Tutorium 2

Prozesse und Parallelität

Aufgabe 1.3 - Parallelisierung

Wie unterscheidet sich die Herangehensweise von *parbegin/parend* und *fork/join*?

Aufgabe 1.3 Lösung - Parallelisierung

- **parbegin/parend**
  - Startet mehrere Prozesse, die unabhängig voneinander arbeiten können
  - Alle parallelen Aufgaben beginnen gleichzeitig
  - Wartet mit *parend*, bis alle parallelen Prozesse abgeschlossen sind, bevor das Programm fortgesetzt wird
- **fork/join**
  - *fork* erstellt dynamisch einen neuen Child Prozess, der parallel ausgeführt werden kann)
  - Mit *join* kann man genauer steuern, wann und auf welche Prozesse gewartet werden soll. Ermöglicht komplexere Abhängigkeiten.

Aufgabe 1.3 - Parallelisierung

Gegeben sei der untenstehende Abhängigkeitsgraph. Setzen Sie diesen mithilfe der aus der Vorlesung bekannten Befehle *fork/join* und *parbegin/parend* in Pseudocode um.

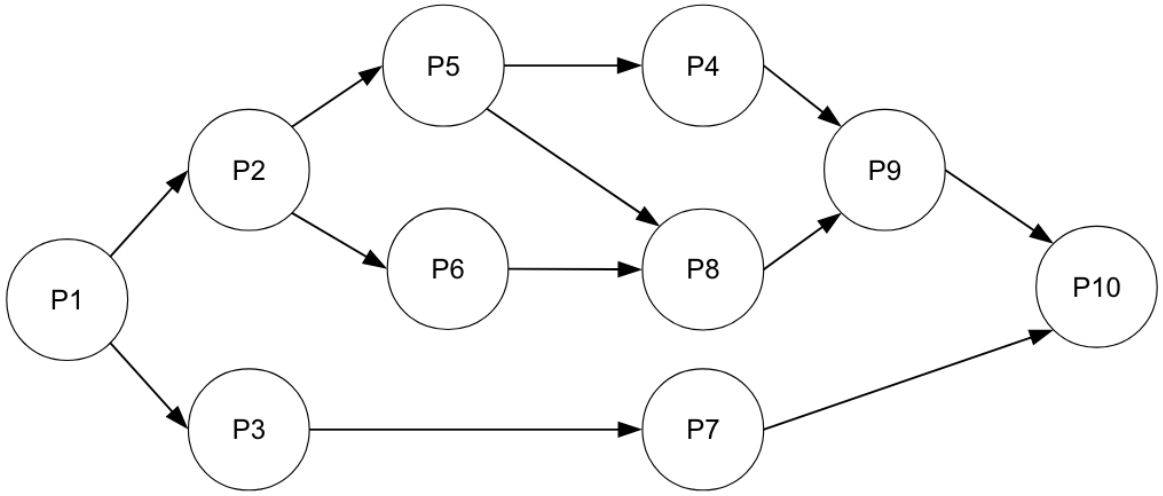


Abbildung 1: Abhängigkeitsgraph

Aufgabe 1.3 Lösung - Parallelisierung

```
parbegin/parend:
----------------

p1;
parbegin
    begin
        p2;
        parbegin
            p5;
            p6;
        parend

        parbegin
            p4;
            p8;
        parend
        p9;
    end
    begin
        p3;
        p7;
    end
parend
p10;
```

```
fork/join:
----------

p1
fork a
p3
p7
join a
p10

a:
p2
fork p6
p5
fork p4
join p6
p8
join p4
p9
end
```

Aufgabe 1.3 - Parallelisierung

Gehen Sie dann wieder den umgekehrten Weg und zeichnen Sie den Abhängigkeitsgraphen aus dem Pseudocode. Haben sich zusätzliche Abhängigkeiten gebildet?

Aufgabe 1.3 Lösung - Parallelisierung

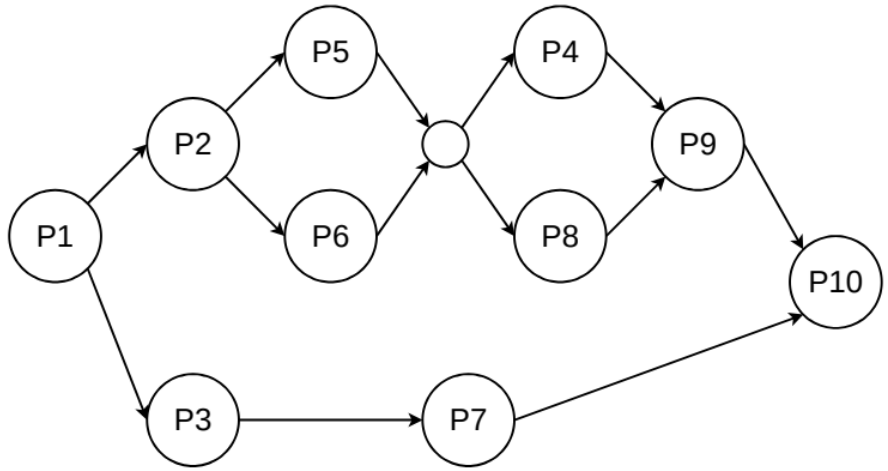


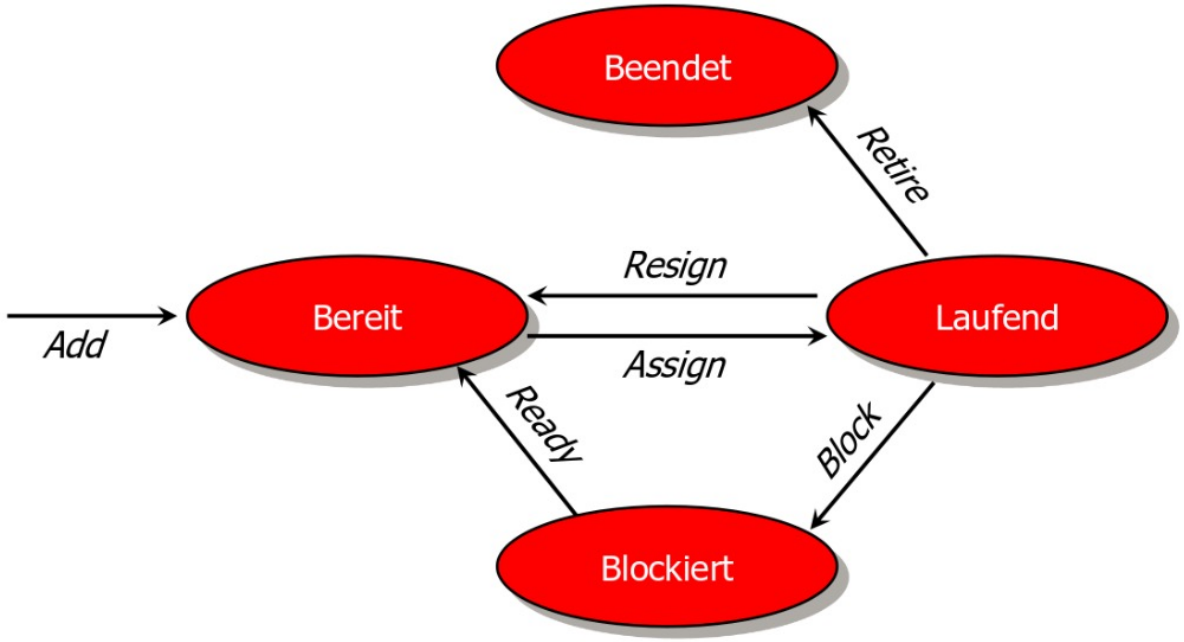
Abbildung 2: Abhängigkeitsgraph nach Rücktransformation von parbegin/parend

- Es hat sich nach der Rucktransformation tatsächlich eine neue Abhängigkeit ergeben. P4 muss nun auch auf P6 warten, was im Originalgraph nicht der Fall war.
- Bei der Rucktransformation von *fork/join* bleibt der Graph genauso erhalten.

Aufgabe 1.4 - Prozessmanagement

Benennen Sie die möglichen Zustände eines Prozesses und Skizzieren Sie die Übergänge.

Aufgabe 1.4 - Prozessmanagement



Aufgabe 1.5 - Prozessmodi

Welche Prozessormodi existieren? Nennen Sie zwei privilegierte Modi und einen nicht privilegierten Modus.

Aufgabe 1.5 Lösung - Prozessmodi

- Beispielhaft alle (relevanten) Modi auf einem ARM7-Prozessor

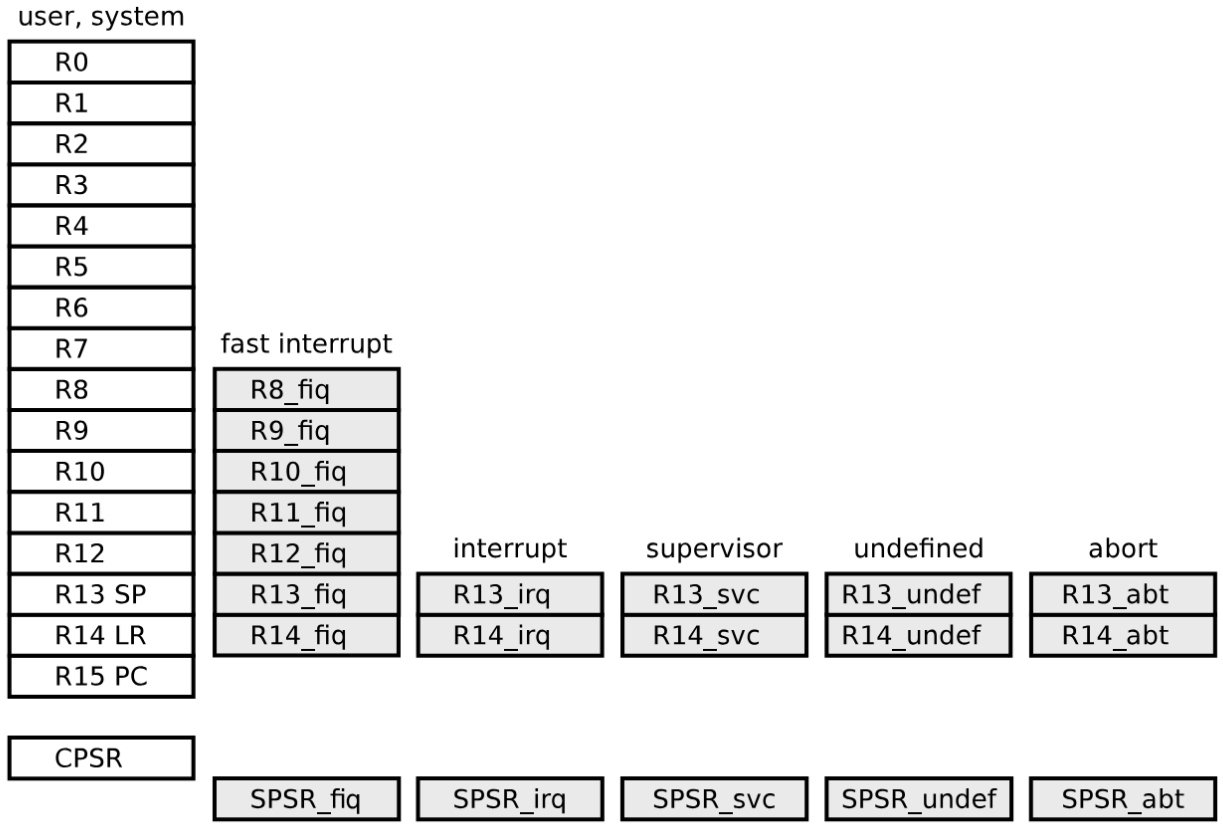
Modus	Kurzform	Modusbits	Beschreibung
User	usr	10000	regulärer unprivilegierter Modus
FIQ	fiq	10001	Behandlung von *Fast Interrupt Requests*
IRQ	irq	10010	Behandlung von normalen *Interrupt Requests*
Supervisor	svc	10011	Behandlung von Systemrufen/Systeminitialisierung
Abort	abt	10111	Behandlung von Speicherzugriffsfehlern
Undefined	und	11011	Behandlung von unbekannten Instruktionen
System	sys	11111	regulärer privilegierter Modus

Aufgabe 1.5 - Prozessmodi

Worin unterscheiden sich nicht privilegierte und privilegierte Modi (Rechte und Anwendung)? Nennen Sie 3 Unterschiede.

Aufgabe 1.5 Lösung - Prozessmodi

- Beispielhaft alle (relevanten) Register auf einem ARM7-Prozessor



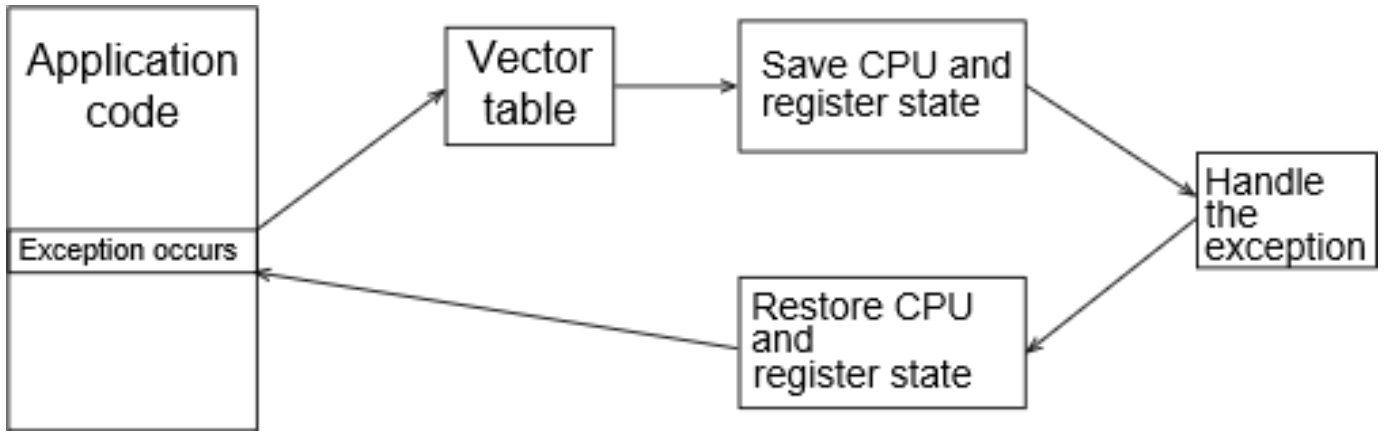
Aufgabe 1.5 - Prozessmodi

Wie wird zwischen privilegierten und nicht privilegierten Modus gewechselt? Nennen Sie drei Beispiele zum Wechseln in einen privilegierten Modus und ein Beispiel zum Wechseln in den unprivilegierten Modus.

Aufgabe 1.6 - Interrupts

Was ist ein Hardware-Interrupt? Wie reagiert das Betriebssystem und der Prozessor auf einen Hardware Interrupt?

Aufgabe 1.6 Lösung - Interrupts

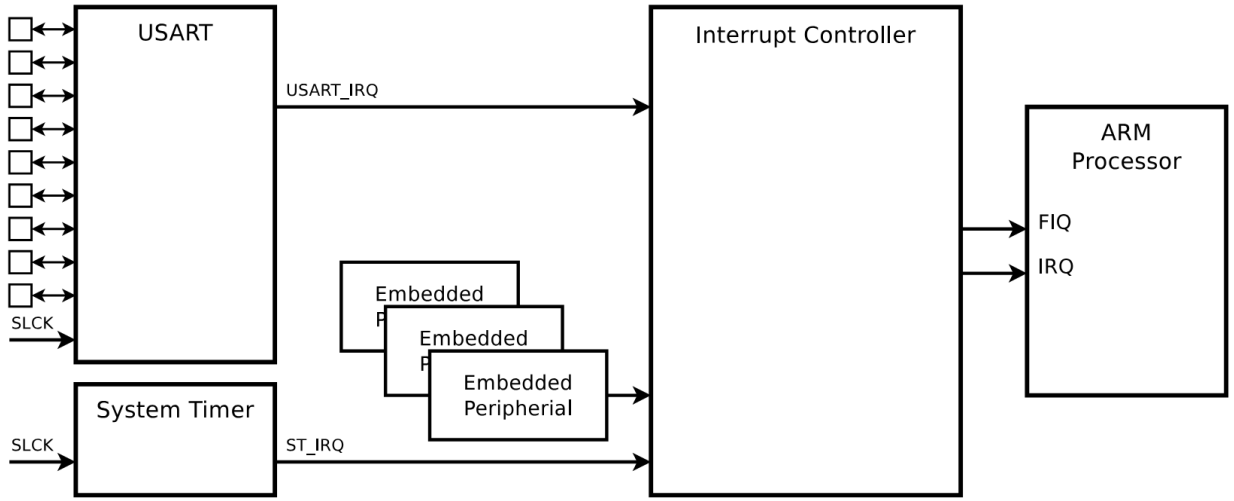


Aufgabe 1.6 - Interrupts

Wie erkennt der Prozessor, dass ein Hardware-Interrupt vorliegt?

Nennen Sie zwei Beispiele, wie Hardware-Interrupts entstehen können

Aufgabe 1.6 Lösung - Interrupts



Aufgabe 1.6 - Interrupts

Wie unterscheiden sich sequentielle Unterbrechungsbehandlung und geschachtelte Unterbrechungsbehandlung? Was passiert, wenn ein Interrupt eintritt, während noch ein vorheriger Interrupt behandelt wird?

Danke, für die Aufmerksamkeit! Bis nächste Woche!