h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

RECHNERARCHITEKTUR

WS2022

Termin5

APCS, Stack, Unterprogramme, Sektionen

Name, Vorname	Matrikelnummer	Anmerkungen
Datum	Raster (z.B. Mi3x)	Testat/Datum

Legende: V:Vorbereitung, D: Durchführung, P: Protokoll/Dokumentation, T: Testat

Ziele:

Verständnis für STACK-Befehle und deren Nutzung bei Unterprogrammen. Verständnis für die APCS (**A**RM **P**rocedure **C**all **S**tandard). Sehen wie der ARM-Befehlssatz im Speicher abgelegt wird. Programme mit möglichst geringer Codegröße zu entwickeln, sowie den Umgang mit einem Debugger/Simulator zu festigen.

Arbeitsverzeichnis:

Kopieren Sie sich das Verzeichnis, welches Ihnen im Praktikum zur Verfügung gestellt wird, in Ihr persönliches Verzeichnis. Dort stehen Ihnen dann alle benötigten Dateien zur Verfügung.

Vorbereitung

Arbeiten Sie sich in folgende Befehle des ARM-Prozessors und in den ARM Procedure Call Standard (APCS) ein:

Instruktion	Bedeutung
STMDB R13, {R4-R8, LR}	Speichert die Registerwerte R4 bis R8 sowie LR (=R14) an die Adresse, die in R13 (=SP) steht als voll absteigender Stack
LDMIA R13, {R4-R8, PC}	Lädt den Speicherinhalt von der Adresse, die in R13 (=SP) steht in Form eines voll absteigenden Stacks in die Register R4 bis R8 sowie PC (=R15)

Aufgabe 1:

In folgenden Tabellen ist jeweils ein Speicherauszug gezeigt.

Welche Werte stehen in den Registern; auf den Speicherstellen nach Ausführung der Blocktransferbefehle?

R9 = 0x8000

Inhalt	Adresse
11	0x8014
10	0x8010
9	0x800C
8	0x8008
7	0x8004
6	0x8000
5	0x7FFC
4	0x7FF8
3	0x7FF4
2	0x7FF0

LDMDA R9, {R1, R2, R6}

R1:

R2:

R6:

R9:

STMIB R9!, {R1, R2, R6}

R1:

R2:

R6:

R9:

Inhalt	Adresse
	0x8014
	0x8010
	0x800C
	0x8008
	0x8004
	0x8000
	0x7FFC
	0x7FF8
	0x7FF4
	0x7FF0

Aufgabe 2:

Schreiben Sie ein beliebiges, kleines Programm in ARM Assembler, das durch Unterprogramme strukturiert wird. Folgende Anforderungen werden an das Programm und die Unterprogramme gestellt:

Die APCS Konvention ist einzuhalten.

Das Hauptprogramm soll (mindestens) drei Unterprogramme (UPx) aufrufen.

- UP1 benutzt Nicht-Scratchregister und stellt keine Blattroutine dar (ruft somit weitere Unterprogramme auf)
- UP2 benutzt nur Scratchregister und stellt eine Blattroutine dar (ruft somit keine weiteren Unterprogramme auf)
- UP3 benutzt Nicht-Scratchregister und stellt keine Blattroutine dar (ruft z.B. UP1 auf)

Aufgabe 3:

Schreiben Sie das Programmbeispiel (Zusatzaufgabe) aus Termin 2 (selbst modifizierender Code) in ARM7-Assembler und testen Sie dieses. Die Umsetzung soll dem MU1-Code möglichst ähnlich sein. Beobachten Sie die sich ändernde Speicherstelle, den sich ändernden Befehl.
Achtung: Sollten Sie mit Haltepunkten (Breakpoints) an der zu untersuchenden Stelle arbeiten.

Beschäftigen Sie sich mit den Problemen, welches dieses Programm machen kann.

Warum funktioniert das Programm im Simulator?

Wie groß darf die Werteliste maximal werden?

Was passiert wenn die Werteliste zu groß ist?

..?

Aufgabe 4:

Berichtigen Sie das Programm nun so, dass der Programmcode keinen sich selbst modifizierenden Code mehr enthält. Das Programm im ROM (Read Only Memory) der .text-Section und die sich ändernden Daten im RAM (Random Access Memory) der .data-Section stehen.

Bericht

Der erforderliche Praktikumsbericht dient zu Ihrer Nachbereitung des Praktikums und wird stichprobenhaft überprüft. Er beinhaltet auch den zeilenweise kommentierten Quelltext.

```
//
// Loesungen zur Aufgabe2
//
//Name:
//Datum:

        .file   "Aufgabe2.S"
        .text
        .align  2
        .globalmain
        .type   main, function
main:



UP1:

UP2:

UP3:

.Lfe1:
        .size   main,.Lfe1-main
```

****

```
//
// Loesungen zur Aufgabe3
//
//Name:
//Datum:

        .file   "Aufgabe3.S"
        .text
        .align  2
        .globalmain
        .type   main, function
main:
// Hier den Code des selbst modifizierenden Code aus Termin2 in ARM7-Assembler



halt:
        b       halt

Total           word   0        ; Summe
Count           word   5        ; Anzahl der Elemente
Table           word   39       ; The numbers to total ...
                word   25       ;
                word   4        ;
                word   98       ;
                word   17       ;
.Lfe1:
        .size   main,.Lfe1-main

*****************************************************************************************

//
// Loesungen zur Aufgabe4
```

```
//
//Name:
//Datum:

        .file   "Aufgabe4.S"
        .text
        .align  2
        .globalmain
        .type   main, function
main:
// Hier den Code des nicht mehr selbst modifizierenden Code einfuegen



        bx      lr

        .data
Total           word   0        ; Summe
Count:          word   5        ; Anzahl der Elemente
Table           word   39       ; The numbers to total ...
                word   25       ;
                word   4        ;
                word   98       ;
                word   17       ;
.Lfe1:
        .size   main,.Lfe1-main
```