



## Lösungsvorschlag: 1. Übung zur Vorlesung Systemprogrammierung I

### Aufgabe 1: Plurix (1)

In dieser Aufgabe sollen Sie sich mit der Plurix-Laufzeitumgebung und dem Plurix-Java-Compiler vertraut machen. Laden Sie sich hierfür den Compiler und die Laufzeitumgebung vom Web-Server der Abteilung herunter und testen Sie beides mit dem beiliegenden Testprogramm.

<http://www-vs.informatik.uni-ulm.de/teach/ss05/sp1/>

### Aufgabe 2: Bildschirm (2 + 2)

a) Schreiben Sie unter Plurix eine Methode, welche es Ihnen erlaubt, Buchstaben und Zahlen am Bildschirm auszugeben. Die Zeichen sollen fortlaufend erscheinen, d.h. bei einem erneuten Aufruf der Methode soll an der nächsten freien Stelle weitergeschrieben werden (Hinweis: Beginn des Textbildschirmes ist bei 0xB8000).

siehe Quellcode

b) Sehen Sie eine Methode vor, um am Anfang der nächsten Zeile weiterzuschreiben.

siehe Quellcode

### Aufgabe 3: Port vs. Mem (1 + 2 + 1 + 2)

a) Lesen Sie unter Plurix die Adressen von 0x0 bis 0x100 mittels `MAGIC.InX(addr)` aus. Geben Sie die erhaltenen Werte am Bildschirm aus.

siehe Quellcode

b) Geben Sie die belegten Ports am Bildschirm aus (Anmerkung: Gehen Sie davon aus, dass ein Port belegt ist, wenn der erhaltene Wert  $\neq 0$  und  $\neq 0xFF$  ist).

siehe Quellcode

c) Lesen Sie nun die Adressen 0x0 bis 0x100 mittels `MAGIC.MemXX[addr]` aus und geben Sie wieder die erhaltenen Werte am Bildschirm aus.

siehe Quellcode

d) Warum unterscheiden sich die erhaltenen Werte? Erklären sie die Funktionsweise von `MAGIC.InX(addr)` und `MAGIC.MemXX[addr]`.

`Magic.In` arbeitet auf dem IO-Adressraum des Rechners, d.h. es werden „reine“ Ports adressiert. Kein Hauptspeicher und auch kein MMIO.

`Magic.Mem` erlaubt eine Adressierung des Hauptspeichers als Array und somit auch den Zugriff auf MemoryMapped Register.

#### **Aufgabe 4: Meta-Aufgabe Interrupts (3)**

Entwickeln Sie für die Studierenden der Vorlesung Systemprogrammierung 1 im Sommersemester 2006 eine schöne(!) Aufgabe, welche den Unterschied zwischen den E/A-Adressen der Interruptcontroller und dem programmierten Offset didaktisch herausarbeitet.

In einem PC sind üblicherweise zwei kaskadierte Interruptcontroller vorhanden. Um einen Interrupt behandeln zu können, benötigen diese Controller unterschiedliche Offsets.

Sind die Adressen und Offsets für einen PIC (Master oder Slave) programmierbar?

Die Standard-Interruptcontroller in einem PC besitzen Register, in welchen der Offset in die IDT eingetragen werden kann.

- a) Muß der Offset des aktuellen Interrupts bekannt sein, um dessen Ende zu signalisieren?
- b) Ist die Adresse zur Übermittlung des EOI Befehls an den Controller abhängig vom Offset des aktuellen Interrupts?