OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe für Eingebettete Systeme und Betriebssysteme
Prof. Dr. Jörg Kaiser

Theoretisches Aufgabenblatt 0

Abgabetermin: 28.04.-2.05.2014

1. Was beschreibt die Von-Neumann Architektur und wie unterscheidet sie sich von der Harvard-Architektur?

2. Welches Problem wird durch den Begriff *Semantisch Lücke* beschrieben und wie kann es gelöst werden?

3. Was ist ein *Betriebssystem*, welche Aufgaben erfüllt es und wonach können Betriebssysteme klassifiziert werden?

4. Eine Befragung ergab die in Tabelle 1 dargestellte folgende Liste von Systemen:

	OS	BP	IP	SP	MP	SU	MU	TS	Bemerkung
Android									
DOS									
GRUB									
KDE									
Lejos									
Minix									
Symbian									
Unix									
Windows CE									
Windows 3.1									
Zeta									

Tabelle 1: Liste der Systeme;
OS=Operating System, BP=Batch Programming, IP=Interactive Programming, SP=Single Programm System, MP=Multi Programm System, SU=Single User System, MU=Multi User System, TS=Time Sharing

Klassifizieren Sie die genannten Systeme nach *Betriebssystem* bzw. *kein Betriebssystem*. Ordnen Sie die Systeme den nachfolgenden Kategorien zu (*Mehrfachnennungen sind möglich*). Begründen Sie die Zuordnung!

5. Was ist unter Diensten und Betriebsmitteln zu verstehen und wie werden sie verwendet?

6. Was versteht man unter dem Begriff *Timesharing* und warum bieten viele Betriebssysteme *Timesharing-* und *Batchbetrieb* gleichzeitig an?

7. Vergleiche *Betriebssysteme* und *Virtuelle Maschinen* worin bestehen Gemeinsamkeiten und was sind Unterschiede?

8. Gegeben seien die 32-Bit-Assembler-Quelltexte 1 und 2, welche in AT&T-Syntax gehalten sind. Welche C-Konstrukte stellen diese Ausschnitte jeweils dar?

```
1   movl    $0x0,0x18(%esp)
2   jmp     pos2
3 pos1:
4   mov     0x18(%esp),%eax
5   mov     %eax,(%esp)
6   call    func
7   addl    $0x1,0x18(%esp)
8 pos2:
9   cmpl    $0x9,0x18(%esp)
10  jle     pos1
```

Quelltext 1: Assemblerquelltext zur Analyse I

```
1   cmpl    $0x5,0x18(%esp)
2   jne     pos3
3   call    doSomething
4 pos3:
```

Quelltext 2: Assemblerquelltext zur Analyse II

9. Gegeben sei ein Auszug aus einem Assemblerquelltext, in dem eine Funktion aufgerufen wird.

```
1   ...
2   push %ebx
3   pushl $0x1337
4   push %eax
5   push %ecx
6   call func
7   addl  $12, %esp
8 L2:
9   ...
10 func:
11  push %ebp
12  movl %esp, %ebp
13  subl  $0x12, %esp
14 L1:
15  ...
16  leave
17  ret
18  ...
```

Quelltext 3: Assemblercode für eine Stack-Analyse

   Beschreibe, wie viele *Parameter/lokale Variablen* die Funktion hat und was während des Funktionsaufrufs passiert.

10. Skizziere den Stackaufbau beim Erreichen der Labels `L1` und `L2` aus Aufgabe 9.