



# h\_da

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**  
FACHBEREICH INFORMATIK

## RECHNERARCHITEKTUR

WS2024

Termin 4

LOAD, STORE, bedingte Befehle, Speicherbereiche, ASCII-Tabelle,  
UTF-8

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 raWS2024, 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 Beispiel der folgenden Befehle des ARM-Prozessors ein:

Instruktion	Bedeutung
ADDNE R1, R2, #1	$R1 := R2 + 1$ , falls das Z-Bit im Prozessorstatuswort <b>nicht</b> gesetzt ist
LDR R1, [R2]	$R1 := \text{mem}_{32}[R2]$
LDREQ R1, [R2]	$R1 := \text{mem}_{32}[R2]$ , falls das Z-Bit im Prozessorstatuswort gesetzt ist
LDRB R1, [R2]	$R1 := \text{mem}_8[R2]$
STR R1, [R2]	$\text{mem}_{32}[R2] := R1$
STRB R1, [R2]	$\text{mem}_8[R2] := R1$
ADR R1, Marke	$R1 := \text{PC} + (\text{Offset zur Marke})$
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 <b>nicht</b> gesetzt ist
LDR R1, = Marke	$R1 := \text{mem}_{32}[\text{PC} + (\text{Offset zur Hilfsmarke})]$ , 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_____
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 man die Condition-Code-Flags NZCV (Bedingungsbits) des Prozessorstatuswort (CPSR) setzen?

- 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 0x13, wenn der Befehl "SUBEQS R1, R1, R1" ausgeführt wird.

Was steht danach im Register R1? \_\_\_\_\_

Was steht danach im CPSR? \_\_\_\_\_

Weisen Sie Ihre Antwort nach.

### **Aufgabe 3:**

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.

Im gegebenen String sind die Umlaute ä, ü, ö, ß, Ä, Ö, Ü entsprechend ae, ue, oe, sz, Ae, Ue, Oe umschrieben.

Zur Lösung der Aufgabe gibt es einen vorbereiteten Unterordner Aufgabe3 mit Programmgerüsten und einem Makefile.

### **Aufgabe 4:**

Dokumentieren Sie die Tests die gemacht wurden, um eine fehlerfreie Funktionalität der Programme nachzuweisen.

### **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.

```
// 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:
        adr      r1, marke

        bx      lr
marke:
        .word   0x12345678

.Lfe1:
        .size   main,.Lfe1-main

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

// End of File
```

```
// 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 ASCII nach UTF-8 zu wandeln

@ Adresse des zu bearbeitenden String ist zu uebergeben

wandel:

```
@    ..
    bx lr
```

.Lfe1:

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

```
// Programmrahmen zur Kontrolle einer Loesung von Aufgabe 3 vom Termin4 "wandel.S"
```

```
// von: Manfred Pester
```

```
// vom: 17.10.2024
```

```
//
```

```
#include <stdio.h>
```

```
// Funktionsprototypen fuer die zu entwickelnde Assembleroutine
unsigned int wandel(char*);
```

```
// String ohne Umlaute
```

```
char f[] = "AeOeUeszaeoeue\n";
```

```
int main()
```

```
{
    printf("String ohne Umlaute vor der Wandlung\n");
    printf(f);
    wandel(f);
    printf("String nach der Wandlung nun mit UTF-8 Kodierung und Umlauten\n");
    printf(f);
    printf("\n");
    return 0;
}
```