TI II: Rechnerarchitektur SS 2007 Übungsblatt Nr. 2

Prof. Dr.-Ing. Jochen Schiller, AG Technische Informatik, Freie Universität Berlin

Ausgabe am 14.05.2007 — Abgabe spätestens 25.05.2007, 12:00 Uhr

Bitte bei der Abgabe beide Namen/Matr.Nr. der Mitglieder einer Gruppe, NUMMER DER ÜBUNG/TEILAUFGABE und DATUM auf den Lösungsblättern **nicht vergessen**! Darauf achten, dass die Lösungen beim richtigen Tutor/der richtigen Tutorin abgegeben werden.
Zu spät abgegebene Lösungen werden nicht mehr berücksichtigt!

---

1. Aufgabe: Leistung (2.5 Punkte)

Nehmen wir an für die Ausführungszeit wurden folgende Werte gemessen:

Programm	Computer A	Computer B
1	2s	5s
2	5s	2s

Sie die folgenden Aussagen wahr oder falsch?

- A ist im Hinblick auf Programm 1 schneller als B?
- B ist im Hinblick auf Programm 2 schneller als A?
- A ist bei einer Last, bei der Programm 1 und 2 gleich häufig ausgeführt werden, schneller als B.
- A ist bei einer Last, bei der Programm 1 doppelt so häufig ausgeführt wird wie Programm 2, schneller als B.
- A ist bei einer Last, bei der Programm 2 halb so häufig ausgeführt wird wie Programm 1, langsamer als B.

2. Aufgabe: Zahlendarstellung (8 Punkte)

Berechnen Sie die Ergebnisse (Ergebnis zur Basis 10 angeben):

- $0x45 + 13444_8 - 10001110_2 + 11110_{10}$
- $111011 * 2^2 + 11110 * 2^4 – 101 * 2^2 * 11 * 2^1$
- $11010100000_2 * 2^{-3} + 332545_8 * 8_{10} – 0xFEEF$
- $1101.1001_2 + 11.11_2 + 111.11100_2 * 2^{-3}$

Achtung! Der Rechenweg muss bei allen Aufgaben stets erkennbar sein.

3. Aufgabe: MIPS Assembler 1 (4 Punkte)

Schreiben Sie ein Programm, das die Fibonacci-Zahlen berechnet. Benutzen Sie hierfür MIPS Assembler Instruktionen. Im Register `$t0` ist angegeben, die wievielte Zahl ausgegeben werden soll. Das Ergebnis soll in Register `$t1` stehen und das erreichen des Labels `end`: das Ende des Programms verdeutlichen.

4. Aufgabe: MIPS Assembler 2 (6 Punkte)

Was berechnet das folgende Programm? Dokumentieren Sie dazu zeilenweise und/oder geben Sie den Algorithmus in Pseudocode an. Wir gehen von 8 Bit breiten Registern aus. Die Eingaben befinden sich in den Registern $t1 und $t2. Hinweis: Der Algorithmus steht (bis auf eine minimale Änderung) im Skript!

```
init:
     addi $t0, $zero, 0         #
     addi $t7, $zero, 8         #
     addi $t4, $zero, 1         #

label1:
     beq  $t7, $zero, end       #
     andi $t3, $t1, 1           #
     bne $t3, $t4, label2       #
     add $t0, $t0, $t2          #
label2:
     jal shift
     j label1

end:                            # hier wird nur noch der MIPS angehalten
li   $v0, 10                    # system call for exit
     syscall                    # hier anhalten; wir sind fertig


shift:                          #
     andi $t3, $t0, 1           #
     srl  $t0, $t0, 1           #
     srl  $t1, $t1, 1           #
     bne  $t3, $t4, label3      #
     addi $t1, $t1, 128         #
label3:
    subi $t7, $t7, 1            #
    jr $ra                      #
```

5. Aufgabe: Extremer RISC-Prozessor (10 Punkte)

Ein ganz einfacher RISC-Prozessor habe nur den einen Befehl `DBNZ R, L` mit der Bedeutung „Decrement and Branch if Not Zero“ wobei zunächst das Register `R` dekrementiert werden soll und anschließend nur dann ein Sprung zur Marke `L` ausgeführt wird, wenn `R` ungleich Null ist. Das Zahlenformat des Prozessors seien Integer im 2er-Komplement. Für die folgenden Aufgaben können Sie neu definierte Befehle als Makros wieder verwenden. Weiterhin steht ihnen eine beliebige Zahl an Registern zur Verfügung, welche jedoch initial einen nicht weiter definierten Wert enthalten. Geben Sie für folgende Befehle möglichst optimale Makros an:

a) `CLEAR R` – Setzt das Register `R` auf den Wert 0 (1 Punkt)
b) `GOTO label` – Führt einen Sprung zur durch *label* markierten Adresse aus. (1 Punkt)
c) `NEG R, S` – Schreibt den negierten Wert von R in S (1 Punkt)
d) `SUB R, S` – Subtrahiert `R` von `S`, wobei das Ergebnis am Schluss in `S` steht (2 Punkte)
e) `ADD X, Y, Z` – Addiert `X` und `Y` und schreibt das Ergebnis in `Z` (2 Punkte)
f) `MOV A, B` – Verschiebt den Inhalt von A nach B (1 Punkt)
g) `COPY A, B` – Kopiert den Inhalt von `A` nach `B` (`A` muss nach der Operation seinen alten Wert haben; nicht erneut `MOV` angeben!) (2 Punkte)