h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

RECHNERARCHITEKTUR

SS2016

Termin 5

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. Ziel ist es Programme mit möglichst geringer Codegröße zu implementieren, sowie der Umgang mit einem Debugger/Simulator.

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
STMFD R13, {R1-R4, LR}	Speichert die Registerwerte R1 bis R4 sowie LR (=R14) an die Adresse, die in R13 (=SP) steht als voll absteigender Stack
LDMFD R13, {R1-R4, PC}	Lädt den Speicherinhalt von der Adresse, die in R13 (=SP) steht in Form eines voll absteigenden Stacks in die Register R1 bis R4 sowie PC (=R15)

Aufgabe 1:

In folgenden Tabellen ist jeweils ein Speicherauszug gezeigt. Welche Werte stehen in den Registern nach Ausführung des Blocktransferbefehls? R9 = 0x8000

Inhalt	Adresse
11	.....
10	....
9	.....
8	….
7	0x8004
6	0x8000
5	....
4	....
3	....
2	....

LDMDA R9, {R1, R2, R6} LDMFD R9, {R1, R2, R6}

R1: R1:

R2: R2:

R6: R6:

Aufgabe 2:

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

Die APCS Konvention wird eingehalten

Das Programm benutzt (mindestens) drei Unterprogramme (UP):

- UP1 benutzt nur 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

Aufgabe 3:

Schreiben Sie das Programmbeispiel aus Termin 2 (selbst modifizierender Code) in ARM7-Assembler und testen Sie dieses. Beobachten Sie die sich ändernden Speicherstellen/Befehle.

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

Warum funktioniert das Programm im Simulator?

Wie groß dürfte die Werteliste werden?

...?

Aufgabe 4:

Berichtigen Sie das Programm nun so, dass der Programmcode (kein sich selbst modifizierenden Code) im ROM (Read Only Memory) der .text-Section und die sich ändernden Daten im RAM (Random Acces 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 Aufgabe3
//
//Name:
//Datum:

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



        bx      lr

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
        .global main
        .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
```