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

Rechnerorganisation im WS 2017/18

Musterlösungen zum 4. Übungsblatt

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

Lösung 1 (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 2 (5 Punkte)

1. Die Programmadressen sind die im Programm (Maschinenbefehle, Adresswerte) verwendeten Adressen. [2 P.]
   Die logischen Adresse ist die Adresse, die aus einer Programmadresse unter Berücksichtigung der durch den Befehl verlängten Adressrechnung bestimmt wird (z.B. Addition von Basisadresse). Es handelt sich jedoch immer noch um eine virtuelle Adresse im Adressraum des entsprechenden Prozesses.
   Die physikalische Adresse ist die reale Position der Daten im Speicher, die die CPU zur Adressierung auf dem Datenbus verwendet.

2. Die Übersetzung der logischen Adresse zu einer physikalischen Adresse nimmt die Speicherverwaltungseinheit (*memory management unit*, MMU) vor. Die genannten Adressen stehen nur in einem rechnerischen Zusammenhang und müssen nicht die gleiche Länge haben (aktueller AMD64: nur 48 Bitleitungen für physische Adresse). [1 P.]

3. Ein superskalarer Prozessor nutzt Parallelisierung auf Befehlsebene aus und kann in einem Taktschritt mehrere Befehle gleichzeitig auf redundanten Funktionseinheiten ausführen (im Gegensatz zu normalem Pipelining kann deshalb sogar ein IPC-Wert kleiner 1 erreicht werden). [2 P.]
   Bei der MIMA wird hingegen maximal alle 7 Takte ein neuer Befehl geladen. Daher handelt es sich offensichtlich nicht um eine superskalare Architektur.

Lösung 3 (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.

Lösung 4 (4 Punkte)

	R-Typ	I-Typ	J-Typ	Pseudobefehl
`j`			X	
`lw`		X		X
`addu`	X			
`neg`				X
`bne`		X		
`ori`		X		
`beqz`				X
`add`	X			

Hinweis: Bei `lw` mit einer Konstanten, die zu groß ist, um als 16-Bit-Wert dargestellt zu werden, handelt es sich um einen Pseudobefehl (z.B. `lw $0, 0x12345($t1)`), ansonsten um einen Befehl vom Typ I. Jede der beiden Lösungsmöglichkeiten wird als korrekt akzeptiert.

Lösung 5 (6 Punkte)

1.
```
          blt $s3, $s4, label
          sub $s3, $s4, $s5
label:    ...
```

2.
```
          sle $s5, $s4, $s3
```

3.
```
          beq $s4, $s5, label1
          sub $s4, $s3, $s5
          j label2
label1: add $s4, $s5, $s3
label2: ....
```

Lösung 6 (4 Punkte)

Berechnung des Betrags einer Zahl.

Pseudobefehl: `abs $a0, $s0`

```
sra $a0, $s0, 31    # setzt $a0 auf 111....111, falls $s0 negativ ist
                    # und auf 00...000, falls $s0 positiv ist
xor $s0, $a0, $s0   # A XOR 1 = !A, dadurch wird $s0 binär negiert,
                    # falls $s0 negativ ist
sub $a0, $s0, $a0   # da 11...11 = -1 ist wird im Falle, dass $s0 negativ
                    # ist 1 auf $a0 addiert und somit wurde das Zweierkomplement
                    # gebildet falls $s0 negativ ist
```