Übung 6 - Betriebssysteme II im FS 2017

1 Ein- und Ausgabe

Sie sollen ein Programm schreiben, das Interrupts simuliert.

1.1 Speicherbus

Entwerfen Sie eine Klasse, die einen Speicherbus abbildet. Der Speicherbus soll 6 Funktionen anbieten:

1. Setzen der Adresse
2. Lesen der Adresse
3. Setzen des Datums
4. Lesen des Datums
5. Setzen des Modus
6. Lesen des Modus

Adressen und Daten sollen jeweils 16 Bit breit sein. Adresse und Modus soll nur von der CPU gesetzt werden können. Modus gibt an, ob die CPU lesen oder schreiben möchte.

1.2 Geräte und Speicher

Entwerfen Sie eine Klasse, deren Objekte Geräte darstellen. Ein Gerät soll mit einer Instanz der Speicherbus-Klasse verknüpft werden können. Die Klasse soll ausserdem eine abstrakte Funktion „run“ erhalten, die keine Parameter und keinen Rückgabewert hat. In dieser Funktion soll das Verhalten des Geräts (abgeleiteter Klassen) simuliert werden.

Entwerfen Sie eine Klasse, die den Speicher simuliert. Speicher ist in diesem Zusammenhang ein Gerät, das in „run“ die Adresse und den Modus vom Speicherbus liest und dann

1. das Datum vom Speicherbus liest und an der Adresse speichert, wenn Modus schreiben anzeigt, oder
2. das Datum an der Adresse auf den Speicherbus legt, wenn Modus lesen anzeigt.

Um es einfach zu halten, soll die Adresse sich immer auf ein 16-Bit breites Datum beziehen, so dass Sie im Speicher einfach ein Array anlegen können, dessen Indizes die Adressen sind.

Entwerfen Sie dann eine Geräte-Klasse, die in „run“:

1. ein Zeichen vom Speicherbus liest und in der Konsole entsprechend ausgibt. (In Java können Sie dafür `System.out.print` verwenden.)

Stefan Richter ■ stefan.richter@hsr.ch
HSR Hochschule für Technik Rapperswil ■ Informatik ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch

2. ein Zeichen von der Konsole einlesen und auf den Speicherbus legen kann (In Java können Sie dafür `if (System.in.available () > 0) { return System.in.read (); }` verwenden.)

Überlegen Sie sich dazu ein Protokoll, das mit den Informationen des Speicherbusses auskommt.

1.3 CPU

Entwerfen Sie eine Klasse, die eine CPU simuliert. Die CPU soll auch von der Geräte-Klasse abgeleitet sein. In „run“ soll die CPU eine Instruktion vom Speicherbus anfordern; die CPU benötigt also einen Instruction Pointer, der in jedem Aufruf von „run“ erhöht werden muss. Eine Instruktion soll 16 Bit lang sein; jeder Operand soll 16 Bit umfassen. Definieren Sie folgende Instruktionen:

1. Transferiere Wert von Adresse in Register.
2. Transferiere Wert aus Register an Adresse.
3. Transferiere Wert von Register zu Register.
4. Springe an Adresse (= Setze Instruction Pointer auf Adresse)

1.4 Simulator

Schreiben Sie einen Simulator, der „run“ auf jedem Gerät, einschliesslich Speicher und CPU, in einer Endlosschleife nacheinander aufruft. Fügen Sie je nach Bedarf eine Wartepause ein, um die Ausführung zu verlangsamen.

Füllen Sie (statisch) den Speicher mit einem Programm, das die von Ihnen definierten Instruktionen verwendet. Das Programm soll in einer Endlosschleife „.“ ausgeben.

Initialisieren Sie die CPU mit dem Startpunkt Ihres Programms. Führen Sie dann den Simulator aus.

1.5 Eingabe

Erweitern Sie Ihr Programm, so dass es auf eine Eingabe vom Benutzer wartet. Diese Eingabe soll ein einzelnes Zeichen sein. Dann soll das Programm dieses Zeichen ausgeben. Das soll zunächst mit Busy-Wait Polling geschehen.

Stefan Richter ■ stefan.richter@hsr.ch
HSR Hochschule für Technik Rapperswil ■ Informatik ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch

1.6 Interrupt

Fügen Sie ein Interrupt-Flag zu Ihrer CPU hinzu. Der Einfachheit halber soll das Gerät dieses Flag direkt setzen dürfen, wenn es einen Interrupt signalisieren möchte. Wir definieren ausserdem, dass die Interrupt-Vector-Table aus einem einzigen Eintrag besteht und an Adresse 0 startet.

Die CPU soll in „run“ überprüfen, ob der Interrupt gesetzt ist und dann den Instruction Pointer auf die Adresse umsetzen.

Schreiben Sie dann einen Interrupt-Handler, der das eingegebene Zeichen vom Gerät in den Speicher kopiert, so dass das Programm ab der nächsten Ausgabe dieses Zeichen ausgibt.

Stefan Richter ■ stefan.richter@hsr.ch
HSR Hochschule für Technik Rapperswil ■ Informatik ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch