Technische Informatik II: Übungen 4

November 14, 2002

Das folgende beschreibt ein Interrupt-Verfahren:

1. Der Geräte-Controller asseriert eine Interrupt-Zeil auf dem Systembut, um die Interrupt-Folge zu starten
2. Sobald die CPU bereit ist, den Interrupt abzuarbeiten, asseriert sie ein Interrupt-Bestätigungssignal auf dem Bus
3. Erkennt der Geräte-Controller, dass sein Interrupt-Signal bestätigt wurde, gibt er eine kleine Ganzzahl auf die Datenleitung aus, um sich zu identifizieren. Diese Zahl wird als **Interrupt-Vector** bezeichnet.
4. Die CPU entfernt den Interrupt-Vector vom Bus und speichert ihn vorübergehend.
5. Die CPU schiebt den Programmzähler und das PSW (Programmstatuswort) auf dem Stack
6. Die CPU ermittelt einen neuen Programmzähler, indem sie den Interrupt-Vector als Index zu einer Tabelle unten im Speicher benutzt. Ist der Programmzähler z.B. 4 Byte, entspricht der Interrupt-Vector n der Adresse 4n. Dieser neue Programmzähler zeigt auf den Anfang der Interrupt-Routine, die Interrupts für Geräte veranlasst. Oft wird das PSW ebenfalls geladen bzw. modifiziert.
7. Die Interrupt-Routine speichert zunächst alle Register, damit sie später wiederhergestellt werden können. Sie können auf dem Stack oder in einer Systemtabelle gespeichert werden.
8. Ein Interrupt-Vektor wird im allgemeinen von allen Geräten eines bestimmten Typs benutzt. Deshalb ist noch nicht bekannt, welches Terminal den Interrups veranlasst hat. Die Terminal-Nummer wird aus einem Geräteregister ausgelesen.
9. An diesem Punkt können weitere Informationen über den Interrupt, z.B. Statuscodes, eingelesen werden.
10. Falls ein I/O-Fehler aufgetreten ist, kann er an diesem Punkt gehandhabt werden.

11. Die globalen Variablen *ptr* und *count* werden aktualisiert. *ptr* wird inkrementiert, um auf das nächste Byte zu zeigen; *count* wird decrementiert, um darauf hinzuweisen, dass 1 Byte weniger auszugeben ist. Ist *count* immer noch grösser als 0, since mehr Zeichen auszugeben. Das Zeichen, auf das *ptr* jetzt verweist, wird in das Ausgabepufferregister kopiert.

12. Falls nötig, wird ein spezieller Code ausgegeben, um dem Gerät bzw. Interrupt-Controller mitzuteilen, dass der Interrupt verarbeitet wurde.

13. Alle gespeicherten Register werden wiederhergestellt.

14. Die Instruktion *RETURN FROM INTERRUPT* wird ausgeführt, so dass die CPU weider in den Modus und Status zurückversetzt wird, in denen sie sich bei Eintritt des Interrupts befunden hat. Der Computer fährt dort fort, wo er wegen des Interrupts unterbrochen hat.

Zu diesem Verfahren folgende Fragen:

1. Welche Art Geräte bespricht das Interrupt-Verfahren?

2. Welche Verfahrensschritte werden von SW kontrolliert bzw. in HW implemeniert?

3. Wie muss das Verfahren modifiziert werden, um das Interrupt-Verfahren eines "Invalid Opcode"-Interrupts durchzuführen?

4. Die gleiche Frage für den Fall eines "Schreiben der Festplatte"- Interrupt.

5. Was bedeuten die Begriffe "Stack", "Interrupt-Vector", "PSW", "Interrupt-Routine", "Statuscodes"?

6. Was sind *ptr* und *count*, wo befinden sie sich, und wie werden sie benutzt?

7. Warum stellen I/O-Geräte den Interrupt-Vector auf den Bus? Wäre es möglich, diese Informationen statt dessen in einer Tabelle im Speicher abzulegen?

8. Warum haben Interrupt-Routine Prioritäten, während normale Prozeduren keine Prioritäten haben?