

ARM® および Thumb®-2 命令セット クイックリファレンスカード

表の凡例	
Rm {, <opsh>}	表「レジスタと定数によるシフトのオプション」を参照して下さい。
<Operand2>	表「フレキシブル第2オペランド」を参照して下さい。シフトおよびロテートは Operand2 の一部としてのみ使用できます。
<fields>	表「PSR フィールド」を参照して下さい。
<PSR>	APSR (アプリケーションプログラムステータスレジスタ)、CPSR (カレントプロセッサステータスレジスタ)、または SPSR (保存プロセッサステータスレジスタ) を指定します。 フラグはアーキテクチャ v4 以前では予測できません。アーキテクチャ v5 以降では変化しません。
C*, V*	Rs またはシフト値をイミディエート値で指定できます。シフトの種類ごとに指定できる値は、表「レジスタと定数によるシフトのオプション」に記載されている値と同じです。
<Rs sh>	
x, y	B はハーフレジスタ [15:0] を示し、T は [31:16] を示します。
<imm8m>	ARM : 8 ビット値を偶数ビット数分右にロテートして形成される 32 ビット定数です。 Thumb : 8 ビット値を任意のビット数分左にシフトして形成される 32 ビット定数か、0xXYXYXYXY、0x00XY00XY、または 0xXY00XY00 形式のいずれかのビットパターンです。
<prefix>	表「並列命令の接頭文字」を参照して下さい。
{IA IB DA DB}	ポストインクリメント (IA)、プレインクリメント (IB)、ポストデクリメント (DA)、またはブレデクリメント (DB) です。 IB および DA は Thumb 状態では使用できません。省略すると、デフォルトで IA になります。
<size>	B、SB、H、または SH はそれぞれ、バイト、符号付きバイト、ハーフワード、符号付きハーフワードを意味します。 SB および SH は、STR 命令では使用できません。
<reglist>	コンマで区切られたレジスタのリストを中括弧 ((および)) で囲んで指定します。
<reglist-PC>	<reglist> と同じですが、PC は指定できません。
<reglist+PC>	<reglist> と同じで、PC も指定できます。
<フラグ>	nzcvq (ALU フラグ PSR[31:27]) または g (SIMD GE フラグ PSR[19:16]) を指定します。
\$	表「ARM アーキテクチャのバージョン」を参照して下さい。
+/-	+ または - (+ は省略できます)。
<iflags>	割り込みフラグです。a、i、f (アボート、割り込み、高速割り込み) を 1 つ以上指定します。
<p_mode>	表「プロセッサモード」を参照して下さい。
SPm	<p_mode> で指定されたプロセッサモードの SP です。
<lsb>	ビットフィールドの最下位ビットです。
<width>	ビットフィールドの幅です。<width> + <lsb> <= 32 になる必要があります。
{X}	X を指定した場合、RsX は 16 ビット分ロテートされた Rs です。それ以外の場合、RsX は Rs と同じです。
{!}	! を付けた (プレインデックス) の場合はデータ転送後にベースレジスタを更新します。
{S}	S を指定した場合は条件フラグを更新します。
{T}	T を指定した場合はユーザーモードの特権です。
{R}	R を指定した場合は結果が最も近い値に丸められ、それ以外の場合は切り捨てられます。

演算		\$	アセンブラ	S 指定時に更新するフラグ	アクション	注釈
加算	加算		ADD{S} Rd, Rn, <Operand2>	.N Z C V	Rd := Rn + Operand2	N
	キャリー付き		ADC{S} Rd, Rn, <Operand2>	N Z C V	Rd := Rn + Operand2 + Carry	N
	ワイド	T2	ADD Rd, Rn, #<imm12>		Rd := Rn + imm12, imm12 の範囲は 0 ~ 4095 です。	T, P
	サチュレート { 倍演算 }	5E	Q{D}ADD Rd, Rm, Rn		Rd := SAT(Rm + Rn) doubled: Rd := SAT(Rm + SAT(Rn * 2))	Q
アドレス	フォームの PC 相対アドレス		ADR Rd, <label>		Rd := <label>, 現在の命令からの <label> 範囲については、注 L を参照して下さい。	N, L
減算	減算		SUB{S} Rd, Rn, <Operand2>	.N Z C V	Rd := Rn - Operand2	N
	キャリー付き		SBC{S} Rd, Rn, <Operand2>	N Z C V	Rd := Rn - Operand2 - NOT(Carry)	N
	ワイド	T2	SUB Rd, Rn, #<imm12>		Rd := Rn - imm12, imm12 の範囲は 0 ~ 4095 です。	T, P
	逆減算		RSB{S} Rd, Rn, <Operand2>	N Z C V	Rd := Operand2 - Rn	N
	キャリー付き逆減算		RSC{S} Rd, Rn, <Operand2>	N Z C V	Rd := Operand2 - Rn - NOT(Carry)	A
サチュレート { 倍演算 }	5E	Q{D}SUB Rd, Rm, Rn		Rd := SAT(Rm - Rn) doubled: Rd := SAT(Rm - SAT(Rn * 2))	Q	
スタックを返さない例外		SUBS PC, LR, #<imm8>	N Z C V	PC := LR - imm8, CPSR = SPSR(現在のモード), imm8 の範囲は 0 ~ 255 です。		
並列算術演算	ハーフワード単位の加算	6	<prefix>>ADD16 Rd, Rn, Rm		Rd[31:16] := Rn[31:16] + Rm[31:16], Rd[15:0] := Rn[15:0] + Rm[15:0]	G
	ハーフワード単位の減算	6	<prefix>>SUB16 Rd, Rn, Rm		Rd[31:16] := Rn[31:16] - Rm[31:16], Rd[15:0] := Rn[15:0] - Rm[15:0]	G
	バイト単位の加算	6	<prefix>>ADD8 Rd, Rn, Rm		Rd[31:24] := Rn[31:24] + Rm[31:24], Rd[23:16] := Rn[23:16] + Rm[23:16], Rd[15:8] := Rn[15:8] + Rm[15:8], Rd[7:0] := Rn[7:0] + Rm[7:0]	G
	バイト単位の減算	6	<prefix>>SUB8 Rd, Rn, Rm		Rd[31:24] := Rn[31:24] - Rm[31:24], Rd[23:16] := Rn[23:16] - Rm[23:16], Rd[15:8] := Rn[15:8] - Rm[15:8], Rd[7:0] := Rn[7:0] - Rm[7:0]	G
	ハーフワード単位の交換、加算、減算	6	<prefix>>ASX Rd, Rn, Rm		Rd[31:16] := Rn[31:16] + Rm[15:0], Rd[15:0] := Rn[15:0] - Rm[31:16]	G
	ハーフワード単位の交換、減算、加算	6	<prefix>>SAX Rd, Rn, Rm		Rd[31:16] := Rn[31:16] - Rm[15:0], Rd[15:0] := Rn[15:0] + Rm[31:16]	G
差分の絶対値の符号なし合計	6	USAD8 Rd, Rm, Rs		Rd := Abs(Rm[31:24] - Rs[31:24]) + Abs(Rm[23:16] - Rs[23:16]) + Abs(Rm[15:8] - Rs[15:8]) + Abs(Rm[7:0] - Rs[7:0])		
乗算と累積加算	6	USADA8 Rd, Rm, Rs, Rn		Rd := Rn + Abs(Rm[31:24] - Rs[31:24]) + Abs(Rm[23:16] - Rs[23:16]) + Abs(Rm[15:8] - Rs[15:8]) + Abs(Rm[7:0] - Rs[7:0])		
サチュレート	符号付きサチュレートワード (右シフト)	6	SSAT Rd, #<sat>, Rm{, ASR <sh>}		Rd := SignedSat((Rm ASR sh), sat), <sat> の範囲は 1 ~ 32, <sh> の範囲は 1 ~ 31 です。	Q, R
	符号付きサチュレートワード (左シフト)	6	SSAT Rd, #<sat>, Rm{, LSL <sh>}		Rd := SignedSat((Rm LSL sh), sat), <sat> の範囲は 1 ~ 32, <sh> の範囲は 0 ~ 31 です。	Q
	符号付きサチュレート 2 ハーフワード	6	SSAT16 Rd, #<sat>, Rm		Rd[31:16] := SignedSat(Rm[31:16], sat), Rd[15:0] := SignedSat(Rm[15:0], sat), <sat> の範囲は 1 ~ 16 です。	Q
	符号なしサチュレートワード (右シフト)	6	USAT Rd, #<sat>, Rm{, ASR <sh>}		Rd := UnsignedSat((Rm ASR sh), sat), <sat> の範囲は 0 ~ 31, <sh> の範囲は 1 ~ 31 です。	Q, R
	符号なしサチュレートワード (左シフト)	6	USAT Rd, #<sat>, Rm{, LSL <sh>}		Rd := UnsignedSat((Rm LSL sh), sat), <sat> の範囲は 0 ~ 31, <sh> の範囲は 0 ~ 31 です。	Q
	符号なしサチュレート 2 ハーフワード	6	USAT16 Rd, #<sat>, Rm		Rd[31:16] := UnsignedSat(Rm[31:16], sat), Rd[15:0] := UnsignedSat(Rm[15:0], sat), <sat> の範囲は 0 ~ 15 です。	Q

ARM および Thumb-2 命令セット クイックリファレンスカード

演算		\$	アセンブラ	S 指定時に更新するフラグ	アクション	注釈
乗算	乗算		MUL{S} Rd, Rm, Rs	N Z C*	Rd := (Rm * Rs)[31:0] (Thumb-2 では、Rs が Rd の場合、S を使用できません)	N, S
	乗算と累積加算		MLA{S} Rd, Rm, Rs, Rn	N Z C*	Rd := (Rn + (Rm * Rs))[31:0]	S
	乗算と減算	T2	MLS Rd, Rm, Rs, Rn		Rd := (Rn - (Rm * Rs))[31:0]	
	符号なし long 乗算		UMULL{S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := unsigned(Rm * Rs)	s
	符号なし long 累積加算		UMLAL{S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := unsigned(RdHi, RdLo + Rm * Rs)	s
	符号なし倍精度 long 累積加算	6	UMAAL RdLo, RdHi, Rm, Rs		RdHi, RdLo := unsigned(RdHi + RdLo + Rm * Rs)	
	符号付き long 乗算		SMULL{S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := signed(Rm * Rs)	s
	デュアル符号付き乗算と積の減算と long 累積加算		SMLAL{S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := signed(RdHi, RdLo + Rm * Rs)	s
	16 * 16 ビット	5E	SMULxy Rd, Rm, Rs		Rd := Rm[x] * Rs[y]	
	32 * 16 ビット	5E	SMULWy Rd, Rm, Rs		Rd := (Rm * Rs[y])[47:16]	
	16 * 16 ビットと累積加算	5E	SMLAxy Rd, Rm, Rs, Rn		Rd := Rn + Rm[x] * Rs[y]	Q
	32 * 16 ビットと累積加算	5E	SMLAWy Rd, Rm, Rs, Rn		Rd := Rn + (Rm * Rs[y])[47:16]	Q
	16 * 16 ビットと long 累積加算	5E	SMLALxy RdLo, RdHi, Rm, Rs		RdHi, RdLo := RdHi, RdLo + Rm[x] * Rs[y]	
	デュアル符号付き乗算と積の加算	6	SMUAD{X} Rd, Rm, Rs		Rd := Rm[15:0] * RsX[15:0] + Rm[31:16] * RsX[31:16]	Q
	乗算と累積加算	6	SMLAD{X} Rd, Rm, Rs, Rn		Rd := Rn + Rm[15:0] * RsX[15:0] + Rm[31:16] * RsX[31:16]	Q
	デュアル符号付き乗算と積の減算と long 累積加算	6	SMLALD{X} RdLo, RdHi, Rm, Rs		RdHi, RdLo := RdHi, RdLo + Rm[15:0] * RsX[15:0] + Rm[31:16] * RsX[31:16]	
	デュアル符号付き乗算と積の減算	6	SMUSD{X} Rd, Rm, Rs		Rd := Rm[15:0] * RsX[15:0] - Rm[31:16] * RsX[31:16]	Q
	乗算と累積加算	6	SMLS{X} Rd, Rm, Rs, Rn		Rd := Rn + Rm[15:0] * RsX[15:0] - Rm[31:16] * RsX[31:16]	Q
	デュアル符号付き乗算と積の減算と long 累積加算	6	SMLS{X} RdLo, RdHi, Rm, Rs		RdHi, RdLo := RdHi, RdLo + Rm[15:0] * RsX[15:0] - Rm[31:16] * RsX[31:16]	
	符号付き上位ワード乗算	6	SMMUL{R} Rd, Rm, Rs		Rd := (Rm * Rs)[63:32]	
乗算と累積加算	6	SMMLA{R} Rd, Rm, Rs, Rn		Rd := Rn + (Rm * Rs)[63:32]		
乗算と減算	6	SMMLS{R} Rd, Rm, Rs, Rn		Rd := Rn - (Rm * Rs)[63:32]		
内部 40 ビット累積加算	XS	MIA Rd, Rm, Rs		Ac := Ac + Rm * Rs		
バックハーフワード	XS	MIAPH Ac, Rm, Rs		Ac := Ac + Rm[15:0] * Rs[15:0] + Rm[31:16] * Rs[31:16]		
ハーフワード	XS	MIAXy Ac, Rm, Rs		Ac := Ac + Rm[x] * Rs[y]		
除算	符号付きまたは符号なし	RM	<op> Rd, Rn, Rm		Rd := Rn / Rm <op> は SDIV (符号付き) または UDIV (符号なし) です。 シフト命令も参照して下さい。	T
データの移動	データ		MOV{S} Rd, <Operand2>	N Z C	Rd := Operand2	N
	ビット反転		MVN{S} Rd, <Operand2>	N Z C	Rd := 0xFFFFFFFF EOR Operand2	N
	上位ハーフワード	T2	MOVT Rd, #<imm16>		Rd[31:16] := imm16, Rd[15:0] には影響しません。imm16 の範囲は 0 ~ 65535 です。	
	ワイド	T2	MOV Rd, #<imm16>		Rd[15:0] := imm16, Rd[31:16] = 0, imm16 の範囲は 0 ~ 65535 です。	
	40 ビット累算器からレジスタへ レジスタから 40 ビット累算器へ	XS XS	MRA RdLo, RdHi, Ac MAR Ac, RdLo, RdHi		RdLo := Ac[31:0], RdHi := Ac[39:32] Ac[31:0] := RdLo, Ac[39:32] := RdHi[7:0]	
シフト	算術右シフト		ASR{S} Rd, Rm, <Rs sh>	N Z C	Rd := ASR(Rm, Rs sh) MOV{S} Rd, Rm, ASR <Rs sh> と同じ	N
	論理左シフト		LSL{S} Rd, Rm, <Rs sh>	N Z C	Rd := LSL(Rm, Rs sh) MOV{S} Rd, Rm, LSL <Rs sh> と同じ	N
	論理右シフト		LSR{S} Rd, Rm, <Rs sh>	N Z C	Rd := LSR(Rm, Rs sh) MOV{S} Rd, Rm, LSR <Rs sh> と同じ	N
	右ローテート		ROR{S} Rd, Rm, <Rs sh>	N Z C	Rd := ROR(Rm, Rs sh) MOV{S} Rd, Rm, ROR <Rs sh> と同じ	N
	拡張付き右ローテート		RRX{S} Rd, Rm	N Z C	Rd := RRX(Rm) MOV{S} Rd, Rm, RRX と同じ	N
	先行ゼロカウント	5	CLZ Rd, Rm		Rd := Rm に含まれる先行ゼロの数	
比較	比較		CMP Rn, <Operand2>	N Z C V	Rn - Operand2 を実行して CPSR フラグを更新します。	N
	否定		CMN Rn, <Operand2>	N Z C V	Rn + Operand2 を実行して CPSR フラグを更新します。	N
論理演算	テスト		TST Rn, <Operand2>	N Z C	Rn AND Operand2 を実行して CPSR フラグを更新します。	N
	等価テスト		TEQ Rn, <Operand2>	N Z C	Rn EOR Operand2 を実行して CPSR フラグを更新します。	N
	AND		AND{S} Rd, Rn, <Operand2>	N Z C	Rd := Rn AND Operand2	N
	EOR		EOR{S} Rd, Rn, <Operand2>	N Z C	Rd := Rn EOR Operand2	N
	ORR		ORR{S} Rd, Rn, <Operand2>	N Z C	Rd := Rn OR Operand2	N
	ORN	T2	ORN{S} Rd, Rn, <Operand2>	N Z C	Rd := Rn OR NOT Operand2	T
	ビットクリア		BIC{S} Rd, Rn, <Operand2>	N Z C	Rd := Rn AND NOT Operand2	N

ARM および Thumb-2 命令セット クイックリファレンスカード

演算	\$	アセンブラ	アクション	注釈
ビットフィールド	T2	BFC Rd, #<lsb>, #<width>	Rd[(width+lsb-1):lsb] := 0, Rd の他のビットには影響しません。	
	T2	BFI Rd, Rn, #<lsb>, #<width>	Rd[(width+lsb-1):lsb] := Rn[(width-1):0], Rd の他のビットには影響しません。	
	T2	SBFX Rd, Rn, #<lsb>, #<width>	Rd[(width-1):0] := Rn[(width+lsb-1):lsb], Rd[31:width] := Replicate(Rn[width+lsb-1])	
パック	T2	UBFX Rd, Rn, #<lsb>, #<width>	Rd[(width-1):0] := Rn[(width+lsb-1):lsb], Rd[31:width] := Replicate(0)	
	6	PKHBT Rd, Rn, Rm{, LSL #<sh>}	Rd[15:0] := Rn[15:0], Rd[31:16] := (Rm LSL sh)[31:16]. sh の範囲は 0 ~ 31 です。	
ハーフワードのバック (下位ハーフワード + 上位ハーフワード)	6	PKHTB Rd, Rn, Rm{, ASR #<sh>}	Rd[31:16] := Rn[31:16], Rd[15:0] := (Rm ASR sh)[15:0]. sh の範囲は 1 ~ 32 です。	
	6	SXTB Rd, Rm{, ROR #<sh>}	Rd[31:0] := SignExtend((Rm ROR (8 * sh))[15:0]). sh の範囲は 0 ~ 3 です。	N
符号付き拡張	6	SXTB16 Rd, Rm{, ROR #<sh>}	Rd[31:16] := SignExtend((Rm ROR (8 * sh))[23:16]), Rd[15:0] := SignExtend((Rm ROR (8 * sh))[7:0]). sh の範囲は 0 ~ 3 です。	N
	6	SXTB Rd, Rm{, ROR #<sh>}	Rd[31:0] := SignExtend((Rm ROR (8 * sh))[7:0]). sh の範囲は 0 ~ 3 です。	N
符号なし拡張	6	UXTB Rd, Rm{, ROR #<sh>}	Rd[31:0] := ZeroExtend((Rm ROR (8 * sh))[15:0]). sh の範囲は 0 ~ 3 です。	N
	6	UXTB16 Rd, Rm{, ROR #<sh>}	Rd[31:16] := ZeroExtend((Rm ROR (8 * sh))[23:16]), Rd[15:0] := ZeroExtend((Rm ROR (8 * sh))[7:0]). sh の範囲は 0 ~ 3 です。	N
加算を伴う符号付き拡張	6	SXTAH Rd, Rn, Rm{, ROR #<sh>}	Rd[31:0] := Rn[31:0] + SignExtend((Rm ROR (8 * sh))[15:0]). sh の範囲は 0 ~ 3 です。	
	6	SXTAB16 Rd, Rn, Rm{, ROR #<sh>}	Rd[31:16] := Rn[31:16] + SignExtend((Rm ROR (8 * sh))[23:16]), Rd[15:0] := Rn[15:0] + SignExtend((Rm ROR (8 * sh))[7:0]). sh の範囲は 0 ~ 3 です。	
加算を伴う符号なし拡張	6	SXTAB Rd, Rn, Rm{, ROR #<sh>}	Rd[31:0] := Rn[31:0] + SignExtend((Rm ROR (8 * sh))[7:0]). sh の範囲は 0 ~ 3 です。	
	6	UXTAH Rd, Rn, Rm{, ROR #<sh>}	Rd[31:0] := Rn[31:0] + ZeroExtend((Rm ROR (8 * sh))[15:0]). sh の範囲は 0 ~ 3 です。	
加算を伴う符号なし拡張	6	UXTAB16 Rd, Rn, Rm{, ROR #<sh>}	Rd[31:16] := Rn[31:16] + ZeroExtend((Rm ROR (8 * sh))[23:16]), Rd[15:0] := Rn[15:0] + ZeroExtend((Rm ROR (8 * sh))[7:0]). sh の範囲は 0 ~ 3 です。	
	6	UXTAB Rd, Rn, Rm{, ROR #<sh>}	Rd[31:0] := Rn[31:0] + ZeroExtend((Rm ROR (8 * sh))[7:0]). sh の範囲は 0 ~ 3 です。	
反転	T2	RBIT Rd, Rm	For (i = 0; i < 32; i++) : Rd[i] = Rm[31-i]	
	6	REV Rd, Rm	Rd[31:24] := Rm[7:0], Rd[23:16] := Rm[15:8], Rd[15:8] := Rm[23:16], Rd[7:0] := Rm[31:24]	N
	6	REV16 Rd, Rm	Rd[15:8] := Rm[7:0], Rd[7:0] := Rm[15:8], Rd[31:24] := Rm[23:16], Rd[23:16] := Rm[31:24]	N
	6	REVSH Rd, Rm	Rd[15:8] := Rm[7:0], Rd[7:0] := Rm[15:8], Rd[31:16] := Rm[7] * ÿ	N
選択	6	SEL Rd, Rn, Rm	GE[0] = 1 の場合は Rd[7:0] := Rn[7:0], それ以外の場合は Rd[7:0] := Rm[7:0]. GE[1], GE[2], GE[3] でも同様にビット [15:8], [23:16], [31:24] が選択されます。	
If-Then	T2	IT{pattern} {cond}	pattern に応じて、後続の命令を最大 4 つまで条件付き命令にします。pattern は最大 3 文字で構成される文字列です。各文字は T (Then) または E (Else) です。IT の後の最初の命令には条件 cond が付きます。その後の命令には、パターンに対応する文字が T の場合は条件 cond が付き、対応する文字が E の場合は cond の逆が付きます。使用できる条件コードについては、表「条件フィールド」を参照して下さい。	T, U
分岐	分岐	B <label>	PC := label, label はこの命令のアドレスから 7 32MB (T2: 7 16MB, T: -252 ~ +256B) 以内です。	N, B
	リンク付き	BL <label>	LR := 次の命令のアドレス、PC := label, label はこの命令のアドレスから 7 32MB (T2: 7 16MB) 以内です。	
	分岐と切り替え	BX Rm	PC := Rm, Rm[0] が 1 の場合は Thumb、Rm[0] が 0 の場合は ARM に切り替えます。	N
	リンク付きと切り替え (1)	BLX <label>	LR := 次の命令のアドレス、PC := label, 命令セットを変更します。label はこの命令のアドレスから 7 32MB (T2: 7 16MB) 以内です。	C
	リンク付きと切り替え (2)	BLX Rm	LR := 次の命令のアドレス、PC := Rm[31:1]. Rm[0] が 1 の場合は Thumb に、Rm[0] が 0 の場合は ARM に変更します。	N
	分岐と Jazelle 状態への変更	BXJ Rm	可能な場合は Jazelle 状態に変更します。	
比較し、0 の (または 0 でない) 場合に分岐	T2	CB{N}Z Rn, <label>	Rn (== 0 または Rn != 0) の場合 PC := label, label はこの命令のアドレスから +4 ~ 130 です。	N, T, U
テーブル分岐バイト	T2	TBB [Rn, Rm]	PC = PC + ZeroExtend(Memory(Rn + Rm, 1) << 1). 分岐の範囲は 4 ~ 512 です。Rn には PC を指定できます。	T, U
テーブル分岐ハーフワード	T2	TBH [Rn, Rm, LSL #1]	PC = PC + ZeroExtend(Memory(Rn + Rm << 1, 2) << 1). 分岐の範囲は 4 ~ 131072 です。Rn には PC を指定できます。	T, U
PSR との移動	PSR からレジスタへ	MRS Rd, <PSR>	Rd := PSR	
	レジスタフラグから APSR フラグへ	MSR APSR_<flags>, Rm	APSR_<flags> := Rm	
	イミディエートフラグから APSR フラグへ	MSR APSR_<flags>, #<imm8m>	APSR_<flags> := imm8_r	
	レジスタから PSR へ	MSR <PSR>_<fields>, Rm	PSR := Rm (選択したバイトのみ)	
イミディエート値を PSR へ	MSR <PSR>_<fields>, #<imm8m>	PSR := imm8_r (選択したバイトのみ)		
プロセッサ状態の変更	プロセッサ状態の変更	6 CPSID <iflags> {, #<p_mode>}	指定された割り込みをディセーブルにします。オプションでモードを変更できます。	U, N
		6 CPSIE <iflags> {, #<p_mode>}	指定された割り込みを有効にします。オプションでモードを変更できます。	U, N
	プロセッサモードの変更	6 CPS #<p_mode>		U
	エンディアン方式の設定	6 SETEND <endianness>	ロードおよびストア用にエンディアン方式を設定します。<endianness> は BE (ビッグエンディアン) または LE (リトルエンディアン) のいずれかを指定できます。	U, N

ARM および Thumb-2 命令セット

クイックリファレンスカード

単一データ項目のロードとストア	\$	アセンブラ	<op> が LDR の場合のアクション	<op> が STR の場合のアクション	注釈
ワード、バイト、またはハーフワードのロードまたはストア	イミディエートオフセット	<op>{size}{T} Rd, [Rn {, #<offset>}]{!}	Rd := [address, size]	[address, size] := Rd	1、N
	ポストインデックス、イミディエート値	<op>{size}{T} Rd, [Rn], #<offset>	Rd := [address, size]	[address, size] := Rd	2
	レジスタオフセット	<op>{size} Rd, [Rn, +/-Rm {, <opsh>}]{!}	Rd := [address, size]	[address, size] := Rd	3、N
	ポストインデックス、レジスタ PC 相対	<op>{size}{T} Rd, [Rn], +/-Rm {, <opsh>} <op>{size} Rd, <label>	Rd := [address, size] Rd := [label, size]	[address, size] := Rd 該当なし	4 5、N
ダブルワードのロードまたはストア	イミディエートオフセット	<op>D Rd1, Rd2, [Rn {, #<offset>}]{!}	Rd1 := [address], Rd2 := [address + 4]	[address] := Rd1, [address + 4] := Rd2	6、9
	ポストインデックス、イミディエート値	<op>D Rd1, Rd2, [Rn], #<offset>	Rd1 := [address], Rd2 := [address + 4]	[address] := Rd1, [address + 4] := Rd2	6、9
	レジスタオフセット	<op>D Rd1, Rd2, [Rn, +/-Rm {, <opsh>}]{!}	Rd1 := [address], Rd2 := [address + 4]	[address] := Rd1, [address + 4] := Rd2	7、9
	ポストインデックス、レジスタ PC 相対	<op>D Rd1, Rd2, [Rn], +/-Rm {, <opsh>} <op>D Rd1, Rd2, <label>	Rd1 := [address], Rd2 := [address + 4] Rd1 := [label], Rd2 := [label + 4]	[address] := Rd1, [address + 4] := Rd2 該当なし	7、9 8、9

データまたは命令のプリロード	\$ (PLD)	\$ (PLI)	\$ (PLDW)	アセンブラ	<op> が PLD の場合のアクション	<op> が PLI の場合のアクション	<op> が PLDW の場合のアクション	注釈
イミディエートオフセット	5E	7	7MP	<op> [Rn {, #<offset>}]	[address, 32] (データ) のプリロード	[address, 32] (命令) のプリロード	[address, 32] (データ) を書き込むためのプリロード	1、C
レジスタオフセット	5E	7	7MP	<op> [Rn, +/-Rm {, <opsh>}]	[address, 32] (データ) のプリロード	[address, 32] (命令) のプリロード	[address, 32] (データ) を書き込むためのプリロード	3、C
PC 相対	5E	7		<op> <label>	[label, 32] (データ) のプリロード	[label, 32] (命令) のプリロード		5、C

その他のメモリ操作	\$	アセンブラ	アクション	注釈
多重ロード	データブロックのロード	LDM{IA IB DA DB} Rn{!}, <reglist>-PC	[Rn] からレジスタリストをロード	N、I
	復帰 (と切り替え)	LDM{IA IB DA DB} Rn{!}, <reglist>+PC	レジスタのロード。PC := [address][31:1] (§ 5T: [address][0] が 1 の場合は Thumb に切り替え)	I
	CPSR の復元	LDM{IA IB DA DB} Rn{!}, <reglist>+PC>A	レジスタのロード、分岐 (§ 5T: および切り替え)、CPSR := SPSR、例外モードのみ。	I
	ユーザモードのレジスタ	LDM{IA IB DA DB} Rn, <reglist>-PC>A	[Rn] からユーザモードレジスタリストをロード。特権モードのみ。	I
ポップ		POP <reglist>	LDM SP!, <reglist> の標準構造形式	N
排他的ロード	セマフォ操作	LDREX Rd, [Rn]	Rd := [Rn], アドレスに排他アクセスを示すタグを付けます。共有アドレスでない場合は、未解決のタグが設定されます。 Rd, Rn には PC を指定できません。	
	ハーフワードまたはバイト	LDREX{H B} Rd, [Rn]	Rd[15:0] := [Rn] または Rd[7:0] := [Rn], アドレスに排他アクセスを示すタグを付けます。共有アドレスでない場合は、未解決のタグが設定されます。Rd, Rn には PC を指定できません。	9
	ダブルワード	LDREXD Rd1, Rd2, [Rn]	Rd1 := [Rn], Rd2 := [Rn+4], アドレスに排他アクセスを示すタグを付けます。共有アドレスでない場合は、未解決のタグが設定されます。Rd1, Rd2, Rn には PC を指定できません。	
多重ストア	ブッシュ、またはデータブロックのストア	STM{IA IB DA DB} Rn{!}, <reglist>	レジスタリストを [Rn] にストア	N、I
	ユーザモードのレジスタ	STM{IA IB DA DB} Rn{!}, <reglist>A	ユーザモードレジスタリストを [Rn] にストア。特権モードのみ。	I
ブッシュ		PUSH <reglist>	STMDB SP!, <reglist> の標準構造形式	N
排他的ストア	セマフォ操作	STREX Rd, Rm, [Rn]	許可される場合は、[Rn] := Rm、排他アクセスタグのクリア、Rd := 0、それ以外の場合は、Rd := 1、Rd、Rm、Rn には PC を指定できません。	
	ハーフワードまたはバイト	STREX{H B} Rd, Rm, [Rn]	許可される場合は、[Rn] := Rm[15:0] または [Rn] := Rm[7:0]、排他アクセスタグのクリア、Rd := 0、それ以外の場合は、Rd := 1、Rd、Rm、Rn には PC を指定できません。	10
	ダブルワード	STREXD Rd, Rm1, Rm2, [Rn]	許可される場合は、[Rn] := Rm1, [Rn+4] := Rm2、排他アクセスタグのクリア、Rd := 0、それ以外の場合は、Rd := 1、Rd、Rm1、Rm2、Rn には PC を指定できません。	
排他的クリア		CLREX	ローカルプロセッサの排他アクセスタグのクリア	C

注: ロード、ストア、およびプリロード操作オプションの使用可否と範囲

注	ARM のワード、B、D	ARM の SB、H、SH	ARM の T、BT	Thumb-2 のワード、B、SB、H、SH、D	Thumb-2 の T、BT、SBT、HT、SHT
1	オフセット: -4095 ~ +4095	オフセット: -255 ~ +255	該当なし	オフセット: ライトバックの場合は -255 ~ +255、それ以外の場合は -255 ~ +4095	オフセット: 0 ~ +255、ライトバックは許可されません
2	オフセット: -4095 ~ +4095	オフセット: -255 ~ +255	オフセット: -4095 ~ +4095	オフセット: -255 ~ +255	該当なし
3	{, <opsh>} の全範囲	{, <opsh>} は使用できません	該当なし	<opsh> は LSL #<sh> に制限されます (<sh> の範囲は 0 ~ 3)	該当なし
4	{, <opsh>} の全範囲	{, <opsh>} は使用できません	該当なし	{, <opsh>} の全範囲	該当なし
5	現在の命令から +/- 4092 の範囲内の label	該当なし	該当なし	現在の命令から +/- 4092 の範囲内の label	該当なし
6	オフセット: -255 ~ +255	-	-	オフセット: -1020 ~ +1020、4 の倍数で指定して下さい	-
7	{, <opsh>} は使用できません	-	-	該当なし	-
8	現在の命令から +/- 252 の範囲内の label	-	-	該当なし	-
9	Rd1 は偶数番号のレジスタでかつ r14 は指定できません、Rd2 == Rd1 + 1	-	-	Rd1 != PC、Rd2 != PC	-
10	Rm1 は偶数番号のレジスタでかつ r14 は指定できません、Rm2 == Rm1 + 1	-	-	Rm1 != PC、Rm2 != PC	-

ARM および Thumb-2 命令セット

クイックリファレンスカード

コプロセッサ命令	§	アセンブラ	アクション	注釈
データ操作		CDP{2} <copr>, <op1>, CRd, CRn, CRm{, <op2>}		C2
コプロセッサから ARM レジスタへの移動		MRC{2} <copr>, <op1>, Rd, CRn, CRm{, <op2>}		C2
2つの ARM レジスタの移動	5E	MRRRC <copr>, <op1>, Rd, Rn, CRm		C2
2つの ARM レジスタの移動 (代替方法)	6	MRRC2 <copr>, <op1>, Rd, Rn, CRm		C
ARM レジスタからコプロセッサへの移動		MCR{2} <copr>, <op1>, Rd, CRn, CRm{, <op2>}		C2
2つの ARM レジスタの移動	5E	MCRR <copr>, <op1>, Rd, Rn, CRm		C2
2つの ARM レジスタの移動 (代替方法)	6	MCRR2 <copr>, <op1>, Rd, Rn, CRm		C
ロードとストア、ブレインデクス		<op>{2} <copr>, CRd, [Rn, #+/-<offset8*4>]{!}	op : LDC または STC. オフセット : 0 ~ 1020 の 4 の倍数。	C2
ロードとストア、ゼロオフセット		<op>{2} <copr>, CRd, [Rn] {, 8-bit copro. option}	op : LDC または STC.	C2
ロードとストア、ポストインデクス		<op>{2} <copr>, CRd, [Rn], #+/-<offset8*4>	op : LDC または STC. オフセット : 0 ~ 1020 の 4 の倍数。	C2

その他の命令	§	アセンブラ	アクション	注釈
ワードのスワップ		SWP Rd, Rm, [Rn]	temp := [Rn], [Rn] := Rm, Rd := temp.	A, D
バイトのスワップ		SWPB Rd, Rm, [Rn]	temp := ZeroExtend([Rn][7:0]), [Rn][7:0] := Rm[7:0], Rd := temp	A, D
復帰状態のストア	6	SRS{IA IB DA DB} SP{!}, #<p_mode>	[SPm] := LR, [SPm + 4] := CPSR	C, I
例外からの復帰	6	RFE{IA IB DA DB} Rn{!}	PC := [Rn], CPSR := [Rn + 4]	C, I
ブレークポイント	5	BKPT <imm16>	プリフェッチアバートまたはデバッグ状態への移行。16 ビットのビットフィールドが命令にエンコードされます。	C, N
セキュアモニターコール	Z	SMC <imm4>	セキュアモニターコール例外。4 ビットのビットフィールドが命令にエンコードされます。以前の SMI です。	
スーパーバイザコール		SVC <imm24>	スーパーバイザコール例外。24 ビットのビットフィールドが命令にエンコードされます。以前の SWI です。	N
操作なし	6K	NOP	なし。時間も消費しない可能性があります。	N, V
ヒント	7	DBG	デバッグシステムおよび関連するシステムにヒントを提供します。	
データメモリバリア	7	DMB	メモリアクセスの観察の順序が保証されます。	C
データ同期バリア	7	DSB	メモリアクセスの完了が保証されます。	C
命令同期バリア	7	ISB	プロセッサパイプラインと分岐予測ロジックをフラッシュします。	C
イベントの設定	6K	SEV	マルチプロセッサシステムのイベントを信号で送信します。未実装の場合は NOP になります。	N
イベント待機	6K	WFE	イベント、IRQ、FIQ、不正確なアバート、デバッグエントリ要求を待機します。未実装の場合は NOP になります。	N
割り込み待機	6K	WFI	IRQ、FIQ、不正確なアバート、デバッグエントリ要求を待機します。未実装の場合は NOP になります。	N
明け渡し	6K	YIELD	他のスレッドに制御を明け渡します。未実装の場合は NOP になります。	N

注釈				
A	Thumb 状態では使用できません。	P	Thumb 状態では Rn に PC を指定できます。	
B	Thumb 状態では、IT ブロックに含めなくても、条件付きにできます。	Q	サチュレーション (加算または減算) やオーバーフロー (乗算) が発生すると、Q フラグを設定します。MRS と MSR を使用して Q フラグの読み出しとリセットを行います。	
C	ARM 状態では、条件コードを使用できません。	R	ARM 命令では <sh> の範囲は 1 ~ 32 です。	
C2	オプション 2 は ARMv5 から使用できます。これにより代替方法の操作が得られます。ARM 状態では、この代替方法に条件コードを使用できません。	S	S 修飾子は Thumb-2 命令では使用できません。	
D	廃止される予定です。代わりに、LDREX および STREX を使用して下さい。	T	ARM 状態では使用できません。	
G	個別の演算結果をベースに、CPSR の 4 つの GE フラグを更新します。	U	IT ブロックでは使用できません。ARM または Thumb 状態では、条件コードを使用できません。	
I	IA がデフォルトで、通常は省略します。	V	NOP 命令を使用できない場合は、アセンブラによって適切な命令が挿入されます。	
L	ARM: <imm8m>. 16 ビットの Thumb : 0 ~ 1020 の 4 の倍数。32 ビットの Thumb : 0 ~ 4095。			
N	この命令の一部またはすべての形式は、Thumb-2 コードの 16 ビット (ナロー) 命令です。詳細については、『Thumb 16 ビット命令セット クイックリファレンスカード』を参照して下さい。			

ARM および Thumb-2 命令セット クイックリファレンスカード

ARM アーキテクチャのバージョン	
<i>n</i>	ARM アーキテクチャのバージョン <i>n</i> 以上
<i>nT</i> , <i>nJ</i>	ARM アーキテクチャのバージョン <i>n</i> 以上の T または J バリエーション
5E	ARM v5E および 6 以上
T2	ARM v6 以上の Thumb-2 のすべてのバージョン
6K	ARM 命令は ARMv6K 以上、Thumb は ARMv7
7MP	マルチプロセス拡張を実装した ARMv7 アーキテクチャ
Z	ARM v6 以上のセキュリティ拡張機能のすべてのバージョン
RM	ARMv7-R と ARMv7-M のみ
XS	XScale コプロセッサ命令

フレキシブル第 2 オペランド	
イミディエート値	#<imm8m>
レジスタと定数によるシフトのオプション (以下を参照して下さい)	Rm {, <opsh>}
レジスタ、レジスタによる論理左シフト	Rm, LSL Rs
レジスタ、レジスタによる論理右シフト	Rm, LSR Rs
レジスタ、レジスタによる算術右シフト	Rm, ASR Rs
レジスタ、レジスタによる右ロテート	Rm, ROR Rs

レジスタと定数によるシフトのオプション		
(シフトなし)	Rm	Rm, LSL #0 と同じ
論理左シフト	Rm, LSL #<shift>	0 ~ 31 のシフトが可能です。
論理右シフト	Rm, LSR #<shift>	1 ~ 32 のシフトが可能です。
算術右シフト	Rm, ASR #<shift>	1 ~ 32 のシフトが可能です。
右ロテート	Rm, ROR #<shift>	1 ~ 31 のシフトが可能です。
拡張付き右ロテート	Rm, RRX	

PSR フィールド (少なくとも 1 つの接尾文字を使用)		
接尾文字	意味	
c	制御フィールドマスクバイト	PSR[7:0]
f	フラグフィールドマスクバイト	PSR[31:24]
s	状態フィールドマスクバイト	PSR[23:16]
x	拡張フィールドマスクバイト	PSR[15:8]

著作権

® または ™ のマークが付いた言葉およびロゴは、この著作権情報で別段に規定されている場合を除き、ARM Limited の EU およびその他の国における登録商標または商標です。本書に記載されている他の製品名は、各社の所有する商標です。本書に記載されている情報の全部または一部、ならびに本書で紹介する製品は、著作権所有者の文書による事前の許可を得ない限り、転用・複製することを禁じます。本書に記載されている製品は、今後も継続的に開発・改良の対象となります。本書に含まれる製品およびその利用方法についての情報は、ARM が利用者の利益のために提供するものです。したがって当社では、製品の市販性または利用の適切性を含め、暗示的・明示的に関係なく一切の責任を負いません。本リファレンスカードは、本製品の利用者をサポートすることだけを目的としています。本リファレンスカードに記載されている情報の使用、情報の誤りまたは省略、あるいは本製品の誤使用によって発生したいかなる損失・損傷についても、ARM Limited は一切責任を負いません。

条件フィールド		
ニーモニック	説明	説明 (VFP)
EQ	等しい	等しい
NE	等しくない	等しくない、または順番がない
CS / HS	キャリー設定 / 大きいか等しい (符号なし)	大きいか等しい、または順番がない
CC / LO	キャリークリア / 小さい (符号なし)	小さい
MI	否定	小さい
PL	正または 0	大きいか等しい、または順番がない
VS	オーバフロー	順番がない (NaN オペランドが含まれている)
VC	オーバフローなし	順番がある
HI	大きい (符号なし)	大きい、または順番がない
LS	小さいか等しい (符号なし)	小さいか等しい
GE	大きいか等しい (符号付き)	大きいか等しい
LT	小さい (符号付き)	小さい、または順番がない
GT	大きい (符号付き)	大きい
LE	小さいか等しい (符号付き)	小さいか等しい、または順番がない
AL	常時 (通常省略)	常時 (通常省略)

すべての ARM 命令 (注 C または注 U が指定されている命令を除く) には、命令ニーモニックの後 (つまり、このカードで示されているように、命令の最初のスペースの前) に上記の条件コードの 1 つを含めることができます。この条件は命令にエンコードされます。すべての Thumb-2 命令 (注 U が指定された命令を除く) には、命令ニーモニックの後に上記の条件コードの 1 つを含めることができます。この条件は、先行する IT 命令にエンコードされます (条件分岐命令の場合を除く)。命令内の条件コードは先行する IT 命令の条件コードと一致する必要があります。Thumb-2 がないプロセッサでは、条件コードを付けることができる Thumb 命令は B <label> のみです。

プロセッサモード	
16	ユーザ
17	FIQ 高速割り込み
18	IRQ 割り込み
19	スーパーバイザ
23	7ホート
27	未定義
31	システム

並列命令の接頭文字	
S	符号付き算術演算モジュロ 2 ⁸ または 2 ¹⁶ を示し、CPSR GE ビットをセットします。
Q	符号付きサチュレート算術演算を示します。
SH	符号付き演算を行い、結果を半分にします。
U	符号なし算術演算モジュロ 2 ⁸ または 2 ¹⁶ を示し、CPSR GE ビットをセットします。
UQ	符号なしサチュレート算術演算を示します。
UH	符号なし算術演算を行い、結果を半分にします。

文書番号

ARM QRC 0001M

変更履歴

発行	日付	変更	発行	日付	変更
A	1995 年 6 月	第 1 版	B	1996 年 9 月	第 2 版
C	1998 年 11 月	第 3 版	D	1999 年 10 月	第 4 版
E	2000 年 10 月	第 5 版	F	2001 年 9 月	第 6 版
G	2003 年 1 月	第 7 版	H	2003 年 10 月	第 8 版
I	2004 年 12 月	第 9 版	J	2005 年 5 月	RVCT 2.2 SP1 リリース
K	2006 年 3 月	RVCT 3.0 リリース	L	2007 年 3 月	RVCT 3.1 リリース
M	2008 年 9 月	RVCT 4.0 リリース			