h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

RECHNERARCHITEKTUR

WS2019

Termin 4

LOAD, STORE, bedingte Befehle, Speicherbereiche

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

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

Ziele:

Verständnis für LOAD und STORE Befehle, bedingte Befehle und die verschiedenen Speicherbereiche. Ziel ist die Implementierung mit möglichst geringer Codegröße sowie der Umgang mit einem Debugger/ Simulator und der Entwicklungsumgebung.

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 die Gruppe der LOAD und STORE Befehle, bedingte Befehle und Verzweigungsbefehle am Beipiel der folgenden Befehle des ARM-Prozessors ein:

Instruktion	Bedeutung
ADDNE R1, R2, #1	R1 := R2 + 1, falls das Z-Bit im Prozessorstatuswort **nicht** gesetzt ist
LDR R1, [R2]	R1 := mem_{32}[R2]
LDREQ R1, [R2]	R1 := mem_{32}[R2], falls das Z-Bit im Prozessorstatuswort gesetzt ist
LDRB R1, [R2]	R1 := mem_{8}[R2]
STR R1, [R2]	mem_{32}[R2] := R1
STRB R1, [R2]	mem_{8}[R2] := R1
ADR R1, Marke	R1:=PC+(Offset zur Marke)
B Marke	PC wird auf Adresse der Marke gesetzt
BEQ Marke	PC wird auf Adresse der Marke gesetzt, falls das Z-Bit im Prozessorstatuswort gesetzt ist
BNE Marke	PC wird auf Adresse der Marke gesetzt, falls das Z-Bit im Prozessorstatuswort **nicht** gesetzt ist
LDR R1, = Marke	R1 := mem_{32}[PC+(Offset zur Hilfsmarke)] , dies ist eine Pseudoinstruktion

Aufgabe 1:

Auf welchen Adressen wird der Inhalt von Register r1 gespeichert? Ergänzen Sie sie Kommentarzeilen.

```
mov     r0, #0
str     r1, [r0], #4      // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
eor     r0, r0, r0
str     r1, [r0, #4]      // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
mov     r0, #0
str     r1, [r0]!         // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
sub     r0, r0, r0
str     r1, [r0, #4]!     // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
and     r0, r0, #0
strb    r1, [r0, #1]!     // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
mov     r1, #4
strb    r1, [r0, r1]!     // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
```

Aufgabe 2:

Bearbeiten Sie schriftlich die Fragen.

a) Auf welche Weise kann man die Condition-Code-Flags NZCV (Bedingungsbits) des Prozessorstatuswort (CPSR) setzen?

b) Wie wird die Pseudoinstruktion "ADR R1, Marke" vom Assembler umgesetzt? Schreiben Sie hierzu den Befehl in einen der vorgegebenen Programmrahmen und schauen Sie ihn sich im Debugger in der Mixed-Darstellung an. Vollziehen Sie die Umsetzung des Compiler nach und informieren Sie sich auch über Pipelining.

c) Das Prozessorstatuswort hat den Wert 0x8000013, wenn der Befehl "SUBEQ R1, R1, R1" ausgeführt wird. Was steht danach im Register R1? Weisen Sie Ihre Antwort mit einem Programm nach.

Aufgabe 3:

Es ist ein Programm "kopieren" zu entwickeln, welches eine Zeichenkette von StringA nach StringB kopiert. In der Zeichenkette StringA stehen nur Buchstaben "a..z" und A..Z". Die Zeichenkette hat als Ende-Kennung ein Nullzeichen, ist also nullterminiert. Die verwendeten Strings beinhalten maximal 255 Zeichen.

Aufgabe 4:

Nach dem Kopiervorgang sollen In StringB alle Buchstaben in Großbuchstaben A..Z gewandelt werden. Es ist ein Programm "grossschreibung" zu entwickeln.

Aufgabe 5:

In einem weiteren Programm "sortieren" soll StringB aufsteigend sortiert werden. Es gibt verschiedene Sortieralgorithmen (z.B. Bubblesort).

Aufgabe 6:

Dokumentieren Sie die Tests die gemacht wurden, um eine fehlerfreie Funktionalität der Programme nach zu weisen.

Bericht

Der erforderliche Praktikumsbericht dient zu Ihrer Nachbereitung des Praktikums und wird stichprobenhaft überprüft. Er hat auch den zeilenweisen kommentierten Quelltext zu beinhalten. Haben Sie Ihre Berichte zu den Praktikumsterminen dabei.

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

        .file   "aufgabe1.S"
        .text                   @ legt eine Textsection fuer ProgrammCode + Konstanten an
        .align  2               @ sorgt dafuer, dass nachfolgende Anweisungen auf einer durch 4 teilbaren Adresse liegen
                                @ unteren 2 Bit sind 0
        .global main            @ nimmt das Symbol main in die globale Sysmboltabelle auf
        .type   main,function
main:
        mov     r0, #0
        str     r1, [r0], #4            // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
        eor     r0, r0, r0
        str     r1, [r0, #4]            // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
        mov     r0, #0
        str     r1, [r0]!               // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
        sub     r0, r0, r0
        str     r1, [r0, #4]!           // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
        and     r0, r0, #0
        strb    r1, [r0, #1]!           // Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
        mov     r1, #4
        strb    r1, [r0, r1]!// Inhalt von r1 auf Adresse 0x_____ danach steht in r0 0x____
        bx      lr
.Lfe1:
        .size   main,.Lfe1-main

// End of File
```

****

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

        .file   "aufgabe2.S"
        .text                   @ legt eine Textsection fuer ProgrammCode + Konstanten an
        .align  2               @ sorgt dafuer, dass nachfolgende Anweisungen auf einer durch 4 teilbaren Adresse liegen
                                @ unteren 2 Bit sind 0
        .global main            @ nimmt das Symbol main in die globale Sysmboltabelle auf
        .type   main,function
main:




        bx      lr


.Lfe1:
        .size   main,.Lfe1-main

// End of File
```

****

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

        .file   "aufgabe3.S"
        .text           @ legt eine Textsection fuer ProgrammCode + Konstanten an
        .align          @ sorgt dafuer, dass nachfolgende Anweisungen auf einer durch 4 teilbaren Adresse liegen
                        @ unteren 2 Bit sind 0
        .global main    @ nimmt das Symbol main in die globale Sysmboltabelle auf
        .type   main, function
main:
        push    {lr}            @ Ruecksprungadresse und evtl. weitere Register sichern
        ..
        bl      kopieren
        ..
        bl      grossschreibung
        ..
        bl      sortieren
        pop     {pc}

kopieren:
@ hier Ihr Programm zum Kopieren eines String
        ...
        ...

grossschreibung:
@ hier Ihr Programm  um aus Kleinbuchstaben Grossbuchstaben zu machen
        ...
        ...

sortieren:
@ hier Ihr Programm um  sortieren
        ...
        ...


Adr_StringA:    .word   StringA         @ Hilfsvariable um an Adressen aus anderen Segmenten zu kommen
Adr_StringB:    .word   StringB         @ Hilfsvariable um an Adressen aus anderen Segmenten zu kommen


.Lfe1:
        .size   main,.Lfe1-main         @ Programmgroesse berechnen

// .data-Section fuer initialisierte Daten
        .data
//  Liste von Zeichen
StringA:  .byte   'D','i','e','s','s','i','n','d','v','i','e','l','e','B','u','c','h','s','t','a','b','e','n', 0


// .comm-Section fuer nicht initialisierte Daten
        .comm   StringB, 256    @ Speicherbereich fuer zu sortierenden StringB

// End of File
```

****