Praktikum #43

Kachelverwaltung, Netzwerktreiber

Zielsetzung

Ziel dieser Phase ist die Verwaltung von physikalischen Kacheln sowie das Einbinden des Netzwerktreibers. Der Netzwerktreiber soll noch nicht gestartet werden, dies sowie Tests bzgl. der Lauffähigkeit des Netzwerktreibers sind Bestandteil der nächsten Phase.

Voraussetzungen für den Netzwerktreiber

Der Netzwerktreiber erfordert einige Methoden in SYSTEM.memory und SYSTEM.interrupts. Diese müssen gegebenenfalls dem eigenen Interface angepasst werden (Semantik siehe unten). Außerdem erwartet der Treiber eine Klasse SystemMemory, in der die Speicherstellen für folgende Variablen stehen (die angegebenen Speicherstellen sind die verwendeten unter Plurix):

```
public final static int DNLIST = 0x2100;
public final static int DNLISTINSERT = 0x2104;
public final static int DNLISTCLEARED = 0x2108;
public final static int INBUFFER = 0x210C;
public final static int UPLIST = 0x2110;
public final static int UPLISTLAST = 0x2114;
public final static int InISR = 0x2118;
//frei ab 0x211C
```

Des weiteren wird beim Empfang eines Pakets auf SYSTEM.prot geprüft, welches eine Methode mit folgender Signatur haben muss:

```
public boolean received(NetworkAdapter device, int packID, int bytes)
```

Innerhalb dieser Methode können dann mittels

```
device.getData(packID, oip, size, addr);
```

Daten abgeholt werden. Dabei sind die Parameter oip, size und addr vom Typ int und geben die Position innerhalb des Netzwerkpaketes, die Länge der zu kopierenden Daten sowie die logische Zieladresse für diese Daten.

In der Klasse Intel486 wird eine Methode CopyMemory(source, dest, bytes) erwartet, diese darf beliebig umgeschrieben oder anderweitig implementiert werden.

void memory.ReleasePage(address, isLogical): Seite mit Adresse address freigeben. Ist die Adresse physikalisch, so wird die Kachel freigegeben. Ist die Adresse logisch, so wird außer der adressierten Kachel auch der dazugehörige Eintrag in der PageTable freigegeben.
int memory.Logical2Physical(logAddr): Liefert die physikalische Adresse zur angegebenen logischen. Dabei ist insbesondere darauf zu achten, dass die unteren 12 Bits korrekt übernommen werden.
int memory.GetNonDSMMemory(size): Alloziert einen zusammenhängenden Speicherblock der Größe size außerhalb des DSMs und liefert die Startadresse des allozierten Blocks.
int memory.GetLogDeviceMemory(physAddr, size): Liefert eine logische Adresse zur übergebenen physikalischen Adresse, wobei der Zugriff mindestens für size Bytes gesichert ist. Diese Funktion erfordert also keinen Bereich im Hauptspeicher, sondern ausschließlich eine Umsetzung von (bekannten) physikalischen Adressen auf logische.