

Übungsblatt 9 (Block C – 1)

(16 Punkte)

Abgabe bis spätestens Mittwoch, 11. Januar 2017, 16:00 Uhr.
Besprechung ab Montag, 16. Januar 2017.

Um die Aufgaben zu lösen, sollten Sie den in der Vorlesung vorgestellten MARS Simulator installieren und mit ihm arbeiten. Sie finden die Software unter:

<http://courses.missouristate.edu/KenVollmar/MARS/index.htm>

Installieren Sie den Simulator auf Ihrem Rechner.

Das gesamte RS Team wünscht Ihnen frohe Weihnachten und einen guten Rutsch ins neue Jahr!

9.1 Assemblerprogrammierung (4 Punkte)

In einer Assemblerprozedur soll die folgende Register-Transfer-Anweisung durchgeführt werden:

```
Speicher[0x10010000] := (Speicher[0x10010004] - Speicher[0x10010008] + 8)
                      * Speicher[0x1001000C] + Speicher[0x10010010]
```

Vervollständigen Sie die nachfolgende Sequenz bis einschließlich zum Programmende. Die Verwendung von lw- und sw-Befehlen mit Offset-Werten, die nicht mit 16 Bit dargestellt werden können, ist nicht zulässig. Verwenden Sie möglichst wenige Register und möglichst wenige Befehle!

```
li    $2, 0x10010000 # vorgegeben: Reg[2] := 0x10010000
lw    $3, 0x4($2)   # vorgegeben: Reg[3] := Speicher[0x10010004]
lw    $4, 0x8($2)   # vorgegeben: Reg[4] := Speicher[0x10010008]
```


9.2 Einfache Fallunterscheidung (4 Punkte)

Ein MIPS-Programm soll bei der Auswertung der Klausurkorrekturen helfen. Es soll feststellen, ob jemand bestanden hat (Punkte ≥ 40) oder nicht (Punkte < 40).

Zunächst ist die Variable „Bestanden“ mit 99 belegt, was soviel bedeutet wie „die Note steht noch nicht fest“. Ergänzen Sie das Programmfragment so, dass in „Bestanden“ entweder eine 1 für bestanden oder eine 0 für „nicht bestanden“ steht.

```
.data
Punkte: .word 42      # Klausurpunkte
Bestanden: .word 99   # 0 Nein, 1 Ja, 99 weiß nicht
Grenze: .word 40      # Bestehensgrenze 40 Punkte

.text
.globl main

.
.
.
```

9.3 Fehlersuche (4 Punkte)

Sie haben ein Programm zur iterativen Berechnung der Fakultät erhalten. Leider haben sich 6 Fehler (syntaktische und semantische) eingeschlichen, die Sie finden und beseitigen sollen. Die Anzahl der Programmzeilen soll dabei erhalten bleiben.

Hinweis: Definition der Fakultät: $n! = n \cdot (n-1) \cdot (n-2) \cdots 1$ und $0! = 1$

- Geben Sie die Fehler und das korrigierte Programm an.
- Das Ergebnis in Register 3 soll in einem anderen Programmteil als Zweierkomplementzahl interpretiert werden. Wie groß (dezimal) darf die Eingabe „ein“ sein, damit Register 3 als richtiges Ergebnis verwendet werden kann?

```
.data
ein: .word 5 # Eingabewert vom User (z.B. 5! = 120 = 0x78)
erg: .word 1 # Ergebnis für den User (am Programmende)

.text
.global main
main:
    lw $2, ein      # Eingabe 'holen'
    li $3, 0        # vorbelegen, in $3 könnte ja sonstwas stehen
    bneq $2, $0, fertig # 0! gibt keine Schleife
jump:
    mul $3, $3, $2   # mul erg mit zähler
    subi $2, 1       # zählen
    bgt $2, 1, jump  # Schleifenende, mul mit 1 muss nicht sein
fertig:
    sw erg, $3       # Fakultät nach Berechnung in erg
    li $2, $10       # Programmende
    syscall
```

9.4 Assemblerprogrammierung (4 Punkte)

Implementieren Sie eine Berechnung des Paritätsbits einer Variablen „wert“ (Typ `.word`).

Ein Paritätsbit gibt an, ob die Anzahl von 1-Bits einer Variablen gerade oder ungerade ist. Für eine gerade Anzahl von 1-Bits soll der Wert 0 zurückgegeben werden, bei ungerader Anzahl der Wert 1.

Das ermittelte Paritätsbit soll in der vorgegebenen Variablen „pari“ (Typ `.word`) abgelegt werden.

Beispiel: Für die Zahl $(23)_{10} = (0010111)_2$ ergibt sich 0 als Paritätsbit und für die Zahl $(38)_{10} = (0100110)_2$ ergibt sich 1 als Paritätsbit.

Hinweise: Sie benötigen keinen Multiplikationsbefehl. Der Befehl „srl“ könnte von Nutzen sein (siehe Skript¹ S. 13).

```
.data
wert: .word 38          # Wertebeispiel (ergibt ungerade Parität, also eine 1).
pari: .word 0          # Zu berechnende Parität.

.text
.globl main
main: lw    $2, wert    # Beispielwert in R2

.
.
.
```

Hinweise:

Die Abgaben sollen bis Mittwoch, 11. Januar 2017, 16:00 Uhr in die Briefkästen in der Otto-Hahn-Straße 12 eingeworfen werden.

Die Briefkästen finden Sie in der ersten Etage der Otto-Hahn-Straße 12 am Übergang zum Erdgeschoss der Otto-Hahn-Straße 14. Die Briefkästen sind mit dem Namen der Veranstaltung, der Gruppennummer sowie Zeit und Ort der Übung gekennzeichnet. Für Rechnerstrukturen sind dies die Briefkästen mit den Nummern 28 bis 40.

Schreiben Sie unbedingt Ihren **Namen**, Ihre **Matrikelnummer** und Ihre **Gruppennummer** rechts oben auf Ihre Abgabe. Sie dürfen als Team mit bis zu zwei weiteren Personen abgeben. Geben Sie dann nur eine einzige Lösung ab und schreiben Sie alle Namen und Matrikelnummern des Teams auf die gemeinsame Abgabe.

Heften Sie die Abgabe bitte zusammen (Tacker oder notfalls Büroklammer). Bitte die Abgabe **nicht falten** und **keine Schnellhefter oder Umschläge** abgeben.

Es gibt insgesamt 12 Übungsblätter, die in 3 Blöcke (A, B, C) aufgeteilt sind. In jedem Block müssen Sie 30 Punkte von 64 möglichen Punkten erreichen, um zur Prüfung zugelassen zu werden.

¹<http://ls12-www.cs.tu-dortmund.de/daes/media/documents/teaching/courses/ws1314/rs/script/rs2.pdf>