Versuch 2 Fragen

Vaargk Marko Felix Stephen David

9. Dezember 2019

- Was ist ein Stack?
- Was ist ein Programm/Prozess?
- Was ist ein Scheduler?
- Was ist ein Laufzeitkontext? Wozu dient dieser?
- Warum und was ist ein Leerlaufprozess?
- Was ist ein kritischer Bereich? Was ist ein atomarer Bereich?
- Ist die Funktion os_EnterCriticalSection() ein kritischer Bereich? -nein, da wir das GIEB deaktivieren (dadurch atomar)
- Was passiert bei einem Stack Overflow? Was bedeutet Inkosistenz? -
  Bei einem Stack Overflow speichert der Prozess zu viele Bytes, sodass der zugewiesene Stackbereich nicht ausreicht. Dadurch wird der Stack des "darüber"liegenden Prozesses beschrieben. Inkonsistenz bedeutet, dass sich der Stackbereich eines Prozesses ohne dessen Ausführung ändert.
- Was für ein Verhalten passiert beim Stack Overflow und kann man ihn rückgängig machen? -
  Das Verhalten nach einem Stack Overflow ist undefiniert. Ein Stack Overflow kann nicht rückgängig gemacht werden.
- Wie erkannt man einen Stack overflow? -
  Wir nutzen Hashing (XOR von allen Bytes des Stackbereiches).
- Kann man einen Stack Overflow möglicherweise nicht erkennen? -
  Ja, falls der berechnete Hashwert gleich ist mit dem vorherigen, jedoch sich der Stack verändert hat. (Bspw. 3 Bytes, alle jeweils 1 vorher. Nachher sind 2 Bytes 0 und ein Byte 1. $\Rightarrow$ gleicher Hashwert)

- Es gibt noch bessere Verfahren, um einen Stack Overflow zu erkennen. Was könnte der Grund dafür sein, dass wir diese hier nicht benutzen?
  - Möglicherweise sind die Verfahren aufwendiger und haben auch eine höhere Laufzeit, welche sich bei einem Mikrokontroller, während des Kontextwechsels stark auf die gesamte Effizienz auswirkt.
- Wie ist der SRAM aufgeteilt?
- Wie speichert der Stack? - LIFO Prinzip, also von unten nach oben.
- Wieso speichert der Stack nach diesem Prinzip? - Stack und Heap sollen sich nicht gegenseitig (durch bspw. Overflow) beschreiben können.
- Strings müssen in Flash gespeichert werden sonst Speicherknappheit
- Was speichern wir zu einem Prozess?
- Wenn man einen neuen Prozess anlegt was und in welcher Reihenfolge müssen wir dann machen?
- Wie oft wird der Scheduler aufgerufen?
- Wenn der Scheduler wechselt was und in welcher Reihenfolge passiert dann? Ganz kleinschrittig!!!
- Was passiert beim Prozessstack vorbereiten?
- Werden die Register beim neuen Prozess direkt beschrieben? -nein Was passiert da genau?
- Ganz genau alles zu stackpointer und deren Verwendung wissen. Ganz wichtig Versuchsdokument Seite 17 Punkt 4
- Was passiert beim Scheduler init und genau kleinschrittig und Reihenfolge der Schritte wissen!
- Wie funktioniert der Wechsel zwischen Prozesstacks?
- Was macht restoreContext?
- Was passiert wenn in os_exec die ISR aufgerufen wird? - Nichts da kritischer Bereich
- wie heißt der Befehl mit dem bei restoreContext der PC aus dem Prozessstack geladen wird? - Reti (return interrupt)
- Warum schreiben wir überhaupt ein Multitasking Betriebssystem?
- Warum setzen wir den StackPointer in der ISR auf den ISR stack bottom?

- Was kann passieren, wenn wir den Stackpointer beim Prozesswechsel nicht auf den SchedulerStack zeigen lassen, bzw. Geht das? Warum geht das? Warum machen wir es trotzdem? - mehr verfügbarer Speicher im Schedulerstack als im Prozessstack