



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

PRAKTIKUM
RECHNERARCHITEKTUR
SS2022
Termin 3

ARM: 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

Ziel der folgenden Aufgaben:

Verständnis für arithmetische und logische Operationen und die Flags im Statusregister. Weiteres Ziel ist die selbstständige 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)

Bereiten Sie die folgenden Aufgaben so vor, dass Sie die Ergebnisse und Programme inklusive Dokumentation zum Laborterminal präsentieren können.

Zur Erreichung des Testats müssen Sie die fertigen Programme vorführen und erklären können.

Aufgabe 1:

Was leisten die folgenden beiden Befehle?

negative Zahlen werden positiv, da beim logischen shift nur 0 eingefügt wird.

LSR R0, R1, #3 $R0 = R1 / 8$ Was passiert mit negativen Zahlen?
RSBS R0, R1, R1, LSL #2 $R0 = (R1 * 2^2) - R1, R0 = R1 * 3$

Aufgabe 2:

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

Vermeiden Sie hierbei - sofern möglich - das gleichzeitige Setzen von mehreren Flags und versuchen Sie eine Lösung zu erarbeiten, welche nur das entsprechende Flag setzt!

Sie können beliebige Werte in beliebigen Registern nutzen.

Beschriften und Befüllen Sie die von Ihnen gewählten Register neben der jeweiligen Zeile, damit klar wird, mit welchen Werten Sie rechnen.

Sie können gerne auch #Immediate-Operanden nutzen, anstelle eines zweiten Registers!

Beispiel (finden Sie für das N-Flag eine **eigene** Lösung!):

Registers		Flag
$R_1 = 0x00000001$	$R_2 = 0x80000000$	N: <u>ADDS R0, R1, R2</u>

Registers		Flag
$R_1 = 0x1$	$R_2 =$	N: <u>SUBS R0, R1, #2</u>
$R_1 = 0x0$	$R_2 =$	Z: <u>ADDS R0, R1, #0x0</u>
$R_1 = 0x7FFFFFFF$	$R_2 = 0xFFFFFFFF$	C: <u>ADDS R0, R1, R2</u> Es wird eine weitere Stelle zum anzeigen der Zahl benötigt, aber der Wert ist innerhalb der anzeigbaren Werte
$R_1 = 0x7FFFFFFF$	$R_2 = 0x1$	V: <u>ADDS R0, R1, R2</u> <u>LSRS R0, R0, #1</u> Erst Overflow und Negative Flag setzen, dann mit shift das negative Flag entfernen
(fürs V-Flag wären auch 2 Befehle erlaubt)		

Aufgabe 3:

Füllen Sie die unten stehende Tabelle aus.

Die Register haben folgende Werte: **NOT R1 = 0x00440044**

R0 = 0xAABBCCDD

R1 = 0xFFBBFFBB

R2 = 0xFFFFFFFF

R3 = zum Beispiel Ihre Matrikelnummer (rechtsbündig, Hexadezimalzahl) **1112879 = 0x0010FB2F**

R4 = 0x3

R5 = 0x2

R6 = 0x7FFFFFFF

R7 = 0x80000000

Instruktion	R9 (hexadez.)	Zusatzfrage	Antwort
ANDS R9, R0, R3	0x10C80D	Wie werden die Flags N, Z, C, V gesetzt?	<u>0</u> , <u>0</u> , <u>0</u> , <u>0</u>
EOR R9, R3, R3	0x0	Gilt das Ergebnis für jeden Wert in R3?	Ja/Nein Ja
SUBS R9, R7, #3	0x7FFFFFFD	Wie werden die Flags N, Z, C, V gesetzt?	<u>0</u> , <u>0</u> , <u>1</u> , <u>1</u>
RSBS R9, R5, #3	0x1	Wie werden die Flags N, Z, C, V gesetzt?	<u>0</u> , <u>0</u> , <u>1</u> , <u>0</u>
ADDS R9, R4, #12	0xF	Wie werden die Flags N, Z, C, V gesetzt?	<u>0</u> , <u>0</u> , <u>0</u> , <u>0</u>
ADDS R9, R6, R4	0x80000002	Wie werden die Flags N, Z, C, V gesetzt?	<u>1</u> , <u>0</u> , <u>0</u> , <u>1</u>
TST R4, #1	-	Wie werden die Flags N, Z, C, V gesetzt?	<u>0</u> , <u>0</u> , <u>*</u> , <u>*</u>
TEQ R4, R4	-	Wie werden die Flags N, Z, C, V gesetzt?	<u>0</u> , <u>1</u> , <u>*</u> , <u>*</u>
CMP R5, R4	-	Wie werden die Flags N, Z, C, V gesetzt?	<u>1</u> , <u>0</u> , <u>0</u> , <u>0</u>
CMN R2, R5	-	Wie werden die Flags N, Z, C, V gesetzt?	<u>0</u> , <u>1</u> , <u>1</u> , <u>0</u>
ORR R9, R0, R3	0xAABBFFFF		
MOV R9, #126	0x7E		
BIC R9, R0, R1	0x44		
BIC R9, R2, #15	0xFFFFFFFF0		
MVN R9, R1	0x440044		

NOT #15 = 0xFFFFFFFF0

Aufgabe 4:

Überprüfen Sie mit den gegebenen Programmen „aufgabe1.S“ bis „aufgabe3.S“ Ihre Lösungen der Aufgaben 1 bis 3.

Wechseln Sie in das Verzeichnis raSS2022/Termin3/. Starten Sie den snavigator und legen ein neues Projekt an. Beachten Sie dazu die Dokumentation zur Entwicklungsumgebung.

Sofern die Testprogramme andere Ergebnisse liefern: Analysieren Sie warum dies der Fall ist. Finden Sie die Fehler.

Aufgabe 5:

Es sind in den Registern R0 bis R3 „signed integer“-Werte gegeben. Überlegen Sie sich mindestens vier universell einsetzbare verschiedene Arten, wie Sie eine Vorzeichenumkehr bei Erhaltung des Betrags erreichen können -- zum Beispiel indem Sie die Schritte der 2er-Komplementbildung nachbilden. Programmieren, testen und dokumentieren Sie Ihre Verfahren und Erkenntnisse.

Zusatzaufgabe 1:

Schreiben Sie ein ARM-Assembler-Programm, welches aus jedem Nibble (8 Hex-Ziffern) des im Register 0 gegebenen Wert ASCII-Zeichen codiert und diese in den Registern R1 bis R8 ablegt.

Beispiel 1: R0 = 0x87654321 → R1 = 0x31; R2=0x32; ..; R8=0x38

Beispiel 2: R0 = 0xFEDCBA98 → R1 = 0x38; R2=0x39 ; R3=65; R4=0x42; ..; R8=0x46

Überprüfen Sie Ihr Programm darauf, ob Sie es mit noch weniger Code-Zeilen umsetzen können.

Zusatzaufgabe 2:

Schreiben Sie ein ARM-Assembler-Programm, welches den Inhalt von zwei beliebigen Registern tauscht, ohne zusätzliche (neben den zwei zu tauschenden) Register oder Speicherstellen zu verwenden.

Versuchen Sie so wenige Codezeilen wie möglich zu erreichen.

Der erforderliche Praktikumsbericht dient zu Ihrer Nachbereitung des Praktikums. Haben Sie die Praktikumsberichte, für eine evtl. Kontrolle durch die Betreuer, dabei. Die Erstellung eines Berichts für jede Gruppe ist erlaubt.

Zu Aufgabe 1:

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

```
.file    "aufgabe1.S"
.text    @ legt eine Textsection fuer PrgrammCode + 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:
    ASR    R0, R1, #3      @ ...
    RSBS   R0, R1, R1, LSL #3  @ ...

    bx     lr      @ Ruecksprung zum aufrufenden Programm

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

// End of File
```

Zu Aufgabe 2:

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

```
.file    "aufgabe2.S"
.text    @ legt eine Textsection fuer PrgrammCode + 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:
    MOV    r1, #1
    MOV    r2, #0x80000000

    ADDS   r0, r1, r2    @ ...

// ...

    bx     lr
.Lfe1:
    .size   main,.Lfe1-main

// End of File

*****
```

Zu Aufgabe 3:

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

```
.file "aufgabe3.S"
.text @ legt eine Textsection fuer PrgrammCode + 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:
    push {r4, r5, r6, r7, r9, lr}

    ldr R0, = 0xaabbccdd
    ldr R1, = 0xffbbffbb
    ldr R2, = 0xffffffff
    ldr r3, = 0x123456 @ z.B. Matrikelnummer
    ldr r4, = 0x3
    ldr r5, = 0x2
    ldr r6, = 0x7ffffff
    ldr r7, = 0x80000000

    @ R9 (hexadez.) - N, Z, C, V
    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? ja / nein
    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 @

    pop {r4, r5, r6, r7, r9, pc}
.Lfe1:
    .size main, .Lfe1-main

// End of File
```


zu Aufgabe 5:

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

```
.file "aufgabe5.S"
.text       @ legt eine Textsection fuer PrgrammCode + 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:
    push   {r4, r5, r6, r7, lr}
    mov    r4, #1
    mov    r5, #-1
    mov    r6, #15
    mov    r7, #0x80000000
//..

    pop    {r4, r5, r6, r7, pc}
.Lfe1:
    .size   main,.Lfe1-main

// End of File
```

zu Zusatzaufgabe 1:

```
// Name:      Matrikelnummer:
// Name:      Matrikelnummer:
// Datum:
    .file      "zusatzaufgabe1.S"
    .text      @ legt eine Textsection fuer PrgrammCode + 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:
    ldr r1, =0x876543221
//..

    bx        lr
.Lfe1:
    .size      main,.Lfe1-main

// End of File
```

Makefile für Rechnerarchitekturpraktikum Termin 3 SS2022
von: Manfred Pester
vom: 27.04.2020

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

all: aufgabe1 aufgabe2 aufgabe3 aufgabe5 zusatzaufgabe1

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 aufgabe3.elf

aufgabe5: aufgabe5.S
\$(GCC) -g aufgabe5.S -o aufgabe5.elf

zusatzaufgabe1: zusatzaufgabe1
\$(GCC) -g zusatzaufgabe1.S -o zusatzaufgabe1.elf

clean:
rm *.o
rm *.elf