**4.5.2 Unbedingte Programmverzweigungen**

Unbedingte Programmverzweigungen sind Sprungbefehle. Diese Instruktionen werden verwendet um den

sequenziellen Programmlauf zu ändern. Einige dieser Sprungbefehle werden für den Aufruf eines

Unterprogrammes oder den Wechsel in den Thumb-Modus verwendet.

Durch das Konzept der bedingten Instruktionsausführung stellen diese Verzweigungsbefehle nur einen

Spezialfall der bedingten Instruktionsausführung dar.

Die Syntax der unbedingten Verzweigungsinstruktionen ist:

**<B | BL> <label>**

**BX Rm**

**BLX <label> | Rm**

Das Label verkörpert als symbolische Konstante die Sprungadresse zu der verzweigt werden soll. Sie wird vom

Assembler als pc-relativer Versatz (Offset) von der aktuellen Speicherstelle berechnet und muss innerhalb

32MB liegen. Während bei der B-/BL-Instruktion das Sprungziel als (symbolische) Konstante vorliegen muss,

kann mit den BX-/BLX- Verzweigungen auch registerbasiert eine Verzeigung erfolgen. Dies ist besonders dann

nützlich, wenn ein Sprungziel während des Programmlaufes berechnet wird oder aus einer Tabelle gelesen wird.

Die vier Verzweigungsinstruktionen sind in Tabelle 30 aufgeführt.

**Instruktion Sprungdistanz Beschreibung Wirkung**

B 32MB Generelle Programmverzweigung pc = Offset(label)

BL 32MB Unterprogrammaufruf.

Programmverzweigung mit Speichern der

Rücksprungadresse im Link-Register

pc = Offset(label)

lr=Adresse der nächsten

Instruktion nach BL

BX 32MB Analog B, aber mit Wechselmöglichkeit

in den 16-Bit Thumb-Modus.

pc = Rm & 0xfffffffe

T = Rm &1

BLX 32MB Analog BL, aber mit Wechselmöglichkeit

in den 16-Bit Thumb-Modus.

pc = Rm & 0xfffffffe

T = Rm &1

lr=Adresse der nächsten

Instruktion nach BL

Tabelle 30: Programmverzweigungen

Der Aufruf einer Subroutine (Unterprogramm) erfolgt normalerweise mit einer BL-Instruktion. Bei dieser

Sprungvariante wird in lr-Register (r14) die Adresse der dem Aufruf nachfolgenden Instruktion abgespeichert.

Sie dient als Rücksprungadresse aus der Subroutine.

Die maximal mögliche Sprungdistanz ist durch die Offsetcodierung im Instruktionswort mit 25-Bit (signed) fest

vorgegeben und beträgt im Maximum ca. 32MB.

Beim Wechsel in den Thumb-Modus mit Sprungziel in Rm wird die Sprungadresse +1 angegeben. Der

ARM-Prozessor ignoriert die 1-Ausrichtung indem einen Sprung nach Rm&0xFFFFFFFE ausführt. Enthält Rm

eine gerade Adresse, erfolgt kein Wechsel in den Thumb-Modus.

Wird BLX mit einer Labeladresse benutzt, erfolgt ein Wechsel in den Thumb-Modus ohne dass eine +1-Adresse

angegeben werden muss.

Version 3.2c, August 2011 Seite 84

Hardwarenahme Softwareentwicklung Kapitel 4: Instruktionssatz ARM V5

**Beispiel 4-26:**

Sprünge können grundsätzlich vorwärts und rückwärts im Programmcode erfolgen. Die hierzu notwendigen

Offsetrechnungen werden automatisch richtig vom Assembler durchgeführt. Der folgende Code zeigt wie

unbedingte Sprünge codiert werden können. Der erste Sprung überspringt die ersten drei Additionsbefehle und

führt direkt zur ersten Subtraktion. Der zweite Sprung im Code springt rückwärts und produziert eine nicht

abbrechende Schleife.

B vorwaerts

ADD r1,r2,#4

ADD r0,r6,#2

ADD r3,r7,#4

vorwaerts:

SUB r1,r2,#4

rueckwaerts:

ADD r1,r2,#4

SUB r1,r2,#4

ADD r4,r6,r7

B rueckwaerts

**Beispiel 4-27:**

Der Aufruf eines Unterprogrammes erfolgt konventionsgemäss mit einer BL-Instruktion. Der Rücksprung

erfolgt mit der durch BL in r14 eingespeicherten Rücksprungadresse. Der folgende Code zeigt das Prinzip des

Subroutinenaufrufes mit Rücksprung.

. . .

BL subroutine

ADD r0,r1,r2

. . .

subroutine:

<Subroutinen Code >

MOV pc,lr @Ruecksprung

Der ARM-Assembler kennt keinen speziellen Subroutinenrücksprungbefehl. Dies erfolgt immer mit MOV pc,lr

oder mit dem Lesen der Rücksprungadresse vom Stack.

**4.5.3 Bedingte Programmverzweigungen**

Durch das Konzept der bedingten Instruktionsausführung bei ARM können alle Instruktionen unter

Zugrundelegung einer Bedingung ausgeführt werden. Dies trifft daher auch für Sprungbefehle zu. Alle

Verzeigungsbefehle nach Tabelle 30 können daher durch Ergänzung mit einem Suffix zu bedingten

Verzweigungen ergänzt werden.

Die Syntax für die bedingten Sprungbefehle:

**<B | BL>{<cond>} label**

**BX{<cond>} Rm**

**BLX{<cond>} Rm**

Hierbei ist <cond> eine Bedingung als Suffix nach Tabelle 31. Der Rest ist analog Kapitel 4.5.2.

Version 3.2c, August 2011 Seite 85

Hardwarenahme Softwareentwicklung Kapitel 4: Instruktionssatz ARM V5

Tabelle 31: Bedingungen (Condition Codes) für bedingte Instruktionsausführung.

**Beispiel 4-28:**

Berechnen der Fakultät einer positiven Ganzzahl durch fortgesetzte Multiplikation mit Hilfe einer Subroutine.

Die Subroutine erhält die Zahl als Argument in r0 und führt die Kettenmultiplikation aus. Das Resultat wird r0

an die aufrufende Einheit retourniert.

main:

MOV r0,#5 @ Beispiel 5!

BL FakSub @ (r0!) berechnen

B main @ Endlosschleife

# Subroutine zum Berechnen der Fakultaet durch fortgesetzte Multiplikation

# n!=1\*n\*(n-1)\*(n-2)\*...\*3\*2

# Input r0: Argument n

# Output r0: Resultat n!

# Benutzte Register: r1

FakSub:

MOVS r1,r0 @ Zaehlerregister

MOVLT r0,#0 @ Wenn n<0, dann Resultat 0 (Konvention)

MOVGE r0,#1 @ Resultat von 0! oder Startwert für loop

BLE endFak @ Abbruch falls n<=0

loop:

MUL r0,r1,r0 @ Fortgesetzte Multiplikation n\*(n-1)\*(n-2)\*...\*3\*2

SUB r1,r1,#1 @ Zaehlerregister dekrementieren

CMP r1,#1 @ Zaehlerregister > 1: nächster loop

BGT loop

endFak:

MOV pc,lr @ Ruecksprung

Version