Cortex-M0 Armマシン語表 （asm15、抜粋）

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0	Rd			u8								1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3	Rm				Rd2-0			1,3

※Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles

演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0	Rd			u8								1
Rd -= u8	0	0	1	1	1	Rd			u8								1
Rd = PC + u8	1	0	1	0	0	Rd			u8								1
Rd += Rm	0	1	0	0	0	1	0	0	Rd3	Rm				Rd2-0			1,3
Rd = Rn + u3	0	0	0	1	1	1	0	u3			Rn			Rd			1
Rd = Rn - u3	0	0	0	1	1	1	1	u3			Rn			Rd			1
Rd = Rn + Rm	0	0	0	1	1	0	0	Rm			Rn			Rd			1
Rd = Rn - Rm	0	0	0	1	1	0	1	Rm			Rn			Rd			1
Rd = -Rm	0	1	0	0	0	0	1	0	0	1	Rm			Rd			1
Rd *= Rm	0	1	0	0	0	0	1	1	0	1	Rm			Rd			1
Rd = Rm << u5	0	0	0	0	0	u5					Rm			Rd			1
Rd = Rm >> u5	0	0	0	0	1	u5					Rm			Rd			1
Rd <<= Rm	0	1	0	0	0	0	0	0	1	0	Rm			Rd			1
Rd >>= Rm	0	1	0	0	0	0	0	0	1	1	Rm			Rd			1
Rd = ~Rm	0	1	0	0	0	0	1	1	1	1	Rm			Rd			1
Rd &= Rm	0	1	0	0	0	0	0	0	0	0	Rm			Rd			1
Rd |= Rm	0	1	0	0	0	0	1	1	0	0	Rm			Rd			1
Rd ^= Rm	0	1	0	0	0	0	0	0	0	1	Rm			Rd			1

※Rd3とRd2-0の4bitでRdを指定する、Rd=PCの時3cycles

※Rd=PC+u8: u8は4byte単位

メモリアクセス	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = [Rn + u5]	0	1	1	1	1	u5					Rn			Rd			2
Rd = [Rn + u5]W	1	0	0	0	1	u5					Rn			Rd			2
Rd = [Rn + u5]L	0	1	1	0	1	u5					Rn			Rd			2
Rd = [PC + u8]L	0	1	0	0	1	Rd			u8								2
Rd = [Rn + Rm]	0	1	0	1	1	1	0	Rm			Rn			Rd			2
Rd = [Rn + Rm]C	0	1	0	1	0	1	1	Rm			Rn			Rd			2
Rd = [Rn + Rm]W	0	1	0	1	1	0	1	Rm			Rn			Rd			2
Rd = [Rn + Rm]S	0	1	0	1	1	1	1	Rm			Rn			Rd			2
Rd = [Rn + Rm]L	0	1	0	1	1	0	0	Rm			Rn			Rd			2
[Rn + u5] = Rd	0	1	1	1	0	u5					Rn			Rd			2
[Rn + u5]W = Rd	1	0	0	0	0	u5					Rn			Rd			2
[Rn + u5]L = Rd	0	1	1	0	0	u5					Rn			Rd			2
[Rn + Rm] = Rd	0	1	0	1	0	1	0	Rm			Rn			Rd			2
[Rn + Rm]W = Rd	0	1	0	1	0	0	1	Rm			Rn			Rd			2
[Rn + Rm]L = Rd	0	1	0	1	0	0	0	Rm			Rn			Rd			2

※[]後の記号でメモリサイズと符号を表す（W:2byte、L:4byte、C:符号付き1byte、S:符号付き2byte）

※u5/u8:Wの場合2byte単位、Lの場合4byte単位となる

条件判断	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rn - u8	0	0	1	0	1	Rn			u8								1
Rn - Rm	0	1	0	0	0	1	0	1	Rn3	Rm				Rn2-0			1
Rn - Rm	0	1	0	0	0	0	1	0	1	0	Rm			Rn			1
Rn + Rm	0	1	0	0	0	0	1	0	1	1	Rm			Rn			1
Rn & Rm	0	1	0	0	0	0	1	0	0	0	Rm			Rn			1

※Rn3とRn2-0の4bitでRnを指定する

分岐	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
IF 0 GOTO n8	1	1	0	1	0	0	0	0	n8								1,3
IF !0 GOTO n8	1	1	0	1	0	0	0	1	n8								1,3
IF cond GOTO n8	1	1	0	1	cond				n8								1,3
GOTO n11	1	1	1	0	0	n11											3
GOTO Rm	0	1	0	0	0	1	1	1	0	Rm				0	0	0	3
GOSUB Rm	0	1	0	0	0	1	1	1	1	Rm				0	0	0	3
GOSUB n22	1	1	1	1	0	n22(21-11)											1
-	1	1	1	1	1	n22(10-0)											3
RET (=#4770)	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	3

※n8/n11/n22:飛び先との命令数の差分から-2した数を指定、分岐するとき3cycles

※cond:0-14 (EQ, NE, CS, CC, MI, PL, VS, VC, HI, LS, GE, LT, GT, LE, AL) !を付けて否定

スタック	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
PUSH {regs}	1	0	1	1	0	1	0	LR	R7	R6	R5	R4	R3	R2	R1	R0	1+N
POP {regs}	1	0	1	1	1	1	0	PC	R7	R6	R5	R4	R3	R2	R1	R0	1,4 +N
SP += u7	1	0	1	1	0	0	0	0	0	u7							1
SP -= u7	1	0	1	1	0	0	0	0	1	u7							1
Rd = SP + u8	1	0	1	0	1	Rd			u8								1
Rd = [SP + u8]L	1	0	0	1	1	Rd			u8								2
[SP + u8]L = Rd	1	0	0	1	0	Rd			u8								2

※u7/u8:4byte単位

※PUSH:SPへregsの大きいレジスタから順に積みSPを減らす 例）PUSH {R1,R2}

※POP:SPからregsの小さいレジスタから順に読み込みSPを増やす 例）POP {R1,R2}

※N:指定したレジスタの数、PCへPOPした場合4+Ncycles（それ以外は1+Ncycles）

特殊演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = REV(Rm)	1	0	1	1	1	0	1	0	0	0	Rm			Rd			1
Rd = REV16(Rm)	1	0	1	1	1	0	1	0	0	1	Rm			Rd			1
Rd = REVSH(Rm)	1	0	1	1	1	0	1	0	1	1	Rm			Rd			1
Rd = ASR(Rm, u5)	0	0	0	1	0	u5					Rm			Rd			1
ASR Rd, Rm	0	1	0	0	0	0	0	1	0	0	Rm			Rd			1
BIC Rd, Rm	0	1	0	0	0	0	1	1	1	0	Rm			Rd			1
ROR Rd, Rm	0	1	0	0	0	0	0	1	1	1	Rm			Rd			1
ADC Rd, Rm	0	1	0	0	0	0	0	1	0	1	Rm			Rd			1
SBC Rd, Rm	0	1	0	0	0	0	0	1	1	0	Rm			Rd			1

※BIC:ビットクリア、ASR:符号付き右シフト、ROR:右ローテート

※REV:byteオーダー反転、REV16:byteオーダー反転(2byteずつ)、REVSH:符号付き16bitを反転32bit化

※ADC:キャリー付き足し算、SBC:キャリー付き引き算

メモリアクセス2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
LDM Rn, {regs}	1	1	0	0	1	Rn			R7	R6	R5	R4	R3	R2	R1	R0	1+N
STM Rn, {regs}	1	1	0	0	0	Rn			R7	R6	R5	R4	R3	R2	R1	R0	1+N

※N:指定したレジスタの数（マルチメモリアクセス）
※LDM:アドレスRnからregsの小さいレジスタから順に読み込みRnを進める 例）LDM R0,{R1,R2}
※STM:アドレスRnへregsの小さいレジスタから順に書き込みRnを進める 例）LDM R0,{R1,R2}

その他	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**cycles**
NOP (=0)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
CPSID (=#B672)	1	0	1	1	0	1	1	0	0	1	1	1	0	0	1	0	1
CPSIE (=#B662)	1	0	1	1	0	1	1	0	0	1	1	0	0	0	1	0	1
WFI (=#BF30)	1	0	1	1	1	1	1	1	0	0	1	1	0	0	0	0	2

※CPSID:割込禁止、CPSIE:割込許可、WFI:割込待ち、NOP:なにもしない(no operation)
R0=R0<<0

- マシン語関連ツール
asm15 - Assembler for IchigoJam
cpuemu15 - IchigoJam マシン語エミュレーター alpha1 (説明)

- 連載、IchigoJamではじめる、Armマシン語入門
1. はじめてのマシン語
2. ハンドアセンブルで超速計算！
3. マシン語メモリアクセスで画面超速表示！
4. マシン語でLEDを光らせよう！
5. 楽しさ広がるマルチバイトメモリアクセスとスタック
6. マシン語使いこなしTIPS
7. カジュアルに使うインラインマシン語
8. アセンブラを使って楽しよう
9. マシン語で高速SPI
10. マシン語を制するもの時間を制す
11. 画面をイチゴで埋め尽くす12の方法

DATA: Cortex-M0プロセッサ - Arm (cycles)
Text: CC BY ichigojam.net