Übungsblatt 8

8.1 Memory Alighment

Berechnen Sie die Memory Adressen alle Variabeln mit die annahme das memory Sigment .data start addresse ist 0x10000000:

```
.data
x: .byte 10
y: .word 0xf323
.space 1
.align 3
z: .byte 24
a: .half 14
str: .space 100
c: .word 10
```

8.2 Unterprogramme, System Calls

Es soll ein MIPS32-Programm geschrieben werden, das die Ein- und Ausgabe mit System Calls realisiert und eine Funktion ncstr (Unterprogramm) aufruft.

1. Erstellen Sie ein Unterprogramm, das folgende C-Funktion realisiert:
   `int ncstr(char *str, char c)`
   Die Routine soll in einem C-String an der Adresse `str` die Anzahl der vorkommenden Zeichen `c` zählen und zurückgeben. Beachten Sie die Konventionen zur Übergabe von Argumenten und Rückgabewerten sowie zur Benutzung von Registern! Auf Daten darf nur über die der Funktion übergebenen Argumente zugegriffen werden.

2. Erstellen Sie ein Programm zur Ein- und Ausgabe der Daten, das Ihre Funktion `ncstr` aus Aufgabenteil 1 mit den entsprechenden Argumenten aufruft. Das Programm soll nacheinander:

   a) dazu auffordern, eine Zeichenkette einzugeben,

   b) die Zeichenkette einlesen,

   c) dazu auffordern, ein einzelnes Zeichen `c` einzugeben,

   d) das Zeichen `c` einlesen,

   e) die Funktion `ncstr` mit den entsprechenden Argumenten aufrufen,

f) die Anzahl der Zeichen `c` in der Zeichenkette ausgeben und

g) fragen, ob das Programm noch einmal ausgeführt werden soll.

Ein Dialog könnte etwa so aussehen:

```
Type a string>Mississippi
Type a char>i
4 occurrence(s)
Type a string>Main
Type a char>m
0 occurrence(s)
```

Für die Ausgabe der Prompts und der Anzahl der Zeichen in der Zeichenkette, sowie das Einlesen der Zeichenkette und des einzelnen Zeichens können Sie die System Calls 4, 1, 8 und 12 benutzen. Für die Abfrage, ob das Programm beendet werden soll, kann auch der `ConfirmDialog` benutzt werden, System Call 50.

Hinweis

☞ Sie dürfen nur die Pseudo-Instruktion `la` verwenden.

8.3 Unterprogramme, System Calls

Ein typisches Programm, wie z.B. 'Word.exe' von Microsoft, liegst ja in seiner Binärform vor, d.h. die Datei beinhaltet nur Maschineninstruktionen für die entsprechende CPU.

Diese können wieder in die Assemblerbefehle 'zurückgerechnet' werden und so kann man den Programmablauf verfolgen ('Word.exe' ist hierfür ein schlechtes Beispiel). Die folgenden vier Instruktionen sind MIPS-Maschinencode:

```
1 0x2008ffff
2 0x00a84004
3 0x00881024
4 0x03e00008
```

- Wandeln Sie mittels der MIPS Manuals die Instruktionen in MIPS-Assemblercode um.
- Denken Sie daran, die Operanden in der korrekten Reihenfolge abzubilden.
- Mit dem nun erzeugte Assemblercode, erstellen Sie das zugehörige MIPS-Programm und überprüfen Sie, ob in der Tag der gleiche Binärcode vorliegt.
- Was wird hier berechnet?