Technische Informatik
Dr. Henrik Brosenne
M. Sc. Maria Kosche

Wintersemester 2023/24
Georg-August-Universität Göttingen
Institut für Informatik

Übungsblatt 3

Maschinensprache Fort.

Hinweise zur Abgabe der Lösungen:

Die Abgabe von Lösungen zu allen Übungsblättern ist grundsätzlich freiwillig.

Die Bearbeitungen der Aufgaben können Sie freiwillig und in geeigneter Form in der Stud.IP-Veranstaltung über das **Vips-Modul** zum entsprechenden Aufgabenblatt hochladen. Sie erhalten dann entsprechend ein kurzes Feedback zu Ihrer Abgabe. Sie können Ihre Bearbeitungen gerne mit LaTeX formatieren, es ist aber auch der direkte Upload von Text oder der Upload von Text- und Bilddateien in gängigen Formaten möglich.

Die Aufgaben können auch in Kleingruppen bearbeitet und abgegeben werden. Wenn Sie nicht alleine abgeben, achten Sie bitte darauf, dass Sie sich selbstständig **vor** der Abgabe im Vips-Modul in einer sogenannten Übungsgruppe zusammenschließen. Hierzu können Sie sich in einer der dort vorhandenen Übungsgruppen gemeinsam eintragen. Nur so erhalten alle Zugang zu Feedback und Kommentaren zu der entsprechenden Abgabe.

Am Ende von Übungsblatt 1 finden Sie Hinweise zu den **Pseudocode-Konventionen** für die Übungen.

Aufgabe 1 – Zahlendarstellung

Geben Sie für die folgenden Zahlen die entsprechende Darstellung an.

1. 23 → binär im Zweierkomplement mit 8 Bit
2. -23 → binär im Zweierkomplement mit 8 Bit
3. 23,3 → binär als Gleitkommazahl im IEEE 754 Format mit einfacher Genauigkeit (geben Sie dabei an, was Vorzeichen, Exponent und Mantisse sind)
4. 01000001101110100110011001100110 (IEEE 754 Format) → als Dezimalzahl

Aufgabe 2 – Assembler lesen

Der folgende RV32I-Code verwendet `printf` aus der C-Standard-Bibliothek. Beim Bauen kann anstelle des bisher bekannten Linker-Befehls, das gcc Kommando wie folgt verwendet werden, sodass automatisch mit der C-Standard-Bibliothek gelinked wird. Die Werte für `outputfilename` und `objectfile` müssen, wie bisher vom `ld`-Befehl bekannt, entsprechend angepasst werden.

```
gcc -o outputfilename objectfile.o
```

Der `printf` Befehl nimmt in Register `a0` einen Format-String entgegen und gibt diesen aus. Dabei werden die Vorkommen von `%d` in der Reihenfolge, in der sie im Format-String vorkommen, durch Integer-Werte aus den Argumentregistern `a1-a7` ersetzt. Im Format-String können auch andere Platzhalter als `%d` verwendet werden. Dies und weitere Details können in der Dokumentation zu `printf` nachgelesen werden.

Beschreiben Sie, was der folgende RV32I-Code tut und bilden Sie ihn als Pseudocode nach. Dabei können Sie davon ausgehen, dass `printf` bereits definiert ist und ohne weiteres als Funktion verwendet werden kann. Der Assembler-Code befindet sich auch auf Stud.IP unter Dateien.

```
1   .globl main
2
3   .data
4   format:     .string "%d\n"
5
6   .text
7   main:
8       li      a0, 5
9       jal     foobar
10      mv      a1, a0
11      la      a0, format
12      call    printf
13      li      a0, 0
14      li      a7, 93
15      ecall
16
17  foobar:
18      addi    sp, sp, -8
19      sw      x8, 8(sp)
20      sw      s0, 4(sp)
21      addi    x8, sp, 8
22      li      s0, 0
23  foo:
24      beqz    a0, bar
25      add     s0, s0, a0
26      addi    a0, a0, -1
27      j       foo
28  bar:
29      mv      a0, s0
30      lw      s0, 4(sp)
31      lw      x8, 8(sp)
32      addi    sp, sp, 8
33      ret
```

Aufgabe 3 – Assembler, Rekursion

Bilden Sie die folgende rekursive Funktion in Pseudocode zur Bestimmung des Maximums einer Liste von `int32` Zahlen in Assembler nach. Dabei ist `first` die Adresse der ersten und `last` die Adresse der letzten Zahl der Liste.

```
int max(pointer first, pointer last) {
        if (first == last)
        return load32(first);

        int32 mid = (last-first)/2;

        int32 left = max(first, first+mid);
        int32 right = max(first+mid+1, last);

        if (left > right)
        return left;

        return right;
}
```

1. Beginnen Sie mit der Berechnung für eine Wertübergabe in den Registern `a1` und `a2` und Ergebnisrückgabe im Register `a0`.
2. Bauen Sie den Funktionsrahmen entsprechend der Anforderungen aus Aufgabenteil 1 auf.
3. Schreiben Sie ein Rahmenprogramm, das die Funktion benutzt.

Aufgabe 4

In dieser Aufgabe sollen Sie eine Funktion `Letters` schreiben, welche die Adresse eines Strings als Argument bekommt und die addierte Anzahl der Vorkommen von `'s'` und `'S'` in dem übergebenem String zurückgibt. Wenn ein leerer String übergeben wird, soll `-1` zurückgegeben werden.

Für die Abgabe reicht es, das Programm aus Aufgabenteil 3 abzugeben.

1. Schreiben Sie die beschriebene Funktion als RV32I-Code.
2. Welches Register wird verwendet, um das Argument zu übergeben und welches Register wird für den Rückgabewert verwendet?
3. Passen Sie den folgenden Programmcode an, indem Sie die nötigen Assembleranweisungen hinzufügen, um die Funktion aus Aufgabenteil 1 aufzurufen. Fügen Sie auch, wie in Aufgabe 2 vorgestellt, Code hinzu, der das Ergebnis der Funktion `Letters` für den übergebenen String in `input` ausgibt.

```
.globl main
.data
        input:  .string "Nobody expects the Spanish Inquisition"
.text
main:
```