Prof. Dr. Christoph Scholl
Dr. Tim Welschehold
Alexander Konrad
Niklas Wetzel

Freiburg, 11.11.2022

Betriebssysteme
Musterlösung zu Übungsblatt 4

Aufgabe 1 (3+4 Punkte)

Gegeben sei ein Rechner mit einem Prozessor, der mit einer Taktrate von 800 MHz arbeitet. Es besteht einerseits die Möglichkeit, die Kommunikation mit der Festplatte Interrupt–getrieben abzuwickeln und andererseits die Möglichkeit, DMA (= Direct Memory Access) zu verwenden. Die Festplatte hat eine Datenübertragungsrate von 8MB/sec.

a) Nehmen Sie für die Interrupt–getriebene Kommunikation an, dass die Festplatte jeweils acht 32–Bit–Worte auf einmal zum Prozessor überträgt und dass sie dem Prozessor jeweils durch Auslösen eines Interrupts mitteilt, dass die Daten bereit sind. Der Overhead, der bei dem Prozessor für eine Datenübertragung (einschließlich Interrupt) anfällt, betrage 1000 Taktzyklen. Nehmen Sie weiterhin an, dass die Festplatte nur während 5% der Zeit überhaupt aktiv ist. Wie groß ist in diesem Fall der relative Anteil der CPU-Zeit, der für Datentransfers von Festplatte zum Prozessor aufgewendet wird?

b) Nehmen Sie für den DMA–Transfer an, dass jeweils 16KB–Blöcke auf einmal von der Festplatte zum Speicher übertragen werden. Die Übertragung wird begonnen mit einer Aktivierung des DMA–Controllers durch den Prozessor. Danach überträgt die Festplatte gesteuert vom DMA–Controller Daten direkt zum Speicher. Nehmen Sie der Einfachkeit halber an, dass während dieser Zeit bei der Arbeit des Prozessors keine Konflikte auf dem Systembus entstehen, so dass der Prozessor ohne Beeinträchtigung durch den DMA–Transfer weiterarbeiten kann. Die Beendigung des Transfers meldet der DMA–Controller dem Prozessor über einen Interrupt. Der Prozessor benötige 1500 Taktzyklen, um den DMA–Controller vor einer Übertragung eines Blockes zu aktivieren. Außerdem benötige der Prozessor weitere 500 Taktzyklen zur Bearbeitung des Interrupts, der vom DMA–Controller nach Beendigung der Blockübertragung ausgelöst wird. Berechnen Sie auch hier den relativen Anteil der CPU-Zeit, den der Prozessor für Datentransfers aufwendet, *unter der Annahme, dass die Festplatte ebenfalls nur während 5% der Zeit überhaupt aktiv ist.*

Hinweis: Bitte geben Sie Ihren Rechenweg an!

Lösung:

a)

$$\begin{aligned} 8 \cdot 32\text{Bit} = 2^5\text{Byte} &\;\hat{=}\; \frac{1000}{800\text{MHz}} = \frac{1}{800000}\text{s} \\ \Rightarrow 8\text{MB} = 2^{23}\text{Byte} = 2^{18} \cdot 2^5\text{Byte} &\;\hat{=}\; 2^{18} \cdot \frac{1}{800000}\text{s} \\ \Rightarrow 0.05 \cdot \frac{2^{18} \cdot \frac{1}{800000}\text{s}}{1\text{s}} = \frac{2^{18}}{16000000} &= 0.016384 \end{aligned}$$

b)

$$\begin{aligned} 16\text{KB} = 2^{14}\text{Byte} &\;\hat{=}\; \frac{2000}{800\text{MHz}} = \frac{1}{400000}\text{s} \\ \Rightarrow 8\text{MB} = 2^{23}\text{Byte} = 2^{9} \cdot 2^{14}\text{Byte} &\;\hat{=}\; 2^{9} \cdot \frac{1}{400000}\text{s} \\ \Rightarrow 0.05 \cdot \frac{2^{9} \cdot \frac{1}{400000}\text{s}}{1\text{s}} = \frac{2^{9}}{8000000} &= 0.000064 \end{aligned}$$

[Rechenweg 2.5p/3.5p, richtiges Ergebnis jeweils 0.5p]

Aufgabe 2 (3+4 Punkte)

Betrachten Sie einen Prozessor mit nur einem Interruptlevel, der zur Verarbeitung von Interrupts mit verschiedenen Prioritäten mit einem Interrupt-Controller zusammenarbeitet. Wenn auf dem Prozessor keine Interrupt-Service-Routine (ISR) aktiv ist oder wenn ein am Interrupt-Controller anliegender Interrupt eine höhere Priorität hat als die aktuell auf dem Prozessor laufende ISR, dann signalisiert der Interrupt-Controller dem Prozessor über ein Signal `INT`, dass ein Interrupt anliegt, der den Prozessor unterbrechen darf. Nach Abarbeiten einer ISR signalisiert der Prozessor dem Interrupt-Controller durch Aktivieren eines Signals `/INTA` für genau einen Takt, dass die ISR beendet wurde. Wir nehmen an, dass es maximal 255 (Hardware-)Interrupts mit Prioritäten 0 bis 254 gibt. Solange ein Interrupt INT_j noch nicht verarbeitet ist, darf das entsprechende I/O-Gerät j keinen weiteren Interrupt auslösen. Prozessorinterne Exceptions können Sie bei dieser Aufgabe vernachlässigen.

a) Überlegen Sie sich eine Methode, wie der Interrupt-Controller feststellen kann, dass auf dem Prozessor aktuell gerade keine ISR läuft, die ihm vom Interrupt-Controller mitgeteilt wurde. Sie können dazu einen 8-Bit-Zähler verwenden, der über zwei Signale `up` und `down` gesteuert wird. Wenn `up = 1, down = 0`, dann zählt der Zähler bei steigender Clockflanke hoch, wenn `up = 0, down = 1`, dann zählt der Zähler bei steigender Clockflanke runter. Durch Aktivieren des Eingangs /reset kann der Zähler auf 0 initialisiert werden. Es genügt, wenn Sie die Methode mit Worten skizzieren.

b) Reicht Ihre in Teil a) entwickelte Methode aus, um den Interrupt-Controller korrekt zu implementieren? Wenn nein, skizzieren Sie eine Situation, in der der Interrupt-Controller keine ausreichenden Informationen hat, um zu entscheiden, ob er über INT einen Interrupt signalisieren darf oder nicht. Entwerfen Sie nun eine Lösung, bei der der Interrupt-Controller das Signal INT immer korrekt setzt. Sie können annehmen, dass der Interrupt-Controller intern einen kleinen Speicherbereich mit 256 Speicherzellen hat, der über 8-Bit-Adressen angesprochen werden kann. Die Speicherzellen haben eine Wortbreite von 8 Bit. Weiterhin können Sie den 8-Bit-Zähler aus Teil a) in geeigneter Weise verwenden. Sie können außerdem annehmen,

dass der Interrupt-Controller die Nummer des anliegenden Interrupts mit der höchsten Priorität stets in einem Register IVN speichert und zusätzlich dessen Priorität in einem Register PR speichert.

Lösung:

a) Idee: Zähler wird genutzt um zu ermitteln, ob noch eine ISR auf dem Prozessor läuft oder nicht. Wichtige Punkte die erwähnt werden sollten sind:

- Beim Start des Interrupt-Controllers (oder generell Start der CPU/des Rechners) /reset Signal nutzen um Zähler mit 0 zu initiliasieren. [0.5P]
- Bei Senden von INT (startet ISR auf der CPU) Zähler inkrementieren, indem `up = 1, down = 0` für einen Takt an Zähler angelegt wird. [1P]
- Bei Empfangen von /INTA (aktuelle ISR auf CPU wurde beendet) Zähler dekrementieren, indem `up = 0, down = 1` für einen Takt an den Zähler angelegt wird. [1P]
- Wenn Zähler = 0, dann weiß Controller, dass keine ISR auf der CPU läuft. [0.5P]

b) Nein, reicht nicht aus. Situation: Zähler war mal `>= 2`. Also wurde min. eine ISR von INT_i unterbrochen weil ein Interrupt INT_j mit höherer Priorität ankam. Nachdem INT_j fertig ist, wird in der CPU vorherige ISR von INT_i fortgesetzt. Kommt jetzt ein Interrupt INT_k am Controller an, weiß der Controller nicht, ob die aktuell ausgeführte ISR höhere Priorität hat oder nicht.

- Es muss mehr als ein Interrupt gleichzeitig in seiner ISR angefangen worden sein (Zähler `>= 2`) [0.5P].
- Es muss vor Eintreffen von INT_k min. eine ISR beendet worden sein [0.5P] [insgesamt 1P für Erklärung]

Idee um Problem zu lösen: Nutze den Speicher als Stack für Historie über die übergebenen Interrupts. Oben auf dem Stack liegt immer die Priorität des aktuell in der CPU abgearbeiteten Interrupts (welche immer die höchste Priorität aller schon gestarteten aber noch nicht beendeten ISR sein muss). Wichtige Punkte die erwähnt werden sollten:

- Bei Senden von INT wird Inhalt von Register PR (Priorität des Interrupts, der der CPU übergeben wird) auf den Stack des Controllers (= Speicher des Controllers, nicht der Stack der CPU!!) gelegt. [1P]
- Bei Empfangen von /INTA wird oberster Eintrag vom Stack entfernt. [0.5P]
- Ob ankommender INT_j an CPU weitergeleitet wird, wird durch Vergleich von PR mit obersten Eintrag des Stacks (Adresse liefert Zähler) erreicht. [0.5P]
- Da kein Stackpointer zur Verfügung steht wird Zähler aus a) für die Adressierung des Stacks genutzt. [1P] Falls hier statt des Zählers ein zusätzliches Register als Stackpointer benutzt wird nur [0.5P].

Aufgabe 3 (6 Punkte)

In der Vorlesung haben Sie eine Erweiterung der Kontrolllogik der RETI kennengelernt. Diese ermöglicht zusätzlich zum Normalbetrieb auch eine Interruptbehandlung.

Erweitern Sie den Zusatzautomat zur Interruptbehandlung in Abbildung 1 so, dass innerhalb einer Interrupt-Service-Routine (ISR) der Befehl `INT i` („Software-Interrupt“) vorkommen kann *oder* ein Interrupt während der Abarbeitung der ISR auftreten kann.

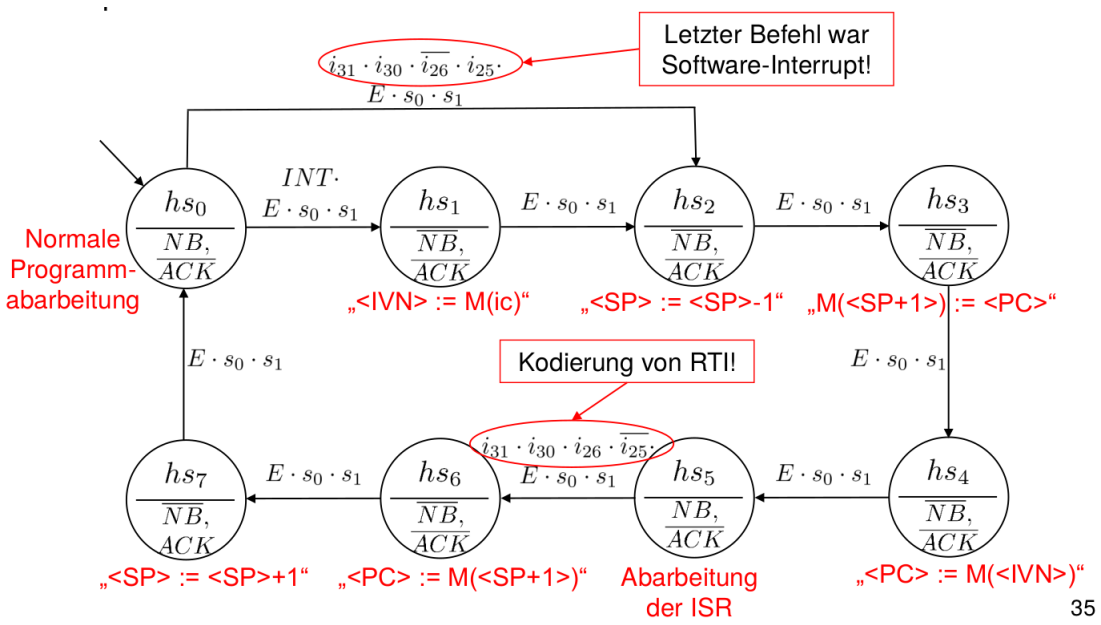

Abbildung 1: Zusatzautomat.

Hinweis: Beachten Sie, dass Sie nach Abarbeitung eines Interrupts nun nicht immer zu hs_0 zurückkehren sollten, sondern gegebenenfalls auch nach hs_5. Beachten Sie außerdem, dass unterbrochene Interruptserviceroutinen ihrerseits wieder unterbrochen werden können etc.. Um sich zu merken, wie viele unterbrochene Interruptserviceroutinen aktuell auf ihre vollständige Abarbeitung warten, können Sie z.B. einen 32-Bit Zähler `CNT` verwenden. Der Einfachheit halber dürfen Sie annehmen, dass dieser Zähler mit 0 initialisiert wird (auch bei Reset der Maschine) und durch ein Signal `CNTinc` bzw. `CNTdec` inkrementiert bzw. dekrementiert werden kann. Weiter dürfen Sie annehmen, dass eine steigende Flanke auf `CNTinc` bzw. `CNTdec` den Zähler inkrementiert bzw. dekrementiert. Außerdem liegt eine 1 an der vom Zähler ausgehenden Leitung `CNTzero` an, sobald der Zählerstand 0 beträgt.

Lösung:

Der erweiterte Zusatzautomat enthält nur neu hinzugekommen Signale. Der Counter wird mit steigenden Flanken inkrementiert/dekrementiert, deshalb muss man die Signale immer wieder zurück setzen. [Bewertung Ermessenspielraum]

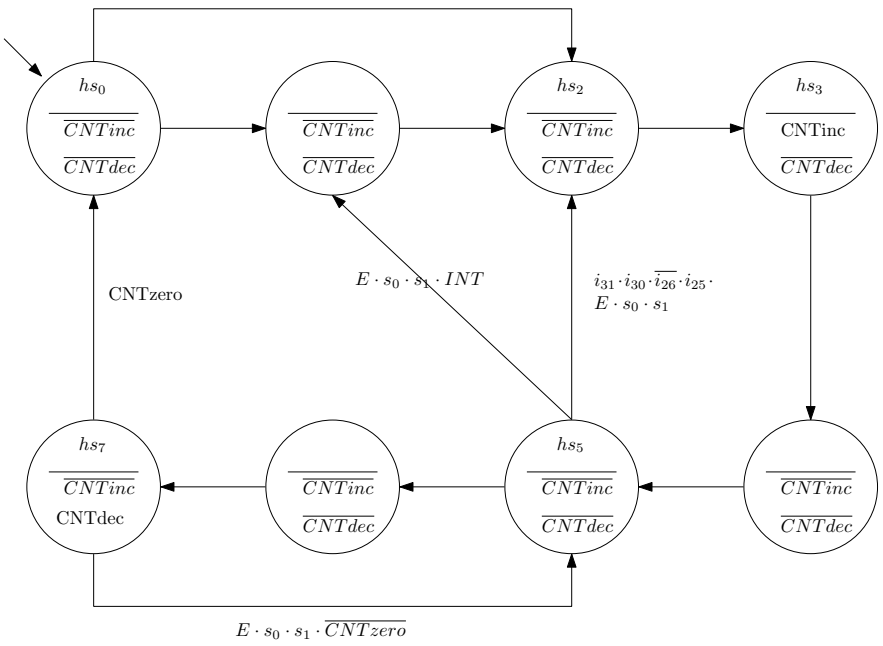

Abbildung 2: Erweiterter Zusatzautomat

Abgabe: als PDF im Übungsportal bis 18.11.2022 um 12:00