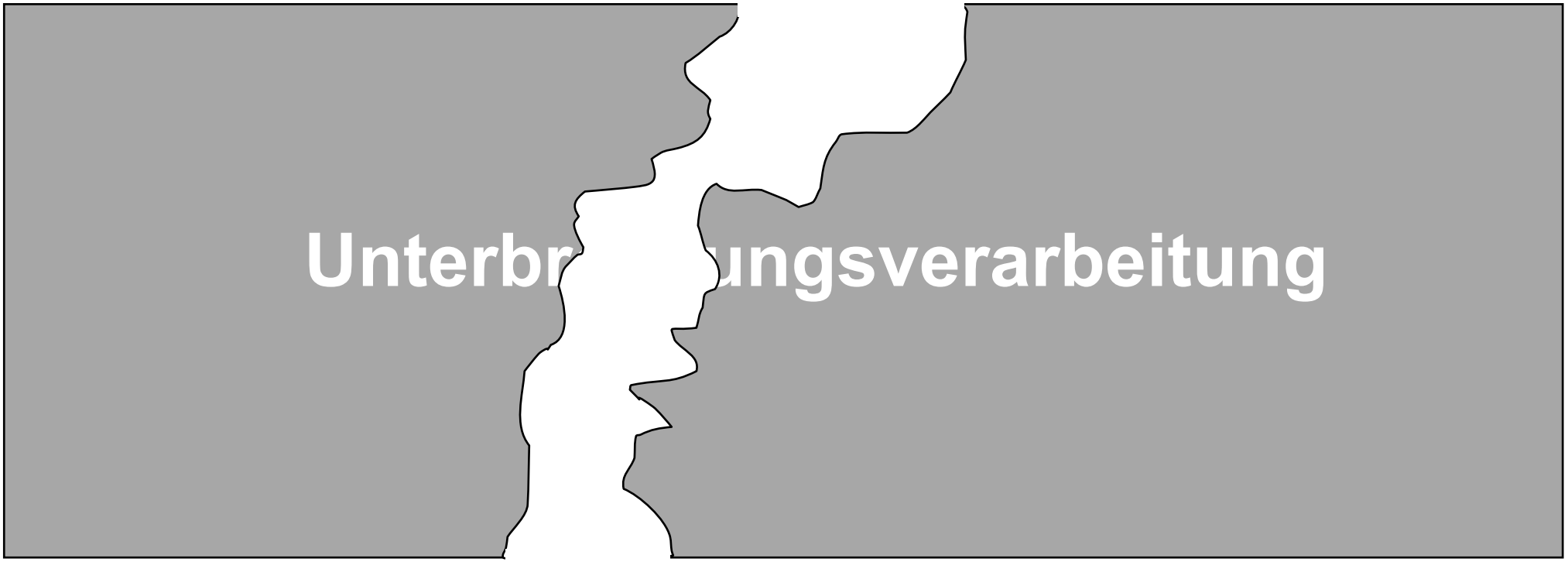

Unterbrechungsverarbeitung

Def.: Interrupt (Systembezogene Unterbrechung):

Ein zur Programmausführung *asynchrones Ereignis*, das die sequentielle Programmausführung unterbricht und die Kontrolle an ein spezielles Programm zur Behandlung des Ereignisses übergibt.

Beispiele: RESET, Power Fail, Echtzeit-Uhr, Steuerung der Ein- und Ausgabe

Welches Problem wird durch die Interrupts gelöst ?

Allgemein: Synchronisation von Prozessoraktivität und externen Ereignissen.

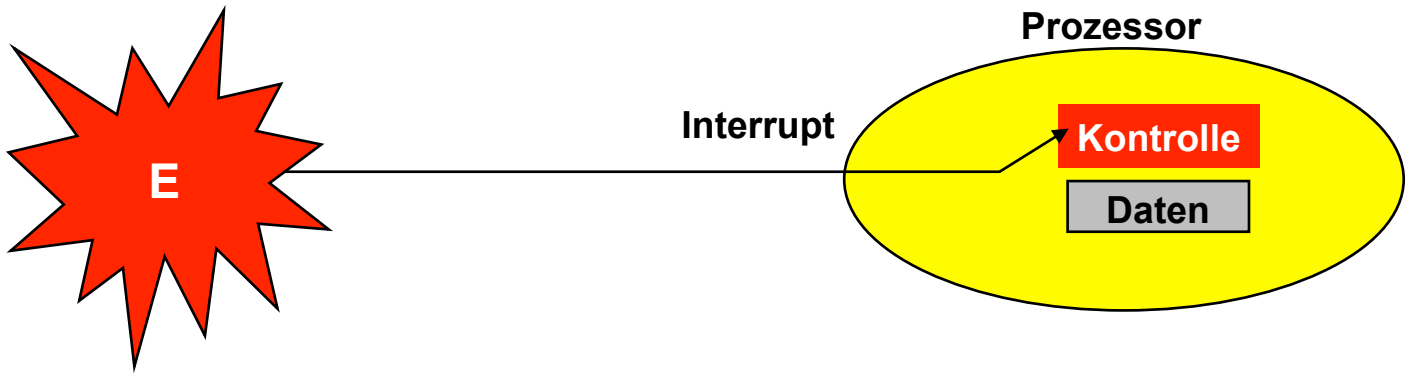
Bisherige Lösung: Explizite Statusabfrage (Polling)

- **explizite Statusabfrage erzeugt einen periodischen Zusatzaufwand**
- **dadurch entsteht der Zielkonflikt zwischen:**

  Häufigkeit der Abfrage und Verzögerung der Bearbeitung eines externen Ereignisses

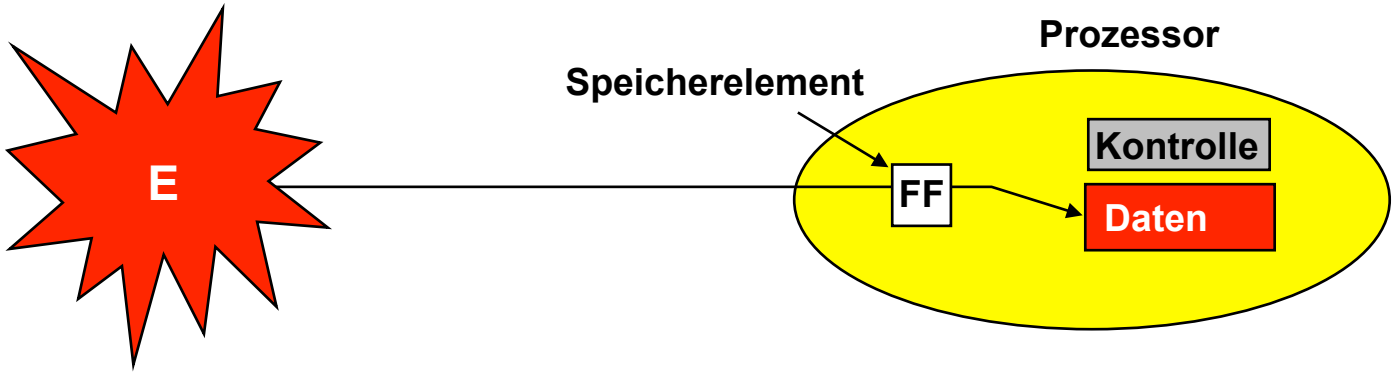
Interrupt kann als Polling auf der Hardware-Ebene betrachtet werden, wo nach jeder Instruktion automatisch überprüft wird, ob eine Unterbrechungsanforderung vorliegt.

Was ist das Problem bei Interrupts?



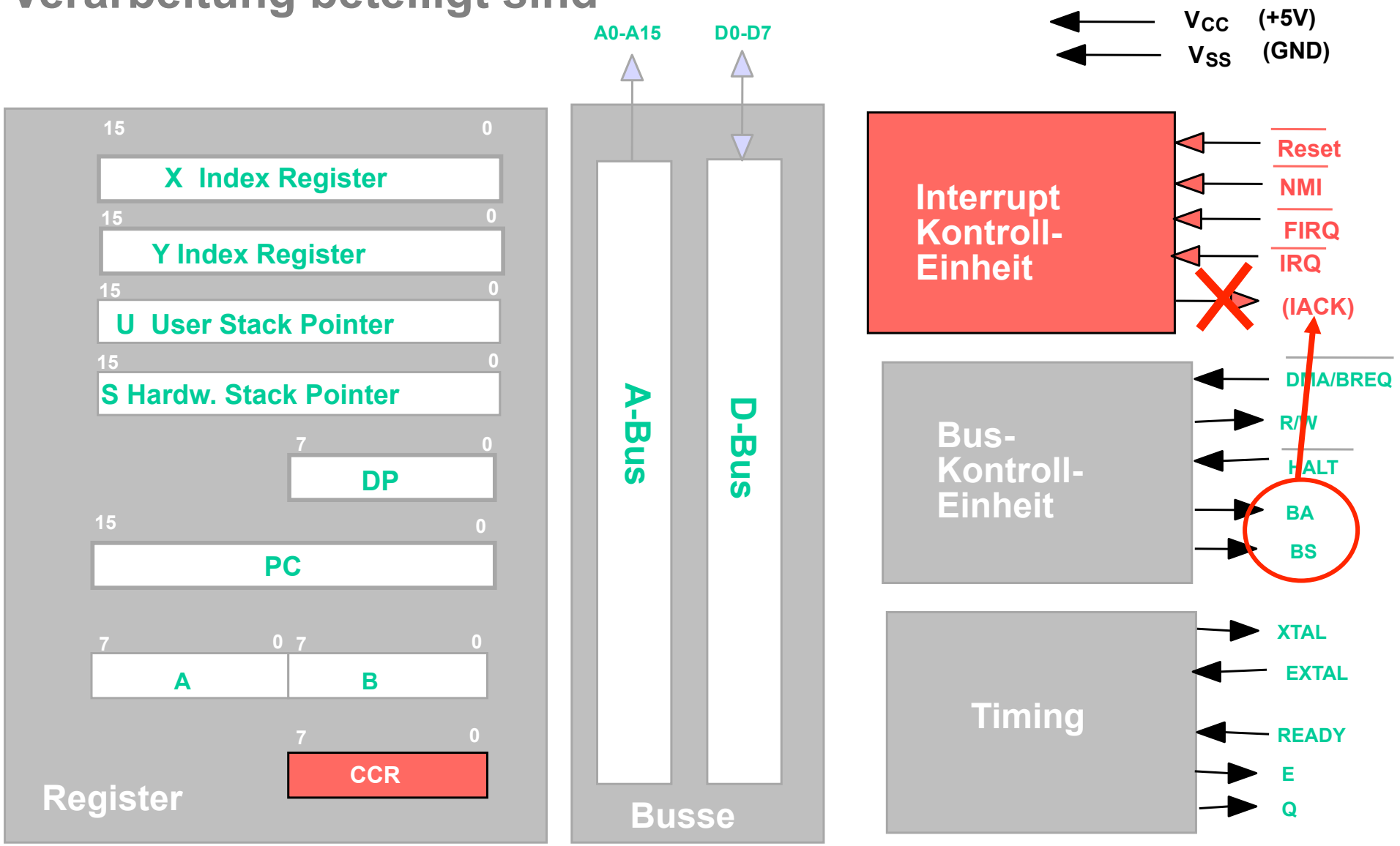
Interrupt-gesteuerte Verarbeitung

externes Ereignis



Daten-gesteuerte Verarbeitung

Komponenten des Motrola 6809 die speziell an der Interrupt-Verarbeitung beteiligt sind

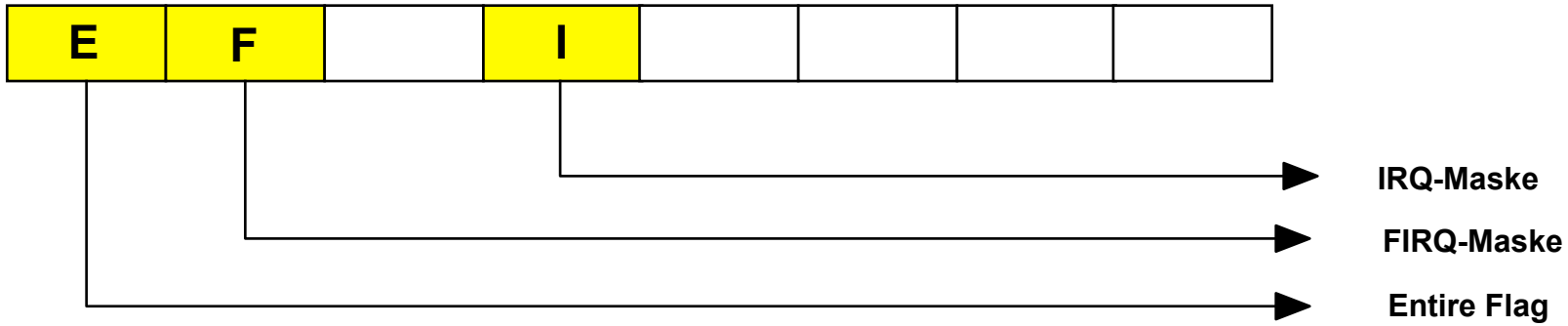


Interruptklassen im MC 6809

RESET:	Löscht den gesamten Prozessorstatus, Transferiert die Kontrolle an die Routine, deren Adresse in FFFE und FFFF steht.
NMI:	Non-Maskable-Interrupt. Kann nicht ausgeschaltet (d.h. im CCR maskiert) werden. Sonst wie IRQ.
SWI:	Software Interrupt (Trap). Die SWI Instruktion erzeugt genau dieselbe Reaktion wie ein Hardware Interrupt. Der Unterschied liegt lediglich darin, daß er nicht durch ein externes Signal, sondern durch den entprechenden Befehl ausgelöst wird, d.h. im strengen Sinne kein asynchrones Ereignis darstellt. Wird beim Debugging ausgenutzt, zur Behandlung von Ausnahmebedingungen und zur Emulation nicht vorhandener Hardware (z.B. Coprozessor-Traps).
IRQ:	Interrupt Request. Alle Register werden auf den Systemstack gerettet. Auf externen Leitungen (Bus Available und Bus State) wird angezeigt, daß der Interrupt akzeptiert wurde. Kontrolle wird zu der, im entsprechenden Interrupt Vektor angegebenen Routine transferiert.
FIRQ:	Fast Interrupt: Nur der PC und das CCR werden gerettet. Sonst wie IRQ.
SWI2 / SWI3:	Wie SWI. Unterschied, I- und F-Bit im CCR werden NICHT modifiziert.

Aktivierung und Status der Unterbrechungsbehandlung (Interrupt)

Condition Code Register



E: Entire Flag, wird bei RTI ausgewertet, um den Registerstatus vor Interrupt wiederherzustellen.
E =1: der gesamte Registersatz ist auf dem Stack gerettet worden
E=0 : PC und CC wurden gerettet.

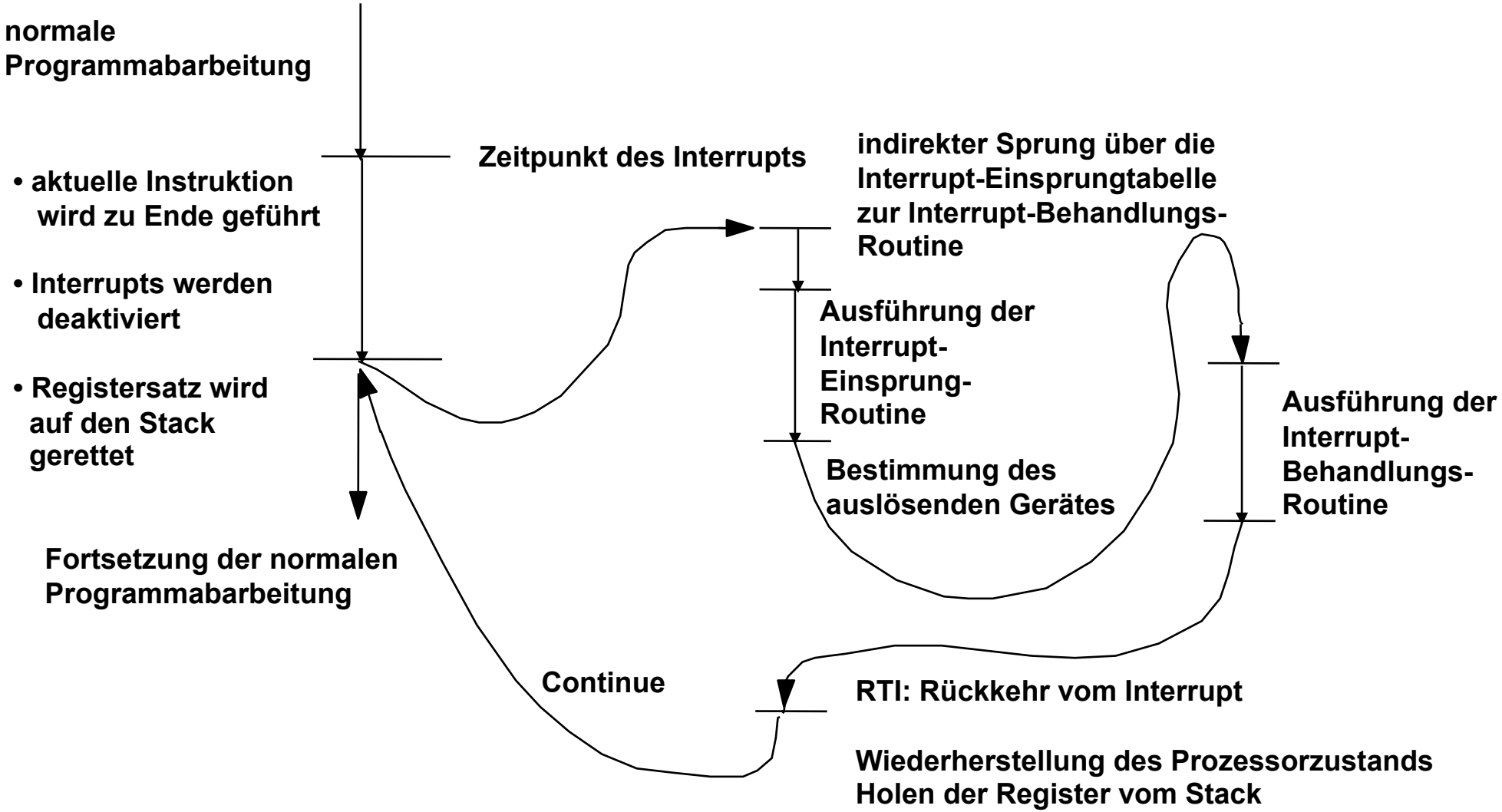
F: Fast Interrupt Request (FIRQ) Maske.
F=1 Interrupts auf der FIRQ-Leitung werden nicht vom Prozessor erkannt und behandelt.
NMI, FIRQ, SWI1 und Reset setzten F=1. IRQ, SWI2 und SWI3 ändern F nicht.

I: Interrupt Request (IRQ) Maske.
I=1 Interrupts auf der IRQ-Leitung werden nicht vom Prozessor erkannt und behandelt.
NMI, IRQ, FIRQ, SWI1 und Reset setzten F=1. SWI2 und SWI3 ändern F nicht.

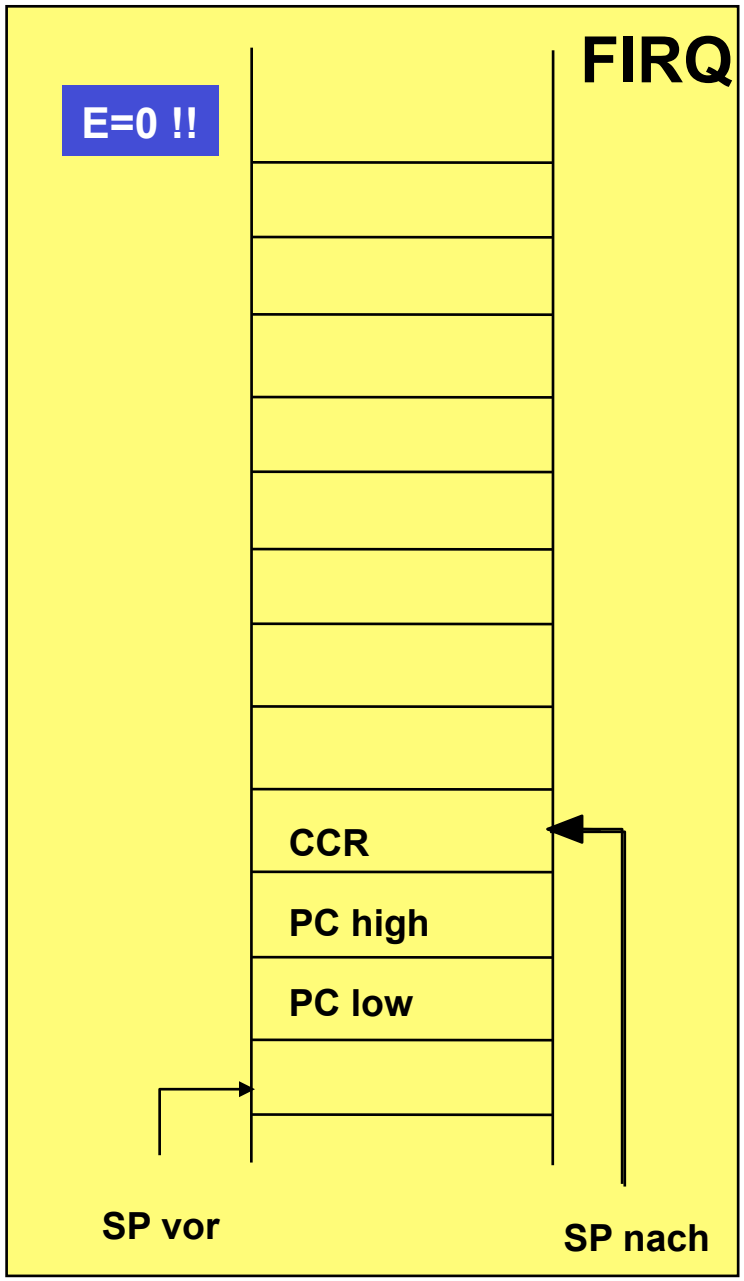
Schritte bei der Behandlung von Interrupts:

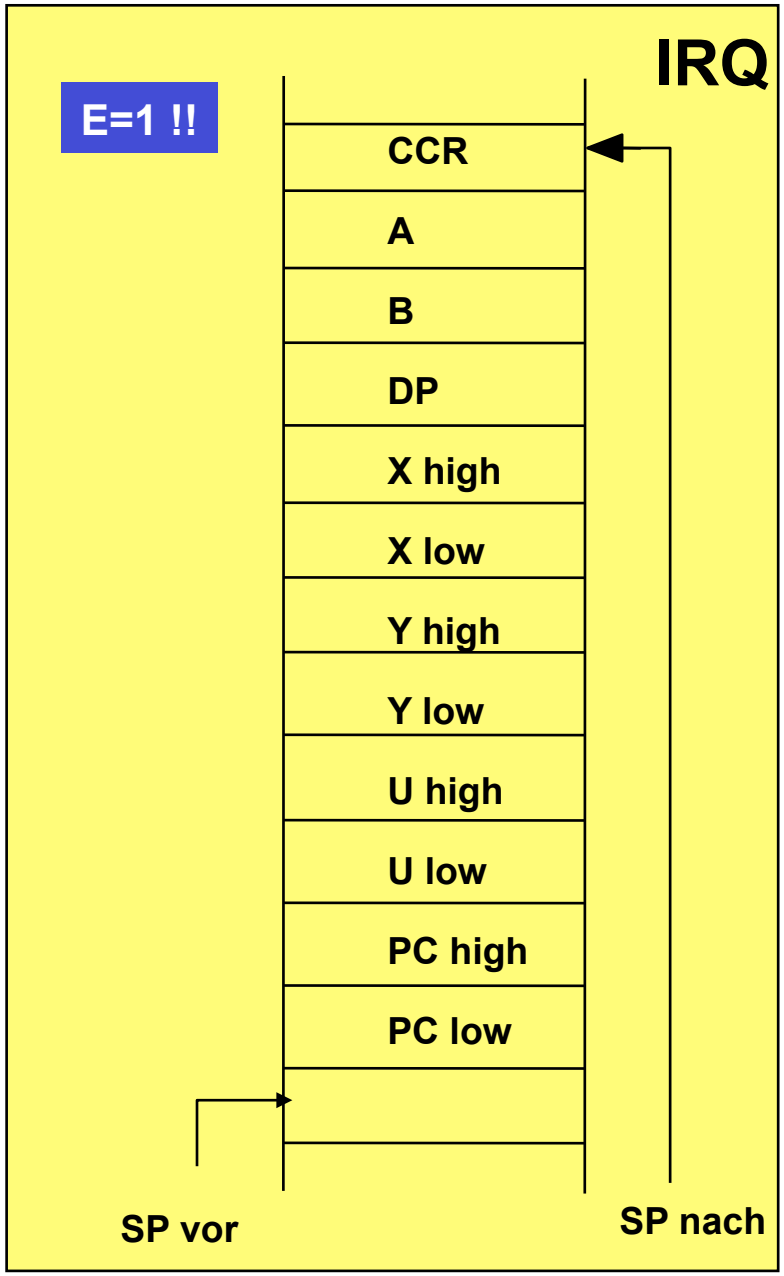
- **Initialisierung**
  - Initialisieren der Interrupt-Einsprungtabelle
  - Aktivieren der entprechenden Interrupts
- **Unterbrechnungsbearbeitung (Interrupt Service)**
  - Erkennen der Interrupt-Anforderung (Interrupt-Request)
  - Beenden der aktuellen Instruktion
  - Verhindern, daß weitere Requests angenommen werden. (Ausnahmen RESET und Power Fail)
  - Retten des Prozessorzustands, mindestens den PC, das CCR und die Register, die durch die Interruptbehandlung überschrieben werden
  - Bestimmen der Routine zur Interruptbehandlung
  - Transfer der Kontrolle an die Routine zur Interruptbehandlung
  - Festellen, wer den Interrupt ausgelöst hat
  - Ausführen der entprechenden Routine zur Interruptbehandlung
  - Ausführung des Rücksprungs aus der Interruptroutine
  - Wiederherstellung des Prozessorzustands
  - Transfer der Kontrolle an das unterbrochene Programm

Interrupt Bearbeitung



Stack - Belegung





Aufwand für die Interrupt-Behandlung:

	Zyklen
Normale Antwortzeit für IRQ und NMI	21
Normale Antwortzeit für FIRQ	12
Antwortzeit für alle Interrupts nach CWAI	9
Ausführungszeit für CWAI	20
Ausführungszeit für RTI, wenn das E-Flag gesetzt ist	15
Ausführungszeit für RTI, wenn das E-Flag nicht gesetzt ist	6
Ausführungszeit für SWI	19
Ausführungszeit für SWI2 und SWI3	20

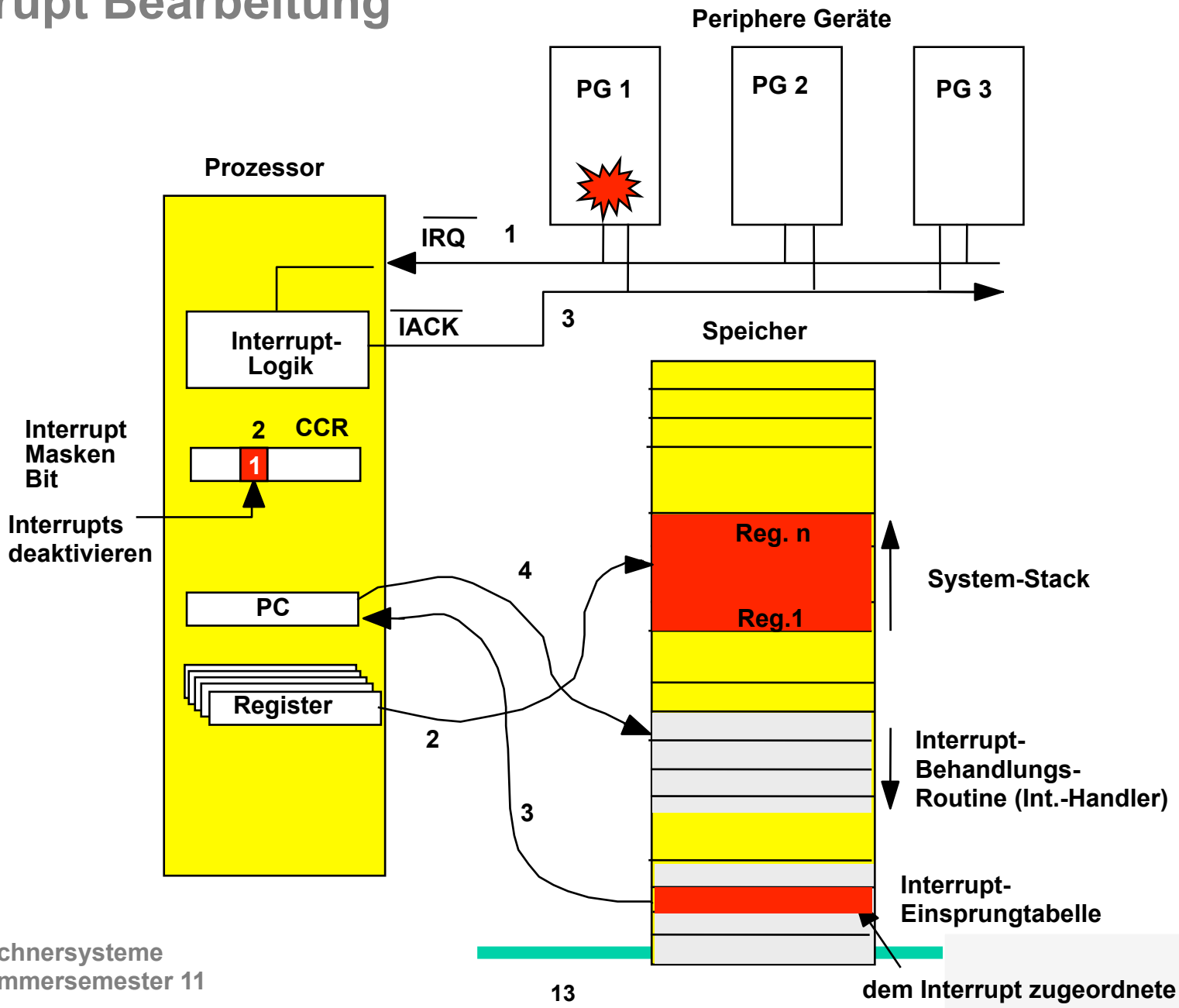
CWAI (Clear and Wait for Interrupt) schreibt den gesamten Prozessorstatus auf den Stack und wendet eine Maske auf die Interrupt Bits des CC-Register an. Das E-Flag wird gesetzt.
Maske: FF:alle disabled, EF: IRQ enabled, BF: FIRQ enabled, AF: FIRQ und IRQ.

6809 Interrupts und Speicherbelegung der Interrupt Vektoren:

Interrupt	MSB	LSB
$\overline{\text{RESET}}$	FFFE	FFFF
NMI	FFFC	FFFD
SWI	FFFA	FFFB
$\overline{\text{IRQ}}$	FFF8	FFF9
FIRQ	FFF6	FFF7
SWI2	FFF4	FFF5
SWI3	FFF2	FFF3
reseviert	FFF0	FFF1

FFFF xx	Einsprungadresse für RESET
FFFE xx	
FFFD xx	Einsprungadresse für NMI
FFFC xx	
FFFB xx	Einsprungadresse für SWI
FFFA xx	
FFF9 xx	Einsprungadresse für IRQ
FFF8 xx	
FFF7 xx	Einsprungadresse für FIRQ
FFF6 xx	
FFF5 xx	Einsprungadresse für SWI2
FFF4 xx	
FFF3 xx	Einsprungadresse für SWI3
FFF2 xx	
FFF1	reserviert
FFF0	

Interrupt Bearbeitung



- **Welches Gerät hat den Interrupt ausgelöst ?**
- **Wie kann ein wichtiger Interrupt von einem weniger wichtigen Interrupt unterschieden werden ?**

Bestimmung der Interrupt-Quelle:

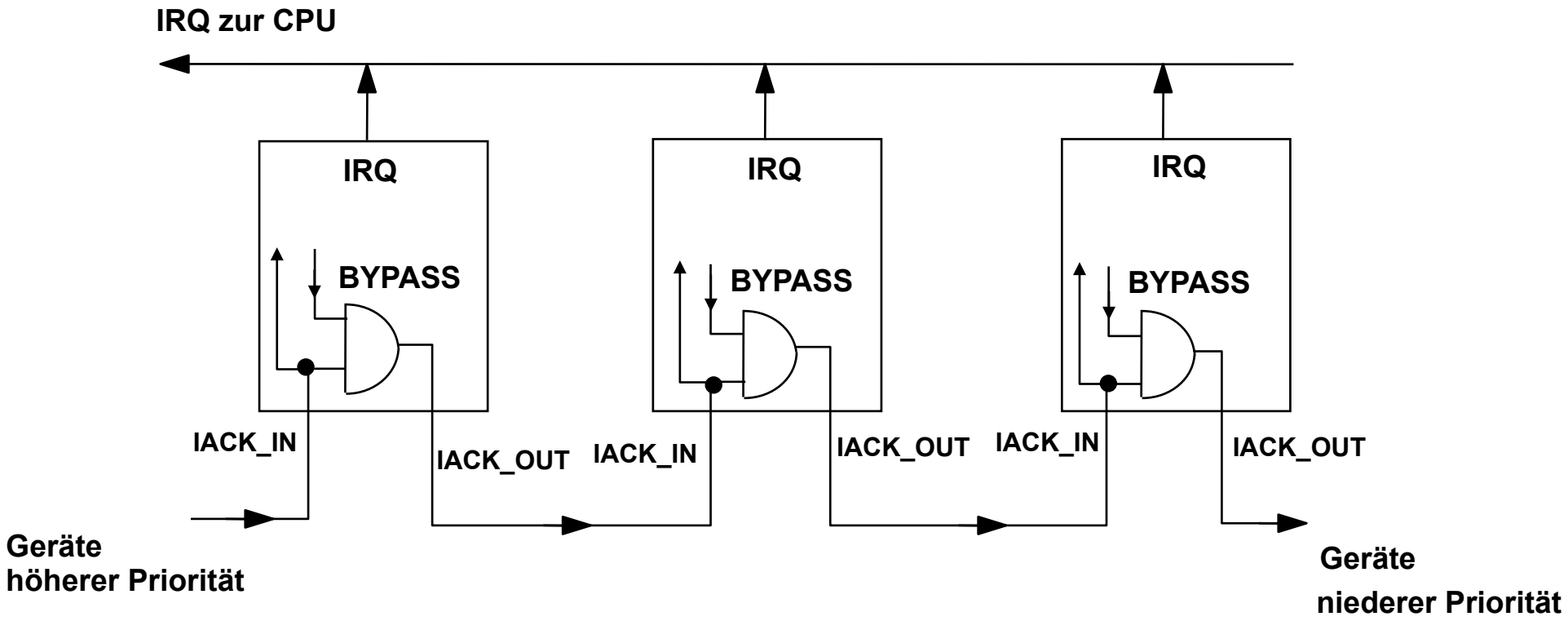
- Nutzung unterschiedlicher Interrupts
- "Polling" aller in Frage kommenden Quellen
- Vektorisierter (Vectored) Interrupt

Prioritätenbestimmung von Interrupts:

- Prioritätsebenen
- "Daisy Chaining" der Interrupt -Acknowlege Leitungen

Daisy-Chaining

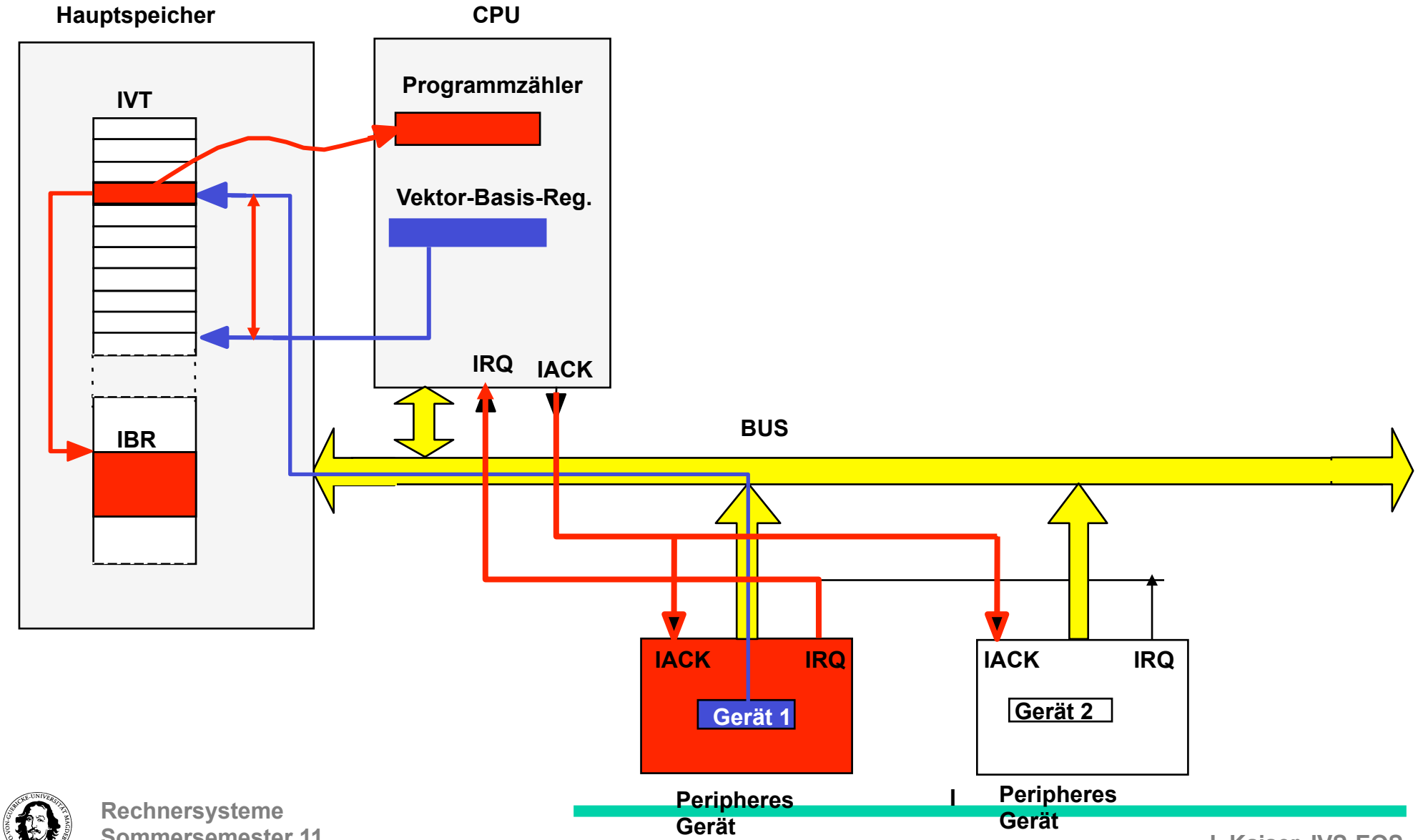
Bestimmung des unterbrechenden Gerätes durch „Hardware Polling“



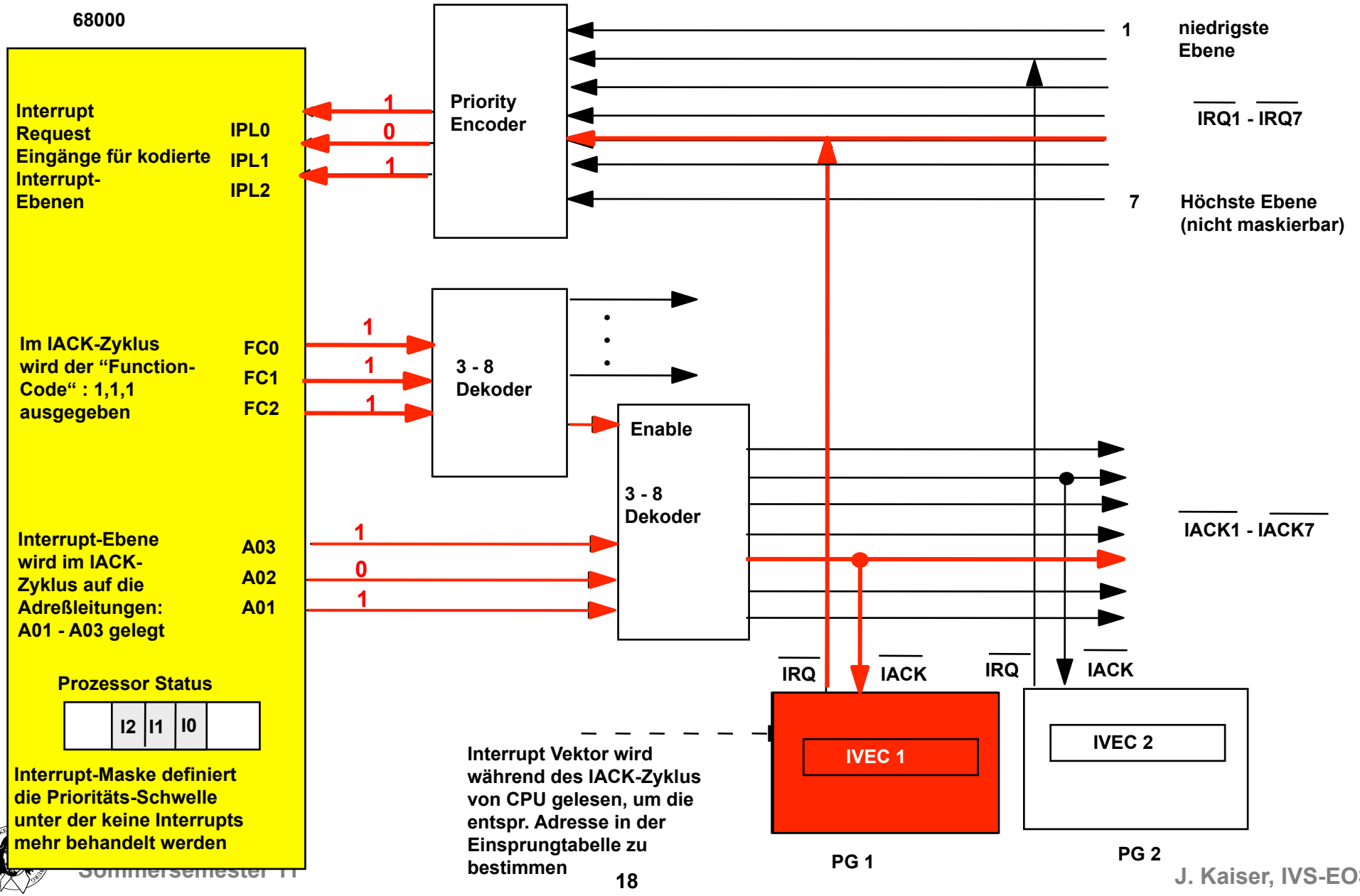
Vektorisierte Unterbrechungsbehandlung

Die unterbrechende Einheit signalisiert in einem speziellen Buszyklus (Interrupt Acknowlege) ihre Identität meist in Form einer Versatzadresse in eine entsprechende Einsprungtabelle.

Vektorisierte Unterbrechungsbearbeitung



Vektorisierte und priorisierte Interrupt Bearbeitung im MC 68000



Debugging:

- Breakpoints
- Single Step / Tracing
- Memory Dump
- Register Dump

Monitor: Meist residentes (im ROM) Programm mit minimaler Funktionalität, das das Laden von Anwenderprogrammen und das Debuggen auf der Maschinenebene erlaubt.

Breakpoint: ist eine bezeichnete Stelle im Programm, an dem die Ausführung automatisch unterbrochen wird, damit der Benutzer den aktuellen Zustand des Systems anschauen und auswerten kann.

Single Step: Dieser Modus erlaubt es, Instruktion nach Instruktion einzeln auszuführen, Register oder Speicherplätze anzuschauen und zu ändern

Trace: Dieser Modus protokolliert jeden Programmschritt aufgrund einer vorher eingegebenen Spezifikation, WAS protokolliert werden soll. Erlaubt die spätere Off-Line-Analyse von Programmen. (Auf den "Trace" können dann Editorfunktionen wie Suchen oder Vergleichen angewandt werden.)

Debugging:

- Breakpoints
- Single Step / Tracing
- Memory Dump
- Register Dump

Ein Breakpoint ist eine bezeichnete Stelle im Programm, an dem die Ausführung automtisch unterbrochen wird, damit der Benutzer den aktuellen Zustand des Systems anschauen und auswerten kann.

Beim Erreichen eines Breakpoints:

- Software Interrupt (SWI)
- Auswahl der entsprechenden Behandlungsroutine
- Ausgabe der Register
- Eintreten in einen Kommandointerpreter

Funktionen: Ändern der Registerinhalte, Anschauen von Speicherinhalten (z.B. Stack), etc.

Beim Verlassem eines Breakpoints:

- Wiedereinsetzen der ursprünglichen Instruktion
- Wiederaufsetzen des Programms VOR dieser Instruktion

```
LDX     $0B, S      Get Return Address from Stack (PC auf Platz -11 und -10 relativ zum Anfang der RL)
LEAX    -2, X       Decrementieren, um VOR der Instruktion aufzusetzen
STX     $0A, S      Zurückschreiben auf den Stack
                    RTI benutzt nun die geänderte Adresse. um den PC zu laden
```

- Return from Interrupt (RTI)

Dieses Verfahren funktioniert nur im RAM, nicht im ROM !

Lernziele

Prinzipien der Unterbrechungsbearbeitung - Asynchronität - Wirkung auf die Programmkontrolle.

Schritte und Ablauf bei der Unterbrechungsverarbeitung.

Unterschiede zwischen programmbezogener und systembezogener Unterbrechung.

Identifizierung des Geräts, das die Unterbrechung ausgelöst hat.

Prioritäten zwischen Interrupts.