Karlsruhe Institute of Technology
Institut für Anthropomatik und Robotik (IAR)

Rechnerorganisation im Wintersemester 2017/18

Aufgaben zu den Tutorien in der Woche vom 04. bis 08. Dezember 2017

Prof. Dr. Wolfgang Karl
Haid-und-Neu-Str. 7
Dr.-Ing. Ömer Terlemez
Adenauerring 2, Geb. 50.20
Email: ti@ira.uka.de
Web: http://ti.ira.uka.de

Lernziele

- MIPS-Assembler

Aufgabe 1

Analysieren Sie das folgende MIPS–Programm.

```
            .data
x:          .word 3
Y:          .word 1, 3, 7

            .text
subroutine: li $v0, 0
            li $t3, 0
marke1:     bge $t3, $a1, marke2
            lw $t0, 0($a0)
            mul $t1, $t0, $t0
            add $v0, $v0, $t1
            addi $a0, $a0, 4
            addi, $t3, $t3, 1
            b marke1
marke2:     jr $ra

            .globl main
main:       la $a0, Y
            lw $a1, x

            jal subroutine

#           Aufruf eines Unterprogramms, welches die
#           Quadratwurzel einer Integerzahl in $v0 berechnet.
#           Das Ergebnis steht in $v0
            move $a0, $v0
            li $v0, 1
            syscall

            li $v0, 10
            syscall
            jr $a
```

1. Welche Funktion hat das Unterprogramm `subroutine`?

   Welche Ausgabe hat das gesamte Programm?
   Hinweis: Bei dem Systemaufruf `print_int` muss das Argument im Register `$a0` stehen.

2. Geben Sie die MIPS-Instruktionen zu den folgenden Pseudoinstruktionen an.

   i.) `b marke`

   ii.) `neg $s3, $s2`

3. Was bewirkt die Assemblerdirektive `.align 3` ?

4. Warum dürfen bei Arithmetikoperationen mit doppelter Genauigkeit nur die Register mit gerader Registernummer verwendet werden?

5. Welche Adressen haben `A, B, C` und `D` im folgenden MIPS-Programmabschnitt.

```
            .data 0x10000003
            .align 3
A:          .byte 6, 5
B:          .word 7, 4
C:          .double 3.1415
D:          .float 2.71828
```

6. Welche Gründe machen die Programmierung der MIPS-Architektur schwierig?

Aufgabe 2

1. Schreiben Sie die folgenden in C gegebenen Kontrollstrukturen in MIPS-Assembler um. Sie dürfen nur die MIPS-Befehle `slt`, `beq` und `bne` verwenden. Zur Speicherung temporärer Variablen verwenden Sie das Register `$at`. Die Variable `a` ist im Register `$t4`, die Variable `b` im Register `$s0` abgelegt.

   i.)
```
if ( a <= b )
{
    ...
}
marke1:
```

```
if ( a >= b )
{
    ...
}
marke2:
```

   iii.)
```
do
{
    marke3:
        ...
} while ( a != b )
```

2. Was sind die Unterschiede zwischen einer statischen und einer dynamischen Speicherallokierung?

Aufgabe 3

Setzen Sie die folgenden C-Kontrollstrukturen mit MIPS-Assembler um (wie in der vorherigen Aufgabe sind kein vollständigen Programm erforderlich). Die Variablen `a`, `b`, `i` und `sum` vom Typ `int32_t` sollen hierbei in den Registern `$s0 - $s3` abgelegt werden. Das Array `array` ist im Datensegment durch eine `array: .word ...` Direktive abgelegt.

1.
```
if (a * 2 < b) {
      a *= 2;
}
```

2.
```
for (a = 10; a >= 0; a -= 2) {
      b += a;
}
```

3.
```
int array[100];
...
sum = 0;
for (i = 0; i < 100; i++)
  sum = sum + array[i];
```

Aufgabe 4

1. Was ändert sich am Assembler-Code der Teilaufgabe 3.3, wenn die Variablen im Datensegment abgelegt und mit Marken entsprechend der Namen der Variablen versehen sind?
2. Beschreiben Sie die Funktion der folgenden MIPS-Befehle:
   i.) `addu $t3, $t2, $t1`
   ii.) `andi $t3, $t2, 0x2000`
   iii.) `slt  $t3, $t2, $t1`
   iv.) `lui  $t3, 0x2000`
3. In welchem Register wird die Rücksprungadresse beim Unterprogrammaufruf gesichert?

Aufgabe 5

1. Geben Sie für das folgende MIPS-Programmstück nach der Ausführung jedes Befehls den Inhalt des jeweiligen Zielregisters in hexadezimaler Schreibweise an.

```
ori   $s1, $zero, 20
sll   $s2, $s1, 3
slti  $s3, $s2, 100
sub   $s4, $s3, $s2
lui   $s5, -7
```

2. Wie ist die Trennung von Programmen und Daten bei der Ihnen bekannten MIPS-R2000-Architektur realisiert?