WICHTIGE ASSEMBLERBEFEHLE Zugelassenes Hilfsmittel für Prüfung MC

Im folgenden wird der flexible 2. Operand als **Op2** abgekürzt. Op2 kann entweder ein Register, ein geshiftetes Register oder ein 12-Bit-Immediatewert sein.

TRANSPORT- UND SPEICHERBEFEHLE

`mov rd, rn`	Registerinhalt kopieren	`Rd ← Rn`
`mov rd, #Imm`	16-Bit-Immediatewert in Register schreiben	`Rd = <Imm>`
`ldr rd, [ra, #offs]`	32 Bit von Adresse *(ra + Offset) lesen. Offset optional.	`Rd = *(Ra + <offs>)`
`str rs, [ra, #offs]`	32-Bit-Registerwert an Adresse *(ra + Offset) schreiben.	`*(Ra + <offs>) = Rs`
`ldrb,strb`	Wie ldr/str aber mit 8 Bit	`//`

RECHENOPERATIONEN, SHIFTS UND BOOLSCHE OPERATIONEN

`add rd, rn, Op2`	32 Bit Addition	`Rd = Rn + Op2`
`adc rd, rn, Op2`	32 Bit Addition mit Carry	`Rd = Rn + Op2 + C`
`sub rd, rn, Op2`	32 Bit Subtraktion	`Rd = Rn − Op2`
`sbc rd, rn, Op2`	32 Bit Subtraktion mit Carry	`Rd = Rn − Op2 + C − 1`
`mul rd, rn, rm`	32 Bit Multiplikation	`Rd = Rn * Rm`
`udiv rd, rn, rm`	32 Bit unsigned Division	`Rd = Rn / Rm`
`sdiv rd, rn, rm`	32 Bit signed Division	`Rd = Rn / Rm`
`lsl rd, rn, #n`	Left-Shift um n Bit	`Rd = Rn << n`
`lsr rd, rn, #n`	Right-Shift um n (logisch)	`Rd = Rn >> n`
`asr rd, rn, #n`	Right-Shift um n (arithmetisch)	`Rd = Rn >> n`
`and rd, rn, Op2`	Bitweises UND	`Rd = Rn & Op2`
`orr rd, rn, Op2`	Bitweises ODER	`Rd = Rn | Op2`
`eor rd, rn, Op2`	Bitweises XODER	`Rd = Rn ^ Op2`
`bic rd, Op2`	Bitweises UND NICHT (Bit clear)	`Rd = Rn & ~(Op2)`

s-Suffix an den Befehlen sorgt für Aktualisierung der PSR-Flags.

SPRÜNGE UND ENTSCHEIDUNGEN

`cmp rn, Op2`	Vergleich (setzt Flags in PSR)	`--`
`b label`	Unbedingter Sprung nach <label>	`PC = label`
`b<cond> label`	Bedingter Sprung nach <label>. Conditions siehe Tabelle unten.	`if (cond)` `  PC = label`
`bl label`	Sprung mit Rücksprungadresse im LR (Funktionsaufruf)	`LR = PC + 4` `PC = label`
`bx rn`	Sprung zu Adresse Rn	`PC ← Rn`

BEDINGUNGEN

`eq / ne`	Equal / Not-Equal	`== / !=  (Z gesetzt?)`
`gt / lt`	Signed Greater / Less Than	`> / <`
`ge / le`	Signed Greater / Less or Equal	`>= / <=`
`mi / pl`	Minus / Plus	`< 0 / > 0 (N gesetzt?)`
`cs / cc`	Carry Set / Carry Clear	`C gesetzt?`