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?

800 MHz = 800.000.000 Hz

8 MB/sec = 8.000.000 B/s

8 32 Bit-Worte = 32 Byte

Festplatte aktiv für 50 Taktzyklen

Festplatte Zeit für 8x 32 Bit

↳ 0,000004 s

↳ 0,0004 ms

1000 Taktzyklen / 800.000.000 Hz

↳ 0,000000125 s für

1000 Taktzyklen

Für 5 % der 1000 Taktzyklen

läuft die Festplatte spricht für:

0,000000625 s

Der relative Anteil der CPU-Zeit

die für den Datentransfer benötigt

wird ist:

0,00015625

↳ 1,5625 %

Das ist das Ergebnis, was rauskommt, wenn man statt Mibibtyte mit Megabyte rechnet. Die Assisnten haben immer noch nicht dazugeschrieben, dass Mibi gemeint ist.

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.*

800 MHz = 800.000.000 Hz

8 MB/sec = 8.000.000 B/s

16 KB = 16 000 Byte

$$\frac{16\,000\ \text{Byte}}{8.000.000\ \text{B/s}} = 0{,}002\ \text{s}$$

↳ Zeit für die vollständige Übertragung

CPU für 2000 Takte

↳ 0,0000025 s

$$\frac{0{,}00000025\,s}{0{,}002\,s} = 0{,}00125$$

$$0{,}00125 \cdot \frac{5}{100} = 0{,}00006\,25$$

$$\downarrow \cdot 100$$

Die relative CPU-Zeit ist $0{,}00625\,\%$.

2/4 für die Mühe und ein Grundverständnis ist da, also 2/7

b)

In a) wurden die Prioritäten der einzelnen Interrupts nicht berücksichtigt, d.h. man hat die laufenden Interrupts nicht mit den neuen Interrupts verglichen.

nur wo ist die a)?
-3

doch kann man. Das Problem ist, dass ein man die Priorität eines unterbrochenen Interrupts nicht kennt, der wieder aufgenommen wird und nicht weiß, ob er eine höhre oder niedrigere Priorität hat als ein neu eintreffender.

Zähler = 0 → Interrupt wird an Prozessor weitergeleitet

Zähler ≠ 0

↳ IVN der einkommenden Interrupts wird im Speicher abgelegt.

Die Priorität des zugehörigen IO-Geäts entspricht der Speicheradresse.

ist etwas aufwändig, den Speicher immer von oben (hohe Prirotät) nach unten (niedrige Priorität) durchgehen, aber da es keine weiteren Angaben gab ist das

Nun wird aus dem Speicher die IVN des aktuell höchst priorisiertesten Interrupts und dessen Prioritäten jeweils in PR und IVN abgelegt. Wenn der esste Interrupt auf die CPU gelegt wird, dann maskiesen wir die IVN in der Speicherzelle von diesem Interrupt mit 256 (max. 255 Interrupts, sprich 256 wird nicht verwendet und kann zum

maskieren des Registers verwendet werden).

Interrupt mit Wert 256 in einem Register

↳ Interrupt mit diesem Wert im Stack des Prozessors zur Bearbeitung (noch kein INTA)

Kommt ein INTA zurück setzen wir das Register zurück und da es nur der höchst priorisierteste Interrupt sein kann finden wir die Adresse noch im PR.

Die IVN des h.p. Interrupts befindet sich nun in beiden Registern.

IVN = 256 → wurde bearbeitet
kein höheres Interrupt existiert
CPU wird nicht unterbrochen

IVN $\neq$ 256 -> Interrupt noch nicht bei der CPU signalisiert

IVN ruft dann CPU auf und maskiert den Interrupt Controller mit 256.

Die Methode ermöglicht also, dass höher priorisierte Interrupts den Prozessor unterbrechen aber die niedrigeren nicht.

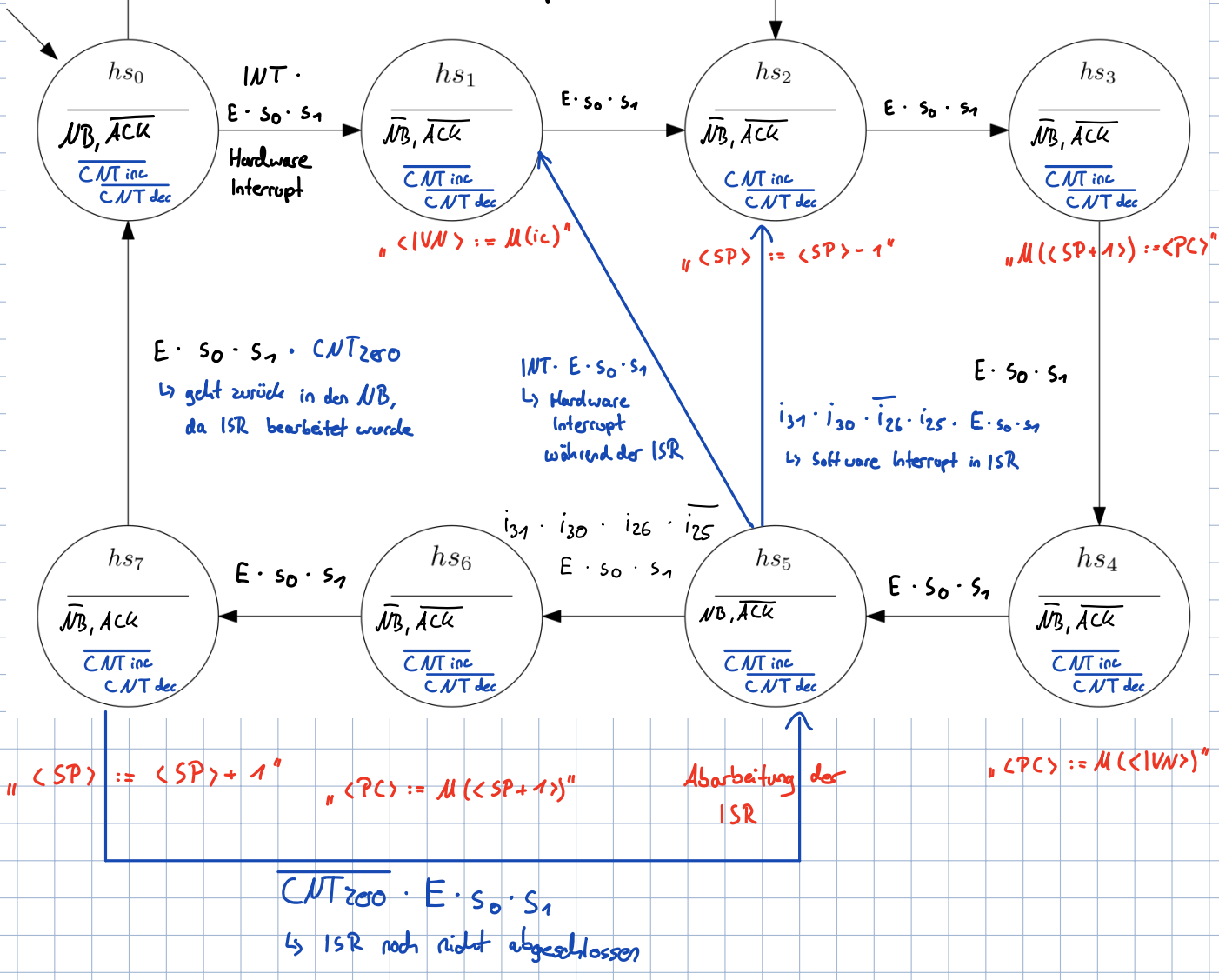
3

$i_{31} \cdot i_{30} \cdot \overline{i_{26}} \cdot i_{25}$

$E \cdot s_0 \cdot s_1$

Software Interrupt

6/6



- hs_2 CNT inc → ISR counter wird erhöht
- hs_7 CNT dec → ISR counter wird verringert