Kap.2
Befehlsschnittstelle

Prozessoren,
externe Sicht

- **2.1** elementare Datentypen, Operationen
- **2.2** logische Speicherorganisation
- **2.3** Maschinenbefehlssatz
- **2.4** Klassifikation von Befehlssätzen
- **2.5** Unterbrechungen
- **2.6** Prozesse

2.5 Unterbrechungen

- Situationen während der Bearbeitung von Programmen, die besondere Behandlung erfordern

Exceptions / Traps

Definition
prozessorinternes Ereignis, welches den normalen Programmablauf unterbricht, heißt **Exception**.

Beispiele
- arithmetischer Überlauf
- Division durch Null
- ungültiger Befehlscode
- ...

Wie werden Exceptions behandelt?

Szenario

Der gerade aktuelle Prozessorbefehl verursachte die Ausnahmesituation.

Behandlung

Je nach Art der Exception und des Prozessors gibt es zwei Möglichkeiten:

- Ein bestimmtes *Flag* in einem *Register* wird gesetzt.
  - Der Programmierer muss den Inhalt des Flags im folgenden Programm auswerten und entsprechend reagieren.
- Es wird eine definierte Ausnahmeroutine für eine bestimmte Exception gestartet (siehe Interrupt).

Interrupts

Definition

externes Ereignis, welches den Programmablauf verändert, heißt **Interrupt**

Interrupts werden durch asynchron zum Prozessor laufende externe Geräte erzeugt und dienen zur Kommunikation mit dem Prozessor.

Beispiele:

- Eingabe über Tastatur/Maus
- Grafikkarte benötigt neue Daten
- Datenübertragung (Festplatte,Netzwerk) beendet
- ...

Interrupts und Exceptions

Zur Namensgebung eine Bemerkung

Interrupts wurden ursprünglich für die Behandlung von Anfragen externer Geräte eingeführt. Derselbe Mechanismus kann auch zur Behandlung von intern generierten Exceptions genutzt werden.

Ereignis	ausgelöst	Bezeichnung
arithmetischer Überlauf	intern	Exception
undefinierter Opcode	intern	Exception
I/O-Request	extern	Interrupt
Hardware-Fehlfunktion	extern	Exception oder Interrupt

Interrupts und Exceptions

Wesentlicher Unterschied

- Exceptions sind **synchron** zum ausgeführten Programm,
- Interrupts hingegen **asynchron**

➢ Exceptions sind reproduzierbar, indem bei gleichem Systemstatus das Programm mit den gleichen Eingaben wiederholt gestartet wird.

Bearbeitung einer Exception

- wie die Interrupts **durch das Betriebssystem**
  (... siehe nächste Folie)

- **durch die Hardware**
  - **Vorteil gegenüber Softwarelösung**
    - Überprüfung auf Vorliegen einer Exception kann in der Regel im gleichen Zyklus erfolgen, in dem der entsprechende Maschinenbefehl abgearbeitet wird.
    - schnellere Abarbeitung der Befehle

Wie werden Interrupts behandelt?

- Ein Interrupt wird asynchron zum gerade laufenden Programm ausgelöst und erfordert eine Behandlung durch das Betriebssystem.

- Es wird eine spezielle Ausnahmeroutine, die **Unterbrechungsbehandlungs-routine** oder auch **Interrupthandler** genannt, aufgerufen.

Beispiel
Der Nutzer drückt eine Taste der Tastatur
- CPU unterbricht aktuelles Programm,
- Tastaturcode wird überprüft,
- weitere Aktionen (z.B. nach Ctrl-Alt-Del) oder Speicherung im Tastaturpuffer,
- Tätigkeit wird an unterbrochener Stelle im alten Zustand fortgesetzt

Interrupthandler: Ursache der Ausnahme bestimmen

Das Betriebssystem benötigt Informationen über die Ursache der Ausnahme.

Zwei mögliche Varianten:

- **Causeregister**
  Bei Auftreten eines Interrupts wird ein bestimmtes Bit (*Flag*) im Causeregister gesetzt. Dieses kann vom Handler abgefragt werden.
- **Vectored Interrupt**
  Dem Betriebssystem wird ein Index aus der **Interrupttabelle** bereitgestellt, in der die Adressen für die zu speziellen Interrupts gehörenden Routinen (**ISR**: **I**nterrupt-**S**ervice-**R**outine) gespeichert sind. Im Ausnahmefall wird die entsprechende Routine aufgerufen und durch den Befehl **RTI** (**R**eturn-**f**rom-**I**nterrupt) verlassen.

MIPS: Exception-Codes im Cause-Register

- CPU, der sogenannte Koprozessor 0, verfügt über spezielle Register für die Ausnahmebehandlung

Register	Nr.	Erläuterung
Status	12	Interrupt Maske und Enable Bits
Cause	13	Exception Type
EPC	14	Adresse des Befehls, der die Exception auslöste

Im Falle einer Exception erfolgt ein Sprung an eine von der CPU-Hardware definierte Adresse (SPIM: 0x8000 0080). Der Exception Code kann nun dem Cause-Register des Koprozessors 0 entnommen werden (Bit 5-2)

MIPS
Exception-Codes

Code	Name	Erläuterung
0	**INT**	**Int**errupt (externe Unterbrechung)
4	**ADDRL**	**Addr**ess error during **l**oading
5	**ADDRS**	**Addr**ess error during **s**toring
6	**IBUS**	Busfehler beim Laden des Befehls
7	**DBUS**	Busfehler beim Speichern von Daten
8	**SYSCALL**	Betriebssystemaufruf durch SYSCALL
9	**BKPT**	**B**rea**kp**oin**t** erreicht
10	**RI**	**R**eserved **I**nstruction: Verwendung eines priviligierten Befehls
12	**OVF**	**Ov**er**f**low: arithmetischer Überlauf

Interrupt-Tabelle (Vektortabelle)

- enthält die Startadressen der **ISR**,
  - jede Zeile muss jeweils eine 4-Byte Adresse aufnehmen können
- Die Indizes werden als **Interrupt-Vektornummern** (**IVN**) oder mit **INT** bezeichnet
- liegt oftmals auf den ersten Adressen des Hauptspeichers

INTEL CPU Exception Table

INT	Function
0	Divide by zero
1	Single step (→ debugging)
2	Non-maskable interrupt (NNI)
3	Breakpoint
4	Overflow
5	Bound range exceeded
6	Invalid opcode
7	Coprocesor not available
8	Double fault exception
9	Coprocessor segment overrun
A	Invalid task state segment (?)
B	Segment is not present
C	Stack exception (Überlauf)
D	General protection exception
E	Page fault
F	Reserved
10	Coprocessor error

IBM PC (Hardware) Interrupt-Tabelle

INT	Function
0	Timer (55 ms – Intervall)
1	Keyboard
3	COM2 and COM4 (serieller Port)
4	COM1 and COM3 (serieller Port)
5	LPT2 (paralleler Port)
6	Floppy Disk
7	LPT1 (paralleler Port)
8	Real Time Clock (CMOS Clock)
C	PS2 – Mausport
D	Numeric Coprocessor Error
E	IDE0 – Festplatte / CDROM / ...
F	IDE1 – Festplatte / CDROM / ...

Maskierte / Nichtmaskierte Interrupts

Prozessoren unterscheiden zwischen
maskierten und nichtmaskierten Interrupts:

- **maskierter Interrupt:**
  - wird nur dann ausgeführt, wenn das **I**nterrupt **E**nable Bit (**IE**) in einem dafür vorgesehenen Register gesetzt ist. (MIPS: Bits 8-15 im Status-Register)
  - Der Nutzer kann den Registerinhalt ändern.
- **nichtmaskierter Interrupt (NMI):**
  - wird auf jeden Fall ausgeführt. Bei NMI handelt es sich üblicherweise um Ausnahmesituationen, die die Funktionalität des Systems gefährden.

Interrupt in Interrupt-Service-Routine

Szenario

Es besteht eine gewisse Wahrscheinlichkeit, daß innerhalb einer ISR (z.B. durch ein I/O-Device verursacht) ein weiteres I/O-Device seinen Interrupt auslöst!

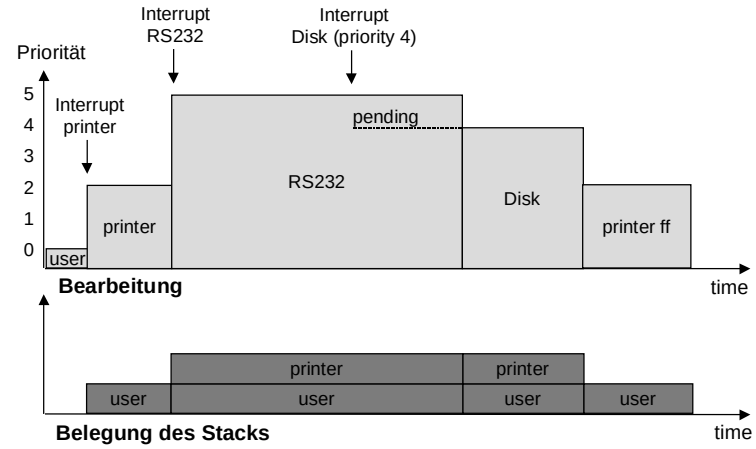
Mögliche Lösungen

- Erster Befehl in der ISR ist Ausmaskierung aller Interrupts
  - ➢ keine weiteren Interrupts möglich
- Vergabe von Interruptprioritäten derart, daß ein kritisches I/O-Device eine höhere Priorität, als ein weniger kritisches Device hat.

Interrupt-Priorität

- Falls ein Device mit Priorität *n* einen Interrupt auslöst, so startet die ISR mit Priorität *n* . Wird ein Interrupt mit einer Priorität *m>n* ausgelöst, so wird die aktuelle ISR unterbrochen und eine neue ISR der Priorität *m* sofort gestartet.
- Der Versuch eines Gerätes mit einer niedrigeren Priorität als *n* die ISR zu unterbrechen, wird ignoriert und der Interruptcode wird zur späteren Ausführung zwischengespeichert (**pending Interrupt**).
- Nach Abarbeitung der ISR geht das System auf die user-Priorität (**userpriority 0**) zurück.

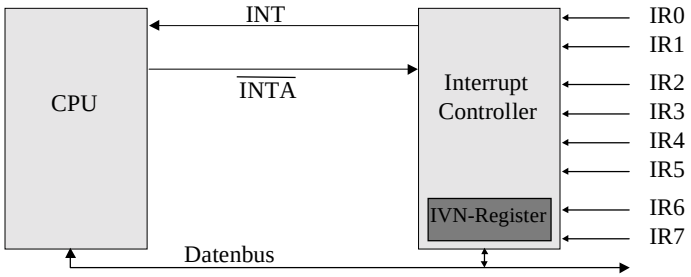
Interrupt-Prioritäten: Illustration



Interrupt-Prioritäten

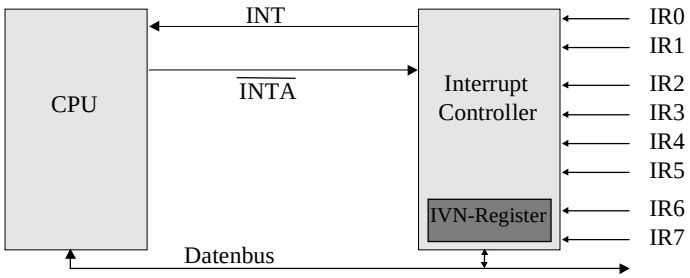
- Prozessoren mit nur einem **Interruptlevel** (x86) ist es selbst nicht möglich, verschiedenen Geräten verschiedene Prioritäten zuzuordnen.
- Abhilfe schafft in diesem Fall ein **Interrupt-Controller**, der extern mit dem Prozessor verbunden wird (z.B. 8259A).
- Dieser externe Controller verwaltet die Prioritäten. Für den Fall dass keine ISR aktiv ist oder ein Interrupt eine laufende ISR unterbrechen darf, signalisiert er dem Prozessor einen Interrupt, falls ein solcher bei ihm anliegt
- Da er vor dem Auslösen eines weiteren Interrupts mit niedriger Priorität wissen muss, ob die ISR beendet wurde, benötigt er eine Rückmeldung vom Prozessor.

Interrupt-Controller



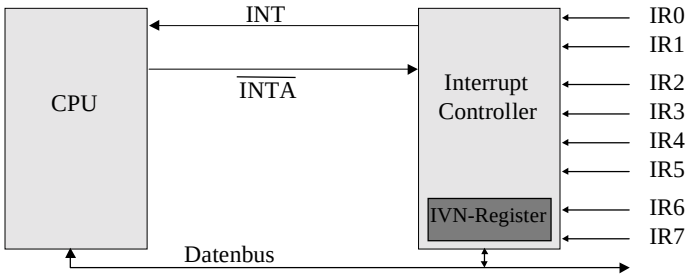
- I/O-Device *j* teilt dem Controller durch Setzen der Signalleitung IR*j* mit, daß eine Ausnahmesituation vorliegt.
- Die Kennung des Interrupts wird in das Register **IVN** eingetragen.
- Der Controller gibt die Meldung durch Setzen des Signals INT an die CPU weiter.
- Ist die CPU bereit, so teilt sie dies dem Controller durch Setzen des Signals INTA mit.
- Der Interrupt-Controller legt den Inhalt seines IVN-Registers auf den Datenbus.

Interrupt-Controller ff



- Der Interrupthandler der CPU nimmt den Inhalt des IVN-Registers vom Bus und rettet ihn in einem eigenen Register. Hiermit weiß der Prozessor, welches periphere Gerät den Interrupt ausgelöst hat.
- Vor der Abarbeitung des Interrupts rettet die CPU den Befehlszähler, sowie Steuer- und Statusregister auf dem Stack.
- Die CPU schlägt in der Interruptvektortabelle nach, um mit Hilfe der IVN die Startadresse der entsprechenden Interrupt-Serviceroutine (ISR) zu finden.

Interrupt-Controller ff



- Die CPU führt die ISR aus. Falls sie ISR Register verwendet, so muß sie deren Inhalt vorher auf dem Stack sichern.
- Vor Beendigung der ISR holt diese die geretteten Registerinhalte vom Stack.
- Die CPU stellt den ehemaligen Zustand (Befehlszähler, Steuer- und Statusregister) wieder her.
- Dem Interrupt-Controller wird die Abarbeitung des Interrupts mitgeteilt.

IBM PC (Hardware) Interrupt-Tabelle

geordnet nach Priorität

INT	function
0	Timer (55 ms Intervall)
1	Keyboard service
8	Real time clock
9	-- Einsteckplätze
10	-- Einsteckplätze
11	-- Einsteckplätze
12	PS2 – Mouseport
13	Numeric coprocessor error
14	IDE0 – Festplatte / CDROM / ...
15	IDE1 – Festplatte / CDROM / ...
3	COM2 oder COM4 (serieller Port)
4	COM1 oder COM3 (serieller Port)
5	LPT2 (paralleler Port)
6	Floppy Disk
7	LPT1 (paralleler Port)