Übungsblatt 2

Ü 2.1 MIPS – Grundlagen

Machen Sie sich mit den im Moodle zur Verfügung gestellten Unterlagen zum MIPS-Assembler vertraut und beantworten Sie folgende Fragestellungen:

(a) Erklären Sie die übliche Speicheraufteilung der MIPS-Architektur. Wo werden lokale Variablen, globale Variablen bzw. der auszuführende Programmcode abgelegt?

(b) Was versteht man unter I-, R- bzw. J-Befehlen? Erklären sie deren Befehlsformat anhand von drei Befehlen ihrer Wahl.

(c) Welche „General Purpose Register" stehen Ihnen in der MIPS-Architektur zur Verfügung? Nach welchen Konventionen sollten diese Register verwendet werden?

Ü 2.2 MIPS – Instruktionen

(a) Geben Sie den Maschinencode für folgende Instruktion in Hexadezimaldarstellung an, und erklären Sie die Bedeutung der einzelnen Bitfelder des Maschinencodes:

```
addi $s0, $s1, 16
```

(b) Was ist der Unterschied zwischen den Befehlen *shift right arithmetic* (`sra`) und *shift right logical* (`srl`)? Warum gibt es keinen Befehl *shift left arithmetic*?

Ü 2.3 MIPS – Ein-/Ausgabe, Verzweigungen

Schreiben Sie ein MIPS-Assembler-Programm, welches von der Tastatur Werte für die Seitenlängen eines Dreiecks (a, b, c) einliest. Sie können davon ausgehen, dass nur positive, ganzzahlige Werte eingegeben werden und c der größte Wert ist. Das Programm soll überprüfen, ob die Dreiecksungleichung gilt (c <= a + b) und eine entsprechende Meldung ausgeben: „Dreiecksungleichung GILT" oder „Dreiecksungleichung gilt NICHT". Testen Sie Ihr Programm mit dem MIPS-Simulator MARS (siehe Moodle) – dies gilt auch für alle folgenden Assembler-Aufgaben!

Hinweis: Die Ein-/Ausgabe in MARS wird über sogenannte Systemaufrufe (System Calls) realisiert. Beachten Sie, dass die Werte in bestimmten Registern ($t0-$t9, $a0-$a3, $v0-$v1) während der Ausführung von Systemaufrufen überschrieben werden können.

Ü 2.4 MIPS – ALU Operationen, Schleifen

Schreiben Sie ein MIPS-Assemblerprogramm, das von der Tastatur eine ganze, vorzeichenbehaftete Zahl (32-Bit Integer, wird intern als Zweierkomplement dargestellt) einliest und als Ergebnis dessen Hexadezimaldarstellung am Bildschirm ausgibt (ohne Zuhilfenahme des in MARS vorhandenen Systemaufrufs Nr. 34).

Beispiele: Eingabe: -2 → Ausgabe: 0xFFFFFFFE, Eingabe: 15 → Ausgabe: 0x0000000F

Ü 2.5 MIPS – Schleifen, Arrays

Gegeben sei folgendes Datensegment mit einem Array `arr` ganzzahliger, positiver Werte und einer Variablen `arrSize`, welche die Arraygröße angibt:

```
.data
arr:    .word 17 23 64 1 37 68 128 10 16 256
arrSize:.word 10
```

Erstellen Sie ein MIPS-Assemblerprogramm, welches für jedes Arrayelement am Bildschirm ausgibt, ob es sich um eine Zweierpotenz handelt oder nicht. Testen Sie Ihr Programm mit unterschiedlichen Werten und Arraygrößen.

Ü 2.6 MIPS – Zeichenketten

Erstellen Sie ein MIPS-Assemblerprogramm, das eine Zeichenkette von der Tastatur einliest, alle Kleinbuchstaben der Zeichenkette durch die entsprechenden Großbuchstaben ersetzt und das Ergebnis am Bildschirm ausgibt. Sie können davon ausgehen, dass die Länge der Zeichenkette nicht größer als 100 (Zeichen) ist.

Beispiel: Eingabe „32 Register!“ → Ausgabe „32 REGISTER!“

Hinweis: Beachten Sie, wie Groß- und Kleinbuchstaben im ASCII Code angeordnet sind.