Karlsruher Institut für Technologie
Institut für Technische Informatik (ITEC)

Rechnerorganisation im WS 2020/21

Musterlösungen zum 4. Übungsblatt

Prof. Dr. Jörg Henkel
Dr.-Ing. Lars Bauer
Roman Lehmann, M. Sc.
Haid-und-Neu-Str. 7,
Geb. 07.21 (Technologiefabrik)
Email: roman.lehmann@kit.edu

Lösung 1 (3 Punkte)

	wahr	falsch
Bei der unmittelbaren Adressierung (*immediate addressing*) enthält der auszuführende Befehl bereits das als Operand zu verwendende Datenwort. Es ist also kein Hauptspeicherzugriff notwendig.	×	
Bei einer Keller-Architektur sind die Operanden einer arithmetischen Operation implizit bekannt. Es muss allerdings der Ort zum Speichern des Ergebnisses explizit spezifiziert werden.		×
Alle Register einer CPU werden zusammengenommen als Prozessorstatuswort bezeichnet.		×
Als Programmiermodell bezeichnet man die Gesamtheit der dem Programmierer zur Verfügung stehenden Register.	×	
Bei der register-indirekten Adressierung sind zwei Speicherzugriffe notwendig um den gewünschten Wert zu lesen bzw. zu schreiben. Deshalb handelt es sich dabei um ein zweistufiges Verfahren zur Speicheradressierung.		×

Lösung 2 (4 Punkte)

1. Eine ausgerichtete Adresse auf einer 32-Bit-Architektur muss ein Vielfaches von 4 sein, d.h. die letzten 2 Bits der Adresse sind Null. [2 P.]

2. Die Byte-Reihenfolge gibt die Anordnung der Bytes eines Datenworts im Speicher an. Es wird unterschieden zwischen Little Endian (das niederwertigste Byte steht an der niederwertigsten Adresse) und Big Endian (das niederwertigste Byte steht an der höchstwertigsten Adresse). [2 P.]
   Sollen Daten architekturübergreifend ausgetauscht werden, so muss die Byte-Reihenfolge in dem Dateiformat oder dem Netzwerkprotokoll klar spezifiziert werden. Auf jeder Architektur muss die native Darstellung dann ggf. in dieses Format umgewandelt werden.

Lösung 3 (3 Punkte)

	CISC	RISC
Erheblicher Einsatz von Pipelining		X
Befehlsworte mit variabler Länge	X	
Viele (General Purpose) Register		X
Mikroprogrammiertes Steuerwerk	X	
Unterstützung vieler Adressierungsarten	X	
Arithmetische Befehle können auf den Hauptspeicher zugreifen	X	

Lösung 4 (3 Punkte)

Das Programm ermittelt das maximale Element im Array a[36, 20, 27, 15, 1,62, 41]. Zur Ermittlung des maximalen Elements wird eine if-then-Schleife benutzt.

```
        .data
a:      .word 36, 20, 27, 15, 1, 62, 41
n:      .word 7

        .text
        .globl main

main:   li $t0, 0               # Array-Index i=0 und in $t0 speichern.
        li $s0, 0               # max=0 und in $s0 ablegen
        lw $s1, n               # Anzahl der Array-Elemente in $s1

m1:     bge $t0, $s1, m3
        mul $t1, $t0, 4         # i auf Wortgrenze skalieren
        lw $t2, a($t1)          # Lade a[i] ins Register $t2
        ble $t2, $s0, m2        # if a[i] <= max, dann "then-Teil"
        move $s0, $t2           # "then-Teil": max = a[i]
m2:     addi $t0, $t0, 1        # Array-Index i inkrementieren
        b m1
m3:     move $a0, $s0           # Ende der Schleife
        li $v0, 1
        syscall
        li $v0, 10
        syscall
```

Lösung 5 (7 Punkte)

1. i.) Funktion des Programmstücks: 3 P.
   Addiert alle ungeraden Zahlen, die kleiner oder gleich n sind.

   ii.) Wert im Register `$v0` nach Abarbeitung des Programmstücks:

   Wenn `$a0 = 9` dann `$v0 = 25`
   Wenn `$a0 = 10` dann `$v0 = 25`

2. i.) `lw $s1, 100($s2)`: 4 P.
   Laden des Wortes (32-Bit) mit der Adresse (100 + Inhalt des Regsites `$s2`) ins Register `$s1`

   ii.) `sw $s1, 100($s2)`:
   Speichern des Wortes im Register `$s1` an der Adresse (100 + Inhalt des Regsites `$s2`)

   iii.) `jal mystery`:
   Unbedingter Sprung zur Marke `mystery` und Speicherung der Adresse des nächsten Befehls (Rücksprungadresse) im Register `$ra`

Lösung 6 (5 Punkte)

1. Ein Systemaufruf wird durch den Maschinenbefehl `syscall` eingeleitet. Durch diesen Befehl springt der Prozessor zu einem vom Betriebssystem definierten Einsprungpunkt zur Bearbeitung des Systemaufrufs. 1 P.

2. Die Systemaufrufe 1 bis 10 lauten: 1 P.

    1. `print_int`
    2. `print_float`
    3. `print_double`
    4. `print_string`
    5. `read_int`
    6. `read_float`
    7. `read_double`
    8. `read_string`
    9. `sbrk` (allokiert Speicher und liefert die Adresse zurück)
    10. `exit` (beendet die Ausführung)

3. Die Nummer des auszuführenden Systemaufrufs wird im Register `$v0` übergeben. 1 P.

4. 2 P.
    - `print_double`: Die auszugebende 8 Byte breite Gleitkommazahl (double) wird in den Registern `$f12` und `$f13` erwartet.
    - `read_string`: In `$a0` wird die Speicheradresse übergeben, an der der gelesene String abgelegt werden soll. In `$a1` wird die Länge des dort reservierten Puffers übergeben, um Pufferüberläufe zu vermeiden. Es werden maximal `$a1`-1 Zeichen gelesen und ein Nullzeichen zur Terminierung des Strings angehängt.