Technische Informatik 1 - Übung 1

David Gugelmann
gugdavid@ee.ethz.ch

19. Oktober 2007

Organisatorisches

Eure Hilfsassistenten

- Diego Adolf, dadolf@ee.ethz.ch
- David Gugelmann, gugdavid@ee.ethz.ch

Vorlesungshomepage

- http://www.tik.ee.ethz.ch/tik/education/lectures/TI1/current/

Ablauf der Theorieübungen

zu Hause vor den Übungen...

- Aufgaben anschauen, Fragen notieren
- mit Lösen beginnen

in den Übungsstunden...

- Wir geben einige wichtige Hinweise
- Selbstständiges Lösen und Fragen stellen
- Wir präsentieren Lösungen

MIPS Instruktionsübersicht

- dkrizanc.web.wesleyan.edu/courses/231/07/mips_instructions.pdf

Funktionsaufrufe in MIPS

Problem

- Es gibt nur eine beschränkte Anzahl Register
- Man kann nicht wissen, ob man Inhalt eines Registers überschreiben darf, oder ob das Register von der aufrufenden Instanz noch gebraucht wird

Lösung

- Man sichert den Inhalt von Registern auf den Stack ab (liegt im Hauptspeicher)
- Es gibt eine Konvention, welche Register die aufrufende Fkt. sichert und welche die aufgerufene Fkt.

Funktionsaufrufe in MIPS

Stack

- Wächst von oben nach unten (also von hoher zu tiefer Adresse $\rightarrow$ leere Einträge bei tiefer Adresse)
- Adressangabe in Bytes
- Der Stack Pointer ($sp) zeigt auf den untersten benutzten Platz auf dem Stack
- Der $sp muss somit vor jedem Schreiben dekrementiert werden
- Die aufgerufene Fkt. darf **nie** oberhalb der Adresse des ursprünglichen $sp schreiben (also oberhalb ihres $fp)
- Die aufgerufene Fkt. muss beim Verlassen den $sp wieder auf den ursprünglichen Wert setzen

Beispiel für Funktionsaufruf (nach Skript, slide 3-12)

```
# void swap(int v[], int k): tauscht k. mit (k+1). Element

swap: addi $sp,$sp,-8 # $sp fuer 2 Register dekrementieren
sw $fp,4($sp)   # fp wird in swap geaendert -> absichern
sw $s0,0($sp)   # $s0 wird in swap geaendert -> absichern
addi $fp,$sp,4  # fp zeigt nun auf Rahmenanfang von swap

# BEGIN Funktionsbody
sll $t6,$a1,2   # $t6 = k * 4
add $t6,$a0,$t6 # $t6 = v + (k * 4)
lw $t7,0($t6)   # temp = v[k]
lw $s0,4($t6)   # $s0 = v[k+1]
sw $t7,4($t6)   # v[k+1] = temp
sw $s0,0($t6)   # v[k] = $s0
# END Funktionsbody

addi $sp,$fp,-4 # Vorbereitung fuer Ruecksprung
lw $fp,4($sp)   # Rueckspeichern von fp
lw $s0,0($sp)   # Rueckspeichern von $s0
addi $sp,$sp,8  # Rahmen deallozieren
jr $ra  # Zurueckspringen
```