TU Dortmund, Fakultät für Informatik
Jansen, Kelter

14. Dezember 2011

Rechnerstrukturen im WS 2011/2012
Übungsblatt 10

Aufgabe 1 (Fehlersuche) (4 Punkte)

Sie haben ein Programm zur iterativen Berechnung der Fakultät erhalten. Leider haben sich 4 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 1  #Ergebnis für den User (am Programmende)

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

Aufgabe 2 (Adressrechnung) (4 Punkte)

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

```
.data
x:    .word 42
y:    .word 123
.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 123 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)

Mit dem folgenden Assemblerprogramm soll die Quersumme QS einer Dezimalzahl (wert) berechnet und unter „quer“ im Speicher abgelegt werden. Ergänzen Sie die fehlenden Operanden gemäß den Kommentaren.

```
.data
wert:    .word 1495              # Eingabewert.
quer:    .word 0                 # Ergebnis der Quersumme QS.
.text
.globl main

main:    [                    ]  # lade den Eingabewert in Register $2.
         [                    ]  # Reg [4] für die QS initalisieren.
loop:    [                    ]  # wenn Reg[2] = 0 -> ende.
         [                    ]  # in Reg[3] steht eine 10 (dezimal).
         [                    ]  # wert / 10 und den Teil hinter dem
         [                    ]  # Komma in Reg [3] laden
         [                    ]  # und in Reg [4] aufaddieren.
         [                    ]  # Restzahl in Reg [2] laden.
         [                    ]  # bei loop weiterrechnen.
ende:    [                    ]  # Ergebnis in quer ablegen.

         li $2,10                # Programmende.
         syscall
```

Aufgabe 4 (Belegungsverfahren) (4 Punkte)

a) Die Zeichenkette „Studi“ 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 F8FB?

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

Die Abgaben sollen bis Mittwoch, den 21. Dezember 2011 um 18.00 Uhr in die gelben Briefkästen im MSW16 vor Raum E10. Bitte Name (bei einem 3er-Team alle), Matrikel- und Gruppennummer oben auf der ersten Seite der Lösungen angeben.