



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

RECHNERARCHITEKTUR
WS2016

Termin 3

Arithmetische und logische Operationen

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

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

Ziele:

Verständnis für arithmetische und logische Operationen. Ziel ist die Implementierung mit möglichst geringer Codegröße sowie das Erlernen und Festigen des Umgangs mit einer Entwicklungsumgebung.

Vorbereitung

Arbeiten Sie sich in die datenverarbeitenden Befehle des ARM-Prozessors ein:

Instruktion	Bedeutung
AND	$Rd = Op1 \text{ AND } Op2$
EOR	$Rd = Op1 \text{ EOR } Op2$
SUB	$Rd = Op1 - Op2$
RSB	$Rd = Op2 - Op1$
ADD	$Rd = Op1 + Op2$
ADC	$Rd = Op1 + Op2 + \text{Carry}$
SBC	$Rd = Op1 - Op2 - \text{Carry}$
RSC	$Rd = Op2 - Op1 - \text{Carry}$
TST	setzt Condition Codes bzgl. $Op1 \text{ AND } Op2$
TEQ	setzt Condition Codes bzgl. $Op1 \text{ EOR } Op2$
CMP	setzt Condition Codes bzgl. $Op1 - Op2$
CMN	setzt Condition Codes bzgl. $Op1 + Op2$
ORR	$Rd = Op1 \text{ ORR } Op2$
MOV	$Rd = Op2$
BIC	$Rd = Op1 \text{ AND NOT } Op2$
MVN	$Rd = \text{NOT } Op2$ (Einerkomplement)

Aufgabe 1:

Was leisten die folgenden beiden Befehle?

ADD R0, R1, R1, LSL #4 _multiplikation mit 17 => $R9 = R1 + (R1 \cdot 16) = 17 \cdot R1$

MOV R0, R1, LSR #3 _division durch 8 => $R9 = R1 / 8$

Aufgabe 2:

Überlegen Sie sich, mit welchen Befehlen Sie die einzelnen Flags (NZCV) gesetzt bekommen.

Zum Beispiel setzt der Befehl: ADDS r0, r0, r0 nur das Zero-Flag wenn $R0=0$

__RSBS R9, R5, #3 für $R5 = 0x2$ setzt das C-Flag

__CMP R5, R4 für $R5 = 0x2$; $R4 = 0x3$ setzt das N-Flag

__nur Overflow ???

Aufgabe 3:

Füllen Sie die unten stehende Tabelle aus.

Die Register haben folgende Werte:

R0 = 0xAABBCCDD

R1 = 0xFFBBFFBB

R2 = 0xFFFFFFFF

R3 = zum Beispiel Ihre Matrikelnummer (rechtsbündig, Hexadezimalzahl) **0x123456**

R4 = 0x3

R5 = 0x2

R6 = 0x7ffffff

R7 = 0x80000000

Instruktion	R9 (hexadez.)	Zusatzfrage	Antwort
ANDS R9, R0, R3		Wie werden die Flags N, Z, C, V gesetzt?	
EOR R9, R3, R3		Gilt das Ergebnis für jeden Wert in R3?	
SUBS R9, R7, #3		Wie werden die Flags N, Z, C, V gesetzt?	
RSBS R9, R5, #3		Wie werden die Flags N, Z, C, V gesetzt?	
ADDS R9, R4, #12		Wie werden die Flags N, Z, C, V gesetzt?	
ADDS R9, R6, R4		Wie werden die Flags N, Z, C, V gesetzt?	
TST R4, #1		Wie werden die Flags N, Z, C, V gesetzt?	
TEQ R4, R4		Wie werden die Flags N, Z, C, V gesetzt?	
CMP R5, R4		Wie werden die Flags N, Z, C, V gesetzt?	
CMN R2, R5		Wie werden die Flags N, Z, C, V gesetzt?	
ORR R9, R0, R3			
MOV R9, #126			
BIC R9, R0, R1			
BIC R9, R2, #15			
MVN R9, R1			

Arbeitsverzeichnis:

Melden Sie sich mit Ihrer Matrikelnummer und dem zugehörigen Passwort an. Kopieren Sie sich vom zur Verfügung stehende Netzwerklaufwerk (/mnt/Originale/ra) das Verzeichnis raWS2016 in Ihr Arbeitsverzeichnis. Wechseln Sie in das Verzeichnis ~/raWS2016/Termin3/.

Aufgabe 4:

Überprüfen Sie Ihre Vorbereitung (Aufgaben 1 bis 3), wenn noch nicht geschehen, durch Debuggen entsprechender gegebener oder evtl. selbst geschriebener Programme.

Aufgabe 5:

Schreiben Sie ein Assemblerprogramm (Funktion: int floatoint(float)), welches den ganzzahligen Teil einer einfach genauen 32-Bit-Gleitpunktzahl (wird in Register R0 übergeben) als Integerwert im Register R0 zurück liefert. Testen und dokumentieren Sie Ihr Programm.

Der erforderliche Praktikumsbericht dient zu Ihrer Nachbereitung des Praktikums. Er beinhaltet die Formulierung der Lösungsidee, die Angabe der Größe der Programme in Bytes sowie den zeilenweisen kommentierten Quelltext. Haben Sie die Praktikumsberichte, für eine evtl. Kontrolle durch die Betreuer, dabei. Die Erstellung eines Berichts für jede Gruppe ist erlaubt.

```
//  
// Loesung zu int floattoint(float)  
//  
//Name:  
//Datum:  
    .file    "floattoint.S"  
    .text  
    .align  2  
    .global floattoint  
    .type   floattoint,function  
floattoint:  
// In Register R0 wird die float-Zahl uebergeben  
// Hier das Programm einfuegen  
  
    bx      lr // In Register R0 wird der float-Zahl uebergeben  
.Lfe1:  
    .size   floattoint,.Lfe1-floattoint  
  
// Zum Testen der in Assembler zu schreibenden Funktionen "int floattoint(float)"  
// von Manfred Pester  
// vom 03.08.2011  
//  
int floattoint(float);  
int main (void)  
{  
float    a = 0;  
float    b = 0xfffff;  
float    c = -1;  
float    d = 127;  
float    e = 0.5;  
int      ergebnis;  
    ergebnis = floattoint(a);  
    ergebnis = floattoint(b);  
    ergebnis = floattoint(c);  
    ergebnis = floattoint(d);  
    ergebnis = floattoint(e);  
    return 0;  
}
```

```
# makefile für Rechnerarchitekturpraktikum Termin 3 SS2016  
# von: Manfred Pester  
# vom: 26.06.2014
```

```
# Variable fuer den zu nutzenden Compiler  
GCC = arm-eb63-elf-gcc
```

```
all: aufgabe1 aufgabe2 aufgabe3 floattoint aufgabe5
```

```
aufgabe1: aufgabe1.S  
    $(GCC) -g aufgabe1.S -o aufgabe1.elf
```

```
aufgabe2: aufgabe2.S  
    $(GCC) -g aufgabe2.S -o aufgabe2.elf
```

```
aufgabe3: aufgabe3.S  
    $(GCC) -g aufgabe3.S -o aufgabe2.elf
```

```
floattoint: floattoint.S  
    $(GCC) -g -c floattoint.S
```

```
aufgabe5: aufgabe5.c floattoint.S  
    $(GCC) aufgabe5.c -S  
    $(GCC) -g -c floattoint.S  
    $(GCC) -g aufgabe5.c floattoint.o -o aufgabe5.elf
```

```
clean:  
    rm *.o  
    rm *.elf
```