ARM® Thumb® 16-bit Instruction Set ordered by machine code

Zürcher Hochschule
für Angewandte Wissenschaften
zh
aw
School of Engineering
InES Institute of Embedded Systems

This card lists all Thumb 16-bit instructions ordered by machine code to ease manually disassemble 16-bit Thumb code.
See the respective ***Thumb® 16-bit Instruction Set Quick Reference Card*** for details on the individual instructions.

Version 1.0, 2014-09-14, Andreas Gieriet

0000 – 0x0xxx Instructions
```
0000 0000 00mm mddd    MOVS  Rddd, Rmmm          ; Rddd  = Rmmm                   --> alias for LSLS Rddd,Rmmm,#0
0000 0iii iimm mddd    LSLS  Rddd, Rmmm, #0biiiii; Rddd  = Rmmm LSL #0b0iiiii
0000 1iii iimm mddd    LSRS  Rddd, Rmmm, #0biiiii; Rddd  = Rmmm LSR #0b0iiiii
```

0001 – 0x1xxx Instructions
```
0001 0iii iimm mddd    ASRS  Rddd, Rmmm, #0biiiii; Rddd  = Rmmm ASR #0b0iiiii
0001 100m mmnn nddd    ADDS  Rddd, Rnnn, Rmmm    ; Rddd  = Rnnn  +  Rmmm
0001 101m mmnn nddd    SUBS  Rddd, Rnnn, Rmmm    ; Rddd  = Rnnn  -  Rmmm
0001 110i iinn nddd    ADDS  Rddd, Rnnn, #0biii  ; Rddd  = Rnnn  +  #0b0iii
0001 111i iinn nddd    SUBS  Rddd, Rnnn, #0biii  ; Rddd  = Rnnn  -  #0b0iii
```

0010 – 0x2xxx Instructions
```
0010 0ddd iiii iiii    MOVS  Rddd, #0biiiiiiii   ; Rddd  =          #0b0iiiiiiii
0010 1nnn iiii iiii    CMP   Rnnn, #0biiiiiiii   ; flags = Rnnn  -  #0b0iiiiiiii
```

0011 – 0x3xxx Instructions
```
0011 0ddd iiii iiii    ADDS  Rddd, #0biiiiiiii   ; Rddd  = Rddd  +  #0b0iiiiiiii
0011 1ddd iiii iiii    SUBS  Rddd, #0biiiiiiii   ; Rddd  = Rddd  -  #0b0iiiiiiii
```

0100 – 0x4xxx Instructions
```
0100 0000 00mm mddd    ANDS  Rddd,  Rmmm         ; Rddd  = Rddd  &  Rmmm
0100 0000 01mm mddd    EORS  Rddd,  Rmmm         ; Rddd  = Rddd  ^  Rmmm
0100 0000 10mm mddd    LSLS  Rddd,  Rmmm         ; Rddd  = Rddd LSL Rmmm
0100 0000 11mm mddd    LSRS  Rddd,  Rmmm         ; Rddd  = Rddd LSR Rmmm
0100 0001 00mm mddd    ASRS  Rddd,  Rmmm         ; Rddd  = Rddd ASR Rmmm
0100 0001 01mm mddd    ADCS  Rddd,  Rmmm         ; Rddd  = Rddd  +  Rmmm +  carry
0100 0001 10mm mddd    SBCS  Rddd,  Rmmm         ; Rddd  = Rddd  -  Rmmm - ~carry
0100 0001 11mm mddd    RORS  Rddd,  Rmmm         ; Rddd  = Rddd ROR Rmmm
0100 0010 00mm mddd    TST   Rddd,  Rmmm         ; flags = Rddd  &  Rmmm
0100 0010 01mm mddd    RSBS  Rddd,  Rmmm, #0     ; Rddd  = 0     -  Rmmm   --> alias for NEGS Rddd, Rmmm
0100 0010 10mm mnnn    CMP   Rnnn,  Rmmm         ; flags = Rnnn  -  Rmmm
0100 0010 11mm mnnn    CMN   Rnnn,  Rmmm         ; flags = Rnnn  +  Rmmm
0100 0011 00mm mddd    ORRS  Rddd,  Rmmm         ; Rddd  = Rddd  |  Rmmm
0100 0011 01mm mddd    MULS  Rddd,  Rmmm         ; Rddd  = Rddd  *  Rmmm
0100 0011 10mm mddd    BICS  Rddd,  Rmmm         ; Rddd  = Rddd  & ~Rmmm   --> bit clear
0100 0011 11mm mddd    MVNS  Rddd,  Rmmm         ; Rddd  =         ~Rmmm
0100 0100 dmmm mddd    ADD   Rdddd, Rmmmm        ; Rdddd = Rdddd + Rmmmm
0100 0101 nmmm mnnn    CMP   Rnnnn, Rmmmm        ; flags = Rnnnn - Rmmmm
0100 0110 dmmm mddd    MOV   Rdddd, Rmmmm        ; Rdddd =         Rmmmm
0100 0111 0mmm m...    BX    Rmmmm               ; PC =           Rmmmm&~0b01  (mmmm == 0b1111: unpredictable)
0100 0111 1mmm m...    BLX   Rmmmm               ; LR = PC, PC = Rmmmm&~0b01  (mmmm == 0b1111: unpredictable)
0100 1ttt iiii iiii    LDR   Rttt, =label        ; Rttt = [((PC+2)&~0b011))+0b0iiiiiiii00] --> +1020 max
```

0101 – 0x5xxx Instructions
```
0101 000m mmnn nttt    STR   Rttt, [Rnnn, Rmmm]  ; [Rnnn + Rmmm] = Rttt
0101 001m mmnn nttt    STRH  Rttt, [Rnnn, Rmmm]  ; [Rnnn + Rmmm] = Rttt                 --> low half
0101 010m mmnn nttt    STRB  Rttt, [Rnnn, Rmmm]  ; [Rnnn + Rmmm] = Rttt                 --> low byte
0101 011m mmnn nttt    LDRSB Rttt, [Rnnn, Rmmm]  ; Rttt<sss1> = [Rnnn + Rmmm]<1>        --> low byte
0101 100m mmnn nttt    LDR   Rttt, [Rnnn, Rmmm]  ; Rttt = [Rnnn + Rmmm]
0101 101m mmnn nttt    LDRH  Rttt, [Rnnn, Rmmm]  ; Rttt<0021> = [Rnnn + Rmmm]<21>       --> low half
0101 110m mmnn nttt    LDRB  Rttt, [Rnnn, Rmmm]  ; Rttt<0001> = [Rnnn + Rmmm]<1>        --> low byte
0101 111m mmnn nttt    LDRSH Rttt, [Rnnn, Rmmm]  ; Rttt<ss21> = [Rnnn + Rmmm]<21>       --> low half
```

0110 – 0x6xxx Instructions
```
0110 0iii iinn nttt    STR   Rttt, [Rnnn, #off]  ; [Rnnn + 0b0iiiii00] = Rttt           --> +124 max
0110 1iii iinn nttt    LDR   Rttt, [Rnnn, #off]  ; Rttt = [Rnnn + 0x0iiiii00]           --> +124 max
```

0111 – 0x7xxx Instructions
```
0111 0iii iinn nttt    STRB  Rttt, [Rnnn, #off]  ; [Rnnn + 0b0iiiii] = Rttt             --> +31 max, low byte
0111 1iii iinn nttt    LDRB  Rttt, [Rnnn, #off]  ; Rttt<0001> = [Rnnn + 0x0iiiii]<1>    --> +31 max, low byte
```

1000 – 0x8xxx Instructions
```
1000 0iii iinn nttt    STRH  Rttt, [Rnnn, #off]  ; [Rnnn + 0x0iiiii0] = Rttt            --> +62 max, low half
1000 1iii iinn nttt    LDRH  Rttt, [Rnnn, #off]  ; Rttt<0021> = [Rnnn + 0x0iiiii0]<21> --> +62 max, low half
```

1001 – 0x9xxx Instructions
```
1001 0ttt iiii iiii    STR   Rttt, [SP, #off]; [SP + 0b0iiiiiiii00] = Rttt             --> +1020 max
1001 1ttt iiii iiii    LDR   Rttt, [SP, #off]; Rttt = [SP + 0b0iiiiiiii00]             --> +1020 max
```

1010 – 0xAxxx Instructions
```
1010 0ddd iiii iiii    ADR   Rddd, label     ; Rddd = ((PC+2)&~0b011)+0b0iiiiiiii00   --> +1020 max
1010 1ddd iiii iiii    ADD   Rddd, SP, #off  ; Rddd = SP + 0b0iiiiiiii00              --> +1020 max
```

1011 – 0xBxxx Instructions
```
1011 0000 0iii iiii    ADD   SP, SP, #off    ; SP = SP + 0b0iiiiiii00                 --> +508 max
1011 0000 1iii iiii    SUB   SP, SP, #off    ; SP = SP - 0b0iiiiiii00                 --> +508 max
1011 00i1 iiii innn    CBZ  Rnnn, label      ; if Rnnn==zero, PC = PC + 0x0iiiiii0    --> +126 max
1011 0010 00mm mddd    SXTH Rddd, Rmmm       ; Rddd<ss21> = Rmmm<4321> --> low half
1011 0010 01mm mddd    SXTB Rddd, Rmmm       ; Rddd<sss1> = Rmmm<4321> --> low byte
1011 0010 10mm mddd    UXTH Rddd, Rmmm       ; Rddd<0021> = Rmmm<4321> --> low half
1011 0010 11mm mddd    UXTB Rddd, Rmmm       ; Rddd<0001> = Rmmm<4321> --> low byte
1011 0100 rrrr rrrr    PUSH {reg0-7}         ; rrrrrrrr = Lo reg-mask --> pushes regs to SP (decrements SP)
1011 0101 rrrr rrrr    PUSH {LR,reg0-7]      ; rrrrrrrr = Lo reg-mask --> pushes regs to SP (decrements SP)
1011 0110 0100 xxxx    -                     ; unpredictable
1011 0110 0101 0...    SETEND LE             ; sets little-endian mode in CPSR
1011 0110 0101 1...    SETEND BE             ; sets big-endian mode in CPSR
1011 0110 0110 0aif    CPSIE aif             ; Enable Processor State  --> a=imprecise-abort, i=IRQ, f=FIQ
1011 0110 0111 0aif    CPSID aif             ; Disable Processor State --> a=imprecise-abort, i=IRQ, f=FIQ
1011 0110 011x 1xxx    -                     ; unpredictable
1011 10i1 iiii innn    CBNZ Rnnn, label      ; if Rnnn!=zero, PC = PC + 0x0iiiiii0      --> + 126 max
1011 1010 00mm mddd    REV   Rddd, Rmmm      ; Rddd<4321> = Rmmm<1234> --> reverse all
1011 1010 01mm mddd    REV16 Rddd, Rmmm      ; Rddd<4321> = Rmmm<3412> --> reverse low half, rev. upper half
1011 1010 10xx xxxx    -                     ; undefined
1011 1010 11mm mddd    REVSH Rddd, Rmmm      ; Rddd<4321> = Rmmm<ss12> --> reverse low half, sign extended
1011 1100 rrrr rrrr    POP  {reg0-7}         ; rrrrrrrr = Lo reg-mask  --> pops regs from SP (increments SP)
1011 1101 rrrr rrrr    POP  {PC,reg0-7]      ; rrrrrrrr = Lo reg-mask  --> pops regs from SP (increments SP)
1011 1110 iiii iiii    BKPT #0biiiiiiii      ; breakpoint, arg ignored by HW
1011 1111 0000 0000    NOP                   ; do nothing
1011 1111 0001 0000    YIELD                 ; do nothing, NOP-Hint: signal to HW to suspend/resume threads
1011 1111 0010 0000    WFE                   ; do nothing, NOP-Hint, wait for event
1011 1111 0011 0000    WFI                   ; do nothing, NOP-Hint: wait for interrupt
1011 1111 0100 0000    SEV                   ; do nothing, NOP-Hint: signal event to multi-processor system
1011 1111 cccc mmmm    ITsel cond            ; if-then: sel=mmmm: T=then/E=else, cond=cccc: as for Bcc<11:8>
```

1100 – 0xCxxx Instructions
```
1100 0nnn rrrr rrrr    STMIA Rnnn! {reg0-7}  ; rrrrrrrr = Lo reg-mask, inc Rnnn
1100 1nnn rrrr rrrr    LDMIA Rnnn! {reg0-7}  ; rrrrrrrr = Lo reg-mask, inc Rnnn if Rnnn not in mask
                       LDMIA Rnnn  {reg0-7}  ; rrrrrrrr = Lo reg-mask, load Rnnn if Rnnn in mask
```

1101 – 0xDxxx Instructions
```
1101 0000 iiii iiii    BEQ label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 0001 iiii iiii    BNE label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 0010 iiii iiii    BHS/BCS label         ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 0011 iiii iiii    BLO/BCC label         ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 0100 iiii iiii    BPL label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 0101 iiii iiii    BMI label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 0110 iiii iiii    BVS label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 0111 iiii iiii    BVC label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 1000 iiii iiii    BHI label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 1001 iiii iiii    BLS label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 1010 iiii iiii    BGE label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 1011 iiii iiii    BLT label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 1100 iiii iiii    BGT label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 1101 iiii iiii    BLE label             ; if true, PC = PC + 0biiiiiiii0 --> -256/+254 max
1101 1110 xxxx xxxx    -                     ; undefined --> can be used for instruction emulation
1101 1111 iiii iiii    SVC #0biiiiiiii       ; supervisor call (formerly called SWI), arg ignored by HW
```

1110 – 0xExxx Instructions
```
1110 0iii iiii iiii    B label               ; PC = PC + 0biiiiiiiiiii0       --> -2048/+2046 max
1110 1xxx xxxx xxxx    -                     ; 32 bit instructions
```

1111 – 0xFxxx Instructions
```
1111 xxxx xxxx xxxx    -                     ; 32 bit instructions
```

Notes:
1) a dot means don't care, but must be set to 0.
2) <4321>: word, <21>: low half word, <1>: low byte, <0001>: zero extend byte, <sss1>: sign extend byte, etc.
3) Undefined instructions can be used to emulate instructions (they trigger the undefined exception).
4) Unpredictable instructions do any unpredictable actions and are therefore illegal instructions.
5) Unallocated codes are undefined unless they are explicitly marked as unpredictable.