Q12 Name

Schulaufgabe aus dem Fach Informatik
am 23.02.24

Synchronisation nebenläufiger Prozesse; Funktionsweise eines Computers

Aufgabe 1: (Notwendigkeit der) Synchronisation von Prozessen

Henni und Holli aus der Klasse 6d arbeiten für den Natur-und-Technik-Unterricht gemeinsam an einer Bildschirmpräsentation. Die entsprechende Datei wird immer am Ende der Unterrichtsstunde auf eine einfache Austauschplattform im WWW hochgeladen. Die Kinder können auch von zu Hause aus darauf zugreifen. Nun fällt Henni gegen Abend etwas ein, was sie noch schnell ändern möchte. Deshalb lädt sie die Datei (also eine Kopie davon) herunter, ändert sie und lädt die geänderte Datei wieder hoch (überschreibt also die vorhandene Datei). Auch Holli hat noch eine gute Idee, lädt die Datei herunter, ändert sie und lädt die geänderte Datei wieder hoch.

a) Bei unglücklicher zeitlicher Verzahnung der oben beschriebenen Vorgänge kann ein für die Kinder sicher sehr ärgerliches Problem auftreten. Erklären Sie dieses Problem mithilfe eines Sequenzdiagramms und einer knappen Erläuterung.

5 BE

b) Beschreiben Sie einen „Mechanismus“, mit dessen Hilfe sich Probleme dieser Art vermeiden ließen: Wie müsste er funktionieren? Wie wäre er zu verwenden?

2 BE

Aufgabe 2: Probleme bei der Synchronisation von Prozessen

Ein Erkundungsroboter (der auch in Umgebungen eingesetzt werden kann, in denen keine zuverlässigen Verbindungen über Funk, Infrarotlicht, Laser o. Ä. möglich sind) führt selbstständig Erkundungsaufträge aus, fährt dann zu seiner Basisstation, stellt eine Steckverbindung mit deren dafür bereitstehendem Datenport her, liefert dann die gesammelten Daten ab und nimmt anschließend neue Aufträge entgegen, die aufgrund der gelieferten Daten von der Basisstation berechnet werden. Das Aktivitätsdiagramm (Diagramm 1) auf dem Diagrammblatt stellt dar, wie der Erkundungsroboter bisher programmiert ist.

Nun wird das System erweitert: Es werden weitere Erkundungsroboter in Betrieb genommen, die unterschiedliche Erkundungsaufträge bekommen, aber alle dieselbe Basisstation anfahren. Es kann auch nach wie vor nur ein Roboter gleichzeitig über den Datenport mit der Basisstation verbunden sein. Jetzt wird es immer wieder vorkommen, dass die Basisstation neue Erkundungsaufträge erst erteilen kann, nachdem sie die Daten von mehreren Robotern entgegengenommen (und verarbeitet) hat.

a) Die Roboter werden sich jetzt früher oder später gegenseitig ganz grundlegend behindern werden, falls sie alle das im Aktivitätsdiagramm abgebildete Programm ausführen, selbst wenn es bei 1. wie folgt verändert wird: Die Aktion „an der Basisstation andocken“ wird durch die komplexere Aktion „an der Basisstation andocken, sobald diese frei ist“ ersetzt.

Geben Sie den Namen dieses typischen Problems, das bei an sich korrekter Synchronisation auftreten kann, an und erklären Sie das Problem am gegebenen Beispiel.

3 BE

b) Verändern Sie nun das Aktivitätsdiagramm (direkt auf dem Diagrammblatt) bei 2. so, dass es nicht zu einer derartigen gegenseitigen Behinderung kommt. (Streichen Sie dabei die Teile des Diagramms, die nicht mehr gelten sollen.)

4 BE

c) Vergleichen Sie schließlich Ihre Lösung knapp mit dem „passiven Warten“ eines Threads wie es zum Beispiel in Java/Kotlin im Zusammenhang mit dem Gebrauch eines Monitors möglich ist (⇔ was ist gleich, was verschieden?).

2 BE

Aufgabe 3: Zustände einer Registemaschine und Assemblerprogrammierung

Eine Registermaschine (deren Programmzähler bei der Abarbeitung eines jeden Befehls um 2 erhöht wird) stellt folgende Befehle zur Verfügung:

Befehl	***Erklärung***
LOAD adresse	Wert von der angegebenen Speicheradresse in den Akkumulator laden
LOADI wert	Angegebenen Wert in den Akkumulator laden
STORE adresse	Wert aus dem Akkumulator an der angegebenen Speicheradresse speichern
ADD adresse	Wert an der angegebenen Adresse zum Wert im Akkumulator addieren
ADDI wert	angegebenen Wert zum Wert im Akkumulator addieren
SUB adresse	Wert an der angegebenen Adresse vom Wert im Akkumulator subtrahieren
SUBI wert	angegebenen Wert vom Wert im Akkumulator subtrahieren
MUL adresse	Wert an der angegebenen Adresse mit Wert im Akkumulator multiplizieren
MULI wert	angegebenen Wert mit Wert im Akkumulator multiplizieren
DIV adresse	Wert im Akkumulator durch Wert an der angegebenen Adresse dividieren
DIVI wert	Wert im Akkumulator durch angegebenen Wert dividieren
CMP adresse	Wert an der angegebenen Adresse mit dem Wert im Akkumulator vergleichen (= Ersteren von Letzterem subtrahieren, ohne jedoch das Ergebnis im Akkumulator abzulegen) und Zero-Flag sowie Negative-Flag entsprechend setzen
CMPI wert	Angegebenen Wert mit dem Wert im Akkumulator vergleichen (= Ersteren von Letzterem subtrahieren, ohne jedoch das Ergebnis im Akkumulator abzulegen) und Zero-Flag sowie Negative-Flag entsprechend setzen
JMPZ adresse	bei gesetztem Zero-Flag an die angegebene Adresse springen
JMPN adresse	bei gesetztem Negative-Flag an die angegebene Adresse springen
JMP adresse	an die angegebene Adresse springen
HOLD	Programmabarbeitung beenden

Assemblierte Programme für die Registermaschine werden immer ab Speicherzelle 0 im Arbeitsspeicher abgelegt.

a) Gegeben ist folgendes Assemblerprogramm für die Registermaschine:

```
LOAD 100
CMP 101
JMPZ 12
LOADI 0
STORE 102
HOLD
LOADI 1
STORE 102
HOLD
```

Nun wird das Programm mit dem Wert 5 in den *beiden* Speicherzellen 100 und 101 getestet. Geben Sie in Form einer Zustandstabelle an, welche Zustände die Registermaschine dabei durchläuft.

4 BE

b) Geben Sie in der Programmiersprache Kotlin eine Funktion an, die dasselbe leistet wie das Assemblerprogramm (wenn ihr die Werte aus den Speicherzellen 100 und 101 als Parameter übergeben werden und ihr Rückgabewert dem in Speicherzelle 102 geschriebenen Wert entsprechen soll).

3 BE

c) Notieren Sie ein Assemblerprogramm für die Registermaschine, das die natürliche Zahl, die vor dem Programmstart in Speicherzelle 100 steht, mit der natürlichen Zahl potenziert, die vor Programmstart in Speicherzelle 101 steht. Das Ergebnis der Berechnung soll in Speicherzelle 102 abgelegt werden.

Tipp: Sorgen Sie gleich zu Beginn des Programms dafür, dass ein für die jeweilige Berechnung günstiger Startwert in Speicherzelle 102 geschrieben wird.

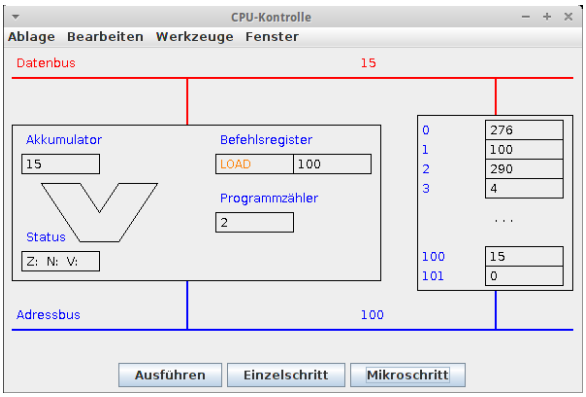
8 BE

Aufgabe 4: Von-Neumann-Architektur und Harvard-Architektur

Diagramm 2 auf dem Diagrammblatt zeigt ein Schema der sogenannten Harvard-Architektur, die zum Beispiel in ‚Rechnern' Anwendung findet, die ausschließlich der Steuerung von Maschinen oder der Signalverarbeitung dienen.

a) Beschriften Sie (direkt auf dem Diagrammblatt) zunächst diejenigen Komponenten, die Ihnen aus der Von-Neumann-Architektur bekannt sind, mit den entsprechenden deutschen Begriffen. Benennen Sie dann den sich ergebenden Unterschied zwischen Von-Neumann-Architektur und Harvard-Architektur so genau wie möglich.

6 BE



b) Die Verwendung der Harvard-Architektur kann gegenüber der Von-Neumann-Architektur einen Geschwindigkeitsvorteil mit sich bringen: Da unter anderem das Bussystem anders gestaltet ist, können nämlich einzelne Schritte des Befehlszyklus, die in der Von-Neumann-Architektur nacheinander stattfinden müssen, in der Harvard-Architektur parallel zueinander stattfinden. Dies wäre zum Beispiel der Fall, wenn auf einer Maschine mit dem in Aufgabe 3 angegebenen Befehlssatz nacheinander die Anweisungen `LOAD 100` und `JMP 14` ausgeführt werden. Parallel zum Schritt „Befehl ausführen" des ersten Befehls könnte hier – wenn „Befehl ausführen" der letzte Schritt des Befehlszyklus ist[1] – bereits ein Schritt des nächsten(!) Befehlszyklus stattfinden. Benennen Sie diesen Schritt und erklären Sie, warum eine gleichzeitige Ausführung in der Von-Neumann-Architektur völlig ausgeschlossen ist (vgl. dazu auch die Abbildung der Minimaschine), in der Harvard-Architektur aber möglich ist.

3 BE

Viel Erfolg!

1 … alle anderen Schritte müssen also vorher erledigt worden sein …