Übungen zu Virtuelle Maschinen, Aufgabe 2

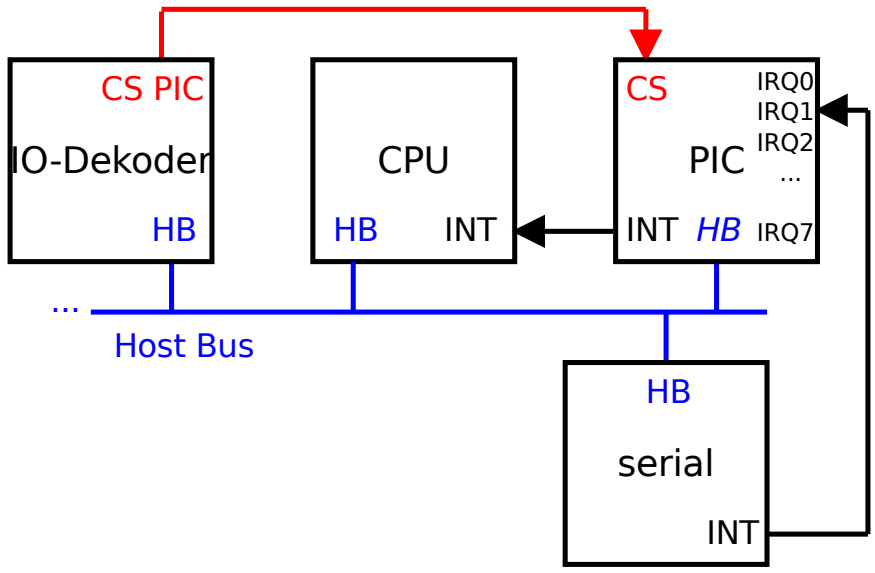
Volkmar Sieh

WS2015/2016

1 Übersicht

In Aufgabe 2 soll die in Aufgabe 1 entwickelte virtuelle Maschine dahingehend erweitert werden, dass Interrupts verarbeitet werden können. Hierzu soll – in verringertem Umfang – ein Interrupt-Controller *PIC 8259a* implementiert werden. Um die Interrupts nutzen zu können muss der CPU-Emulator erweitert werden. Außerdem muss die serielle Schnittstelle dahingehend ausgebaut werden, dass diese Interrupts generieren kann. Da der *PIC 8259a* über I/O-Ports angesteuert wird, muss zusätzlich der *Host-Bus* verändert werden. Die I/O-Dekodierung soll eine neue IO-Dekoder-Komponente erledigen.

Folgendes Bild zeigt eine grobe Übersicht darüber, wie die zu bearbeitenden Komponenten verbunden sein sollen.



2 Organisatorisches

Bitte senden Sie Ihre Lösung bis zum 6.1.2016 per Mail an i4vm@cs.fau.de. Die Projektdateien mit Implementierungsvorschlägen und Testprogrammen finden Sie unter https://www4.cs.fau.de/Lehre/WS15/V_VM/Uebungen/aufgabe2.tar.gz.

3 Hardware-Simulatoren

3.1 CPU-Simulator

Um Interrupts nutzen zu können, muss der CPU-Simulator um Interruptbehandlung erweitert werden. Hierzu sind die Befehle **lidt, iret, sti** und **cli** zu implementiern. Des Weiteren soll – sofern ein Interrupt vorliegt – die entsprechende Behandlungsroutine ausgeführt werden.

Um mit dem *PIC 8259a* zu kommunizieren, werden schließlich die Befehle **inb** und **outb** benötigt.

Bei Aufgabe 1 stellte die `hlt`-Instruktion eine Abbruchbedingung dar. Tritt jedoch eine Unterbrechung während einer `hlt`-Instruktion auf, so wird zunächst die Unterbrechung behandelt und danach die nächste Instruktion ausgeführt. Überlegen Sie sich daher eine sinnvolle alternative Strategie, um die Emulation zu beenden.

3.2 IO-Dekoder

Der IO-Dekoder soll mit dem Host-Bus verbunden werden. Handelt es sich bei einem Buszugriff um einen I/O-Transaktion, so soll der IO-Dekoder die angelegte Port-Adresse auswerten, und für ihn bekannte Komponenten die entsprechende Chip-Select-Leitung der Komponenten aktivieren. Des weiteren sollen – je nach Komponente – die notwendigen Adressleitungen an ebendiese weitergeleitet werden.

In Aufgabe 2 gibt es nur einen einzigen Adressbereich, der dekodiert werden muss: Ein I/O-Zugriff auf Port-Adresse `0x0020` – `0x003f` soll die Chip-Select-Leitung des *PIC 8259a* aktivieren. Die unterste Adressleitung soll außerdem an den Eingang `A0` angeschlossen werden.

3.3 PIC: Intel 8259a

Der *Programmable Interrupt Controller 8259a* soll in eingeschränktem Umfang entsprechend dem Originalchip implementiert werden. Folgende Eigenschaften sollten realisiert werden:

- Fully nested mode
- 8086 Mode
- Edge Triggered Interrupts
- Möglichkeit, Interrupts mittels Interruptmaske zu maskieren
- Automatic end of Interrupt-Modus
- hierzu passende Initialisierungsmöglichkeit durch I/O-Befehle
- Verändern der Interruptmaske mittels I/O-Befehlen

Der *PIC 8259a* darf direkt mit dem Host-Bus verbunden werden. Jedoch soll die Adressdekodierung durch den IO-Dekoder erfolgen. Die Interrupt-Leitung der serielle Schnittstelle sollte an den Eingang `IRQ 1` angeschlossen werden.

Die Dokumentation des *Intel 8259a* finden Sie unter http://pdos.csail.mit.edu/6.828/2005/readings/hardware/8259A.pdf.

3.4 serielle Schnittstelle

Die serielle Debug-Schnittstelle soll um die Fähigkeit erweitert werden, Zeichen von dem Gastsystem mittels Tastatureingabe entgegen zu nehmen. Verwenden Sie hierfür `stdin`.

Wurde ein Zeichen empfangen, so soll ein Interrupt signalisiert werden. Der Empfang von weiteren Zeichen soll in dieser Zeit blockiert werden.

Durch Lesen der Adresse `0x00` der seriellen Schnittstelle kann das empfangen Zeichen ausgelesen werden. Der Interrupt soll daraufhin zurückgenommen werden und die serielle Schnittstelle ist bereit, ein weiteres Zeichen von `stdin` zu empfangen.

Wurde kein Zeichen empfangen, so ist Lesen der Adresse `0x00` undefiniert. Jedoch sollte sich dies nicht auf die weitere Funktionsfähigkeit der seriellen Schnittstelle auswirken.

Hinweis: In den Projektdateien befindet sich eine Implementierung der erweiterten seriellen Schnittstelle, die Sie nach Belieben verwenden können.

4 Verbindungen und Busse

4.1 Host-Bus

Der Host-Bus soll dahingehend erweitert werden, dass ein Interrupt-Acknowledge-Zyklus simuliert werden kann. Außerdem soll der Host-Bus die Möglichkeit bieten, zwischen Speicher- und I/O-Zugriff zu unterscheiden. Ein I/O-Zugriff soll hierei auf 16-Bit Adressbreite beschränkt sein. Auch hier ist – analog zum Speicherzugriff – der Zugriff auf ein Byte an Daten beschränkt.

Tip: Nutzen Sie eine sinnvolle Abstraktion bei der Implementierung der neuen Buszyklen.

4.2 Verbindung zwischen IO-Dekoder und PIC

Überlegen Sie sich eine sinnvolle Variante, den *PIC* für den Interrupt-Acknowledge-Zyklus zum einen mit dem Host-Bus zu verbinden, zum anderen jedoch die Adressdekodierung und Ansteuerung der Chip-Select-Leitung durch den IO-Dekoder erledigen zu lassen.

4.3 Interrupt-Leitungen

Um die Interrupt-Leitung der seriellen Schnittstelle mit dem *Programmable Interrupt Controller* zu verbinden, müssen Sie auch hierfür eine „Leitung“ implementieren, die dies bewerkstelligt. Überlegen Sie sich auch hier eine sinnvolle Abstraktion.

Hinweis: In den Projektdateien befindet sich eine mögliche Variante für Interrupt-Signale in den Dateien `sig_boolean.[ch]`.

4.4 Projektdateien

Im Folgenden finden Sie eine Kurzübersicht über die Projektdateien.

Komponenten/Busse/Hilfsfunktionalität

- `glue-io.[ch]`: Hilfsfunktionen, um File-Deskriptoren auf Aktivität zu überwechen. `glue_io_step()` muss hierfür regelmäßig aufgerufen werden.
- `comp/serial_ctrl.[ch]`: Eine mögliche Implementierung der seriellen Schnittstelle. Verwendet `glue-io`.

- `bus/sig_boolean.[ch]`: Implementierung einer Leitung, welche den Wert `true` oder `false` annehmen kann. Callbacks werden nur bei Änderung des Zustands aufgerufen.

Test-Programme

- `test/test-pic/*`: Testprogramm für den *Programmable Interrupt Controller*. Das `Makefile` erzeugt ein `test-pic.rom`, welches als „Bios“ verwendet werden kann.
- `test/test-pic/x86_cdrom/*`: Erzeugt ein bootbares CDROM Iso-Image, welches in einer virtuelle Maschine oder auf einem echten System verwendet werden kann. Hauptsächlich dafür nützlich, um die Test-programme des PICs zu testen.