付録 1　Thumb-2 命令

●32 ビット命令（1/4）

機能	動作	ニーモニック
シフト・ローテート命令	レジスタの数値により算術右シフト	ASR{S}.W <Rd>, <Rn>, <Rm>
	レジスタの数値によりレジスタ値を論理左シフト	LSL{S}.W <Rd>, <Rn>, <Rm>
	レジスタの数値によりレジスタ値を論理右シフト	LSR{S}.W <Rd>, <Rn>, <Rm>
	レジスタ内の数により、右ローテート	ROR{S}.W <Rd>, <Rn>, <Rm>
	拡張付き右ローテート	RRX{S}.W <Rd>, <Rm>
バリア命令	データメモリバリア	DMB <c>
	データ同期化バリア	DSB <c>
	命令同期化バリア	ISB <c>
ビット操作命令	ビットフィールドをクリア	BFC.W <Rd>, #<lsb>, #<width>
	ビットフィールドを1つのレジスタの値から別のレジスタの値に挿入	BFI.W <Rd>, <Rn>, #<lsb> #<width>
	レジスタの値と12ビットイミディエイト値の否定（1の補数）とをビット単位の論理積	BIC{S}.W <Rd>, <Rn> #<modify_constant(immed_12)>
	レジスタの値とシフトされたレジスタの値の否定（1の補数）とをビット単位の論理積	BIC{S}.W <Rd>, <Rn> <Rm>{, <shift>}
飽和命令	符号付き飽和	SSAT.W <c> <Rd> #<imm>, <Rn>{, <shift>}
	符号なし飽和	USAT <c> <Rd>, #<imm> <Rn>{, <shift>}
レジスタ転送命令	12ビットイミディエイト値をレジスタに移動	MOV{S}.W <Rd> #<modify_constant(immed_12)>
	16ビットイミディエイト値をレジスタの上位ハーフワード[31:16]に移動	MOVT.W <Rd>, #<immed_16>
	16ビットイミディエイト値をレジスタの下位ハーフワード[15:0]に移動し、上位ハーフワード[31:16]をクリア	MOVW.W <Rd>, #<immed_16>
	ステータスレジスタからレジスタに移動	MRS<c> <Rd>, <psr>
	ステータスレジスタに移動	MSR<c> <psr>_<fields>,<Rn>
	シフトしたレジスタの値をレジスタに移動	MOV{S}.W <Rd>, <Rm>{, <shift>}
四則演算命令	レジスタの値、12ビットイミディエイト値、Cビットを加算	ADC{S}.W <Rd>, <Rn> #<modify_constant(immed_12>
	レジスタの値、シフトしたレジスタの値、Cビットを加算	ADC{S}.W <Rd>, <Rn> <Rm>{, <shift>}
	レジスタの値と12ビットイミディエイト値を加算	ADD{S}.W <Rd>, <Rn> #<modify_constant(immed_12)>
	レジスタの値とシフトしたレジスタの値を加算	ADD{S}.W <Rd>, <Rm>{, <shift>}
	レジスタの値と12ビットイミディエイト値を加算	ADDW.W <Rd>, <Rn> #<immed_12>
	2つの符号付きまたは符号なしレジスタ値を乗算し、下位32ビットをレジスタの値に加算（積和）	MLA.W<Rd>, <Rn>, <Rm>, <Racc>
	2つの符号付きまたは符号なしレジスタ値を乗算し、下位32ビットをレジスタの値から減算	MLS.W<Rd>, <Rn>, <Rm>, <Racc>
	2つの符号付きまたは符号なしレジスタ値を乗算	MUL.W <Rd>, <Rn>, <Rm>
	レジスタの値を12ビットイミディエイト値から減算	RSB{S}.W <Rd>, <Rn> #<modify_constant(immed_12)>
	レジスタの値をシフトしたレジスタの値から減算	RSB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
	12ビットイミディエイト値とCビットをレジスタの値から減算	SBC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)
	シフトしたレジスタの値とCビットをレジスタの値から減算	SBC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
	符号付き除算	SDIV<c> <Rd>,<Rn>,<Rm>
	符号付きワードを乗算して、符号拡張された値を、一対のレジスタ値に累算	SMLAL.W<RdLo>,<RdHi>,<Rn>,<Rm>
	2つの符号付きレジスタ値を乗算	SMULL.W<RdLo>,<RdHi>,<Rn>,<Rm>
	12ビットイミディエイト値をレジスタの値から減算	SUB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
	シフトしたレジスタの値をレジスタの値から減算	SUB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}

付録 1　Thumb-2 命令

●32 ビット命令（2/4）

機能	動作	ニーモニック
四則演算命令	12ビットイミディエイト値をレジスタの値から減算	SUBW.W <Rd>, <Rn>, #<immed_12>
	符号なし除算	UDIV<c> <Rd>,<Rn>,<Rm>
	2つの符号なしレジスタ値を乗算して、一対のレジスタ値に累算	UMLAL.W <RdLo>,<RdHi>,<Rn>,<Rm>
	2つの符号なしレジスタ値を乗算	UMULL.W <RdLo>,<RdHi>,<Rn>,<Rm>
ロード・ストア命令	ポストインクリメント（IA）またはプリデクリメント（DB）で、メモリからレジスタへ複数ロード	LDM{IA¦DB}.W <Rn>{!}, <registers>
	ベースレジスタのアドレス +12ビットイミディエイト値のオフセットのメモリからワードをロード	LDR.W <Rxf>, [<Rn>, #<offset_12>]
	ベースレジスタのアドレス + 12ビットイミディエイト値のオフセットのメモリからPCへワードをロード（分岐）	LDR.W PC, [<Rn>, #<offset_12>]
	ポストインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値を オフセットしたメモリからPCへワードをロード（分岐）	LDR.W PC, [Rn], #<+/-<offset_8>
	ポストインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値をオフセットしたメモリからワードをロード	LDR.W <Rxf>, [<Rn>], #+/–<offset_8>
	プレインデクスで、ベースレジスタのアドレスに 8ビットイミディエイト値をオフセットしたメモリからワードをロード	LDR.W <Rxf>, [<Rn>, #<+/–<offset_8>]! LDRT.W <Rxf>, [<Rn>, #<offset_8>]
	プレインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値を オフセットしたメモリからPC へワードをロード（分岐）	LDR.W PC, [<Rn>, #+/–<offset_8>]!
	位置が0、1、2、または3つ左にシフトされたレジスタのアドレスのメモリからワードをロード	LDR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	位置が0、1、2、または3つ左にシフトされたレジスタのアドレスのメモリからPCへワードをロード（分岐）	LDR.W PC, [<Rn>, <Rm>{, LSL #<shift>}]
	PC アドレスに12ビットイミディエイト値をオフセットしたメモリからワードをロード	LDR.W <Rxf>, [PC, #+/–<offset_12>]
	PC アドレスに12ビットイミディエイト値をオフセットしたメモリからPCへワードをロード（分岐）	LDR.W PC, [PC, #+/–<offset_12>]
	ベースレジスタのアドレス+12ビットイミディエイト値のオフセットのメモリからバイト [7:0] をロード	LDRB.W <Rxf>, [<Rn>, #<offset_12>]
	ポストインデクスで、ベースレジスタのアドレスを8ビットイミディエイト値で オフセットしたメモリからバイト[7:0] をロード	LDRB.W <Rxf>. [<Rn>], #+/-<offset_8>
	位置が0、1、2、または3つ左にシフトされたレジスタのアドレスのメモリからバイト[7:0] をロード	LDRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	プレインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値で オフセットしたメモリからバイト[7:0] をロード	LDRB.W <Rxf>, [<Rn>, #<+/–<offset_8>]!
	PC アドレスに12 ビットイミディエイト値でオフセットしたメモリからバイトをロード	LDRB.W <Rxf>, [PC, #+/–<offset_12>]
	プレインデクスで、レジスタのアドレスに、8ビットイミディエイト値×4を オフセットしたメモリからダブルワードをロード	LDRD.W <Rxf>, <Rxf2>, [<Rn>, #+/–<offset_8> ＊4]{!}
	ポストインデクスで、レジスタのアドレスに、8ビットイミディエイト値×4を オフセットしたメモリからダブルワードをロード	LDRD.W <Rxf>, <Rxf2>, [<Rn>], #+/–<offset_8> ＊4
	排他レジスタロードは、ベースレジスタの値とイミディエイト値オフセットからアドレスを計算し、 ワードをメモリからロードして、レジスタに書き込みます。	LDREX<c> <Rt>,[<Rn>{,#<imm>}]
	排他レジスタロードハーフワードは、ベースレジスタの値とイミディエイト値オフセットから アドレスを計算し、ハーフワードをメモリからロードして、レジスタに書き込みます。	LDREXH<c> <Rt>,[<Rn>{,#<imm>}]
	排他レジスタロードバイトは、ベースレジスタの値とイミディエイト値オフセットから アドレスを計算し、バイトをメモリからロードして、レジスタに書き込みます。	LDREXB<c> <Rt>,[<Rn>{,#<imm>}]
	ベースレジスタのアドレス +12ビットイミディエイト値のオフセットのメモリからハーフワード [15:0] をロード	LDRH.W <Rxf>, [<Rn>, #<offset_12>]
	プレインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値でオフセットして ハーフワード[15:0] をロード	LDRH.W <Rxf>, [<Rn>, #<+/–<offset_8>]!
	ポストインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値でオフセットして ハーフワード[15:0] をロード	LDRH.W <Rxf>. [<Rn>], #+/-<offset_8>
	位置が0、1、2、または3つ左にシフトされたレジスタのアドレスのメモリから ハーフワード[15:0] をロード	LDRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	PCアドレスに12ビットイミディエイト値をオフセットしたメモリからハーフワードをロード	LDRH.W <Rxf>, [PC, #+/–<offset_12>]
	ベースレジスタのアドレス +12ビットイミディエイト値のオフセットのメモリから符号付きバイト[7:0] をロード	LDRSB.W <Rxf>, [<Rn>, #<offset_12>]
	ポストインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値を オフセットしたメモリから符号付きバイト[7:0] をロード	LDRSB.W <Rxf>. [<Rn>], #+/-<offset_8>

付録 1　Thumb-2 命令

●32 ビット命令（3/4）

機能	動作	ニーモニック
ロード・ストア命令	プレインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値をオフセットしたメモリから符号付きバイト[7:0] をロード	LDRSB.W <Rxf>, [<Rn>, #<+/–<offset_8>]!
	位置が0、1、2、または3つ左にシフトされたレジスタのアドレスのメモリから符号付きバイト[7:0] をロード	LDRSB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	PCアドレスに12ビットイミディエイト値をオフセットしたメモリから符号付きバイトをロード	LDRSB.W <Rxf>, [PC, #+/–<offset_12>]
	ベースレジスタのアドレス + 12ビットイミディエイト値をオフセットしたメモリから符号付きハーフワード[15:0] をロード	LDRSH.W <Rxf>, [<Rn>, #<offset_12>]
	ポストインデクスで、ベースレジスタのアドレスを8ビットイミディエイト値でオフセットして、符号付きハーフワード[15:0] をロード	LDRSH.W <Rxf>. [<Rn>], #+/-<offset_8>
	プレインデクスで、ベースレジスタのアドレスに8ビットイミディエイト値でオフセットして、符号付きハーフワード[15:0] をロード	LDRSH.W <Rxf>, [<Rn>, #<+/–<offset_8>]!
	位置が0、1、2、または3つ左にシフトされたレジスタのアドレスからメモリ符号付きハーフワード[15:0] をロード	LDRSH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	PCアドレスを12ビットイミディエイト値でオフセットしたメモリから符号付きハーフワードをロード	LDRSH.W <Rxf>, [PC, #+/–<offset_12>]
	複数レジスタのワードを連続メモリ位置へストア	STM{IA¦DB}.W <Rn>{!}, <registers>
	レジスタのアドレス + 12ビットイミディエイト値のオフセットに、レジスタのワードをストア	STR.W <Rxf>, [<Rn>, #<offset_12>]
	ポストインデクスで、レジスタのアドレス + 8ビットイミディエイト値のオフセットに、レジスタのワードをストア	STR.W <Rxf>, [<Rn>], #+/–<offset_8>
	位置が0、1、2、または3つシフトされたレジスタのアドレスに、レジスタのワードをストア	STR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	プレインデクスまたはポストインデクスで、レジスタのアドレス + 8ビットイミディエイト値のオフセットに、レジスタのワードをストア	STR.W <Rxf>,[<Rn>,#+/-<offset_8>]{!} STRT.W<Rxf>, [<Rn>, #<offset_8>]
	プレインデクスで、レジスタのアドレスを8ビットイミディエイト値でオフセットして、レジスタのバイト[7:0] をストア	STRB{T}.W <Rxf>, [<Rn>, #+/–<offset_8>]{!}
	レジスタのアドレス + 12ビットイミディエイト値のオフセットに、レジスタのバイト[7:0] をストア	STRB.W <Rxf>, [<Rn>, #<offset_12>]
	ポストインデクスで、レジスタのアドレスを8 ビットイミディエイト値でオフセットして、レジスタのバイト[7:0] をストア	STRB.W <Rxf>, [<Rn>], #+/–<offset_8>
	位置が0、1、2、または3つシフトされたレジスタのアドレスに、レジスタのバイト[7:0] をストア	STRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	ダブルワードをプレインデクスでストア	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/–<offset_8> ＊4]{!}
	ダブルワードをポストインデクスでストア	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/–<offset_8> ＊4]
	排他レジスタストアは、ベースレジスタの値とイミディエイト値オフセットからアドレスを計算し、実行中のプロセッサがアドレス指定された持っている場合、ワードをレジスタからメモリにストメモリに対する排他アクセスをアします。	STREX<c><Rd>,<Rt>, [<Rn>{,#<imm>}]
	排他レジスタストアバイトは、ベースレジスタの値からアドレスを導出し、実行中のプロセッサがアドレス指定されたメモリに対する排他アクセスを持っている場合、バイトをレジスタからメモリにストアします。	STREXB <c> <Rd>,<Rt>,[<Rn>]
	排他レジスタストアハーフワードは、ベースレジスタの値からアドレスを導出し、実行中のプロセッサがアドレス指定されたメモリに対する排他アクセスを持っている場合、ハーフワードをレジスタからメモリにストアします。	STREXH <c> <Rd>,<Rt>,[<Rn>]
	レジスタのアドレス + 12ビットイミディエイト値のオフセットに、レジスタのハーフワード[15:0] をストア	STRH.W <Rxf>, [<Rn>, #<offset_12>]
	位置が0、1、2、または3つシフトされたレジスタのアドレスに、レジスタのハーフワード[15:0] をストア	STRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
	プレインデクスで、レジスタのアドレスを8ビットイミディエイト値でオフセットして、レジスタのハーフワード[15:0]をストア	STRH{T}.W <Rxf>, [<Rn>, #+/–<offset_8>]{!}
	ポストインデクスで、レジスタのアドレスを8ビットイミディエイト値でオフセットして、レジスタのハーフワード[15:0]をストア	STRH.W <Rxf>, [<Rn>], #+/–<offset_8>
低消費電力モード命令	イベント待ち	WFE.W
	割り込み待ち	WFI.W

付録 1　Thumb-2 命令

●32 ビット命令（4/4）

機能	動作	ニーモニック
比較命令	レジスタの値と12ビットイミディエイト値の2の補数とを比較	CMN.W <Rn>, #<modify_constant(immed_12)>
	レジスタの値とシフトされたレジスタの値の2の補数とを比較	CMN.W <Rn>, <Rm>{, <shift>}
	レジスタの値と12ビットイミディエイト値とを比較	CMP.W <Rn>, #<modify_constant(immed_12)>
	レジスタの値とシフトされたレジスタの値とを比較	CMP.W <Rn>, <Rm>{, <shift>}
符号拡張命令	選択されたビットをレジスタにコピーし、符号拡張	SBFX.W <Rd>, <Rn>, #<lsb>, #<width>
	バイトを32ビットに符号拡張	SXTB.W <Rd>, <Rm>{, <rotation>}
	ハーフワードを32ビットに符号拡張	SXTH.W <Rd>, <Rm>{, <rotation>}
	レジスタの値のビットフィールドをレジスタにコピーし、32ビットにゼロ拡張	UBFX.W <Rd>, <Rn>, #<lsb>, #<width>
	符号なしバイトをレジスタにコピーして、32ビットにゼロ拡張	UXTB.W <Rd>, <Rm>{, <rotation>}
	符号なしハーフワードをレジスタにコピーして、32ビットにゼロ拡張	UXTH.W <Rd>, <Rm>{, <rotation>}
分岐命令	条件分岐	B{cond}.W <label>
	リンク付き分岐	BL <label>
	リンク付き分岐（イミディエイト値）	BL <c> <label>
	無条件分岐	B.W <label>
	バイトでテーブル分岐	TBB [<Rn>, <Rm>]
	ハーフワードでテーブル分岐	TBH [<Rn>, <Rm>, LSL #1]
論理演算命令	レジスタの値と12ビットイミディエイト値をビット単位論理積	AND{S}.W <Rd>, <Rn>, #<modify_constant(immed_12>
	レジスタの値とシフトされたレジスタの値をビット単位論理積	AND{S}.W <Rd>, <Rn>, Rm>{, <shift>}
	レジスタの値と12ビットイミディエイト値とを排他的論理和	EOR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
	レジスタの値とシフトされたレジスタの値とを排他的論理和	EOR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
	レジスタの値と12ビットイミディエイト値のNOTとを論理和	ORN{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
	レジスタの値とシフトしたレジスタの値のNOTとを論理和	ORN[S].W <Rd>, <Rn>, <Rm>{, <shift>}
	レジスタの値と12ビットイミディエイト値を論理和	ORR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
	レジスタの値とシフトしたレジスタの値を論理和	ORR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
	ビットの順序を反転	RBIT.W <Rd>, <Rm>
	ワード中のバイト順を反転	REV.W <Rd>, <Rm>
	各ハーフワード中のバイト順をそれぞれ反転	REV16.W <Rd>, <Rn>
	下位ハーフワードのバイト順を反転して、符号拡張	REVSH.W <Rd>, <Rn>
	レジスタの値と12ビットイミディエイト値で排他的論理和	TEQ.W <Rn>, #<modify_constant(immed_12)>
	レジスタの値とシフトしたレジスタの値で排他的論理和	TEQ.W <Rn>, <Rm>{, <shift}
	レジスタの値と12ビットイミディエイト値で論理積	TST.W <Rn>, #<modify_constant(immed_12)>
	レジスタの値とシフトしたレジスタの値で論理積	TST.W <Rn>, <Rm>{, <shift>}
レジスタ転送命令	排他クリアは、実行中のプロセッサのローカルレコードで、 アドレスが排他アクセスの要求を受けているものをクリアします。	CLREX <c>
	レジスタの値に含まれる先行ゼロの数を返す	CLZ.W <Rd>, <Rn>
	無操作	NOP.W
	イベント送信	SEV<c>