Betriebsystemepraktikum

Wintersemester 2015/16
Klausur 25.02.2016

Mitschrift

27. Februar 2016

Hinweis
Dies ist eine Mitschrift der Klausur vom 25.02.2016.
Kein Anspruch auf Vollständig- oder Richtigkeit.

Inhaltsverzeichnis

Name: Matrikelnummer:

1 Multiple Choice

Es ist anzukreuzen, ob folgende Aussagen wahr(w) oder falsch(f) sind. Es gibt für jede Teilaufgabe nur dann Punkte, wenn alle Antworten der Teilaufgabe richtig sind.

1.1 Prozessor Modus

w	f	
□	□	Der User Modus ist der einzige nicht priviligierte Modus
□	□	Der User Modus kann nur durch eine Ausnahme verlassen werden
□	□	Nach einem Reset befindet sich die CPU im Supervisor Modus

1.2 Ausnahmen

w	f	
□	□	Im Ausnahmemodus ist immer zu erkennen, um welche Ausnahme es sich handelt.
□	□	Im Ausnahmemodus werden automatisch Register gesichert

1.3 AIC

w	f	
□	□	Um den AIC zu nutzen, muss die MMU aktiviert sein
□	□	Die AIC steuert die IRQ Leitung
□	□	Mit der AIC lassen sich IRQ Quellen bestimmen
□	□	Die AIC speichert automatisch den Modus und das Linkregister

1.4 LDM/STM (Load/Store Multiple Registres)

w	f	
□	□	LDM/STM kann nur im Supervisor Modus benutzt werden
□	□	LDM/STM kann nicht im User Modus benutzt werden
□	□	Um LDM/STM nutzen zu können, müssen Caches(?) deaktiviert werden
□	□	mit LDM/STM kann ein Stack realisiert werden

1.5 Stacks

w	f	
□	□	Auf dem Stack können lokale Variablen gespeichert werden
□	□	durch modusspezifische Stacks werden Stackoverflows verhindert
□	□	Jeder Thread besitzt einen eigenen Stack
□	□	Der MMU ist die Basisadresse des Stacks mitzuteilen

1.6 SBit

w f

□ □ Das SBit führt einen Rücksprung aus

□ □ Das SBit realisiert einen Interrupt

□ □ Das SBit kann genutzt werden, um eine Thumboperation auszuführen

□ □ Das Sbit schaltet die CPU aus

1.7 MMU

w f

□ □ Die MMU realisiert die Überstzung von virtuellen/logischen Adressen in physikalische

□ □ Die MMU ist ein essentieller Bestandteil des ARM Kerns

□ □ Die MMU ist ein transparenter Bestandteil des ARM Kerns

□ □ Die MMU wird über die Befehle MMUSTM und MMULD angesprochen

1.8 Remaping

w f

□ □ Ein Remaping ist notwendig, um die IVT zu schreiben

□ □ Für ein Remaping muss die MMU aktiviert sein

□ □ Mit einem Remaping lassen sich leicht verschachtelte Interrupts realisieren

□ □ Durch ein Remaping werden automatisch Register gesichert

1.9 preämptives Multitasking

w f

□ □ Thread haben einen eigenen Adressraum.

□ □ Thread müssen regelmäßig dem Kernel sagen, dass sie aufgeben.

□ □

1.10 Supervisor Stack

w f

□ □ Der Supervisor Stack muss 4 bzw 8 Bit aligned sein

□ □ Der Supervisor Stack muss durch den Bootloader initialisiert werden

2 Moduswechsel

2.1

Welche Schritte sind notwendig, um einen Moduswechsel auszuführen?
Es reicht die Beschreibung der Befehle. Keine ASM (Assembler) Befehle gefordert

2.2 ?

3 AAPCS Register

3.1 ?

3.2

Welche Register müssen nach AAPCS Aufrufkonvention gesichert werden? Begründung
Quellcode gegeben. STM/LDM Befehle sollten durch Registernamen ergänzt werden

4 Ausnahmen

4.1

Nenne 4 beliebige Ausnahmen und beschreibe wann sie auftreten

4.2

Wozu dient ein Software Interrupt und wozu wurde er im Praktikum verwendet ?

5 MMU

5.1

Welche Zugriffsrechte können in einer L1-Tabelle für ein laufendes Userprogramm stehen?

5.2

Beschreibe (ohne TLB, Cache, ...) wie eine virtuelle/logische Adresse von der MMU übersetzt wird

6 Komplexe Aufgabe

Es gibt die Funktion

execute_later(int delay_in_ms, void (*action)(void));

Diese Funktion soll eine Augabe Millisekunden genau verzögert ausführen. Dafür soll der System Timer verwendet werden.

6.1

Beschreibe die Datenstruktur die execute_later verwenden soll und was passieren soll ? (bei Aufruf, bei Timer)

6.2

Welche Schritte sind auf Hardwareseite nötig, um den System Timer zu verwenden (bzw. SystemTimerInterrupts)?

6.3

Die Funktion execute_later soll so erweitert werden, dass die verzögerte Funktion genau einen Parameter übergeben bekommt. Welche Änderungen sind nötig?

7 Printf - Systemcall

Es wurde printf als Systemcall umgesetzt. Ob das eine gute Idee ist, sei dahin gestellt. Printf soll beliebig viele Parameter bekommen können. Die Anzahl der Parameter sei, wenn nötig, bekannt.

7.1

Erstelle eine Aufrufkonvention (Welcher Parameter steht wo?)

7.2

Printf bekommt Speicheradressen. Wie kann sich das System, wenn Datentyp und Größe(?) bekannt sind, vor DataAborts oder Kompromittierung schützen? (Speicheradresse überprüfen)