Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Hagen Peters, Prof. Dr.-Ing. Norbert Luttenberger, Miro Spönemann

Institut für Informatik

2. Übung zur Vorlesung „Organisation und Architektur von Rechnern“

Integer Daten

Abgabe am Montag, 25. April – 16:00

Hinweise:

- Der Termin für die Abgabe Ihrer Lösungen ist immer Montags 16 Uhr, die neuen Aufgaben erhalten Sie am Montag Nachmittag.
- Nach dem Verstreichen des Abgabezeitpunkts ist keine Abgabe mehr möglich; in diesem Fall senden Sie die Lösungen per Email an Ihren Übungsgruppenleiter und rechnen Sie mit deutlichem Punktabzug. Sie können aber eine bereits abgegebene Lösung bis zum Abgabezeitpunkt verändern.
- Bei Programmieraufgaben geben Sie immer sinnvoll kommentierte und strukturierte Quelltextdateien ab. Bei Abgabetyp *code* ist deren Inhalt in die entsprechenden Textfelder zu kopieren; bei Typ *upload* sind die Dateien selbst hochzuladen.
- Geben Sie außerdem nur Programmcode ab, der sich erfolgreich kompilieren lässt. Testen Sie dies vorher!
- Falls in der jeweiligen Aufgabe nicht anders angegeben, kann es auch Punktabzug geben, wenn der Compiler Warnungen anzeigt. Lassen Sie mit der Compileroption `-Wall` alle Warnungen anzeigen.

Aufgabe 1 - Quadrate

3 Punkte

Für zwei Variablen `x` und `y` vom Typ `int` betrachten Sie folgenden C-Code.

```
y = x * x;
```

Schreiben Sie ein Programm, das den größten `int`-Wert `x` bestimmt, für den `y` größer als 0 ist.

Aufgabe 2 - Bit-Darstellung

3 Punkte

Ergänzen Sie das unten stehende Programm um eine Funktion

```
int create_int(int bits[])
```

welche einen Bit-Vektor in die entsprechende Integer-Zahl im 2-Komplement umrechnet. Der Bit-Vektor enthält 32 einzelne Bit-Werte (d.h. jeder Wert ist 0 oder 1), wobei der Eintrag mit der niedrigsten Wertung den Index 0 und der mit der höchsten Wertung den Index 31 hat.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int bits[32];
    int i, x;
    // initialize random seed
    srand(time(NULL));

    // generate a random bit vector
    for (i = 31; i >= 0; i--) {
        bits[i] = rand() % 2;
        printf("%d", bits[i]);
    }

    // create and print an integer from the bit vector
    x = create_int(bits);
    printf(" = %d\n", x);
    return 0;
}
```

Das Programm erstellt einen zufälligen Bitvektor, wandelt diesen um und gibt das Ergebnis auf der Konsole aus.

Aufgabe 3 - Byte-Reihenfolge

4 Punkte

(a/b) Schreiben Sie eine Funktion `is_little_endian()`, die ermittelt welche Byte-Reihenfolge Ihr Rechner benutzt. Die Funktion soll **1** zurückgeben, wenn sie auf einer *little endian* Architektur übersetzt und ausgeführt wird, und **0** wenn sie auf einer *big endian* Architektur übersetzt und ausgeführt wird.

- Ihre Funktion sollte unabhängig von der verwendeten Wortlänge funktionieren.
- Teilen Sie Ihr Programm in zwei Dateien:
  - Teil **a**: `is_little_endian.c` enthält die Funktion `is_little_endian()`
  - Teil **b**: `is_little_endian_main.c` enthält eine einfache *main*-Funktion, welche das Ergebnis von `is_little_endian()` ausgibt.
- Sie können das Programm in die ausführbare Datei `is_little_endian` übersetzen, indem Sie dem Compiler beide Quelldateien übergeben:

  ```
  gcc -o is_little_endian is_little_endian.c is_little_endian_main.c
  ```

(c) Testen Sie das Programm auf Architekturen mit unterschiedlicher Byte-Reihenfolge, z.B. auf den Informatik-Servern `elrond` und `bilbo`, und geben Sie die jeweiligen Programmausgaben ab.

Wenn Sie sich an einem Terminal im Grundausbildungspool einloggen, sind Sie bereits auf dem Server `bilbo`. Von außerhalb können Sie sich per SSH auf einen der Informatik-Server einloggen:

```
ssh <username>@bilbo.informatik.uni-kiel.de
```

Dabei steht `<username>` für Ihren Login-Namen des Informatik-Accounts. Um Dateien auf den Server zu kopieren verwenden Sie `scp`. Die aktuelle Architektur erfahren Sie auch mit dem Kommando `uname -a`.