h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

RECHNERARCHITEKTUR

WS2025

Termin 4

LOAD, STORE, bedingte Befehle, Speicherbereiche, ASCII-Tabelle, APCS

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

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

Ziele:

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

Arbeitsverzeichnis:

Kopieren Sie sich das Verzeichnis raWS2025, welches Ihnen im Praktikum zur Verfügung gestellt wird, in Ihr persönliches Verzeichnis. Dort stehen Ihnen dann benötigte Dateien zur Verfügung.

Vorbereitung

Arbeiten Sie sich in die Gruppe der LOAD und STORE Befehle, bedingte Befehle und Verzweigungsbefehle am Beispiel 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 := mem32[PC+(Offset zur Hilfsmarke)] , dies ist eine Pseudoinstruktion

Aufgabe 1:

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

```
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], #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____
ands    r0, r0, #0
strb    r1, [r0, #2]!     // 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 bzw. mit welchen Befehlen kann der Zustand der Condition-Code-Flags NZCV (Bedingungsbits) des Prozessorstatuswort (CPSR) geändert werden?

b) Wie werden Pseudoinstruktion wie z.B.:

```
ADR  R1, Marke
LDR  R2, =Marke
```

vom Compiler umgesetzt? Schreiben Sie hierzu die Befehle in einen der vorgegebenen Programmrahmen und schauen Sie sich das Ergebnis im Debugger in der Mixed-Darstellung an. Vollziehen Sie die Umsetzung des Compiler nach und informieren Sie sich auch über Pipelining.
Was passiert wenn die Marke sich nicht in der Sektion .text befindet?
Wie kommen wir dann an die Adresse von Marken/Labels?

c) Das Prozessorstatuswort (CPSR) hat den Wert 0x40000013, wenn der Befehl “SUBEQS R1, R1, R1” ausgeführt wird.
Was steht danach im Register R1? _____
Was steht danach im CPSR? _____
Weisen Sie Ihre Antworten nach.

Aufgabe 3: *(Achtung: Ankündigung des Lehrenden beachten es gibt evtl. andere Aufgaben)*

Es ist ein Programm „wandel.S" zu entwickeln, welches einen nullterminierten String, in welchem nur Zeichen aus dem 7-bit ASCII-Zeichensatz verwendet sind, in einen String nach UTF-8 Kodierung wandelt und die Anzahl der gewandelten Umlaute zurück gibt.

In möglichen Strings sind die Umlaute ä, ü, ö, Ä, Ö, Ü entsprechend ae, ue, oe, Ae, Ue, Oe umschrieben. Das „ß" soll wenn nötig mit „sz" umschrieben sein.

Zur Lösung der Aufgabe gibt es einen vorbereiteten Unterordner Aufgabe3 mit Programmgerüsten (main.c und wandel.S) und einem Makefile. Wechseln Sie in den Unterordner, starten Sie dort snavigator, legen Sie ein neues Projekt an und entwickeln Ihre Lösung.

Aufgabe 4:

Dokumentieren Sie die Tests die gemacht werden, um eine fehlerfreie Funktionalität nachzuweisen. Beschreiben Sie Ihre Lösung. Welche Grenzen und evtl. Sonderfälle können auftreten?

Wie groß dürfen die zu wandelten Strings sein?

Kann es Fälle geben in denen die Wandlung zu Fehlern führen kann?

..

Bericht

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

Programmgerüst zu Aufgabe 1:

```
// 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
```

Programmgerüst zu Aufgabe 2:

```
// 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 Symboltabelle auf
        .type   main,function
main:
        adr     r1,  marke




        bx      lr
marke:
        .word   0x12345678


.Lfe1:
        .size   main,.Lfe1-main

// .data-Section fuer initialisierte Daten
        .data
marke1:
        .word   0x87654321

// .comm-Section fuer nicht initialisierte Daten
        .comm marke2,

// End of File
```

Assembler-Programmgerüst zu Aufgabe 3:

```
// Loesung zu Aufgabe 3 und folgende
// Name:                Matrikelnummer:
// Name:                Matrikelnummer:
// Datum:
        .file "wandel.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 wandel  @ nimmt das Symbol wandel in die globale Symboltabelle auf
        .type wandel, function

@ hier Ihr Programm um einen String von 7-bit ASCII nach UTF-8 zu wandeln
@ Adressen der Strings werden übergeben
@ Anzahl der gewandelten Zeichen wird zurück gegeben
wandel:
@       ..
        bx lr

.Lfe1:
        .size main,.Lfe1-main
// End of File
```

C-Programmgerüst zu Aufgabe 3:

```
// Programmrahmen zur Kontrolle einer Lösung von Aufgabe 3 vom Termin4 "wandel.S"
// von: Manfred Pester
// vom: 13.01.2025
//

#include <stdio.h>

// Funktionsprototypen fuer die zu entwickelnde Assemblerroutine
unsigned int wandel(char*, char*);

// Variablen
unsigned int AdgZ = 0;  // Anzahl der gewandelten Zeichen

// String ohne Umlaute nur Zeichen aus dem 7Bit ASCII-Zeichensatz
char ascii_string[] = "AeOeUeszaeoeue\n";

// String mit Umlauten nach UTF-8 Zeichensatz
char utf8_string[sizeof(ascii_string)];

int main()
{
      printf("String ohne Umlaute vor der Wandlung\n");
      printf(ascii_string);
      AdgZ = wandel( ascii_string, utf8_string);
      printf("String nach der Wandlung nun mit UTF-8 Kodierung und Umlauten\n");
      printf("%s", utf8_string);
      printf("\n");
      printf(„Es wurden %d Zeichen gewandelt.\n\r",  AdgZ);
      return 0;
}
```