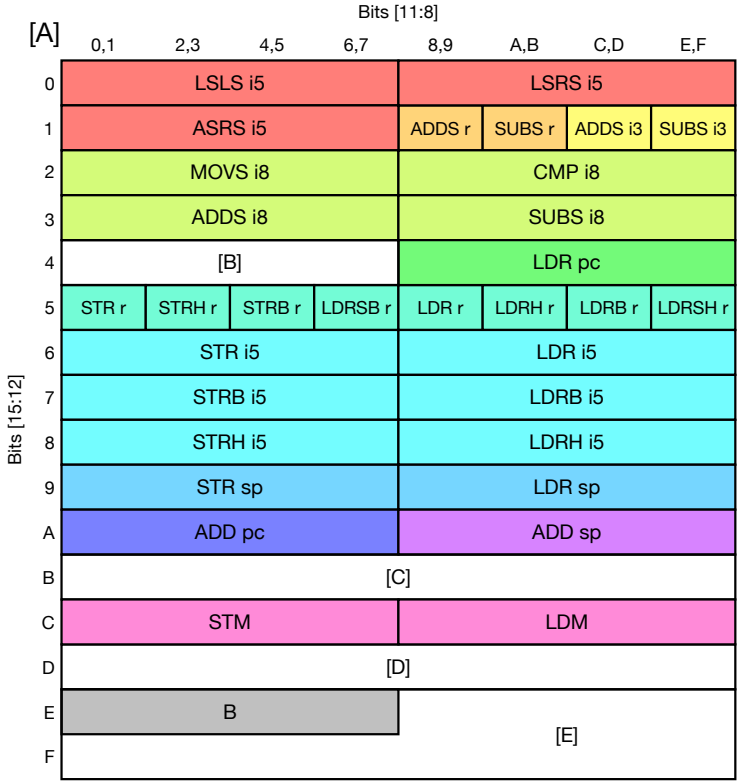
[A]



LSLS/LSRS/ASRS r1, r2, #imm5

ADDS/SUBS r1, r2, r3

ADDS/SUBS r1, r2, #imm3

MOVS/CMP/ADDS/SUBS r1, #imm8

LDR r1, [pc, #4*imm8] (*aka* LDR r1, =const)

STRx/LDRx r1, [r2, r3]

STRx/LDRx r1, [r2, #s*imm5]

STR/LDR r1, [sp, #4*imm8]

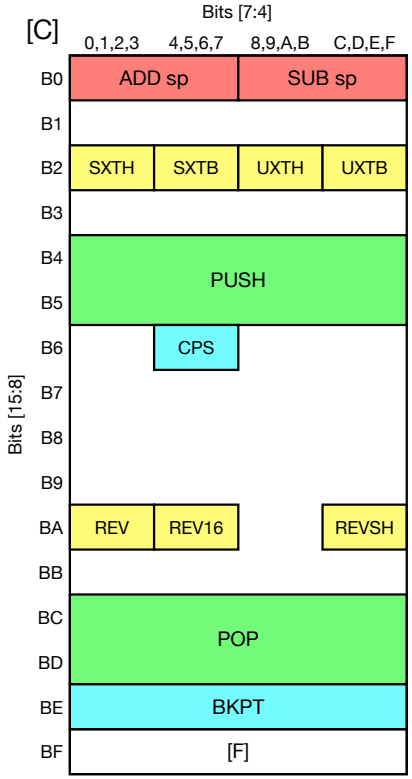
ADD r1, pc, #4*imm8 (*aka* ADR r1, label)

ADD r1, sp, #4*imm8

STM/LDM r1!, {regs}

B 2*disp11

[C]



ADD/SUB sp, sp, #4*imm7

SXTH/SXTB/UXTH/UXTB r1, r2

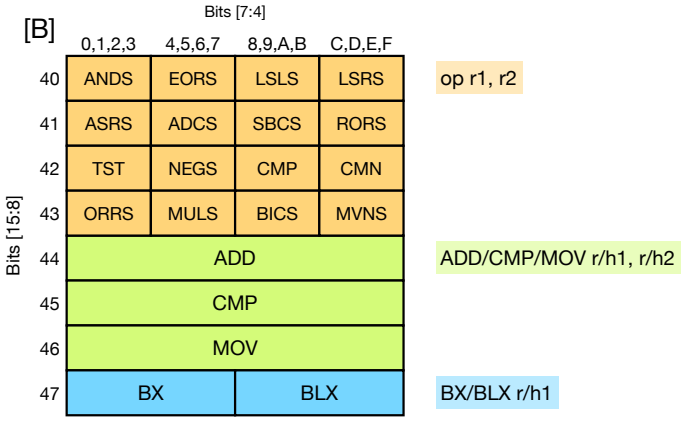
PUSH/POP {regs}

CPSIE/CPSID i

REV/REV16/REVSH r1, r2

BKPT #imm8

[B]

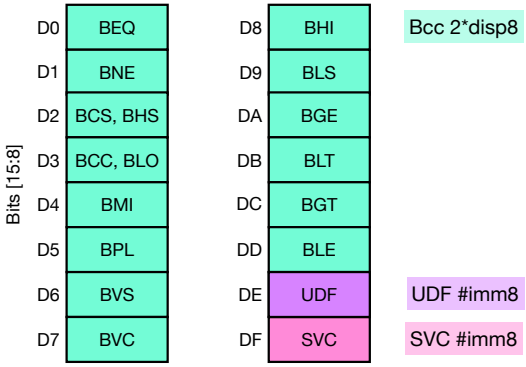


op r1, r2

ADD/CMP/MOV r/h1, r/h2

BX/BLX r/h1

[D]



Bcc 2*disp8

UDF #imm8

SVC #imm8

[E] 32-bit instructions:

F38x	88xx	MSR special, r1
F3EF	8xxx	MRS r1, special
F3BF	8F4x	DSB
F3BF	8F5x	DMB
F3BF	8F6x	ISB
Fxxx	Fxxx	BL 2*disp22

[F] Special instructions:

BF00	NOP
BF10	YIELD
BF20	WFE
BF30	WFI
BF40	SEV

Thumb code quick reference

	Syntax	Action	Flags	Notes
Move				
Immediate	`movs` r*d*, *#imm*	R*d* := *imm*	NZ	Range 0–255
Reg to reg	`movs` r*d*, r*m*	R*d* := R*m*	NZ	Synonym for `lsls` r*d*, r*m*, #0
High regs	`mov` r*d*, r*m*	R*d* := R*m*		Regs R8–R12, *sp*, *lr*, *pc* allowed
Add				
Register	`adds` r*d*, r*n*, r*m*	R*d* := R*n* + R*m*	NZCV	
Immediate	`adds` r*d*, r*n*, *#imm*	R*d* := R*n* + *imm*	NZCV	Range 0–7 or 0–255 if R*n* ≡ R*d*
With carry	`adcs` r*d*, r*d*, r*m*	R*d* := R*d* + R*m* + *C*	NZCV	
Value to *sp*	`add sp, sp,` *#imm*	*sp* := *sp* + *imm*		Range 0–508 (word aligned)
Form addr from *sp*	`add` r*d*, `sp,` *#imm*	R*d* := *sp* + *imm*		Range 0–1020 (word aligned)
Form addr from *pc*	`adr` r*d*, *label*	R*d* := *label*		Range *pc* to *pc* + 1020
Subtract				
Register	`subs` r*d*, r*n*, r*m*	R*d* := R*n* − R*m*	NZCV	
Immediate	`subs` r*d*, r*n*, *#imm*	R*d* := R*n* − *imm*	NZCV	Range 0–7 or 0–255 if R*n* ≡ R*d*
With carry	`sbcs` r*d*, r*d*, r*m*	R*d* := R*d*−R*m*−(1−*C*)	NZCV	
Value from *sp*	`sub sp, sp,` *#imm*	*sp* := *sp* − *imm*		Range 0–508 (word aligned)
Negate	`negs` r*d*, r*m*	R*d* := −R*m*	NZCV	
Multiply				
Register	`muls` r*d*, r*n*, r*d*	R*d* := R*n* ∗ R*d*	NZ	
Compare				
Register	`cmp` r*n*, r*m*	R*n* − R*m*	NZCV	Updates flags from result
Immediate	`cmp` r*n*, *#imm*	R*n* − *imm*	NZCV	Range 0–255
Bitwise				
AND	`ands` r*d*, r*d*, r*m*	R*d* := R*d* AND R*m*	NZ	
Exclusive OR	`eors` r*d*, r*d*, r*m*	R*d* := R*d* XOR R*m*	NZ	
OR	`orrs` r*d*, r*d*, r*m*	R*d* := R*d* OR R*m*	NZ	
Bit clear	`bics` r*d*, r*d*, r*m*	R*d* := R*d* AND NOT R*m*	NZ	
Move NOT	`mvns` r*d*, r*m*	R*d* := NOT R*m*	NZ	
Test bits	`tst` r*n*, r*m*	R*n* AND R*m*	NZ	Updates flags from result
Shift				
Logical shift left	`lsls` r*d*, r*m*, *#imm*	R*d* := R*m* ≪ *imm*	NZC	} C flag set to last bit shifted out, or unchanged if shift is zero
	`lsls` r*d*, r*d*, r*m*	R*d* := R*d* ≪ R*m*	NZC	
Logical shift right	`lsrs` r*d*, r*m*, *#imm*	R*d* := R*m* ≫ *imm*	NZC	
	`lsrs` r*d*, r*d*, r*m*	R*d* := R*d* ≫ R*m*	NZC	
Arith shift right	`asrs` r*d*, r*m*, *#imm*	R*d* := R*m* ASR *imm*	NZC	
	`asrs` r*d*, r*d*, r*m*	R*d* := R*d* ASR R*m*	NZC	
Rotate right	`rors` r*d*, r*d*, r*m*	R*d* := R*d* ROR R*m*	NZC	
Load				
Word, imm offset	`ldr` r*t*, `[`r*n*,*#imm*`]`	R*t* := Mem$_4$[R*n*+*imm*]		Range 0–124, mult of 4
Word, reg offset	`ldr` r*t*, `[`r*n*,r*m*`]`	R*t* := Mem$_4$[R*n*+R*m*]		
Halfword, immed	`ldrh` r*t*, `[`r*n*,*#imm*`]`	R*t* := Mem$_2$[R*n*+*imm*]		Range 0–62, mult of 2
Halfword, register	`ldrh` r*t*, `[`r*n*,r*m*`]`	R*t* := Mem$_2$[R*n*+R*m*]		
Signed halfword	`ldrsh` r*t*, `[`r*n*,r*m*`]`	R*t* := *sext*(Mem$_2$[R*n*+R*m*])		
Byte, imm offset	`ldrb` r*t*, `[`r*n*,*#imm*`]`	R*t* := Mem$_1$[R*n*+*imm*]		Range 0–31

Thumb code quick reference (*continued*)

	Syntax	Action	Notes
Load (*continued*)			
Byte, reg offset	`ldrb` r*t*, `[`r*n*,r*m*`]`	R*t* := Mem$_1$[R*n*+R*m*]	
Signed byte	`ldrsb` r*t*, `[`r*n*,r*m*`]`	R*t* := *sext*(Mem$_1$[R*n*+R*m*])	
PC-relative	`ldr` r*t*, *label*	R*t* := Mem$_4$[*label*]	Range *pc* to *pc* + 1020
SP-relative	`ldr` r*t*, `[sp,`*#imm*`]`	R*t* := Mem$_4$[*sp*+*imm*]	Range 0–1020, mult of 4
Store			
Word, imm offset	`str` r*t*, `[`r*n*,*#imm*`]`	Mem$_4$[R*n*+*imm*] := R*t*	Range 0–124, mult of 4
Word, reg offset	`str` r*t*, `[`r*n*,r*m*`]`	Mem$_4$[R*n*+R*m*] := R*t*	
Halfword, immed	`strh` r*t*, `[`r*n*,*#imm*`]`	Mem$_2$[R*n*+*imm*] := R*t*[15:0]	
Halfword, register	`strh` r*t*, `[`r*n*,r*m*`]`	Mem$_2$[R*n*+R*m*] := R*t*[15:0]	
Byte, imm offset	`strb` r*t*, `[`r*n*,*#imm*`]`	Mem$_1$[R*n*+*imm*] := R*t*[7:0]	Range 0–31
Byte, reg offset	`strb` r*t*, `[`r*n*,r*m*`]`	Mem$_1$[R*n*+R*m*] := R*t*[7:0]	
SP-relative	`str` r*t*, `[sp,`*#imm*`]`	Mem$_4$[*sp*+*imm*] := R*t*	Range 0–1020, mult of 4
Push and pop			
Push	`push {`*regset*`}`		} Subset of R0–R7
Push with link	`push {`*regset*`, lr}`		
Pop	`pop {`*regset*`}`		
Pop and return	`pop {`*regset*`, pc}`		
Branch			
—if equal	`beq` *label*	*pc* := *label* —if *Z*	Range −252 to +258 bytes
—if not equal	`bne` *label*	—if !*Z*	
—if higher or same	`bhs` *label*	—if *C*	Synonym for `bcs`
—if lower	`blo` *label*	—if !*C*	Synonym for `bcc`
—if minus	`bmi` *label*	—if *N*	
—if plus	`bpl` *label*	—if !*N*	
—if overflow	`bvs` *label*	—if *V*	
—if not overflow	`bvc` *label*	—if !*V*	
—if higher	`bhi` *label*	—if *C* && !*Z*	
—if lower or same	`bls` *label*	—if !*C* ‖ *Z*	
—if greater or eq	`bge` *label*	—if *N* == *V*	
—if less than	`blt` *label*	—if *N* != *V*	
—if greater than	`bgt` *label*	—if !*Z* && *N* == *V*	
—if less or eq	`ble` *label*	—if *Z* ‖ *N* != *V*	
Unconditional	`b` *label*	*pc* := *label*	Range ±2 KB
Branch with link	`bl` *label*	*lr* := *next*; *pc* := *label*	
Branch to reg	`bx` r*m*	*pc* := R*m*	} High regs allowed
Branch reg & link	`blx` r*m*	*lr* := *next*; *pc* := R*m*	
No operation	`nop`		
Extend			
Signed byte	`sxtb` r*d*, r*m*	R*d* := *sext*(R*m*[7:0])	
Unsigned byte	`uxtb` r*d*, r*m*	R*d* := R*m*[7:0]	
Signed halfword	`sxth` r*d*, r*m*	R*d* := *sext*(R*m*[15:0])	
Unsigned halfword	`uxth` r*d*, r*m*	R*d* := R*m*[15:0]	