele) in die nachfolgende Tabelle ein und kontrollieren Sie Ablauf und Operationsausführung im Schrittbetrieb: 2P

- Befehlsfolge am Schleifenanfang (Zähler initialisieren):

Adresse	Befehl	ausgeführte Operation

- Befehlsfolge am Schleifenende (Zähler erhöhen und bedingter Sprung zum Beginn des Schleifenkörpers):

Adresse	Befehl	ausgeführte Operation

d) Mit welcher Befehlsfolge wird in der Anweisung

```
if (dat[idx]<= 8)
```

der Wert »dat[idx]« in ein Register geladen und in welches Register? Tragen Sie die Befehle, deren Adressen und die ausgeführten Operationen in die nachfolgende Tabelle ein und kontrollieren Sie die Operationsausführung im Schrittbetrieb: 2P

Adresse	Befehl	ausgeführte Operation

e) Mit welcher Befehlsfolge wird in der Anweisung

```
if (dat[idx]<= 8){ct1++;}
```

die bedingte Increment-Operation »ct1++«, wenn der Wert »dat[idx]« größer 8 ist, übersprungen? Tragen Sie die Befehle, deren Adressen und die ausgeführten Operationen (inc. Sprungbedingungen und Sprungziele) in die nachfolgende Tabelle ein und kontrollieren Sie Ablauf und Operationsausführung im Schrittbetrieb: 1P

Adresse	Befehl	ausgeführte Operation

Punkte Aufgabe 4.3	