TU Dortmund, Fakultät für Informatik
Jansen, Kelter, Munteanu

15. Dezember 2010

Rechnerstrukturen im WS 2010/2011
Übungsblatt 10

Aufgabe 1 (Fehlersuche) (4 Punkte)

Sie haben ein Programm zur iterativen Berechnung der Fakultät erhalten. Leider haben sich 7 Fehler eingeschlichen (syntaktische und semantische Fehler), die Sie finden und beseitigen sollen. Die Anzahl der Programmzeilen soll dabei erhalten bleiben.
Hinweis: n! = n * (n-1) * (n-2) * ... * 1 und 0! = 1.

a) Geben Sie die Fehler und das korrigierte Programm an.

b) Das Ergebnis in Reg 3 soll in einem anderen Programmteil als Zweierkomplementzahl interpretiert werden. Wie groß (dezimal) darf die Eingabe „ein“ sein, damit Reg 3 als richtiges Ergebnis verwendet werden kann?

```
.data
ein:    .word 3  #Eingabe vom User (z.B. 3)
erg:    .word 0  #Ergebnis für den User (am Programmende)
fak:    .word 0  #Fakultät intern berechnen (im Prog benutzt)

.text
.glob main
main:   lw $2, ein        #Eingabe „holen“
        lw $3, fak        #Fakultät weiterberechnen
        beq $2,$0, fertig #zähler=0
jump:   mul $3,$3,$2
        addi $2,$2,1      #zählen
        bge $2,0, jmp
        sw erg, $3        #Fakultät nach Berechnung in erg
fertig: li $2,1           #Programmende
        syscall
```

Aufgabe 2 (Adressrechnung) (4 Punkte)

Gegeben sei folgender MIPS-Assemblerprogrammrahmen, der einige Register- und Speicherinhalte setzt:

```
.data
x:    .word 42
y:    .word 999
.text
.globl main
main: la $2,x
      la $3,y
      li $4,8
      . . .       # Hier steht Ihr Ladebefehl
      li $2,10              #Programmende
      syscall
```

Geben Sie für die nachfolgenden vier Adressierungsarten je einen Ladebefehl in MIPS-Syntax an, so dass obiges Programm – jeweils ergänzt um den entsprechenden Ladebefehl – in das Register $5 den Wert 999 lädt. Es ist erlaubt die Label x und y stellvertretend für deren Speicheradressen zu verwenden. Auf andere Register als $2 – $5 darf nicht zugegriffen werden. Weiterhin darf bei der relativen Adressierung und für den Offset nicht die Konstante 0 benutzt werden.

a) Unmittelbare Adressierung

b) Direkte Adressierung

c) Register-indirekte Adressierung

d) Relative Adressierung

Aufgabe 3 (Assemblerprogrammierung) (4 Punkte)

Durch das folgende Assemblerprogramm sollen alle Elemente eines Feldes der Reihe nach wortweise kopiert werden. Es stehen nur die Register $3 und $4 zur Verfügung, und das Feld hat mindestens eins und weniger als 256 Elemente. Jedes Element belegt vier Bytes. Ergänzen Sie die fehlenden Operanden gemäß den Kommentaren.

```
.data
Ziel:       .space 2000          # Genügend Platz zum Kopieren
Feld:       .word 3,9,-2,10      # Werte der Feldelemente (Beispiele).
Anzahl:     .word 4              # Anzahl der Feldelemente (Beispiel).
.text
.globl main

main:   lw      [          ]     # Lade Anzahl der Feldelemente in $3.

        subi    [          ]     # Berechne Index - 1 des letzten Feldelements.

        li      [          ]     # $4 für mul mit Konstante vorbereiten.

        mul     [          ]     # Berechne Offset des (letzten) Index.

Schieb: lw      [          ]     # Lade Speicher(Feld + Offset) in $4.

        sw      [          ]     # Kopiere Feldelement nach Ziel.

        sub     [          ]     # Berechne Offset des nächsten Index.

        bge     [          ]     # Falls Offset >= 0 springe zurück.

        li   $2,10               # Programmende
        syscall
```

Aufgabe 4 (Belegungsverfahren) (4 Punkte)

a) Die Zeichenkette „iPhone“ sei in einem Speicher (32-Bit Aufteilung) mit dem Belegungsverfahren „little endian“ ab der Adresse F8F8 abgelegt. Welches Zeichen enthält das Byte mit der Adresse F8F9?

b) Wie wird auf das Speicherbelegungsverfahren „big endian“ hardwaremäßig umgeschaltet?

Die Abgaben sollen bis Mittwoch, den 05. Januar 2011 um 20.00 Uhr in die Briefkästen im Pavillon 6 eingeworfen werden. Bitte Name (bei einem 3er-Team alle), Matrikel- und Gruppennummer oben auf der ersten Seite der Lösungen angeben.