
6. Übung

- ❑ Wdh. Unterprogrammaufrufe
- ❑ Unterbrechungen und Ausnahmen
 - ❑ Fallstudie MIPS



Einfacher Unterprogrammaufruf

```
main:          jal subroutine
```

```
subroutine:    # keine weiteren  
              # Unterprogrammaufrufe  
  
              # für lokale Variablen  
              # nur $t0 bis $t9  
  
              jr $ra
```

Beispiel für einen Unterprogrammaufruf

```
# Hauptprogramm
```

```
    .globl main
main:    subu $sp, $sp, 8
        sw $ra, 0($sp)
        sw $fp, 4($sp)
        addu $fp, $sp, 8
```

```
    jal subroutine
```

```
        lw $ra, 0($sp)
        lw $fp, 4($sp)
        addu $sp, $sp, 8
        jr $ra
```

**Unterprogrammaufrufe
mit jal-Befehl !**

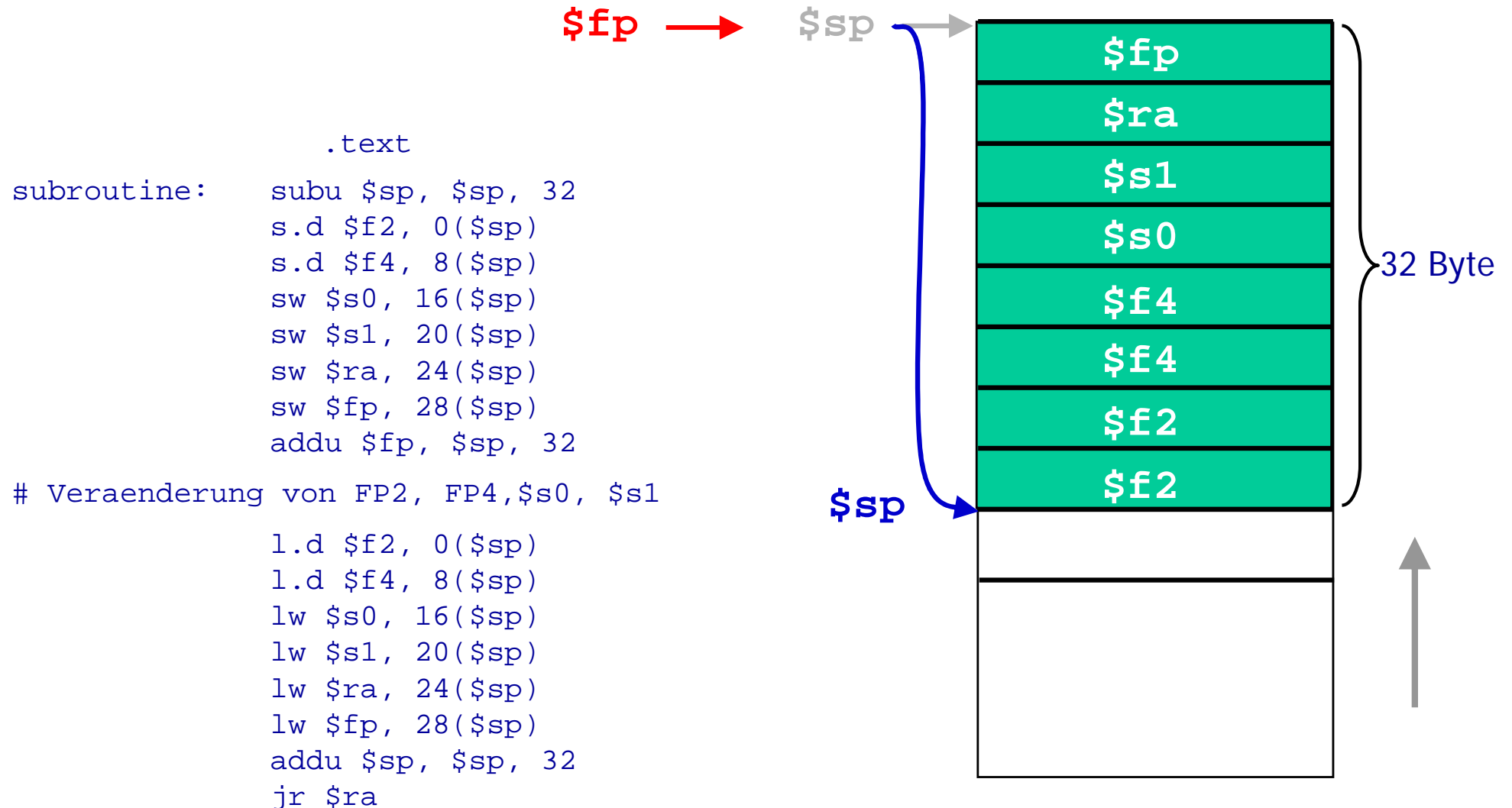
```
subroutine:    .text
              subu $sp, $sp, 32
              s.d $f2, 0($sp)
              s.d $f4, 8($sp)
              sw $s0, 16($sp)
              sw $s1, 20($sp)
              sw $ra, 24($sp)
              sw $fp, 28($sp)
              addu $fp, $sp, 32
```

```
# Veraenderung von $fp2,$fp4,$s0,$s1
```

```
        l.d $f2, 0($sp)
        l.d $f4, 8($sp)
        lw $s0, 16($sp)
        lw $s1, 20($sp)
        lw $ra, 24($sp)
        lw $fp, 28($sp)
        addu $sp, $sp, 32
        jr $ra
```



Beispiel für einen Unterprogrammaufruf



Unterprogrammaufruf

Aufrufendes Programm



Unterprogramm



Behandlung von Ausnahmesituationen

- ❑ Während des Betriebs eines Mikroprozessorsystems können Ausnahmesituationen (Exceptions) auftreten
- ❑ Eine Ausnahmesituation erfordert eine vorübergehende Unterbrechung oder gar den **Abbruch** des laufenden Programms
- ❑ **Ursachen:**
 - Fehler im System bei der Ausführung des Anwenderprogramms oder Fehler der Hardware
 - Wunsch externer Systemkomponenten, die Aufmerksamkeit des Prozessors zu erhalten
- ❑ Die Ausnahmebehandlung erfolgt durch eine Ausnahmeroutine (Interrupt Service Routine), deren Aktivierung durch eine Hardware-Komponente (Unterbrechungs-System, *Interrupt System*) im Steuerwerk unterstützt wird.
- ❑ Die Ausnahmeroutine hat Ähnlichkeit mit dem Aufbau eines Unterprogramms. Es gibt jedoch wesentliche Unterschiede:



Ausnahmeroutine/Unterprogramm

- Aktivierung:
 - *call subroutine* bei Unterprogramm
 - Hardware-Aktivierung durch *Externes Signal* bei Ausnahmeroutine
- Beendigung:
 - *ret*-Befehl bei Unterprogrammen (*return from subroutine*)
 - *reti*-Befehl bei Ausnahmebehandlung (*return from interrupt*)
- Einsprungsadresse ins Unterprogramm direkt im Programm, bei Ausnahmebehandlung über Interrupt-Tabelle
- Unterprogrammaufruf sichert meist nur den PC auf den Stack, Ausnahmebehandlungs-Aufruf meist auch das PSW
- Unterprogrammaufrufe werden immer durchgeführt, die meisten Ausnahmebehandlungen werden nur aktiviert, falls das *Interrupt-Enable-Bit* im Steuerregister (PSW) gesetzt ist



Unterbrechungen und Ausnahmen

□ Unterbrechung (Interrupt):

- zum Programmablauf **asynchrone** Ereignisse.
- Sie werden durch die Hardware erzeugt.
- Prozessorexterne Ursachen
 - Beispiel: Ein- und Ausgabegeräte, z.B. die Tastatur

□ Ausnahme (Exception, Trap):

- zum Programmablauf **synchrone** Ereignisse.
- Sie treten an fest vorgegebenen Stellen im Programm ein.
- Prozessorinterne Ursachen
 - Beispiel: arithmetische Fehler, ungültiger Befehl, ...



Prozessorexterne Ursachen

➤ **RESET:**

Rücksetzen des Mikrorechnersystems, z. B. ausgelöst durch Taste, Schwankungen der Betriebsspannung, Watch-Dog, ...

➤ **HALT:**

Anhalten des Prozessors, z. B. zur Vermeidung von Zugriffskonflikten auf dem Systembus bei DMA-Zugriffen, Paritätsfehler

➤ **ERROR:**

Aufruf einer Fehlerbehandlungsroutine, z. B. bei Bus-Fehlern oder Zugriffe auf geschützte Datenbereiche



Prozessorexterne Ursachen

- **(Hardware)-Interrupt:** Unterbrechungsanforderung von einem peripheren Gerät, z. B. um verfügbare Daten eines Eingabegerätes anzukündigen

2 Arten: maskierbar/nicht maskierbar (NMI)

- **Nicht maskierbare Interrupts (NMI):** werden nach Abschluss des gerade ausgeführten Befehls unbedingt durchgeführt (wichtige Ausnahmesituationen, z. B. Zusammenbruch der Betriebsspannung)
- **Maskierbare Interrupts (IRQ):** werden nur dann ausgeführt, wenn im Steuerregister des Prozessors das *Interrupt-Enable* Flag (IE-Bit) gesetzt ist.



Prozessorinterne Ursachen

Prozessorinterne Ursachen: Bei Prozessoren, die auf dem Chip interruptfähige Komponenten besitzen (z. B. serielle oder parallele Schnittstellen, Zeitgeber, ..). Treten synchron zum Programmablauf auf.

- **Software Interrupts:** durch SWI- (*software interrupt*) oder INT-Befehl im Programm ausgelöste Unterbrechungen
- **Traps:** Ausnahmesituationen durch prozessorinterne Ereignisse, z. B. Overflow, Divide by Zero, Stack Overflow, ungültiger OpCode, Seitenfehler (*Page Trap*), Einzelschritt-Unterbrachung (Trace)



Interrupt-Vektortabelle

Interrupt-Vektortabelle (Ausnahme-Vektortabelle):

- Festwertspeicher an spezieller Speicheradresse (Oft in einer der untersten Speicherseiten)
- Enthält die Startadressen der Behandlungsroutinen
- Interrupt-Quelle liefert bei Interrupt eine Interrupt-Vektor-Nummer (IVN), welche den Eintrag in der Interruptvektor-Tabelle charakterisiert
- Basis-Adressregister enthält die Basisadresse der Tabelle



Beispiel einer Vektortabelle

Index	Ausnahmesituation	
0	divide by 0	Division durch 0
1	overflow	Zahlenbereichüberschreitung
2	array bound check	Indexbereichüberschreitung
3	invalid opcode	illegaler Befehlscode
4	SVC (supervisor call)	Betriebssystem-Aufruf
5	privilege violation	unerlaubter Aufruf einer privilegierten Operation
6	trace	Einzelschritt-Modus
7	
-	(weiter Traps)
i	
i+1	RESET	Rücksetzen
i+2	BERR	Busfehler
i+3	NMI	nicht maskierbarer Interrupt
-	
k	

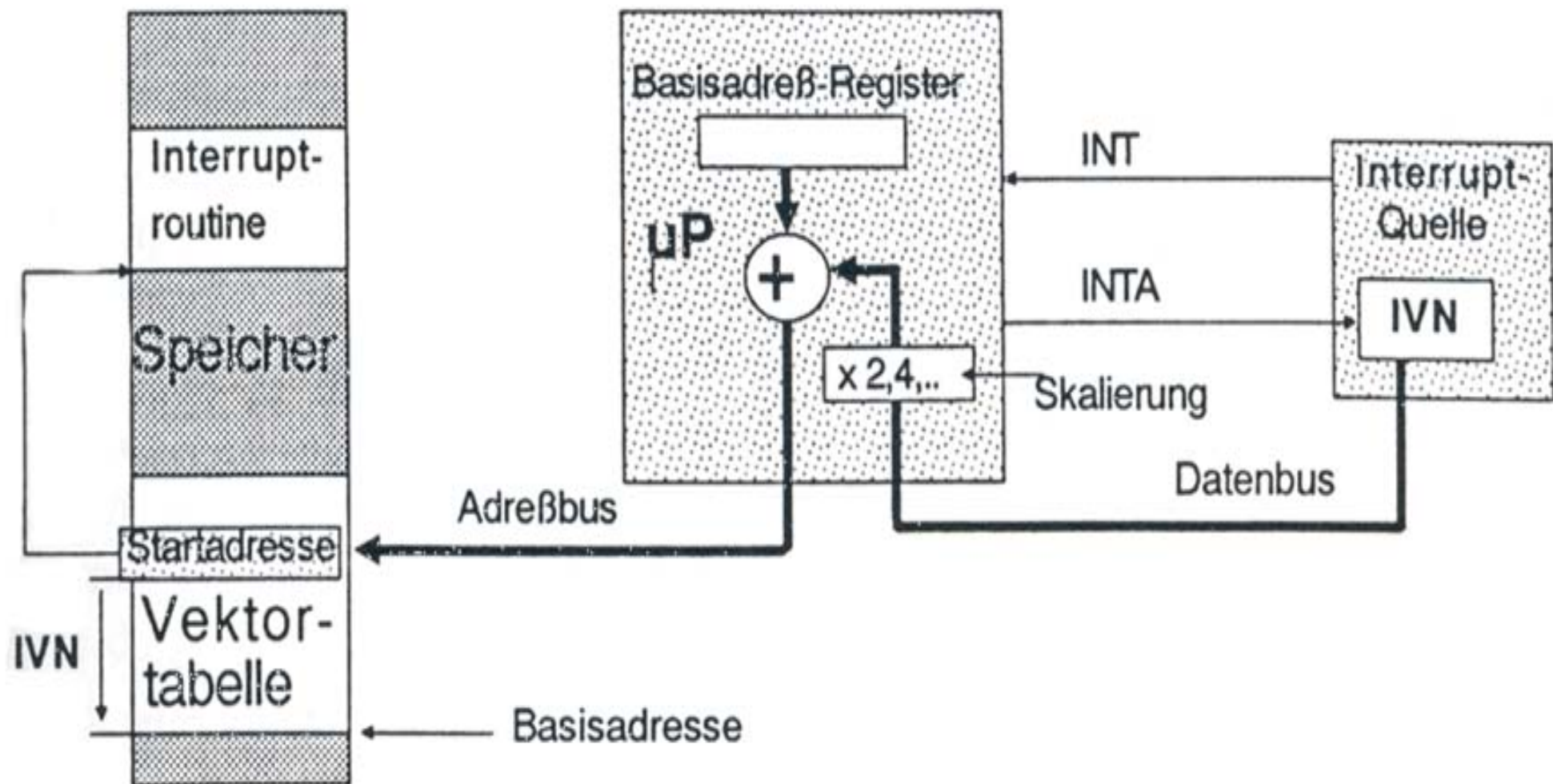


Beispiel einer Vektortabelle

Index	Ausnahmesituation	
k+1	error	Coprozessor-Fehler
-	Weitere Coprozessor-Meldungen
l		
l+1	page fault	Seitenfehler
-	Weitere Fehler der Speicherverwaltung
m	
m+1	Reserviert für zukünftige Erweiterungen und für Testzwecke des Herstellers
-	
n	
n+1	user vectors	Durch Systementwickler frei zuzuordnende, maskierbare Interrupts
-	
255	



Berechnung der Startadresse der Interrupt Service Routine



Ablauf einer Interrupt Service Routine

- Interrupt-Aktivierung
- Beenden des gerade in Ausführung befindlichen Befehls
- Feststellen, ob Software-Interrupt oder interner/externer Hardware-Interrupt vorliegt
- Feststellen, ob das Interrupt Enable Bit gesetzt ist
➔ Interrupt zugelassen
- Falls HW-Interrupt: Quelle des Interrupts finden, INTA-Leitung (Interrupt Acknowledge) aktivieren
- PSW und PC auf den Stack sichern
- Interrupt Enable Bit zurücksetzen (und damit weitere Unterbrechungen verhindern)



Ablauf einer Interrupt Service Routine

- Startadresse der Interrupt-Service-Routine ermitteln und in den PC laden
- Interrupt Service Routine ausführen:
 - meist werden zuerst die benutzte Register auf den Stack gesichert
 - Interrupt Enable Bit wieder setzen
 - Am Ende der Interrupt Service Routine: IRET-Befehl
- PSW und PC werden wiederhergestellt und mit dem unterbrochenen Programm wird fortgefahren.



MIPS - Ausnahmen

Code	Beschreibung	
0	externe Unterbrechung	
4	fehlerhafte Adresse beim Laden oder Befehl-Holen	
5	fehlerhafte Adresse beim Speichern	
6	Busfehler beim Befehl-Holen	
7	Busfehler beim Laden oder Speichern von Daten	
8	Systemaufruf	} Software
9	Programmunterbrechung (breakpoint) zur Fehlersuche	
10	reservierter Befehl	
12	arithmetischer Überlauf bei Ganzzahlberechnungen	
14	ungültiges Fließkomma-Ergebnis	
15	Division durch Null	
16	Überlauf bei Fließkomma-Berechnung	
17	Unterlauf bei Fließkomma-Berechnung	



Beispiele für Ausnahmen

Externe Unterbrechung:

- Drücken einer Taste
- Maus wurde bewegt
- Timer ist abgelaufen
- Peripheriegerät (z. B. Drucker) ist fertig
- serielle Schnittstelle hat ein Wort empfangen

Fehlerhafte Adresse:

```
                .data
strng:          .asciiz "arglwuz = "
result:         .space 23

                .text
                .globl main
main:           ...
                lw $a0, result
```



Beispiele für Ausnahmen

Busfehler:

- Time-Out
- Paritätsfehler
- ungültige Port- oder Speicher-Adressen

Programmunterbrechung (breakpoint):

Ausgabe von Register- oder Speicherinhalten oder des Prozessorstatus zur Fehlersuche

Systemaufruf (syscall)

Software-Interrupt zur Ausführung von Unterprogrammen des Betriebssystems (vergleichbar mit INT n beim 8086)



Beispiele für Ausnahmen

Reservierter Befehl:

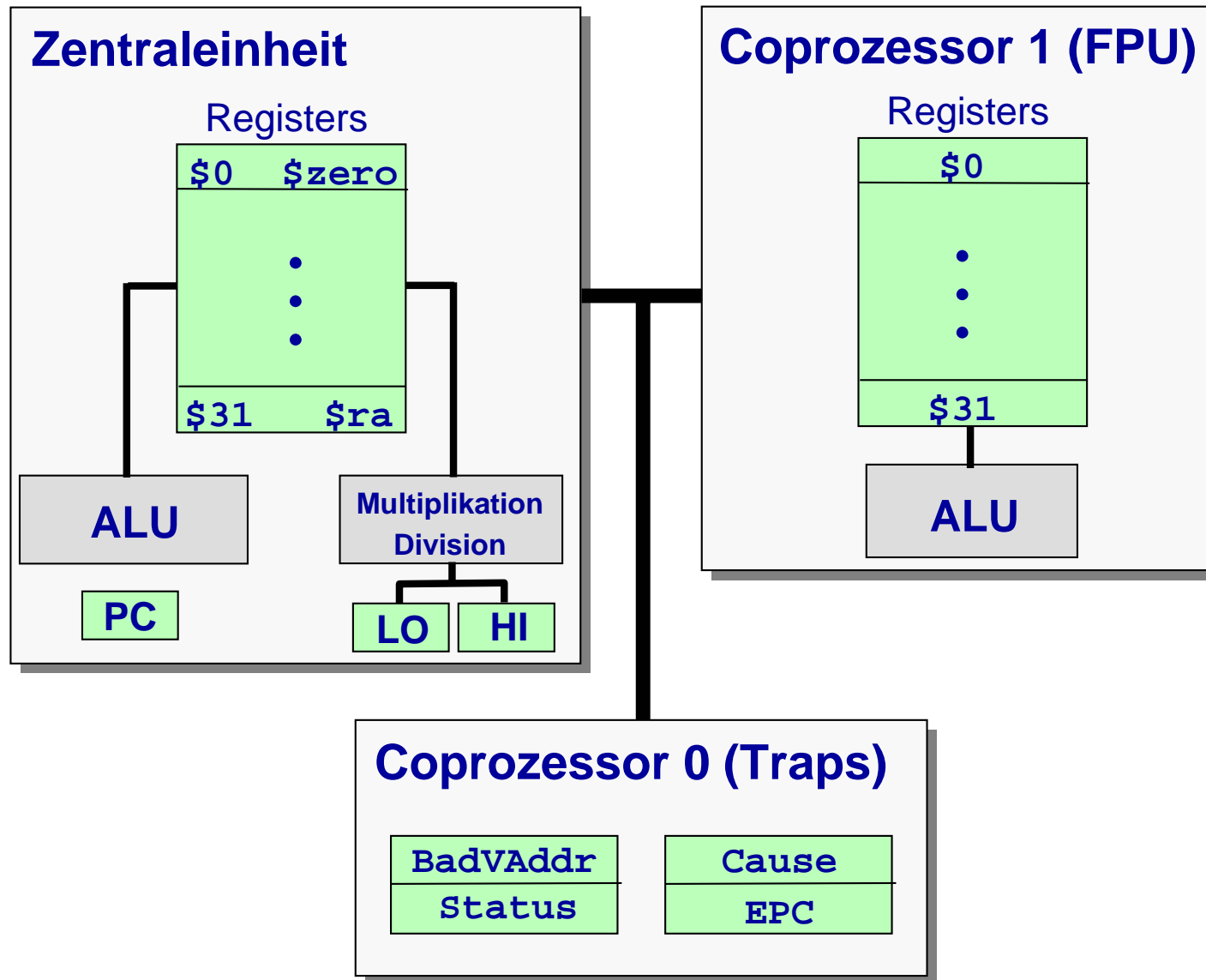
Es wurde versucht, einen Befehl auszuführen, dessen Bits 31...26 nicht als Opcode definiert sind

Arithmetischer Überlauf bei Ganzzahlberechnungen:

Ergebnis einer ADD, ADDI oder SUB Instruktion lässt sich nicht als 32-Bit Zweierkomplement darstellen



Aufbau des MIPS-Prozessors



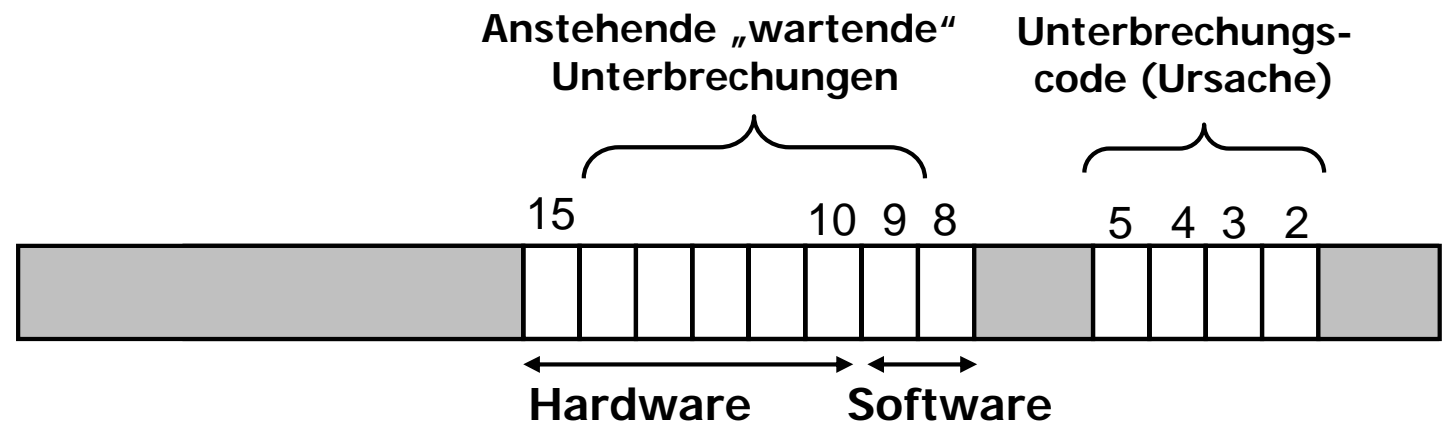
Register des Coprozessors 0

Coprozessor 0 enthält vier Register zur Behandlung von Ausnahmen und Unterbrechungen:

➤ **BadVAddr (Reg.-Nr. 8):**

Falls die Unterbrechung durch einen Speicherzugriff verursacht wurde, ist hier die Speicheradresse enthalten

➤ **Cause (Reg.-Nr. 13):**

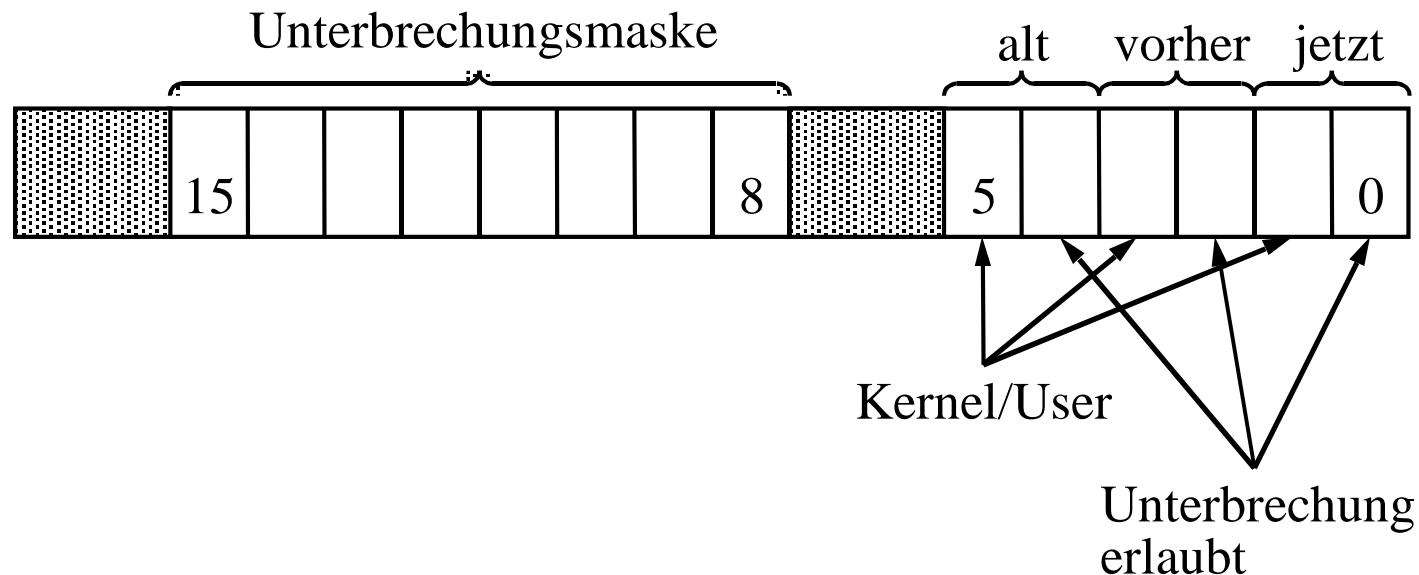


Register des Coprozessors 0

➤ EPC (Reg.-Nr. 14):

Speicheradresse, in der sich der Befehl befindet, der zur Unterbrechung geführt hat

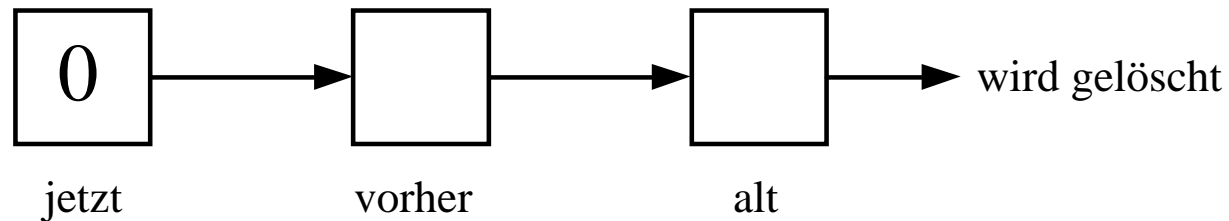
➤ Status (Reg.-Nr. 12):



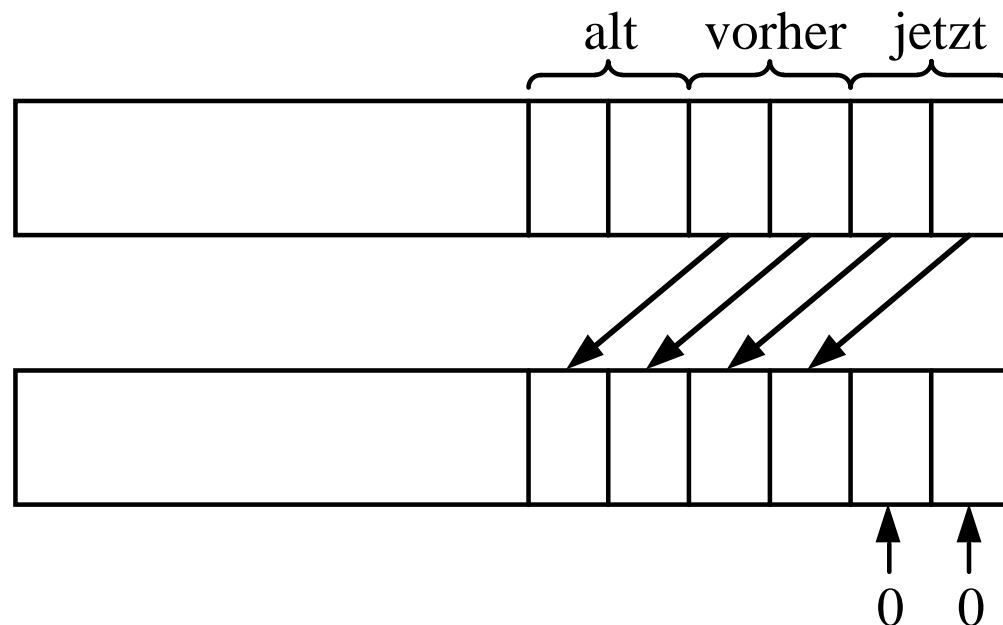
Die Unterbrechungsmaske bestimmt, welche Unterbrechungen zugelassen sind

Register des Coprozessors 0

Speichern der Kernel/User- und Interrupt-Enable-Bits:

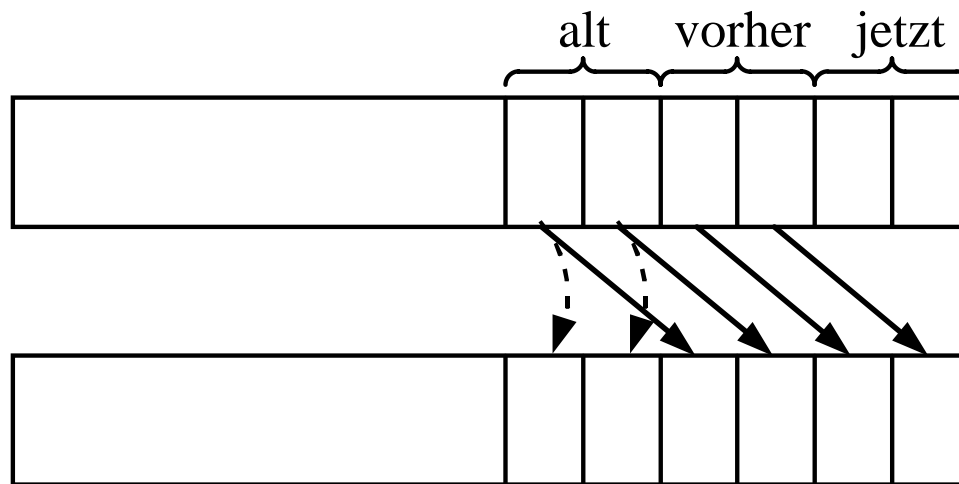


Beim Auftreten einer Unterbrechung:



Register des Coprozessors 0

Rücksprung aus einer Unterbrechung (rfe Instruktion):



Ausnahmebehandlung (Interrupt-Handler)

Bei Ausnahme oder Unterbrechung erfolgt Sprung nach
Adresse $8000\ 0080_{16}$

```
.ktext 0x80000080
```

Benutzung des Stapels zur Sicherung von Daten nicht erlaubt,
daher Sicherung von Registern im Kerneldatensegment

```
sw $a0, a0_save  
sw $a1, a1_save  
sw $ra, ra_save
```

Register \$at sichern

```
.set noat  
sw $at, at_save  
.set at
```



Ausnahmebehandlung (Interrupt-Handler)

Laden des Cause- und EPC-Registers

```
mfc0 $k0, $13  
mfc0 $k1, $14
```

Unterbrechungen ignorieren

```
bgt $k0, 0x44, fertig
```

Spezifische Maßnahmen

```
... (Benutzung von $a0 und $a1)  
jal print_excp
```



Abschluss der Ausnahmebehandlung

```
fertig:    lw $a0, a0_save
           lw $a1, a1_save
           lw $ra, ra_save
           .set noat
           lw $at, at_save
           .set at
```

Statusregister wiederherstellen

```
rfe
```

Fehlerhafte Instruktion soll nicht nochmals ausgeführt werden

```
addiu $k1, $k1, 4
```

Standard-Traphandler

```
s1: .word 0          # Speicher zum Sichern von Registern;
s2: .word 0          #
    .ktext 0x80000080 # Code gehört zum Kernel Textsegment
    .set noat        # keine Nutzung von $at durch Assembler
move $k1 $at         # Retten von $at in $k1 (für Kernel)
    .set at          # Nutzung durch von $at wieder möglich
sw $v0 s1            # Rette $v0 in feste Speicherzelle
sw $a0 s2            # Rette $a0 in feste Speicherzelle
                    # Nutzung von $at, $v0 und $a0

mfc0 $k0 $13         # Lade Unterbrechungsursache
...                  # Verzweige abhängig von $k0
...                  # Lösche Ursachenregister
lw $v0 s1            # Rückschreiben von $v0
lw $a0 s2            # Rückschreiben von $a0
    .set noat
move $at $k1         # Restore $at
    .set at
rfe                  # Restaurieren von Statusregistern
mfc0 $k0 $14         # Hole alten Wert des PC
addiu $k0 $k0 4      # Erhöhe um 4
jr $k0               # Führe unterbrochenes Programm fort
```

