

PRAKTIKUM
RECHNERARCHITEKTUR

Termin 3

ARM: Arithmetische und logische Operationen

Termin 3
ARM: Arithmetische und logische Operationen

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 zum Praktikumstermin präsentieren können.

Termin 3
ARM: Arithmetische und logische Operationen

Aufgabe 1:

Was leisten die folgenden beiden Befehle?

LSR R0,R1,#2 _____

ADD R0,R1,R1,LSL#3 _____

Aufgabe 2:

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

Im Register R0 steht 0x1 und im Register R1 steht 0x80000000.

Beispiel: ADDS R2,R0,R1 @setztz.B.nurdasNegativ-Flag

Aufgabe 3:

Füllen Sie die untenstehende Tabelle aus.

Die Register haben folgende Werte:

R0 = 0xAABBCCDD

R1 = 0xFFBBFFBB

R2 = 0xFFFFFFFF

R3 = 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?	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			

Aufgabe 4:

Überprüfen Sie mit den gegebenen Programmen „aufgabe1.S“ bis „aufgabe3.S“ Ihre Lösungen der Aufgaben 1 bis 3.
Sofern die Testprogramme andere Ergebnisse liefern: Analysieren Sie, warum dies der Fall ist.

Aufgabe 5:

Es sind in den Registern R0 bis R3 Werte gegeben, deren Vorzeichen Sie auf verschiedene Arten (z.B. 2K-Wandlung) umkehren sollen. Überlegen Sie sich mindestens drei weitere universell einsetzbare Verfahren, programmieren und testen Sie Ihre Verfahren.

Aufgabe 6:

Schreiben Sie ein ARM-Assembler-Programm, welches Byte1 und Byte 4, sowie Byte 2 und Byte 3 in einem Register vertauscht. Beispiel: 0x1234ABCD -> 0xCDAB3412.

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

Zusatzaufgabe 1:

Schreiben Sie ein ARM-Assembler-Programm, welches den Inhalt von zwei beliebigen Registern tauscht, ohne für den Tausch selbst zusätzliche (neben den zwei zu tauschenden) Register oder Speicherstellen zu verwenden. Versuchen Sie, so wenige Codezeilen wie möglich zu benötigen.

Zu Aufgabe 1:

```
global _start
_start:
main:

        LSR    R0,R1,#2        @ ...
        ADD    R0,R1,R1,LSL#3  @ ...

        bx     lr              @ Ruecksprung zum aufrufenden Programm

_end:
*****
```

Zu Aufgabe 2:

```
global _start
_start:
main:

        MOV    r0,#1
        MOV    r1,#0x80000000
@ ...
        ADDS   r2,r1,r0        @ ...

        bx     lr

_end:
*****
```

Zu Aufgabe 3:

```

.global _start
_start:
main:   push    {r4, r5, r6, r7, r9, lr}
        ldr     R0, =0xaabbccdd
        ldr     R1, =0xffbbffbb
        ldr     R2, =0xffffffff
        ldr     r3, =0x123456
        ldr     r4, =0x3
        ldr     r5, =0x2
        ldr     r6, =0x7fffffff
        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}

_end:

```

zu Aufgabe 5:

```

.global _start
_start:

main:   push    {r4, r5, lr}
        mov     r0, #1
        mov     r1, #-1
        mov     r2, #15
        mov     r3, #0x80000000

        @ ...

        pop     {r4, r5, pc}

_end:

```

zu Aufgabe 6:

```
global
_start:
main: ldr r1, =0x1234ABCD
//..
```

```
        bx      lr
_end:
```
