AVR®命令一式手引書

AVR®命令一式手引書

序説

この手引書は8ビットAVR®デバイスに対して利用可能な全ての命令の概要と説明を与えます。各命令は機能的な説明、それの操作符号(演算子)、構文、ステータス レジスタの最後の状態、周期数を含む自身の項を持ちます。

この手引書はAVRデバイスによって使われる各種アドレス指定種別の説明、最新の全てのAVRデバイスとどの命令が利用可能かを一覧にする追補も含みます。

本書は一般の方々の便宜のため有志により作成されたもので、Microchip社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

目次

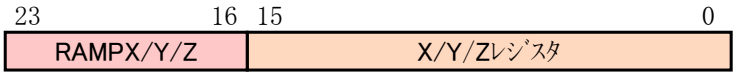
1. 命令一式用語

ステータス レジスタ (SREG)	SREG	: ステータス レジスタ
	C	: キャリー フラグ
	Z	: ゼロ フラグ
	N	: 負フラグ
	V	: 2の補数溢れフラグ
	S	: N + V, 符号付き検査用
	H	: ハーフ キャリー フラグ
	T	: BLDとBSTの命令によって使われる転送ビット
	I	: 全体割り込み許可/禁止ビット
レジスタとオペランド	Rd	: レジスタ ファイルでの転送先(と単一被演算子での転送元)レジスタ
	Rr	: レジスタ ファイルでの転送元レジスタ
	R	: 命令が実行された後の結果
	K	: 定数データ
	k	: 定数アドレス
	b	: レジスタ ファイルまたはI/Oレジスタ内のビット位置 (0～7:3ビット)
	s	: ステータス レジスタ内のビット位置 (0～7:3ビット)
	X,Y,Z	: 間接アドレス レジスタ (X=R27:R26, Y=R29:R28, Z=R31:R30またはメモリが64Kバイトを超える場合にX=RAMPX:R27:R26, Y=RAMPY:R29:R28, Z=RAMPZ:R31:R30)
	A	: I/Oメモリ アドレス
	q	: 間接アドレス用変位量 (主に6ビット)
	UU	: 符号なし×符号なし被演算子
	SS	: 符号付き×符号付き被演算子
	SU	: 符号付き×符号なし被演算子
メモリ空間識別子	DS( )	: データ空間をアドレス指定するポインタを表します。
	PS( )	: プログラム空間をアドレス指定するポインタを表します。
	I/O(A)	: I/O空間のアドレスA
	I/O(A,b)	: I/O空間のアドレスAのバイトのビット位置b
	Rd(n)	: レジスタRdのビットn
演算子	×	: 算術乗算
	+	: 算術加算
	−	: 算術減算
	∧	: 論理積 (**訳注**:本書では基本的にANDと表記)
	∨	: 論理和 (**訳注**:本書では基本的にORと表記)
	⊕	: 排他的論理和 (**訳注**:本書では基本的にXORと表記)
	>>	: 右移動
	<<	: 左移動
	==	: 比較
	←	: 割り当て
	↔	: 交換
	$\bar{x}$	: xの論理的補数(NOT x)
スタック	STACK	: 戻りアドレスと保存(PUSH)するレジスタ用スタック
	SP	: スタック ポインタ
フラグ	⇔	: 命令によって影響を及ぼされるフラグ
	0	: 命令によって解除(0)されるフラグ
	1	: 命令によって設定(1)されるフラグ
	−	: 命令によって影響を及ぼされないフラグ

2. I/O空間に置かれたCPUレジスタ

2.1. RAMPX, RAMPY, RAMPZ

64Kバイトを超えるデータ空間を持つMCUでデータ空間全域の間接アドレス指定と、64Kバイトを超えるプログラム空間を持つMCUで定数データ取得を許す、X,YまたはZレジスタと結合されるレジスタ。

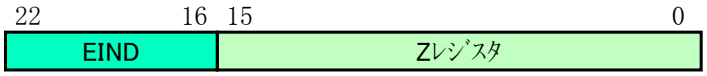


2.2. RAMPD

64Kバイトを超えるデータ空間を持つMCUで、データ空間全域の直接アドレス指定を許す、オペランドの絶対アドレスと結合されるレジスタ。

2.3. EIND

64K語(128Kバイト)を超えるプログラム空間を持つMCUで、プログラム空間全域に対する間接の分岐と呼び出しを許す、Zレジスタと結合されるレジスタ。



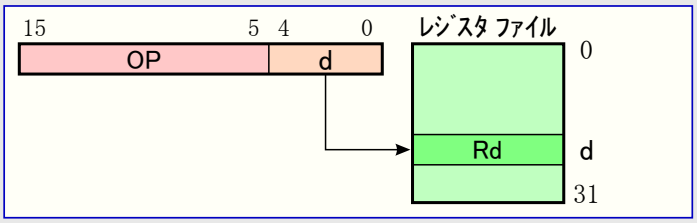
3. ﾌﾟﾛｸﾞﾗﾑ領域とﾃﾞｰﾀ領域のｱﾄﾞﾚｽ指定種別

AVR®強化RISC ﾏｲｸﾛ ｺﾝﾄﾛｰﾗはﾌﾟﾛｸﾞﾗﾑ(ﾌﾗｯｼｭ)ﾒﾓﾘとﾃﾞｰﾀ(SRAM、ﾚｼﾞｽﾀ ﾌｧｲﾙ、I/Oﾚｼﾞｽﾀ、拡張I/Oﾒﾓﾘ)ﾒﾓﾘのｱｸｾｽ用に強力で効率的なｱﾄﾞﾚｽ指定種別を持ちます。本章はAVR構造によって支援される各種ｱﾄﾞﾚｽ指定種別を示します。以下の図中のOPは命令語の動作ｺｰﾄﾞ部分を示しますが、図の単純化のためにｱﾄﾞﾚｽ指定を示すﾋﾞｯﾄ位置が正確ではありません。一般的にRAMENDとFLASHENDは各々ﾃﾞｰﾀ空間とﾌﾟﾛｸﾞﾗﾑ空間の最終位置を表します。

注: ﾃﾞﾊﾞｲｽによっては存在しないｱﾄﾞﾚｽ指定種別もあるので、対応するﾃﾞﾊﾞｲｽの命令要約を参照、確認してください。

3.1. 単一ﾚｼﾞｽﾀ(Rd)直接ｱﾄﾞﾚｽ指定

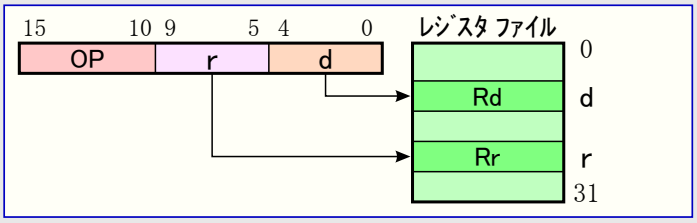
図3-1. 単一ﾚｼﾞｽﾀ直接ｱﾄﾞﾚｽ指定



ｵﾍﾟﾗﾝﾄﾞはﾚｼﾞｽﾀRd(d)を示します。

3.2. ﾚｼﾞｽﾀ間(Rd,Rr)直接ｱﾄﾞﾚｽ指定

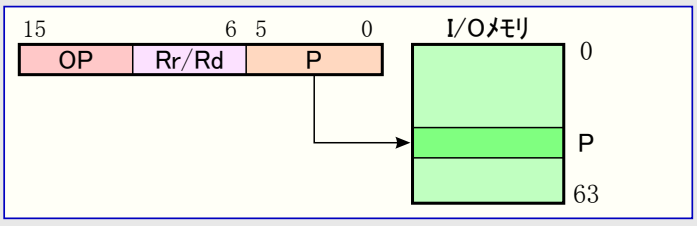
図3-2. ﾚｼﾞｽﾀ間直接ｱﾄﾞﾚｽ指定



ｵﾍﾟﾗﾝﾄﾞはﾚｼﾞｽﾀRr(r)とRd(d)を示し、結果はﾚｼﾞｽﾀRd(d)に格納されます。

3.3. I/O直接ｱﾄﾞﾚｽ指定

図3-3. I/O直接ｱﾄﾞﾚｽ指定

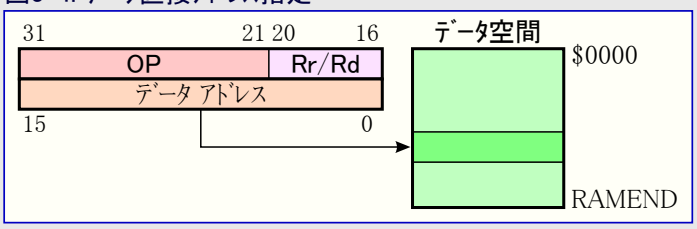


ｵﾍﾟﾗﾝﾄﾞはI/OｱﾄﾞﾚｽPと転送元または転送先となるﾚｼﾞｽﾀRr/Rdを示します。

注: いくつかのAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗはI/O直接ｱﾄﾞﾚｽ指定に対する演算子符号で予約された64位置で支援できるよりも多くの周辺機能部を持ちます。64位置より高い拡張I/Oﾒﾓﾘは I/Oｱﾄﾞﾚｽ指定ではなくﾃﾞｰﾀ ｱﾄﾞﾚｽ指定によってだけ達することができます。

3.4. ﾃﾞｰﾀ直接ｱﾄﾞﾚｽ指定

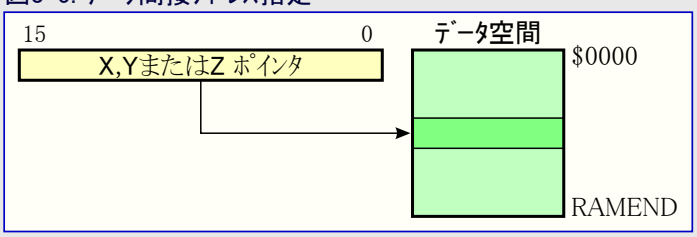
図3-4. ﾃﾞｰﾀ直接ｱﾄﾞﾚｽ指定



ｵﾍﾟﾗﾝﾄﾞは2語命令の下位16ﾋﾞｯﾄでﾃﾞｰﾀ空間のｱﾄﾞﾚｽを示し、Rr/Rdが転送元または転送先となるﾚｼﾞｽﾀを示します。LDS命令は64Kﾊﾞｲﾄを超えるﾒﾓﾘをｱｸｾｽするのにRAMPDﾚｼﾞｽﾀを使います。

3.5. ﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定

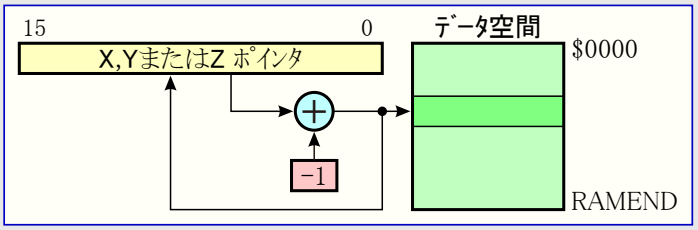
図3-5. ﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定



ｵﾍﾟﾗﾝﾄﾞ ｱﾄﾞﾚｽはX,YまたはZﾎﾟｲﾝﾀの内容です。SRAMのないAVRﾃﾞﾊﾞｲｽではﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定をﾚｼﾞｽﾀ間接ｱﾄﾞﾚｽ指定と呼びます。これはﾃﾞｰﾀ領域の0～31がﾚｼﾞｽﾀ ﾌｧｲﾙのため、ﾚｼﾞｽﾀ間接ｱﾄﾞﾚｽ指定がﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定の一部であるからです。

3.6. 事前減少付きﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定

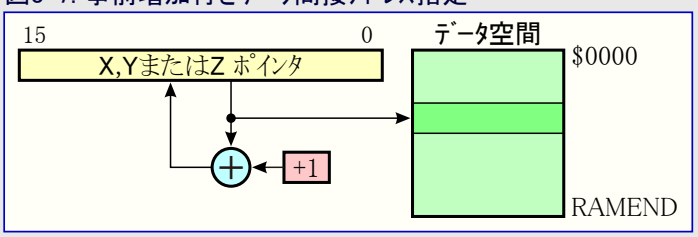
図3-6. 事前減少付きﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定



X,YまたはZﾎﾟｲﾝﾀはｱｸｾｽ動作前に内容が減少(-1)されます。ｵﾍﾟﾗﾝﾄﾞ ｱﾄﾞﾚｽは減少されたX,YまたはZﾎﾟｲﾝﾀの内容です。

3.7. 事後増加付きﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定

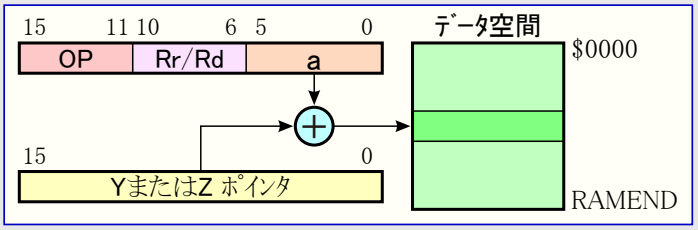
図3-7. 事前増加付きﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定



X,YまたはZﾎﾟｲﾝﾀはｱｸｾｽ動作後に内容が増加(+1)されます。ｵﾍﾟﾗﾝﾄﾞ ｱﾄﾞﾚｽは増加される前のX,YまたはZﾎﾟｲﾝﾀの内容です。

3.8. 変位付きﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定

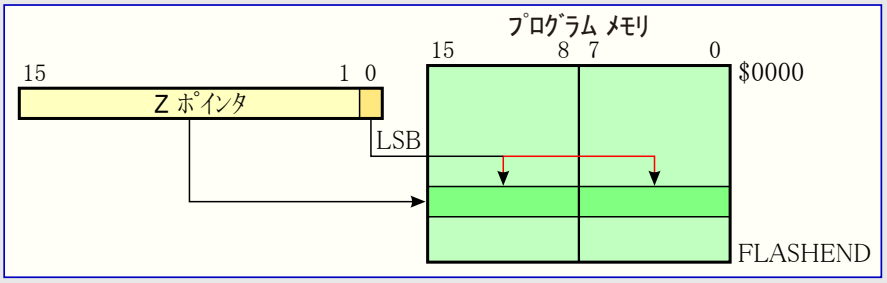
図3-8. 変位付きﾃﾞｰﾀ間接ｱﾄﾞﾚｽ指定



ｵﾍﾟﾗﾝﾄﾞ ｱﾄﾞﾚｽは命令語に含まれる変位(a)をYまたはZﾎﾟｲﾝﾀに加算した結果です。Rr/Rdは転送元または転送先のﾚｼﾞｽﾀを指定します。

3.9. LPM, ELPM, SPM命令を使うﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ定数ｱﾄﾞﾚｽ指定

図3-9. ﾌﾟﾛｸﾞﾗﾑ空間定数ｱﾄﾞﾚｽ指定

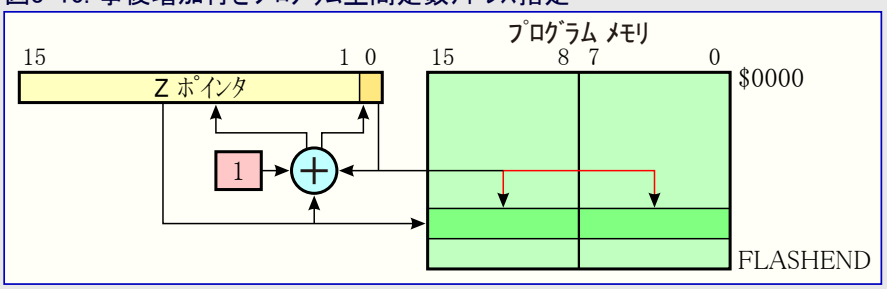


ﾊﾞｲﾄ定数のｱﾄﾞﾚｽ(位置)はZﾎﾟｲﾝﾀの内容で示されます。上位15ﾋﾞｯﾄが語ｱﾄﾞﾚｽを選びます。LPM,ELPM命令では最下位ﾋﾞｯﾄがLSb=0の場合に下位ﾊﾞｲﾄ、LSb=1の場合に上位ﾊﾞｲﾄを選びます。SPM命令ではLSbが解除(0)されるべきです。

ELPM命令が使われる場合、Zﾚｼﾞｽﾀを拡張するのにZﾎﾟｲﾝﾀ拡張ﾚｼﾞｽﾀ(RAMPZ)が使われます。

3.10. LPM Z+, ELPM Z+命令を使う事後増加付きﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ定数ｱﾄﾞﾚｽ指定

図3-10. 事後増加付きﾌﾟﾛｸﾞﾗﾑ空間定数ｱﾄﾞﾚｽ指定

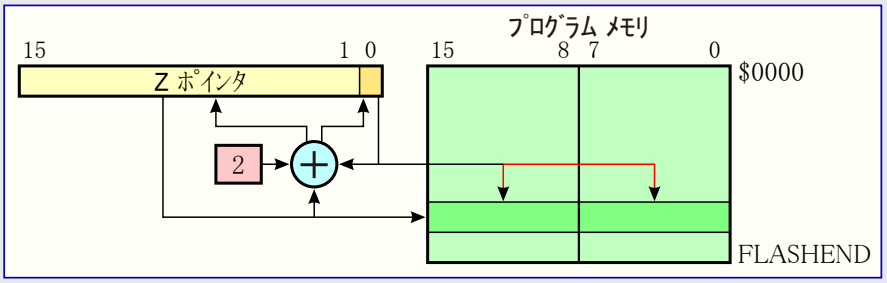


ﾊﾞｲﾄ定数のｱﾄﾞﾚｽ(位置)はZﾎﾟｲﾝﾀの内容で示されます。上位15ﾋﾞｯﾄが語ｱﾄﾞﾚｽを選びます。LPM,ELPM命令では最下位ﾋﾞｯﾄがLSb=0の場合に下位ﾊﾞｲﾄ、LSb=1の場合に上位ﾊﾞｲﾄを選びます。

ELPM Z+命令が使われる場合、Zﾚｼﾞｽﾀを拡張するのにZﾎﾟｲﾝﾀ拡張ﾚｼﾞｽﾀ(RAMPZ)が使われます。

3.11. 事後増加付きSPM命令

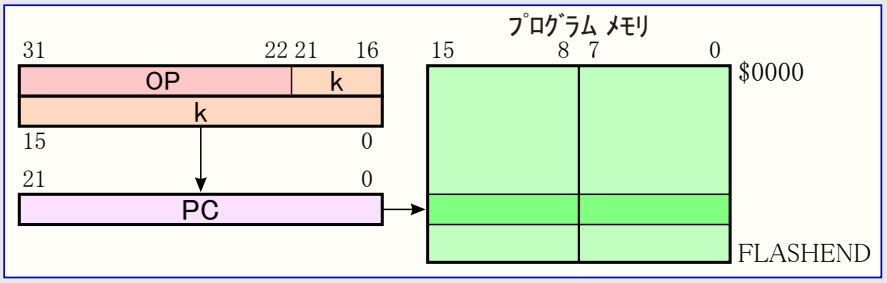
図3-11. 事後増加付きSPM命令



Zﾎﾟｲﾝﾀは操作後に2増やされます。定数ﾊﾞｲﾄ ｱﾄﾞﾚｽは増加前のZﾎﾟｲﾝﾀの内容によって示されます。上位15ﾋﾞｯﾄが語ｱﾄﾞﾚｽを選び、LSbは解除(0)されたままにすべきです。

3.12. JMP, CALL命令によるﾌﾟﾛｸﾞﾗﾑ絶対(直接)ｱﾄﾞﾚｽ指定

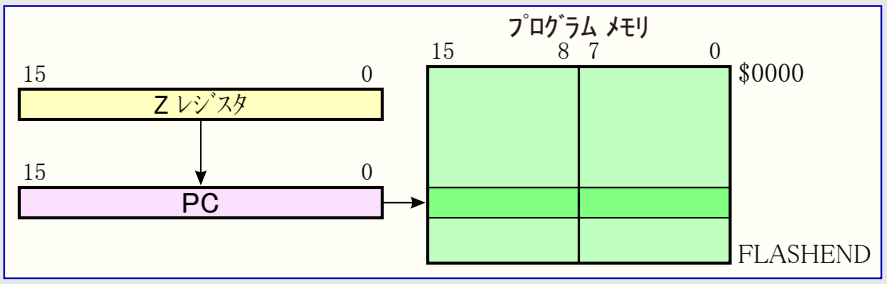
図3-12. ﾌﾟﾛｸﾞﾗﾑ空間絶対ｱﾄﾞﾚｽ指定



ﾌﾟﾛｸﾞﾗﾑ実行は命令語内のｱﾄﾞﾚｽ即値(k)から継続されます。

3.13. IJMP, ICALL命令によるﾌﾟﾛｸﾞﾗﾑ間接ｱﾄﾞﾚｽ指定

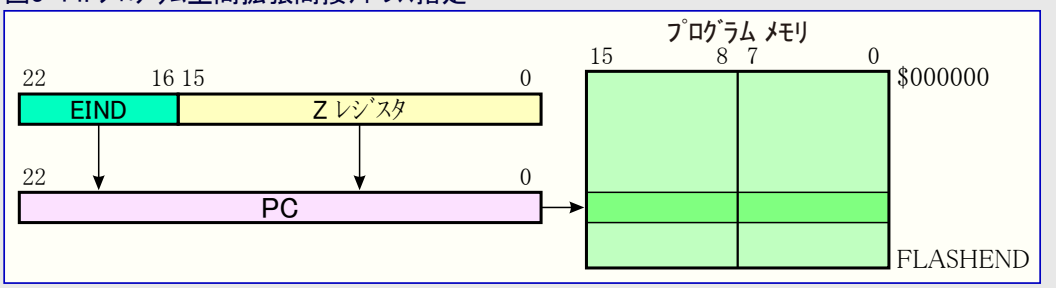
図3-13. ﾌﾟﾛｸﾞﾗﾑ空間間接ｱﾄﾞﾚｽ指定



ﾌﾟﾛｸﾞﾗﾑ実行はZﾚｼﾞｽﾀに含まれるｱﾄﾞﾚｽから継続されます(即ち、PCはZﾚｼﾞｽﾀの内容を設定されます)。

3.14. EIJMP, EICALL命令によるﾌﾟﾛｸﾞﾗﾑ拡張間接ｱﾄﾞﾚｽ指定

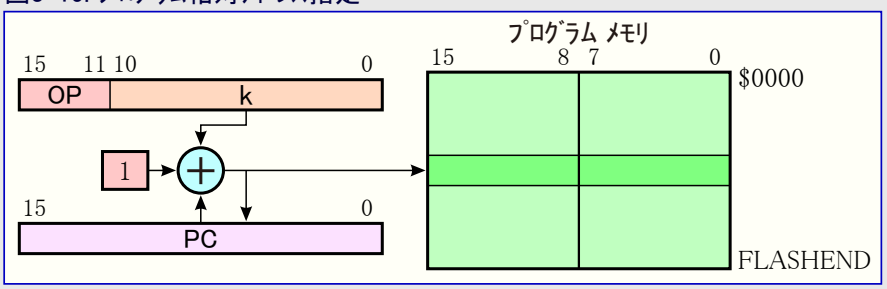
図3-14. ﾌﾟﾛｸﾞﾗﾑ空間拡張間接ｱﾄﾞﾚｽ指定



ﾌﾟﾛｸﾞﾗﾑ実行はZﾚｼﾞｽﾀとEINDﾚｼﾞｽﾀに含まれるｱﾄﾞﾚｽから継続されます(即ち、PCはEINDとZのﾚｼﾞｽﾀの内容を設定されます)。

3.15. RJMP, RCALL命令によるﾌﾟﾛｸﾞﾗﾑ相対ｱﾄﾞﾚｽ指定

図3-15. ﾌﾟﾛｸﾞﾗﾑ相対ｱﾄﾞﾚｽ指定



ﾌﾟﾛｸﾞﾗﾑ実行はPC+k+1のｱﾄﾞﾚｽから継続されます。相対値kは符号付きの-2048～2047です。

(**訳補**) 本命令復号時点のPC値は(命令)事前取得のために本命令の次の位置(本命令位置のﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ値をPCとした場合、PC+1)を示しています。従ってこの時点に於けるPC値+相対値は(PC+1)+kになります。

4. 命令一式要約

その生涯中のAVR CPUの数度の更新は命令一式、特に命令のﾀｲﾐﾝｸﾞに対して結果として異なる趣を生じます。命令一式に於いて全命令が全てのﾃﾞﾊﾞｲｽに含まれる訳ではありませんが、機械語ﾚﾍﾞﾙの互換性は縮小ｺｱ(AVRrc:AVR Reduced Core)に関連する非常に少数の例外付きで全てのCPU版に対してそのままです。下表はAVR 8ﾋﾞｯﾄCPUの主な版を含みます。各種版に加えて、ﾃﾞﾊﾞｲｽﾒﾓﾘ割り当ての大きさに依存する違いがあります。代表的にそれらの違いはC/C++ｺﾝﾊﾟｲﾗによって処理されますが、ｺｰﾄﾞを移植する使用者はｺｰﾄﾞ実行がｸﾛｯｸ周期数で僅かに変わり得ることに注意すべきです。

表4-1. AVR® 8ﾋﾞｯﾄCPUの版

名称	説明
AVR	1995年からの大元の命令一式
AVRe	AVR命令一式は語移動(MOVW)命令で拡張され、ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ取得(LPM)が強化されました。AVRと同じﾀｲﾐﾝｸﾞ。
AVRe+	AVR命令一式は乗算(xMULxx)命令で拡張され、該当する場合、EICALL、EIJMP、ELPMの拡張範囲命令を持ちます。AVR,AVReと同じﾀｲﾐﾝｸﾞ。従って、ｸﾛｯｸ周期数の表一覧はAVReとAVRe+を区別せず、両方を表すのにAVReを使います。
AVRxm	AVRe+命令一式は読み/変更/書き(LAxとXCH)とﾃﾞｰﾀ暗号規格(DES)の命令で拡張されました。SPMはSPM Z+2を含むように拡張されました。ﾀｲﾐﾝｸﾞはAVR,AVRe,AVRe+と比べてかなり違います。
AVRxt	AVRe+とAVRxmの組み合わせ。利用可能な命令はAVRe+と同じですが、AVR,AVRe,AVRe+,AVRxmに比べてﾀｲﾐﾝｸﾞが改良されています。
AVRrc	AVRrcはﾚｼﾞｽﾀ ﾌｧｲﾙで16個のﾚｼﾞｽﾀ(R31～R16)だけを持ち、命令一式が減らされています。ﾀｲﾐﾝｸﾞはAVR,AVRe,AVRe+,AVRxm,AVRxtに比べてかなり違います。更なる詳細については命令一式概要を参照してください。

表4-2. 算術/論理演算命令

ﾆｰﾓﾆｯｸ	ｵﾍﾟﾗﾝﾄﾞ	意味	動作	ﾌﾗｸﾞ	ｸﾛｯｸ数			
					AVRe	AVRxm	AVRxt	AVRrc
ADD	Rd,Rr	汎用ﾚｼﾞｽﾀ間の加算	Rd ← Rd + Rr	I,T,H,S,V,N,Z,C	1	1	1	1
ADC	Rd,Rr	ｷｬﾘｰを含め汎用ﾚｼﾞｽﾀ間の加算	Rd ← Rd + Rr + C	I,T,H,S,V,N,Z,C	1	1	1	1
ADIW	Rd,K	即値の語(ﾜｰﾄﾞ)長加算	Rd+1:Rd ← Rd+1:Rd + K	I,T,H,S,V,N,Z,C	2	2	2	N/A
SUB	Rd,Rr	汎用ﾚｼﾞｽﾀ間の減算	Rd ← Rd − Rr	I,T,H,S,V,N,Z,C	1	1	1	1
SUBI	Rd,K	汎用ﾚｼﾞｽﾀから即値の減算	Rd ← Rd − K	I,T,H,S,V,N,Z,C	1	1	1	1
SBC	Rd,Rr	ｷｬﾘｰを含め汎用ﾚｼﾞｽﾀ間の減算	Rd ← Rd − Rr − C	I,T,H,S,V,N,Z,C	1	1	1	1
SBCI	Rd,K	汎用ﾚｼﾞｽﾀからｷｬﾘｰと即値の減算	Rd ← Rd − K − C	I,T,H,S,V,N,Z,C	1	1	1	1
SBIW	Rd,K	即値の語(ﾜｰﾄﾞ)長減算	Rd+1:Rd ← Rd+1:Rd − K	I,T,H,S,V,N,Z,C	2	2	2	N/A
AND	Rd,Rr	汎用ﾚｼﾞｽﾀ間の論理積	Rd ← Rd AND Rr	I,T,H,S,0,N,Z,C	1	1	1	1
ANDI	Rd,K	汎用ﾚｼﾞｽﾀと即値の論理積	Rd ← Rd AND K	I,T,H,S,0,N,Z,C	1	1	1	1
OR	Rd,Rr	汎用ﾚｼﾞｽﾀ間の論理和	Rd ← Rd OR Rr	I,T,H,S,0,N,Z,C	1	1	1	1
ORI	Rd,K	汎用ﾚｼﾞｽﾀと即値の論理和	Rd ← Rd OR K	I,T,H,S,0,N,Z,C	1	1	1	1
EOR	Rd,Rr	汎用ﾚｼﾞｽﾀ間の排他的論理和	Rd ← Rd XOR Rr	I,T,H,S,0,N,Z,C	1	1	1	1
COM	Rd	1の補数(論理反転)	Rd ← $FF − Rd	I,T,H,S,0,N,Z,1	1	1	1	1
NEG	Rd	2の補数	Rd ← $00 − Rd	I,T,H,S,V,N,Z,C	1	1	1	1
SBR	Rd,K	汎用ﾚｼﾞｽﾀの(複数)ﾋﾞｯﾄ設定(1)	Rd ← Rd OR K	I,T,H,S,0,N,Z,C	1	1	1	1
CBR	Rd,K	汎用ﾚｼﾞｽﾀの(複数)ﾋﾞｯﾄ解除(0)	Rd ← Rd AND ($FF − K)	I,T,H,S,0,N,Z,C	1	1	1	1
INC	Rd	汎用ﾚｼﾞｽﾀの増加(+1)	Rd ← Rd + 1	I,T,H,S,V,N,Z,C	1	1	1	1
DEC	Rd	汎用ﾚｼﾞｽﾀの減少(−1)	Rd ← Rd − 1	I,T,H,S,V,N,Z,C	1	1	1	1
TST	Rd	汎用ﾚｼﾞｽﾀのｾﾞﾛとﾏｲﾅｽ検査	Rd ← Rd AND Rd	I,T,H,S,0,N,Z,C	1	1	1	1
CLR	Rd	汎用ﾚｼﾞｽﾀの全0設定(=$00)	Rd ← Rd XOR Rd	I,T,H,0,0,0,1,C	1	1	1	1
SER	Rd	汎用ﾚｼﾞｽﾀの全1設定(=$FF)	Rd ← $FF	I,T,H,S,V,N,Z,C	1	1	1	1
MUL	Rd,Rr	符号なし間の乗算	R1:R0 ← Rd×Rr (U×U)	I,T,H,S,V,N,Z,C	2	2	2	N/A
MULS	Rd,Rr	符号付き間の乗算	R1:R0 ← Rd×Rr (S×S)	I,T,H,S,V,N,Z,C	2	2	2	N/A
MULSU	Rd,Rr	符号付きと符号なしの乗算	R1:R0 ← Rd×Rr (S×U)	I,T,H,S,V,N,Z,C	2	2	2	N/A
FMUL	Rd,Rr	符号なし間の固定小数点乗算	R1:R0 ← (Rd×Rr)<<1 (U×U)	I,T,H,S,V,N,Z,C	2	2	2	N/A
FMULS	Rd,Rr	符号付き間の固定小数点乗算	R1:R0 ← (Rd×Rr)<<1 (S×S)	I,T,H,S,V,N,Z,C	2	2	2	N/A
FMULSU	Rd,Rr	符号付きと符号なしの固定小数点乗算	R1:R0 ← (Rd×Rr)<<1 (S×U)	I,T,H,S,V,N,Z,C	2	2	2	N/A
DES	K	ﾃﾞｰﾀ暗号化/解読	H=0で、R15～R0 ← 暗号化(R15～R0,K) H=1で、R15～R0 ← 解読(R15～R0,K)	I,T,H,S,V,N,Z,C	N/A	1/2	N/A	N/A

表4-3. 分岐命令

ニーモニック	オペランド	意味	動作	フラグ	クロック数			
					AVRe	AVRxm	AVRxt	AVRrc
RJMP	k	無条件PC相対分岐	PC ← PC + k + 1	I,T,H,S,V,N,Z,C	2	2	2	2
IJMP		無条件Zレジスタ間接分岐	PC(15～0) ← Z, PC(21～16) ← 0	I,T,H,S,V,N,Z,C	2	2	2	2
EIJMP		無条件拡張Zレジスタ間接分岐	PC(15～0) ← Z, PC(21～16) ← EIND	I,T,H,S,V,N,Z,C	2	2	2	N/A
JMP	k	無条件絶対分岐	PC ← k	I,T,H,S,V,N,Z,C	3	3	3	N/A
RCALL	k	PC相対サブルーチン呼び出し	PC ← PC + k + 1	I,T,H,S,V,N,Z,C	3/4(1)	2/3(1)	2/3	3
ICALL		Zレジスタ間接サブルーチン呼び出し	PC(15～0) ← Z, PC(21～16) ← 0	I,T,H,S,V,N,Z,C	3/4(1)	2/3(1)	2/3	3
EICALL		拡張Zレジスタ間接サブルーチン呼び出し	PC(15～0) ← Z, PC(21～16) ← EIND	I,T,H,S,V,N,Z,C	4(1)	3(1)	2/3	N/A
CALL	k	絶対サブルーチン呼び出し	PC ← k	I,T,H,S,V,N,Z,C	4/5(1)	3/4(1)	3/4	N/A
RET		サブルーチンからの復帰	PC ← STACK	I,T,H,S,V,N,Z,C	4/5(1)	4/5(1)	4/5	6
RETI		割り込みからの復帰	PC ← STACK	I,T,H,S,V,N,Z,C	4/5(1)	4/5(1)	4/5	6
CPSE	Rd,Rr	汎用レジスタ間比較、一致でスキップ	Rd=Rrなら、PC ← PC + 2または3	I,T,H,S,V,N,Z,C	1/2,3	1/2,3	1/2,3	1/2
CP	Rd,Rr	汎用レジスタ間の比較	Rd − Rr	I,T,H,S,V,N,Z,C	1	1	1	1
CPC	Rd,Rr	キャリーを含めた汎用レジスタ間の比較	Rd − Rr − C	I,T,H,S,V,N,Z,C	1	1	1	1
CPI	Rd,K	汎用レジスタと即値の比較	Rd − K	I,T,H,S,V,N,Z,C	1	1	1	1
SBRC	Rr,b	汎用レジスタのビットが解除(0)でスキップ	Rr(b)=0なら、PC ← PC + 2または3	I,T,H,S,V,N,Z,C	1/2,3	1/2,3	1/2,3	1/2
SBRS	Rr,b	汎用レジスタのビットが設定(1)でスキップ	Rr(b)=1なら、PC ← PC + 2または3	I,T,H,S,V,N,Z,C	1/2,3	1/2,3	1/2,3	1/2
SBIC	A,b	I/Oレジスタのビットが解除(0)でスキップ	I/O(A,b)=0なら、PC ← PC + 2または3	I,T,H,S,V,N,Z,C	1/2,3	2/3,4	1/2,3	1/2
SBIS	A,b	I/Oレジスタのビットが設定(1)でスキップ	I/O(A,b)=1なら、PC ← PC + 2または3	I,T,H,S,V,N,Z,C	1/2,3	2/3,4	1/2,3	1/2
BRBS	s,k	ステータス フラグが設定(1)で分岐	SREG(b)=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRBC	s,k	ステータス フラグが解除(0)で分岐	SREG(b)=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BREQ	k	一致で分岐	Z=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRNE	k	不一致で分岐	Z=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRCS	k	キャリー フラグが設定(1)で分岐	C=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRCC	k	キャリー フラグが解除(0)で分岐	C=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRSH	k	符号なしの≧で分岐	C=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRLO	k	符号なしの＜で分岐	C=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRMI	k	−(マイナス)で分岐	N=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRPL	k	+(プラス)で分岐	N=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRGE	k	符号付きの≧で分岐	(N XOR V)=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRLT	k	符号付きの＜で分岐	(N XOR V)=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRHS	k	ハーフキャリー フラグが設定(1)で分岐	H=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRHC	k	ハーフキャリー フラグが解除(0)で分岐	H=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRTS	k	一時フラグが設定(1)で分岐	T=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRTC	k	一時フラグが解除(0)で分岐	T=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRVS	k	2の補数溢れフラグが設定(1)で分岐	V=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRVC	k	2の補数溢れフラグが解除(0)で分岐	V=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRIE	k	割り込み許可で分岐	I=1なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2
BRID	k	割り込み禁止で分岐	I=0なら、PC ← PC + k + 1	I,T,H,S,V,N,Z,C	1/2	1/2	1/2	1/2

或る形式			補完(逆)形式			注釈
ニーモニック	一般的な検査	ステータス レジスタ	ニーモニック	一般的な検査	ステータス レジスタ	
BRGE	Rd≧Rr	S=0	BRLT	Rd＜Rr	S=1	符号付き
BRSH		C=0	BRLO		C=1	符号なし
BRNE	Rd≠Rr	Z=0	BREQ	Rd＝Rr	Z=1	符号なし/符号付き
BRBC	−	SREG(s)=0	BRBS	−	SREG(s)=1	−
BRCC		C=0	BRCS		C=1	単純(フラグ直接)
BRPL		N=0	BRMI		N=1	
BRVC		V=0	BRVS		V=1	

注: ステータス レジスタの状態は先行する命令の結果で、更なる情報については命令説明をご覧ください。先行する命令がCP、CPI、SUB、SUBIなら、'一般的な検査'列に従って分岐が起きます。

表4-4. ﾃﾞｰﾀ転送命令

ﾆｰﾓﾆｯｸ	ｵﾍﾟﾗﾝﾄﾞ	意味	動作	ﾌﾗｸﾞ	ｸﾛｯｸ数			
					AVRe	AVRxm	AVRxt	AVRrc
MOV	Rd,Rr	汎用ﾚｼﾞｽﾀ間の複写	Rd ← Rr	I,T,H,S,V,N,Z,C	1	1	1	1
MOVW	Rd,Rr	汎用ﾚｼﾞｽﾀ対間の複写	Rd+1:Rd ← Rr+1:Rr	I,T,H,S,V,N,Z,C	1	1	1	N/A
LDI	Rd,K	即値の取得	Rd ← K	I,T,H,S,V,N,Z,C	1	1	1	1
LDS(*)	Rd,k	ﾃﾞｰﾀ空間から直接取得	Rd ← (k)	I,T,H,S,V,N,Z,C	2(1)	3(1,3)	3(2)	2
LD	Rd,X	Xﾚｼﾞｽﾀ間接で取得	Rd ← (X)	I,T,H,S,V,N,Z,C	2(1)	2(1,3)	2(2)	1/2
LD	Rd,X+	事後増加付きXﾚｼﾞｽﾀ間接で取得	Rd ← (X), X ← X + 1	I,T,H,S,V,N,Z,C	2(1)	2(1,3)	2(2)	2/3
LD	Rd,-X	事前減少付きXﾚｼﾞｽﾀ間接で取得	X ← X - 1, Rd ← (X)	I,T,H,S,V,N,Z,C	2(1)	3(1,3)	2(2)	2/3
LD	Rd,Y	Yﾚｼﾞｽﾀ間接で取得	Rd ← (Y)	I,T,H,S,V,N,Z,C	2(1)	2(1,3)	2(2)	1/2
LD	Rd,Y+	事後増加付きYﾚｼﾞｽﾀ間接で取得	Rd ← (Y), Y ← Y + 1	I,T,H,S,V,N,Z,C	2(1)	2(1,3)	2(2)	2/3
LD	Rd,-Y	事前減少付きYﾚｼﾞｽﾀ間接で取得	Y ← Y - 1, Rd ← (Y)	I,T,H,S,V,N,Z,C	2(1)	3(1,3)	2(2)	2/3
LDD	Rd,Y+q	変位付きYﾚｼﾞｽﾀ間接で取得	Rd ← (Y+q)	I,T,H,S,V,N,Z,C	2(1)	3(1,3)	2(2)	N/A
LD	Rd,Z	Zﾚｼﾞｽﾀ間接で取得	Rd ← (Z)	I,T,H,S,V,N,Z,C	2(1)	2(1,3)	2(2)	1/2
LD	Rd,Z+	事後増加付きZﾚｼﾞｽﾀ間接で取得	Rd ← (Z), Z ← Z + 1	I,T,H,S,V,N,Z,C	2(1)	2(1,3)	2(2)	2/3
LD	Rd,-Z	事前減少付きZﾚｼﾞｽﾀ間接で取得	Z ← Z - 1, Rd ← (Z)	I,T,H,S,V,N,Z,C	2(1)	3(1,3)	2(2)	2/3
LDD	Rd,Z+q	変位付きZﾚｼﾞｽﾀ間接で取得	Rd ← (Z+q)	I,T,H,S,V,N,Z,C	2(1)	3(1,3)	2(2)	N/A
STS(*)	k,Rr	ﾃﾞｰﾀ空間へ直接設定	(k) ← Rr	I,T,H,S,V,N,Z,C	2(1)	2(1)	2(2)	1
ST	X,Rr	Xﾚｼﾞｽﾀ間接で設定	(X) ← Rr	I,T,H,S,V,N,Z,C	2(1)	1(1)	1(2)	1
ST	X+,Rr	事後増加付きXﾚｼﾞｽﾀ間接で設定	(X) ← Rr, X ← X + 1	I,T,H,S,V,N,Z,C	2((1)	1(1)	1(2)	1
ST	-X,Rr	事前減少付きXﾚｼﾞｽﾀ間接で設定	X ← X - 1, (X) ← Rr	I,T,H,S,V,N,Z,C	2(1)	2(1)	1(2)	2
ST	Y,Rr	Yﾚｼﾞｽﾀ間接で設定	(Y) ← Rr	I,T,H,S,V,N,Z,C	2(1)	1(1)	1(2)	1
ST	Y+,Rr	事後増加付きYﾚｼﾞｽﾀ間接で設定	(Y) ← Rr, Y ← Y + 1	I,T,H,S,V,N,Z,C	2(1)	1(1)	1(2)	1
ST	-Y,Rr	事前減少付きYﾚｼﾞｽﾀ間接で設定	Y ← Y - 1, (Y) ← Rr	I,T,H,S,V,N,Z,C	2(1)	2(1)	1(2)	2
STD	Y+q,Rr	変位付きYﾚｼﾞｽﾀ間接で設定	(Y+q) ← Rr	I,T,H,S,V,N,Z,C	2(1)	2(1)	1(2)	N/A
ST	Z,Rr	Zﾚｼﾞｽﾀ間接で設定	(Z) ← Rr	I,T,H,S,V,N,Z,C	2(1)	1(1)	1(2)	1
ST	Z+,Rr	事後増加付きZﾚｼﾞｽﾀ間接で設定	(Z) ← Rr, Z ← Z + 1	I,T,H,S,V,N,Z,C	2(1)	1(1)	1(2)	1
ST	-Z,Rr	事前減少付きZﾚｼﾞｽﾀ間接で設定	Z ← Z - 1, (Z) ← Rr	I,T,H,S,V,N,Z,C	2(1)	2(1)	1(2)	2
STD	Z+q,Rr	変位付きZﾚｼﾞｽﾀ間接で設定	(Z+q) ← Rr	I,T,H,S,V,N,Z,C	2(1)	2(1)	1(2)	N/A
LPM		ﾌﾟﾛｸﾞﾗﾑ領域からZﾚｼﾞｽﾀ間接で取得	R0 ← (Z)	I,T,H,S,V,N,Z,C	3	3	3	N/A
LPM	Rd,Z	同上(任意のﾚｼﾞｽﾀへ)	Rd ← (Z)	I,T,H,S,V,N,Z,C	3	3	3	N/A
LPM	Rd,Z+	同上(事後増加付き)	Rd ← (Z), Z ← Z + 1	I,T,H,S,V,N,Z,C	3	3	3	N/A
ELPM		ﾌﾟﾛｸﾞﾗﾑ領域から拡張Zﾚｼﾞｽﾀ間接で取得	R0 ← (RAMPZ:Z)	I,T,H,S,V,N,Z,C	3	3	3	N/A
ELPM	Rd,Z	同上(任意のﾚｼﾞｽﾀへ)	Rd ← (RAMPZ:Z)	I,T,H,S,V,N,Z,C	3	3	3	N/A
ELPM	Rd,Z+	同上(事後増加付き)	Rd ← (RAMPZ:Z), Z ← Z + 1	I,T,H,S,V,N,Z,C	3	3	3	N/A
SPM		ﾌﾟﾛｸﾞﾗﾑ領域へZﾚｼﾞｽﾀ間接で設定	(RAMPZ:Z) ← R1:R0	I,T,H,S,V,N,Z,C	-(4)	-(4)	-(4)	N/A
SPM	Z+	同上(事後増加(+2)付き)	(RAMPZ:Z) ← R1:R0, Z ← Z + 2	I,T,H,S,V,N,Z,C	N/A	-(4)	-(4)	N/A
IN	Rd,A	I/Oﾚｼﾞｽﾀからの入力	Rd ← I/O(A)	I,T,H,S,V,N,Z,C	1	1	1	1
OUT	A,Rr	I/Oﾚｼﾞｽﾀへの出力	I/O(A) ← Rr	I,T,H,S,V,N,Z,C	1	1	1	1
PUSH	Rr	汎用ﾚｼﾞｽﾀをｽﾀｯｸへ保存	STACK ← Rr	I,T,H,S,V,N,Z,C	2	1(1)	1	1
POP	Rd	ｽﾀｯｸから汎用ﾚｼﾞｽﾀへ復帰	Rd ← STACK	I,T,H,S,V,N,Z,C	2	2(1)	2	3
XCH	Z,Rd	交換	TEMP ← Rd, Rd ← (Z), (Z) ← TEMP	I,T,H,S,V,N,Z,C	N/A	2	N/A	N/A
LAS	Z,Rd	取得と設定	TEMP ← Rd, Rd ← (Z), (Z) ← TEMP OR (Z)	I,T,H,S,V,N,Z,C	N/A	2	N/A	N/A
LAC	Z,Rd	取得と解除	TEMP ← Rd, Rd ← (Z), (Z) ← ($FF-TEMP) AND (Z)	I,T,H,S,V,N,Z,C	N/A	2	N/A	N/A
LAT	Z,Rd	取得と反転	TEMP ← Rd, Rd ← (Z), (Z) ← TEMP EOR (Z)	I,T,H,S,V,N,Z,C	N/A	2	N/A	N/A

(*)：これらの命令はAVRrcｺｱでｱﾄﾞﾚｽ指定範囲が$0040～$00BFに制限され、機械語自体も他と違って1語命令になり、実行ｸﾛｯｸ数も基本的に1つ減少します。詳細についてはAVRrcｺｱに於けるLDS命令とSTS命令を参照してください。

表4-5. ビット操作/検査命令

ニーモニック	オペランド	意味	動作	フラグ	クロック数			
					AVRe	AVRxm	AVRxt	AVRrc
LSL	Rd	論理的左移動(シフト)	Rd(n+1) ← Rd(n), Rd(0) ← 0	I,T,H,S,V,N,Z,C	1	1	1	1
LSR	Rd	論理的右移動(シフト)	Rd(n) ← Rd(n+1), Rd(7) ← 0	I,T,H,S,V,0,Z,C	1	1	1	1
ROL	Rd	キャリーを含めた左回転	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	I,T,H,S,V,N,Z,C	1	1	1	1
ROR	Rd	キャリーを含めた右回転	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	I,T,H,S,V,N,Z,C	1	1	1	1
ASR	Rd	算術的右移動(シフト)	Rd(n) ← Rd(n+1), n=0～6	I,T,H,S,V,N,Z,C	1	1	1	1
SWAP	Rd	ニブル(4ビット)上位/下位交換	Rd(7～4) ⇔ Rd(3～0)	I,T,H,S,V,N,Z,C	1	1	1	1
BSET	s	ステータス レジスタのビット設定(1)	SREG(s) ← 1	1,T,H,S,V,N,Z,C	1	1	1	1
BCLR	s	ステータス レジスタのビット解除(0)	SREG(s) ← 0	0,T,H,S,V,N,Z,C	1	1	1	1
SBI	A,b	I/Oレジスタのビット設定(1)	I/O(A,b) ← 1	I,T,H,S,V,N,Z,C	2	1	1	1
CBI	A,b	I/Oレジスタのビット解除(0)	I/O(A,b) ← 0	I,T,H,S,V,N,Z,C	2	1	1	1
BST	Rr,b	汎用レジスタのビットを一時フラグへ複写	T ← Rr(b)	I,T,H,S,V,N,Z,C	1	1	1	1
BLD	Rd,b	一時フラグを汎用レジスタのビットへ複写	Rd(b) ← T	I,T,H,S,V,N,Z,C	1	1	1	1
SEC		キャリー フラグを設定(1)	C ← 1	I,T,H,S,V,N,Z,1	1	1	1	1
CLC		キャリー フラグを解除(0)	C ← 0	I,T,H,S,V,N,Z,0	1	1	1	1
SEN		負フラグを設定(1)	N ← 1	I,T,H,S,V,1,Z,C	1	1	1	1
CLN		負フラグを解除(0)	N ← 0	I,T,H,S,V,0,Z,C	1	1	1	1
SEZ		ゼロ フラグを設定(1)	Z ← 1	I,T,H,S,V,N,1,C	1	1	1	1
CLZ		ゼロ フラグを解除(0)	Z ← 0	I,T,H,S,V,N,0,C	1	1	1	1
SEI		全体割り込み許可	I ← 1	1,T,H,S,V,N,Z,C	1	1	1	1
CLI		全体割り込み禁止	I ← 0	0,T,H,S,V,N,Z,C	1	1	1	1
SES		符号フラグを設定(1)	S ← 1	I,T,H,1,V,N,Z,C	1	1	1	1
CLS		符号フラグを解除(0)	S ← 0	I,T,H,0,V,N,Z,C	1	1	1	1
SEV		2の補数溢れフラグを設定(1)	V ← 1	I,T,H,S,1,N,Z,C	1	1	1	1
CLV		2の補数溢れフラグを解除(0)	V ← 0	I,T,H,S,0,N,Z,C	1	1	1	1
SET		一時フラグを設定(1)	T ← 1	I,1,H,S,V,N,Z,C	1	1	1	1
CLT		一時フラグを解除(0)	T ← 0	I,0,H,S,V,N,Z,C	1	1	1	1
SEH		ハーフキャリー フラグを設定(1)	H ← 1	I,T,1,S,V,N,Z,C	1	1	1	1
CLH		ハーフキャリー フラグを解除(0)	H ← 0	I,T,0,S,V,N,Z,C	1	1	1	1

表4-6. MCU制御命令

ニーモニック	オペランド	意味	動作	フラグ	クロック数			
					AVRe	AVRxm	AVRxt	AVRrc
BREAK		中断	(デバッグ インターフェース記述を参照)	I,T,H,S,V,N,Z,C	1	1	1	1
NOP		無操作		I,T,H,S,V,N,Z,C	1	1	1	1
SLEEP		休止形態開始	(電力管理と休止形態記述を参照)	I,T,H,S,V,N,Z,C	1	1	1	1
WDR		ウォッチドッグ タイマ リセット	(ウォッチドッグ タイマ記述を参照)	I,T,H,S,V,N,Z,C	1	1	1	1

注

1: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、NVMへのアクセスに対して有効ではありません。NVMをアクセスする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なデバイスのデータシートでNVMCTRL章をご覧ください。

3: LD命令がI/Oレジスタをアクセスする場合、1周期減ります。

4: デバイスのプログラミング時間で変わります。

5. 命令説明

5.1. ADC – キャリーを含めた汎用作業レジスタ間の加算 (Add with Carry)

5.1.1. 説明

2つの汎用作業レジスタ(Rd,Rr)とステータス レジスタ(SREG)のキャリー フラグ(C)の内容を加算し、結果をRdレジスタに配置。

動作

Rd ← Rd + Rr + C

書式		オペランド	プログラム カウンタ
ADC	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ビット)

0	0	0	1	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.1.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H – $Rd3 \cdot \overline{Rr3} + Rr3 \cdot \overline{R3} + \overline{R3} \cdot Rd3$
ビット3からのキャリー発生で設定(1)、その他で解除(0)。

S – $N \oplus V$
2の補数での符号。

V – $Rd7 \cdot \overline{Rr7} \cdot \overline{R7} + \overline{Rd7} \cdot Rr7 \cdot R7$
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N – R7
結果の最上位ビットの複写値。

Z – $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C – $Rd7 \cdot \overline{Rr7} + Rr7 \cdot \overline{R7} + \overline{R7} \cdot Rd7$
最上位ビットからのキャリー発生で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2バイト)

表5-1. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例: R3:R2 ← R3:R2 + R1:R0

```
ADD     R2, R0          ; 下位バイトの加算
ADC     R3, R1          ; 下位バイトからのキャリーを含めて上位バイトの加算
```

5.2. ADD - 汎用作業ﾚｼﾞｽﾀ間の加算 (Add without Carry)

5.2.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)なしで2つの汎用作業ﾚｼﾞｽﾀ(Rd,Rr)の内容を加算し、結果をRdﾚｼﾞｽﾀに配置。

動作

Rd ← Rd + Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
ADD	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	0	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.2.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H - $Rd3 \cdot \overline{Rr3} + Rr3 \cdot \overline{R3} + \overline{R3} \cdot Rd3$
ﾋﾞｯﾄ3からのｷｬﾘｰ発生で設定(1)、その他で解除(0)。

S - N XOR V
2の補数での符号。

V - $Rd7 \cdot \overline{Rr7} \cdot \overline{R7} + \overline{Rd7} \cdot Rr7 \cdot R7$
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C - $Rd7 \cdot \overline{Rr7} + Rr7 \cdot \overline{R7} + \overline{R7} \cdot Rd7$
最上位ﾋﾞｯﾄからのｷｬﾘｰ発生で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-2. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R1, R2          ; R1とR2の加算(R1=R1+R2)
        ADD     R28, R28        ; R28自身の加算(R28=R28+R28)
```

5.3. ADIW – 汎用作業ﾚｼﾞｽﾀ対に即値を加算 (Add Immediate to Word)

5.3.1. 説明

汎用作業ﾚｼﾞｽﾀ対(R[d+1]:Rd)の内容と即値定数(0～63)を加算し、結果をR[d+1]:Rdﾚｼﾞｽﾀに配置。この命令は上位4つの汎用作業ﾚｼﾞｽﾀ対で作動し、ﾎﾟｲﾝﾀ ﾚｼﾞｽﾀでの操作に上手く適合します。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

R[d+1]:Rd ← R[d+1]:Rd + K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
ADIW	Rd,K	d={24,26,28,30}, K=0～63	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	1	0	K5	K4	d2	d1	K3	K2	K1	K0

5.3.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

S – N XOR V
2の補数での符号。

V – $\overline{\text{Rdh7}}$ AND R15
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N – R15
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{\text{R15}}$ AND $\overline{\text{R14}}$ AND $\overline{\text{R13}}$ AND ・・・ $\overline{\text{R1}}$ AND $\overline{\text{R0}}$
結果が$0000で設定(1)、その他で解除(0)。

C – $\overline{\text{R15}}$ AND Rdh7
最上位ﾋﾞｯﾄからのｷｬﾘｰ発生で設定(1)、その他で解除(0)。

R(結果)は演算後のR[d+1]:Rdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-3. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

例:

```
        ADIW    R24, 1          ; R25:R24を+1
        ADIW    ZL, 63          ; R31:R30(Zﾚｼﾞｽﾀ)を+63
```

5.4. AND – 汎用作業ﾚｼﾞｽﾀ間の論理積 (Logical And)

5.4.1. 説明

2つの汎用作業ﾚｼﾞｽﾀ(Rd,Rr)の内容間で論理積(AND)を実行し、結果をRdﾚｼﾞｽﾀに配置。

動作

Rd ← Rd AND Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
AND	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	1	0	0	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.4.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

S – N XOR V
2の補数での符号。

V – 0
解除(0)。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-4. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        AND     R2, R3          ; R2とR3のﾋﾞｯﾄ単位の論理積取得
        LDI     R16, 1          ; R16にﾋﾞｯﾄ遮蔽値(0000 0001)を取得
        AND     R2, R16         ; ﾋﾞｯﾄ0値をR2に抽出
```

5.5. ANDI – 汎用作業ﾚｼﾞｽﾀと即値の論理積 (Logical AND with Immediate)

5.5.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の内容と即値定数(K)の論理積(AND)を実行し、結果を汎用作業ﾚｼﾞｽﾀ(Rd)に配置。

動作

Rd ← Rd AND K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
ANDI	Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	1	1	1	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0

5.5.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

S – N XOR V
2の補数での符号。

V – 0
解除(0)。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-5. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

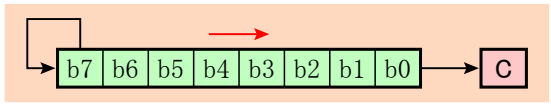
```
        ANDI    R17, $0F        ; 下位ﾆﾌﾞﾙ(4ﾋﾞｯﾄ)のみ有効
        ANDI    R18, $10        ; ﾋﾞｯﾄ4のみ有効
        ANDI    R19, $AA        ; 奇数ﾋﾞｯﾄのみ有効
```

5.6. ASR – 汎用作業ﾚｼﾞｽﾀの算術的右移動(ｼﾌﾄ) (Arithmetic Shift Right)

5.6.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の全ﾋﾞｯﾄの場所を1つ右移動、ﾋﾞｯﾄ7は一定を保たれます。ﾋﾞｯﾄ0はｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)に置かれます。この動作は符号を変えずに符号付きの値を効率的に2で除算します。この場合、ｷｬﾘｰ ﾌﾗｸﾞは結果の丸めに使うことができます。

動作



書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
ASR	Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	1	0	1

5.6.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

S – N XOR V
2の補数での符号。

V – N XOR C (移動後の)

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C – Rd0
移動前のRdの最下位ﾋﾞｯﾄが設定(1)ならば設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-6. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        LDI     R16, $10        ; R16に定数16を設定
        ASR     R16             ; R16＝R16÷2
        LDI     R17, $FC        ; R17に定数-4を設定
        ASR     R17             ; R17＝R17÷2
```

5.7. BCLR - ステータス レジスタ(SREG)のビットを解除(0) (Bit Clear in SREG)

5.7.1. 説明

ステータス レジスタ(SREG)の単一フラグ(ビット)を解除(0)。

動作

SREG(s) ← 0

書式	オペランド	プログラム カウンタ
BCLR s	s=0～7	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	1	0	0	1	S2	S1	S0	1	0	0	0

5.7.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔

I - s=7ならば0、その他で変化なし。
T - s=6ならば0、その他で変化なし。
H - s=5ならば0、その他で変化なし。
S - s=4ならば0、その他で変化なし。
V - s=3ならば0、その他で変化なし。
N - s=2ならば0、その他で変化なし。
Z - s=1ならば0、その他で変化なし。
C - s=0ならば0、その他で変化なし。

命令語数 1 (2バイト)

表5-7. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        BCLR    0               ; キャリー フラグを解除
        BCLR    7               ; 全体割り込み禁止
```

5.8. BLD – SREGのTﾋﾞｯﾄを汎用作業ﾚｼﾞｽﾀのﾋﾞｯﾄに取得 (Bit Load from the T Bit in SREG to a Bit in Register)

5.8.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のT(一時)ﾋﾞｯﾄを汎用作業ﾚｼﾞｽﾀ(Rd)のﾋﾞｯﾄbに複写。

動作

Rd(b) ← T

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
BLD	Rd,b	d=0～31, b=0～7	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	1	0	0	d4	d3	d2	d1	d0	0	b2	b1	b0

5.8.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-8. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例: ﾋﾞｯﾄ複写

```
        BST     R1, 2           ; R1のﾋﾞｯﾄ2をTﾋﾞｯﾄに複写
        BLD     R0, 4           ; TﾋﾞｯﾄをR0のﾋﾞｯﾄ4に複写
```

5.9. BRBC – ステータス レジスタ(SREG)のビット=0で分岐 (Branch if Bit in SREG is Cleared)

5.9.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)の単一ビットを調べ、そのビットが解除(0)されている場合にPC相対分岐。この命令は(PC−63 ≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。

動作

SREG(s)=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式	オペランド	プログラム カウンタ	
BRBC s,k	s=0～7, k=−64～+63	PC ← PC + k + 1	(条件成立時)
		PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	S2	S1	S0

5.9.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-9. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        CPI     R20, 5          ; R20=5か検査
        BRBC    1, NOTEQ        ; 5以外(Z=0)で分岐
                ≀
NOTEQ:  NOP                     ; 処理なし(5以外での分岐先)
```

5.10. BRBS - ステータス レジスタ(SREG)のビット=1で分岐 (Branch if Bit in SREG is Set)

5.10.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)の単一ビットを調べ、そのビットが設定(1)されている場合にPC相対分岐。この命令は(PC-63 ≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。

動作

SREG(s)=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ	
BRBS	s,k	s=0～7, k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	s2	s1	s0

5.10.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-10. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        BST     R0, 3           ; R0のビット3をT(一時)ビットに複写
        BRBS    6, BITSET       ; Tビット=1(R0のビット3=1)で分岐
                ～
BITSET: NOP                     ; 処理なし(R0のビット3=1の分岐先)
```

5.11. BRCC – キャリーなしで分岐 (Branch if Carry Cleared)

5.11.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)のキャリー フラグ(C)を調べ、Cが解除(0)されている場合にPC相対分岐。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。(BRBC 0,k命令、BRSH命令と等価)

動作

C=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式	オペランド	プログラム カウンタ	
BRCC k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
		PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	0	0

5.11.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5−11. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
            ADD     R22, R23        ; R22とR23を加算
            BRCC    NOCARRY         ; 桁溢れなし(C=0)で分岐
                    ～
NOCARRY:    NOP                     ; 処理なし(桁溢れなしでの分岐先)
```

5.12. BRCS - キャリーありで分岐 (Branch if Carry Set)

5.12.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)のキャリー フラグ(C)を調べ、Cが設定(1)されている場合にPC相対分岐。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。(BRBS 0,k命令、BRLO命令と等価)

動作

C=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ	
BRCS	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	0	0

5.12.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-12. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        CPI     R26, $56        ; R26と$56を比較検査
        BRBC    CARRY           ; R26＜$56(C=1)で分岐(符号なし)
                ≀
CARRY:  NOP                     ; 処理なし(R26＜$56での分岐先)
```

5.13. BREAK – 内蔵ﾃﾞﾊﾞｯｸﾞ機能用動作停止 (Break)

5.13.1. 説明

BREAK命令はﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ ｼｽﾃﾑによって使われ、応用ｿﾌﾄｳｪｱによって使われません。BREAK命令が実行されると、AVR CPUは停止状態に設定されます。これは内部資源に対してﾁｯﾌﾟ上ﾃﾞﾊﾞｯｶﾞ ｱｸｾｽを与えます。

ﾃﾞﾊﾞｲｽが施錠されるか、またはﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ ｼｽﾃﾑが許可されない場合、CPUはBREAK命令をNOP命令として扱い、停止状態に移行しません。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

内蔵ﾃﾞﾊﾞｯｸﾞ機能用に動作停止

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
BREAK	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0

5.13.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-13. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

5.14. BREQ – 一致で分岐 (Branch if Equal)

5.14.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｾﾞﾛ ﾌﾗｸﾞ(Z)を調べ、Zが設定(1)されている場合にPC相対分岐。CP, CPI, SUB, SUBI命令のどれかの直後にこの命令が実行される場合、Rdで表された符号付きまたは符号なし2進数がRrで表された符号付きまたは符号なし2進数に等しかった場合にだけ分岐が起きます。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBS 1,k命令と等価)

動作

Z=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BREQ	k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	0	1

5.14.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-14. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        CP      R1, R0          ; R1とR0を比較検査
        BREQ    EQUAL           ; 同一値(Z=1)で分岐
                ⌇
EQUAL:  NOP                     ; 処理なし(同一値での分岐先)
```

5.15. BRGE - 符号付きの≧で分岐 (Branch if Greator or Equal (Signed))

5.15.1. 説明

PC相対条件分岐です。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の符号ﾌﾗｸﾞ(S)を調べ、Sが解除(0)されている場合にPC相対分岐。CP, CPI, SUB, SUBI命令のどれかの直後にこの命令が実行される場合、Rdで表された符号付き2進数がRrで表された符号付き2進数以上だった場合にだけ分岐が起きます。。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBC 4,k命令と等価)

動作

S=0(Rd≧Rr)なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRGE	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	0	0

5.15.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-15. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        CP      R11, R12        ; R11とR12を比較検査
        BRGE    GREATEQ         ; R11≧R12(S=0)で分岐
                ⟨
GREATEQ:NOP                     ; 処理なし(R11≧R12での分岐先)
```

5.16. BRHC - ハーフキャリーなしで分岐 (Branch if Half Carry Flag is Cleared)

5.16.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)のハーフキャリー フラグ(H)を調べ、Hが解除(0)されている場合にPC相対分岐。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。(BRBC 5,k命令と等価)

動作

H=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ	
BRHC	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	0	1

5.16.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-16. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        ADD     R0, R1          ; R0とR1を加算
        BRHC    NOHC            ; 16未満(H=0)で分岐
                ≀
NOHC:   NOP                     ; 処理なし(16未満での分岐先)
```

5.17. BRHS - ハーフキャリーありで分岐 (Branch if Half Carry Flag is Set)

5.17.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)のハーフキャリー フラグ(H)を調べ、Hが設定(1)されている場合にPC相対分岐。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。(BRBS 5,k命令と等価)

動作

H=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ	
BRHS	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	0	1

5.17.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-17. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        ADD     R0, R1          ; R0とR1を加算
        BRHS    HCARRY          ; 16以上(H=1)で分岐
                ≀
HCARRY: NOP                     ; 処理なし(16以上での分岐先)
```

5.18. BRID – 全体割り込み禁止で分岐 (Branch if Global Interrup is Disabled)

5.18.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の全体割り込み許可ﾌﾗｸﾞ(I)を調べ、Iが解除(0)されている場合にPC相対分岐。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBC 7,k命令と等価)

動作

I=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRID	k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	1	1

5.18.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-18. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        BRID    INTDIS          ; 全体割り込み禁止(I=0)で分岐
                ≀
INTDIS: NOP                     ; 処理なし(全体割り込み禁止での分岐先)
```

5.19. BRIE – 全体割り込み許可で分岐 (Branch if Global Interrupt is Enabled)

5.19.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の全体割り込み許可ﾋﾞｯﾄ(I)を調べ、Iが設定(1)されている場合にPC相対分岐。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBS 7,k命令と等価)

動作

I=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRIE	k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	1	1

5.19.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-19. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        BRIE    INTEN           ; 全体割り込み許可(I=1)で分岐
        ～
INTEN:  NOP                     ; 処理なし(全体割り込み許可での分岐先)
```

5.20. BRLO - 符号なしのくで分岐 (Branch if Lower (Unsigned))

5.20.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)を調べ、Cが設定(1)されている場合にPC相対分岐。CP, CPI, SUB, SUBI命令のどれかの直後にこの命令が実行される場合、Rdで表された符号なし2進数がRrで表された符号なし2進数未満の場合にだけ分岐が起きます。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBS 0,k命令、BRCS命令と等価)

動作

C=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRLO	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	0	0

5.20.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-20. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
            EOR     R19, R19        ; R19を初期化(=0)
LOOP:       INC     R19             ; 計数器進行(+1)
                ≀
            CPI     R19, 50         ; 50回経過か検査
            BRLO    LOOP            ; 50回未満(C=1)で分岐
;
            NOP                     ; 処理なし(繰り返し後の処理)
```

5.21. BRLT – 符号付きの＜で分岐 (Branch if Less Than (Signed))

5.21.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の符号ﾌﾗｸﾞ(S)を調べ、Sが設定(1)されている場合にPC相対分岐。CP, CPI, SUB, SUBI命令のどれかの直後にこの命令が実行される場合、Rdで表された符号付き2進数がRrで表された符号付き2進数未満だった場合にだけ分岐が起きます。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBS 4,k命令と等価)

動作

S=1(Rd＜Rr)なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRLT	k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	0	0

5.21.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-21. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        CP      R26, R1         ; R16とR1を比較検査
        BRLT    LESS            ; R16＜R1(S=1)で分岐
                ≀
LESS:   NOP                     ; 処理なし(R16＜R1での分岐先)
```

5.22. BRMI – マイナス(–)で分岐 (Branch if Minus)

5.22.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)の負フラグ(N)を調べ、Nが設定(1)されている場合にPC相対分岐。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。(BRBS 2,k命令と等価)

動作

N=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ	
BRMI	k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	1	0

5.22.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語(ワード)数 1 (2バイト)

表5-22. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        SUBI    R18, 4          ; R18から4を減算
        BRMI    MINUS           ; 負(N=1)で分岐
                ≀
MINUS:  NOP                     ; 処理なし(負での分岐先)
```

5.23. BRNE – 不一致で分岐 (Branch if Not Equal)

5.23.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)のゼロ フラグ(Z)を調べ、Zが解除(0)されている場合にPC相対分岐。CP, CPI, SUB, SUBI命令のどれかの直後にこの命令が実行される場合、Rdで表された符号付きまたは符号なし2進数がRrで表された符号付きまたは符号なし2進数に等しくなかった場合にだけ分岐が起きます。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。(BRBC 1,k命令と等価)

動作

Z=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ	
BRNE	k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	0	1

5.23.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-23. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
            EOR     R27, R27        ; R27を初期化(=0)
LOOP:       INC     R27             ; 計数器進行(+1)
              ≀
            CPI     R27, 5          ; 5回目か検査
            BRNE    LOOP            ; 5回目以外(Z=0)で分岐
;
            NOP                     ; 処理なし(5回目の処理)
```

5.24. BRPL – ﾌﾟﾗｽ(+)で分岐 (Branch if Plus)

5.24.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の負ﾌﾗｸﾞ(N)を調べ、Nが解除(0)されている場合にPC相対分岐。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBC 2,k命令と等価)

動作

N=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRPL	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	1	0

5.24.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-24. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        SUBI    R26, 50         ; R26から50を減算
        BRPL    PLUS            ; 正(N=0)で分岐
                ≀
PLUS:   NOP                     ; 処理なし(正での分岐先)
```

5.25. BRSH - 符号なしの≧で分岐 (Branch if Same or Higher (Unsigned))

5.25.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)を調べ、Cが解除(0)されている場合にPC相対分岐。CP, CPI, SUB, SUBI命令のどれかの直後にこの命令が実行される場合、Rdで表された符号なし2進数がRrで表された符号なし2進数以上だった場合にだけ分岐が起きます。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBC 0,k命令、BRCC命令と等価)

動作

C=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRSH	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	0	0

5.25.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-25. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        SUBI    R19, 4          ; R19から4を減算
        BRSH    HIGHEQ          ; 4以上(C=0)で分岐
                ｝
HIGHEQ: NOP                     ; 処理なし(4以上での分岐先)
```

5.26. BRTC - Tﾋﾞｯﾄ=0で分岐 (Branch if T Bit is Cleared)

5.26.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の一時ﾋﾞｯﾄ(T)を調べ、Tが解除(0)されている場合にPC相対分岐。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBC 6,k命令と等価)

動作

T=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRTC	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	1	0

5.26.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-26. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        BST     R1, 5           ; R1のﾋﾞｯﾄ5をT(一時)ﾋﾞｯﾄに複写
        BRTC    TCLEAR          ; R1のﾋﾞｯﾄ5=0(T=0)で分岐
            ≀
TCLEAR: NOP                     ; 処理なし(R1のﾋﾞｯﾄ5=0での分岐先)
```

5.27. BRTS – Tﾋﾞｯﾄ=1で分岐 (Branch if the T Bit is Set)

5.27.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の一時ﾋﾞｯﾄ(T)を調べ、Tが設定(1)されている場合にPC相対分岐。この命令は(PC−63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBS 6,k命令と等価)

動作

T=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRTS	k	k=−64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	1	0

5.27.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-27. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        BST     R3, 5           ; R3のﾋﾞｯﾄ5をTﾋﾞｯﾄに複写
        BRTS    TSET            ; R3のﾋﾞｯﾄ5=1(T=1)で分岐
            ≀
TSET:   NOP                     ; 処理なし(R3のﾋﾞｯﾄ5=1での分岐先)
```

5.28. BRVC - 2の補数溢れなしで分岐 (Branch if Overflow Cleared)

5.28.1. 説明

PC相対条件分岐。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の2の補数溢れﾌﾗｸﾞ(V)を調べ、Vが解除(0)されている場合にPC相対分岐。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。ｵﾍﾟﾗﾝﾄﾞのkはPCからのｵﾌｾｯﾄ(差分)で2の補数形式で表されます。(BRBC 3,k命令と等価)

動作

V=0なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	
BRVC	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	1	1

5.28.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-28. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
            ADD     R3, R4          ; R3とR4を加算
            BRVC    NOOVER          ; 溢れなし(V=0)で分岐
                ≀
NOOVER:     NOP                     ; 処理なし(溢れなしでの分岐先)
```

5.29. BRVS - 2の補数溢れありで分岐 (Branch if Overflow Set)

5.29.1. 説明

PC相対条件分岐。ステータス レジスタ(SREG)の2の補数溢れフラグ(V)を調べ、Vが設定(1)されている場合にPC相対分岐。この命令は(PC-63≦分岐先≦PC+64)のどちらかの方向にPC相対で分岐します。オペランドのkはPCからのオフセット(差分)で2の補数形式で表されます。(BRBS 3,k命令と等価)

動作

V=1なら、PC ← PC + k + 1, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ	
BRVS	k	k=-64～+63	PC ← PC + k + 1	(条件成立時)
			PC ← PC + 1	(条件不成立時)

機械語 (16ビット)

1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	1	1

5.29.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-29. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立	1	1	1	1
	条件成立	2	2	2	2

例:

```
        ADD     R3, R4          ; R3とR4を加算
        BRVS    OVER            ; 溢れ(V=1)で分岐
                ≀
OVER:   NOP                     ; 処理なし(溢れでの分岐先)
```

5.30. BSET – ステータス レジスタ(SREG)のビットを設定(1) (Bit Set in SREG)

5.30.1. 説明

ステータス レジスタ(SREG)の単一のフラグまたはビットを設定(1)。

動作

SREG(s) ← 1

書式	オペランド	プログラム カウンタ
BSET s	s=0～7	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	1	0	0	0	S2	S1	S0	1	0	0	0

5.30.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
X	X	X	X	X	X	X	X

I – s=7ならば1、その他で変化なし。
T – s=6ならば1、その他で変化なし。
H – s=5ならば1、その他で変化なし。
S – s=4ならば1、その他で変化なし。
V – s=3ならば1、その他で変化なし。
N – s=2ならば1、その他で変化なし。
Z – s=1ならば1、その他で変化なし。
C – s=0ならば1、その他で変化なし。

命令語数 1 (2バイト)

表5-30. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        BSET    6               ; T(一時)ビットを設定(1)
        BSET    7               ; 全体割り込み許可
```

5.31. BST – 汎用作業ﾚｼﾞｽﾀのﾋﾞｯﾄをSREGのTﾋﾞｯﾄに設定 (Bit Store from Bit in Register to T Bit in SREG)

5.31.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)のﾋﾞｯﾄbをｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の一時ﾋﾞｯﾄ(T)に格納。

動作

T ← Rd(b)

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
BST	Rd,b	d=0～31, b=0～7	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	1	0	1	d4	d3	d2	d1	d0	0	b2	b1	b0

5.31.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

T – Rdの対応ﾋﾞｯﾄ=0で0、その他で1。

命令語数 1 (2ﾊﾞｲﾄ)

表5-31. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例: ﾋﾞｯﾄ複写

```
        BST     R1, 2           ; R1のﾋﾞｯﾄ2をTﾋﾞｯﾄに複写
        BLD     R0, 4           ; TﾋﾞｯﾄをR0のﾋﾞｯﾄ4に複写
```

5.32. CALL - 絶対ｱﾄﾞﾚｽ指定によるｻﾌﾞﾙｰﾁﾝ呼び出し (Long Call to a Subroutine)

5.32.1. 説明

ﾌﾟﾛｸﾞﾗﾑ空間全体でｻﾌﾞﾙｰﾁﾝ呼び出し。(CALL命令後の命令への)復帰ｱﾄﾞﾚｽがｽﾀｯｸ上に格納されます(RCALL命令もご覧ください)。ｽﾀｯｸ ﾎﾟｲﾝﾀ(SP)はCALL中に事後減少機構を使います。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作	注釈
PC ← k	① 16ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは64K語(128Kﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。 ② 22ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは4M語(8Mﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ
CALL	k	① 0≦k＜64K	PC ← k	STACK ← PC + 2 SP ← SP - 2 (2ﾊﾞｲﾄ,16ﾋﾞｯﾄ)
		② 0≦k＜4M	PC ← k	STACK ← PC + 2 SP ← SP - 3 (3ﾊﾞｲﾄ,22ﾋﾞｯﾄ)

機械語 (32ﾋﾞｯﾄ)

1	0	0	1	0	1	0	k21	k20	k19	k18	k17	1	1	1	k16
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0

5.32.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数　2 (4ﾊﾞｲﾄ)

表5-32. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	16ﾋﾞｯﾄPC	4 (注)	3 (注)	3	利用不可
	22ﾋﾞｯﾄPC	5 (注)	4 (注)	4	利用不可

注: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

例:

```
            MOV     R16, R0         ; R0をR16に複写
            CALL    CHECK           ; 検査(ｻﾌﾞﾙｰﾁﾝ呼び出し)
            NOP                     ; (継続)

CHECK:      CPI     R16, $41        ; 値検査
            BREQ    ERROR           ; 一致で分岐
            RET                     ; 不一致で呼び出し元へ復帰

ERROR:      RJMP    ERROR           ; 無限繰り返し
```

5.33. CBI - I/Oレジスタのビット解除(0) (Clear Bit in I/O Register)

5.33.1. 説明

I/Oレジスタで指定ビットを解除(0)。この命令はI/Oレジスタの下位32バイト(I/Oアドレス0～31)で動きます。

動作

I/O(A,b) ← 0

書式		オペランド	プログラム カウンタ
CBI	A,b	A=0～31, b=0～7	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	1	0	0	0	A4	A3	A2	A1	A0	b2	b1	b0

5.33.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-33. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	1	1	1

例:

```
        CBI     $12, 7          ; アドレス$0012のビット7を解除(0)
```

5.34. CBR - 汎用作業ﾚｼﾞｽﾀの(複数)ﾋﾞｯﾄ解除(0) (Clear Bits in Register)

5.34.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)で指定した(複数)ﾋﾞｯﾄを解除(0)。Rdﾚｼﾞｽﾀの内容と定数遮蔽Kのﾋﾞｯﾄ反転値間で論理積(AND)を実行。(ANDI Rd,!K命令と等価)

動作

Rd ← Rd AND ($FF - K)

書式	オﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CBR Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	1	1	1	$\overline{K7}$	$\overline{K6}$	$\overline{K5}$	$\overline{K4}$	d3	d2	d1	d0	$\overline{K3}$	$\overline{K2}$	$\overline{K1}$	$\overline{K0}$

5.34.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S - N XOR V
2の補数での符号。

V - 0
解除(0)。

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-34. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        CBR     R16, $F0        ; 上位ﾆﾌﾞﾙ(4ﾋﾞｯﾄ)を解除
        CBR     R18, 1          ; R18のﾋﾞｯﾄ0を解除
```

5.35. CLC – ｷｬﾘｰ ﾌﾗｸﾞを解除(0) (Clear Carry Flag)

5.35.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)を解除(0)。(BCLR 0命令と等価)

動作

C ← 0

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLC	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0

5.35.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	0

C – 0
解除(0)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-35. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R0, R0          ; R0を2倍
        CLC                     ; ｷｬﾘｰ ﾌﾗｸﾞを解除
```

5.36. CLH - ハーフキャリー フラグを解除(0) (Clear Half Carry Flag)

5.36.1. 説明

ステータス レジスタ(SREG)のハーフキャリー フラグ(H)を解除(0)。(BCLR 5命令と等価)

動作

H ← 0

書式	オペランド	プログラム カウンタ
CLH	なし	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0

5.36.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	0	-	-	-	-	-

H - 0
解除(0)。

命令語数 1 (2バイト)

表5-36. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
CLH                     ; ハーフキャリー フラグを解除
```

5.37. CLI - 全体割り込み許可ﾌﾗｸﾞを解除(0) (Clear Global Interrupt Flag)

5.37.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の全体割り込み許可ﾌﾗｸﾞ(I)を解除(0)。割り込みが直ちに禁止されます。割り込みは例えCLI命令と同時に起きたとしても、CLI命令後に実行されません。(BCLR 7命令と等価)

動作

I ← 0

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLI	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	1	1	1	1	0	0	0

5.37.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
0	-	-	-	-	-	-	-

I - 0
解除(0)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-37. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        IN      TEMP, SREG      ; SREG値格納(TEMPは使用者によって定義されていなければなりません。)
        CLI                     ; 時間制限手順の間、割り込み禁止
        SBI     EECR, EEMWE     ; EEPROM書き込み開始
        SBI     EECR, EEWE
        OUT     SREG, TEMP      ; SREG値(Iﾌﾗｸﾞ)復元
```

5.38. CLN – 負ﾌﾗｸﾞを解除(0) (Clear Negative Flag)

5.38.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の負ﾌﾗｸﾞ(N)を解除(0)。(BCLR 2命令と等価)

動作

N ← 0

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLN	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	0

5.38.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	0	–	–

N – 0
解除(0)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-38. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        CLN                     ; 負ﾌﾗｸﾞを解除(0)
```

5.39. CLR - 汎用作業ﾚｼﾞｽﾀを解除($00) (Clear Register)

5.39.1. 説明

汎用作業ﾚｼﾞｽﾀを解除($00)。(EOR Rd,Rd命令と等価)

動作

Rd ← Rd XOR Rr

書式	オﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLR Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	1	0	0	1	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0

5.39.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	0	0	0	1	-

S - 0
解除(0)。

V - 0
解除(0)。

N - 0
解除(0)。

Z - 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-39. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        CLR     R18             ; R18に0を設定
LOOP:   INC     R18             ; 計数器進行(+1)
                ≀
        CPI     R18, 100        ; 100回経過か検査
        BRNE    LOOP            ; 100回未満で継続
```

5.40. CLS – 符号ﾌﾗｸﾞを解除(0) (Clear Signed Flag)

5.40.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の符号ﾌﾗｸﾞ(S)を解除(0)。(BCLR 4命令と等価)

動作

S ← 0

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLS	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	1	0	0	1	0	0	0

5.40.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	0	–	–	–	–

S – 0
解除(0)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-40. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        CLS                     ; 符号ﾌﾗｸﾞを解除
```

5.41. CLT – Tﾋﾞｯﾄを解除(0) (Clear T Bit)

5.41.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の一時ﾋﾞｯﾄ(T)を解除(0)。(BCLR 6命令と等価)

動作

T ← 0

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLT	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	1	1	0	1	0	0	0

5.41.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	0	–	–	–	–	–	–

T – 0
解除(0)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-41. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
CLT                    ; Tﾋﾞｯﾄを解除
```

5.42. CLV – 2の補数溢れﾌﾗｸﾞを解除(0) (Clear Overflow Flag)

5.42.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の2の補数溢れﾌﾗｸﾞ(V)を解除(0)。(BCLR 3命令と等価)

動作

V ← 0

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLV	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	0	1	1	1	0	0	0

5.42.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	0	–	–	–

V – 0
解除(0)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-42. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        CLV                     ; 溢れﾌﾗｸﾞを解除
```

5.43. CLZ – ｾﾞﾛ ﾌﾗｸﾞを解除(0) (Clear Zero Flag)

5.43.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｾﾞﾛ ﾌﾗｸﾞ(Z)を解除(0)。(BCLR 1命令と等価)

動作

Z ← 0

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CLZ	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0

5.43.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	0	–

Z – 0
解除(0)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-43. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        CLZ                     ; ｾﾞﾛ ﾌﾗｸﾞを解除
```

5.44. COM - 1の補数取得(ﾋﾞｯﾄ論理反転) (One's Complement)

5.44.1. 説明

この命令は汎用ﾚｼﾞｽﾀ(Rd)の1の補数を実行します(全ﾋﾞｯﾄ論理反転)。

動作

Rd ← $FF - Rd

書式	オペランド	プログラム カウンタ
COM Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	0	0

5.44.2. ステータス レジスタ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	1

S - N XOR V
2の補数での符号。

V - 0
解除(0)。

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C - 1
設定(1)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-44. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        COM     R4          ; R4=$FFか検査
        BREQ    MATCH       ; R4=$FF(Z=1)で分岐
                ≀
MATCH:  NOP                 ; 処理なし(R4=$FFでの分岐先)
```

5.45. CP – 汎用作業ﾚｼﾞｽﾀ間の比較 (Compare)

5.45.1. 説明

この命令は2つの汎用作業ﾚｼﾞｽﾀ(Rd,Rr)間の比較を実行します。どのﾚｼﾞｽﾀも変わりません。この命令後に全ての条件分岐を使うことができます。

動作

Rd – Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CP	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	1	0	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.45.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H – $\overline{Rd3}$ AND Rr3 OR Rr3 AND R3 OR R3 AND $\overline{Rd3}$
ﾋﾞｯﾄ3からの借入有りで設定(1)、その他で解除(0)。

S – N XOR V
2の補数での符号。

V – Rd7 AND $\overline{Rr7}$ AND $\overline{R7}$ OR $\overline{Rd7}$ AND Rr7 AND R7
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C – $\overline{Rd7}$ AND Rr7 OR Rr7 AND R7 OR R7 AND $\overline{Rd7}$
Rd＜Rrで設定(1)、その他で解除(0)。

演算後のR(結果)

命令語数 1 (2ﾊﾞｲﾄ)

表5-45. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            CP      R4, R19         ; R4とR19を比較
            BRNE    NOMATCH         ; 不一致(Z=0)で分岐
                    ～
NOMATCH:    NOP                     ; 処理なし(不一致での分岐先)
```

5.46. CPC - キャリーを含めた汎用作業レジスタ間の比較 (Compare with Carry)

5.46.1. 説明

この命令は2つの汎用作業レジスタ(Rd,Rr)間の比較を実行し、直前のキャリーも考慮に入れます。どのレジスタも変わりません。この命令後に全ての条件分岐を使うことができます。

動作

Rd - Rr - C

書式		オペランド	プログラム カウンタ
CPC	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ビット)

0	0	0	0	0	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.46.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H - $\overline{Rd3}$ AND Rr3 OR Rr3 AND R3 OR R3 AND $\overline{Rd3}$
ビット3からの借入有りで設定(1)、その他で解除(0)。

S - N XOR V
2の補数での符号。

V - Rd7 AND $\overline{Rr7}$ AND $\overline{R7}$ OR $\overline{Rd7}$ AND Rr7 AND R7
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N - R7
結果の最上位ビットの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$ AND Z
結果が$00で直前の状態を保持、その他で解除(0)。

C - $\overline{Rd7}$ AND Rr7 OR Rr7 AND R7 OR R7 AND $\overline{Rd7}$
Rd＜(Rr+C)で設定(1)、その他で解除(0)。

演算後のR(結果)

命令語数 1 (2バイト)

表5-46. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            CP      R2, R0          ; 下位バイトの比較
            CPC     R3, R1          ; 上位バイトの比較
            BRNE    NOMATCH         ; 不一致(Z=0)で分岐
               ⟨
NOMATCH:    NOP                     ; 処理なし(不一致での分岐先)
```

5.47. CPI – 汎用作業ﾚｼﾞｽﾀと即値定数の比較 (Compare with Immediate)

5.47.1. 説明

この命令は汎用作業ﾚｼﾞｽﾀ(Rd)の内容と即値定数(K)韓の比較を実行します。どのﾚｼﾞｽﾀも変わりません。この命令後に全ての条件分岐を使うことができます。

動作

Rd – K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CPI	Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	1	1	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0

5.47.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H – $\overline{Rd3}$ AND K3 OR K3 AND R3 OR R3 AND $\overline{Rd3}$
ﾋﾞｯﾄ3からの借入有りで設定(1)、その他で解除(0)。

S – N XOR V
2の補数での符号。

V – Rd7 AND $\overline{K7}$ AND $\overline{R7}$ OR $\overline{Rd7}$ AND K7 AND R7
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C – $\overline{Rd7}$ AND K7 OR K7 AND R7 OR R7 AND $\overline{Rd7}$
符号なしのRd＜Kで設定(1)、その他で解除(0)。

演算後のR(結果)

命令語数 1 (2ﾊﾞｲﾄ)

表5-47. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            CPI     R19, 3          ; R19=3か検査
            BRNE    NOMATCH         ; 3以外(Z=0)で分岐
                    ～
NOMATCH:    NOP                     ; 処理なし(3以外での分岐先)
```

5.48. CPSE - 汎用作業ﾚｼﾞｽﾀ間一致でｽｷｯﾌﾟ (Compare Skip if Equal)

5.48.1. 説明

この命令は2つの汎用作業ﾚｼﾞｽﾀ(Rd,Rr)間の比較を実行し、Rd=Rrの場合に次の命令をｽｷｯﾌﾟ(非実行に)します。

動作

Rd=Rrなら、PC ← PC + 2または3, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
CPSE	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1　(不一致) PC ← PC + 2　(一致、次が1語命令) PC ← PC + 3　(一致、次が2語命令)

機械語 (16ﾋﾞｯﾄ)

0	0	0	1	0	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.48.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数　1 (2ﾊﾞｲﾄ)

表5-48. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立(ｽｷｯﾌﾟなし)	1	1	1	1
	条件成立、次が1語命令	2	2	2	2
	条件成立、次が2語命令	3	3	3	利用不可

例:

```
        INC     R4          ; R4を進行(+1)
        CPSE    R4, R0      ; R4＝R0でｽｷｯﾌﾟ
        NEG     R4          ; R4≠R0でR4符号反転
        NOP                 ; 処理なし(処理継続)
```

5.49. DEC – 汎用作業ﾚｼﾞｽﾀを減少(−1) (Decrement)

5.49.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の内容から1を減算し、結果をRdﾚｼﾞｽﾀに配置。

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)はこの操作によって影響を及ぼされず、従って、倍精度演算の繰り返し計数器に使うことをDEC命令に許します。

符号なし値での操作時、BREQとBRNEの条件分岐だけが矛盾なく実行することを期待できます。2の補数値での操作時、全ての符号付き条件分岐が利用可能です。

動作

Rd ← Rd − 1

書式	オﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
DEC Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	1	0	1	0

5.49.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	–

S – N XOR V
2の補数での符号。

V – $\overline{R7}$ AND R6 AND R5 AND R4 AND R3 AND R2 AND R1 AND R0
2の補数での溢れ発生で設定(1)、その他で解除(0)。元のRdが$80の場合のみ2の補数での溢れが発生します。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-49. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        LDI     R17, 10         ; 計数器(R17)に初期値(10)を設定
LOOP:   ADD     R1, R2          ; R1にR2を加算
        DEC     R17             ; 計数器進行(-1)
        BRNE    LOOP            ; 10回まで継続
```

5.50. DES – データ暗号化規格 (Data Encryption Standard)

5.50.1. 説明

この単位部はDES反復を実行する、AVR CPUへの命令一式拡張です。64ビットのデータ塊(平文または暗号文)はR0～R7レジスタに配置され、そしてデータのLSBはR0のLSBに、データのMSBはR7のMSBに配置されます。(パリティ ビットを含む)完全な64ビット鍵は鍵のLSBがR8のLSB、鍵のMSBがR15のMSBで構成されるR8～R15レジスタに配置されます。1つのDES命令実行はDES演算法での1巡を実行します。正しい暗号文または平文にするためには16回実行されなければなりません。中間結果は各DES命令後でレジスタファイル(R0～R15)に格納されます。命令のオペランド(K)はどの周回が実行されるのかを決め、ハーフキャリー(H)フラグが暗号化または解読のどちらが実行されるのかを決めます。

DES演算法は"Specifications for the Data Encryption Standard"(Federal Information Processing Standards Publication 46)で記述されます。初期順列と反転初期順列が各反復で実行されるため、この実装での中間結果は規格と異なります。これは最終的な平文または暗号文に影響を及ぼさず、実行時間を減少します。

動作

H=0なら、暗号化1巡(R7～R0,R15～R8,K)
H=1なら、解読1巡(R7～R0,R15～R8,K)

書式	オペランド	プログラム カウンタ
DES K	K=0～15	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	1	0	0	K3	K2	K1	K0	1	0	1	1

5.50.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-50. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	利用不可	1/2	利用不可	利用不可

注: 非DES命令が先行するDES命令の場合に2周期、さもなければ1周期が必要とされます。

例:

```
        DES     0               ; 1巡目
        DES     1               ; 2巡目
            ⟩
        DES     14              ; 15巡目
        DES     15              ; 16巡目(完了)
```

5.51. EICALL – 拡張間接サブルーチン呼び出し (Extended Indirect Call to Subroutine)

5.51.1. 説明

I/O空間のEINDレジスタとレジスタ ファイルの(16ビット)Zレジスタによって指示されるサブルーチンの間接呼び出し。この命令は4M語プログラム空間全域への間接呼び出しを許します。ICALL命令もご覧ください。スタック ポインタはEICALL中に事後減少機構を使います。

この命令は全てのデバイスで利用可能な訳ではありません。「追補A」を参照してください。

動作

PC(21～16) ← EIND
PC(15～0) ← Z(15～0)

書式	オペランド	プログラム カウンタ	スタック
EICALL	なし	PC(21～16) ← EIND PC(15～0) ← Z(15～0)	STACK ← PC + 1 SP ← SP – 3 (3バイト,22ビット)

機械語 (16ビット)

1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1

5.51.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-51. 実行周期数 (注1)

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	4 (注2)	3 (注2)	3	利用不可

注1: この命令は22ビットPCを持つデバイスにだけ実装されます。

注2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

例:

```
        LDI     R16, $05        ; アドレス拡張バイトを取得
        OUT     EIND, R16       ; アドレス拡張バイトを間接ポインタに設定
        LDI     R30, $00        ; アドレス下位2バイトをZレジスタに設定
        LDI     R31, $10        ;
        EICALL                  ; $051000番地を呼び出し
```

5.52. EIJMP – 拡張間接無条件分岐 (Extended Indirect Jump)

5.52.1. 説明

I/O空間のEINDﾚｼﾞｽﾀとﾚｼﾞｽﾀ ﾌｧｲﾙの(16ﾋﾞｯﾄ)Zﾚｼﾞｽﾀによって指示されるｱﾄﾞﾚｽへの間接無条件分岐。この命令は4M語ﾌﾟﾛｸﾞﾗﾑ空間全域への間接無条件分岐を許します。IJMP命令もご覧ください。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

PC(21～16) ← EIND
PC(15～0) ← Z(15～0)

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
EIJMP	なし	PC(21～16) ← EIND PC(15～0) ← Z(15～0)

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1

5.52.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-52. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

例:

```
        LDI     R16, $05        ; ｱﾄﾞﾚｽ拡張ﾊﾞｲﾄを取得
        OUT     EIND, R16       ; ｱﾄﾞﾚｽ拡張ﾊﾞｲﾄを間接ﾎﾟｲﾝﾀに設定
        LDI     R30, $00        ; ｱﾄﾞﾚｽ下位2ﾊﾞｲﾄをZﾚｼﾞｽﾀに設定
        LDI     R31, $10        ;
        EIJMP                   ; $051000番地へ分岐
```

5.53. ELPM – 拡張間接によるﾌﾟﾛｸﾞﾗﾑ空間からの取得 (Extended Load Program Memory)

5.53.1. 説明

ZﾚｼﾞｽﾀとI/O空間内のRAMPZﾚｼﾞｽﾀによって指示される1ﾊﾞｲﾄを取得し、このﾊﾞｲﾄを転送先汎用作業ﾚｼﾞｽﾀ(Rd)に置きます。この命令は100%空間効率的な定数初期化や定数ﾃﾞｰﾀ取得が特徴です。ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘが16ﾋﾞｯﾄ語で構成される一方でZﾎﾟｲﾝﾀはﾊﾞｲﾄ ｱﾄﾞﾚｽを指示します。従って、Zﾎﾟｲﾝﾀの最下位ﾋﾞｯﾄは下位ﾊﾞｲﾄ(ZLSb=0)または上位ﾊﾞｲﾄ(ZLSb=1)のどちらかを選びます。この命令はﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ空間全体をｱﾄﾞﾚｽ指定することができます。Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀは操作によって無変化のまま、または増加のどちらかにすることができます。増加はRAMPZとZﾎﾟｲﾝﾀ ﾚｼﾞｽﾀの24ﾋﾞｯﾄ連結全体に適用されます。

自己ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ能力を持つﾃﾞﾊﾞｲｽはﾋｭｰｽﾞ ﾋﾞｯﾄや施錠ﾋﾞｯﾄの値を読み出すELPM命令を使うことができます。詳細な記述についてはﾃﾞﾊﾞｲｽの文書を参照してください。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

注: 右に示す記述での実行結果は不定となります。

```
ELPM   R30,Z+
ELPM   R31,Z+
```

動作	注釈
① R0 ← PS(RAMPZ:Z)	; R0暗黙指定
② Rd ← PS(RAMPZ:Z)	;
③ Rd ← PS(RAMPZ:Z), RAMPZ:Z ← RAMPZ:Z + 1	; 事後増加

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
① ELPM	なし(R0暗黙指定)	PC ← PC + 1
② ELPM Rd,Z	d=0～31	PC ← PC + 1
③ ELPM Rd,Z+	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	①
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	1	0	②
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	1	1	③

5.53.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2ﾊﾞｲﾄ)

表5-53. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	3	3	3	利用不可

例:

```
            LDI     ZL, BYTE3(TABLE<<1)     ; R30(ZL)にﾊﾞｲﾄ単位ｱﾄﾞﾚｽ拡張ﾊﾞｲﾄ値を取得
            OUT     RAMPZ, ZL               ; ｱﾄﾞﾚｽ拡張ﾊﾞｲﾄをRAMPZに設定
            LDI     ZH, BYTE2(TABLE<<1)     ; Zﾚｼﾞｽﾀ上位にﾊﾞｲﾄ単位ｱﾄﾞﾚｽ上位ﾊﾞｲﾄ値を設定
            LDI     ZL, BYTE1(TABLE<<1)     ; Zﾚｼﾞｽﾀ下位にﾊﾞｲﾄ単位ｱﾄﾞﾚｽ下位ﾊﾞｲﾄ値と下位ﾊﾞｲﾄ指定を設定
            ELPM    R16, Z+                 ; TABLEの下位ﾊﾞｲﾄをR16に取得後ﾎﾟｲﾝﾀを進行
                    ⟨
;
TABLE:      .DW     $3738                   ; ZﾚｼﾞｽﾀのLSB=0の時に$38、LSB=1の時に$37がｱﾄﾞﾚｽ指定されます。
```

5.54. EOR – 汎用作業ﾚｼﾞｽﾀ間の排他的論理和 (Exclusive OR)

5.54.1. 説明

2つの汎用作業ﾚｼﾞｽﾀ(Rd,Rr)の内容間で排他的論理和(XOR)を実行し、結果をRdﾚｼﾞｽﾀに置きます。

動作

Rd ← Rd XOR Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
EOR	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	1	0	0	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.54.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

S – N XOR V
2の補数での符号。

V – 0
解除(0)。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-54. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

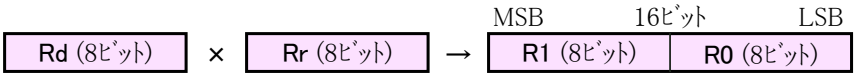
例:

```
        EOR     R1, R1          ; R1を解除($00)
        EOR     R2, R8          ; R2とR8のﾋﾞｯﾄ単位の排他的論理和取得
```

5.55. FMUL – 符号なし小数乗算 (Fractional Multiply Unsigned)

5.55.1. 説明

この命令は8ビット×8ビット=16ビットの符号なし乗算を実行し、結果を1ビット左移動します。



整数部の2進値をN、小数部の2進値をQとして、小数点数を(N.Q)と表します。2つの数値形式(N1.Q1)と(N2.Q2)の乗算結果は((N1+N2).(Q1+Q2))の形式になります。信号処理系の応用では入力が(1.7)形式で結果が(2.14)形式になり、入力と同じ形式にするために左移動が必要になります。FMUL命令はMUL命令と同一の実行時間でこの移動操作も行います。

(1.7)形式は最も一般的な符号付き数値の形式です。FMULは符号なし乗算を行います。FMULは16ビットの(1.15)形式の符号付き乗算で結果が(1.31)形式になる場合の一部分の演算によく使われます。

注: FMUL操作の結果は(1.15)形式の数として解釈される場合に2の補数溢れに悩まされるかもしれません。移動前の乗算のMSbが考慮されなければならず、これはキャリー フラグで見つかります。以降の例をご覧ください。

RdとRrはビット7と6間が小数点となる符号なし小数点数です。結果はR1(上位)とR0(下位)が連結した16ビットで、ビット15と14間が小数点となる符号なし小数点数です。

この命令は全てのデバイスで利用可能な訳ではありません。「追補A」を参照してください。

動作

R1:R0 ← Rd × Rr　(符号なし(1.15) ← 符号なし(1.7) × 符号なし(1.7))

書式	オペランド	プログラム カウンタ
FMUL Rd,Rr	d=16～23, r=16～23	PC ← PC + 1

機械語 (16ビット)

0	0	0	0	0	0	1	1	0	d2	d1	d0	1	r2	r1	r0

5.55.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

Z – $\overline{R15}$ AND $\overline{R14}$ AND $\overline{R13}$ AND ・・・ $\overline{R1}$ AND $\overline{R0}$
結果が$0000で設定(1)、その他で解除(0)。

C – R16
移動前の結果のビット15=1で設定(1)、その他で解除(0)。

R(結果)は演算後のR1:R0と等しくなります。

命令語数　1 (2バイト)

表5-55. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

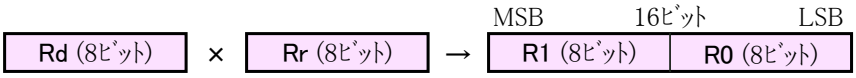
例: 符号付き小数(16×16＝32ビット)　R19:R18×R17:R16＝(R23:R22×R21:R20)<<1

```
        CLR     R2              ; R2を0に設定
        FMULS   R23, R21        ; 符号付き非乗数上位×符号付き乗数上位
        MOVW    R18, R0         ; 3,4バイト目の結果を取得
        FMUL    R22, R20        ; 符号なし非乗数下位×符号なし乗数下位
        ADC     R18, R2         ; 2バイト目からのキャリー補正
        MOVW    R16, R0         ; 1,2バイト目の結果を取得
        FMULSU  R23, R20        ; 符号付き非乗数上位×符号なし乗数下位
        SBC     R19, R2         ; 3バイト目からのキャリー補正
        ADD     R17, R0         ; 2バイト目の結果取得
        ADC     R18, R1         ; 3バイト目の結果取得
        ADC     R19, R2         ; 4バイト目の結果取得
        FMULSU  R21, R22        ; 符号なし非乗数下位×符号付き乗数上位
        SBC     R19, R2         ; 3バイト目からのキャリー補正
        ADD     R17, R0         ; 2バイト目の結果取得
        ADC     R18, R1         ; 3バイト目の結果取得
        ADC     R19, R2         ; 4バイト目の結果取得
```

5.56. FMULS – 符号付き小数乗算 (Fractional Multiply Signed)

5.56.1. 説明

この命令は8ビット×8ビット=16ビットの符号付き乗算をを実行し、結果を1ビット左移動します。



整数部の2進値をN、小数部の2進値をQとして、小数点数を(N.Q)と表します。2つの数値形式(N1.Q1)と(N2.Q2)の乗算結果は((N1+N2).(Q1+Q2))の形式になります。信号処理系の応用では入力が(1.7)形式で結果が(2.14)形式になり、入力と同じ形式にするために左移動が必要になります。FMULS命令はMULS命令と同一の実行時間でこの移動操作も行います。

RdとRrはビット7と6間が小数点となる符号付き小数点数です。結果はR1(上位)とR0(下位)が連結した16ビットで、ビット15と14間が小数点となる符号付き小数点数です。

注: -1($80)と-1($80)の乗算時、移動操作の結果は-1($8000)です。従って移動操作は2の補数溢れを与えます。これはソフトウェアによって調べて処理されなければなりません。

この命令は全てのデバイスで利用可能な訳ではありません。「追補A」を参照してください。

動作

R1:R0 ← Rd × Rr　(符号付き(1.15) ← 符号付き(1.7) × 符号付き(1.7))

書式	オペランド	プログラム カウンタ
FMULS　Rd,Rr	d=16～23, r=16～23	PC ← PC + 1

機械語 (16ビット)

0	0	0	0	0	0	1	1	1	d2	d1	d0	0	r2	r1	r0

5.56.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

Z – $\overline{R15}$ AND $\overline{R14}$ AND $\overline{R13}$ AND ・・・ $\overline{R1}$ AND $\overline{R0}$
結果が$0000で設定(1)、その他で解除(0)。

C – R16
移動前の結果のビット15=1で設定(1)、その他で解除(0)。

R(結果)は演算後のR1:R0と等しくなります。

命令語数　1 (2バイト)

表5-56. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

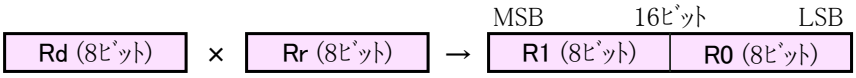
例:

```
        FMULS   R23, R22        ; R23とR22の符号付き乗算
        MOVW    R22, R0         ; 結果をR23:R22に取得
```

5.57. FMULSU - 符号付きと符号なしの小数乗算 (Fractional Multiply Signed with Unsigned)

5.57.1. 説明

符号付き8ﾋﾞｯﾄ×符号なし8ﾋﾞｯﾄ＝符号付き16ﾋﾞｯﾄの乗算を行い、結果を1ﾋﾞｯﾄ左移動(ｼﾌﾄ)します。



整数部の2進値をN、小数部の2進値をQとして、小数点数を(N.Q)と表します。2つの数値形式(N1.Q1)と(N2.Q2)の乗算結果は((N1+N2).(Q1+Q2))の形式になります。信号処理系の応用では入力が(1.7)形式で結果が(2.14)形式になり、入力と同じ形式にするために左移動が必要になります。FMULSU命令はMULSU命令と同一の実行時間でこの移動操作も行います。

(1.7)形式は最も一般的な符号付き数値の形式です。FMULSUは符号付きと符号なしの乗算を行います。FMULSUは16ﾋﾞｯﾄの(1.15)形式の符号付き乗算で結果が(1.31)形式になる場合の一部分の演算によく使われます。

注: FMULSU操作の結果は(1.15)形式の数として解釈される場合に2の補数溢れに悩まされるかもしれません。移動前の乗算のMSbが考慮されなければならず、これはｷｬﾘｰ ﾌﾗｸﾞで見つかります。以降の例をご覧ください。

Rdはﾋﾞｯﾄ7と6間が小数点となる符号付き小数点数で、同様にRrは符号なし小数点数です。結果はR1(上位)とR0(下位)が連結した16ﾋﾞｯﾄで、ﾋﾞｯﾄ15と14間が小数点となる符号なし小数点数です。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

R1:R0 ← Rd × Rr　(符号付き(1.15) ← 符号付き(1.7) × 符号なし(1.7))

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
FMULSU Rd,Rr	d=16～23, r=16～23	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	0	0	0	1	1	1	d2	d1	d0	1	r2	r1	r0

5.57.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	⇔	⇔

Z - $\overline{R15}$ AND $\overline{R14}$ AND $\overline{R13}$ AND ･･･ $\overline{R1}$ AND $\overline{R0}$
結果が$0000で設定(1)、その他で解除(0)。

C - R16
移動前の結果のﾋﾞｯﾄ15=1で設定(1)、その他で解除(0)。

R(結果)は演算後のR1:R0と等しくなります。

命令語数　1 (2ﾊﾞｲﾄ)

表5-57. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

例: 符号付き小数(16×16＝32ﾋﾞｯﾄ)　R19:R18×R17:R16＝(R23:R22×R21:R20)<<1

```
        CLR     R2              ; R2を0に設定
        FMULS   R23, R21        ; 符号付き非乗数上位×符号付き乗数上位
        MOVW    R18, R0         ; 3,4ﾊﾞｲﾄ目の結果を取得
        FMUL    R22, R20        ; 符号なし非乗数下位×符号なし乗数下位
        ADC     R18, R2         ; 2ﾊﾞｲﾄ目からのｷｬﾘｰ補正
        MOVW    R16, R0         ; 1,2ﾊﾞｲﾄ目の結果を取得
        FMULSU  R23, R20        ; 符号付き非乗数上位×符号なし乗数下位
        SBC     R19, R2         ; 3ﾊﾞｲﾄ目からのｷｬﾘｰ補正
        ADD     R17, R0         ; 2ﾊﾞｲﾄ目の結果取得
        ADC     R18, R1         ; 3ﾊﾞｲﾄ目の結果取得
        ADC     R19, R2         ; 4ﾊﾞｲﾄ目の結果取得
        FMULSU  R21, R22        ; 符号なし非乗数下位×符号付き乗数上位
        SBC     R19, R2         ; 3ﾊﾞｲﾄ目からのｷｬﾘｰ補正
        ADD     R17, R0         ; 2ﾊﾞｲﾄ目の結果取得
        ADC     R18, R1         ; 3ﾊﾞｲﾄ目の結果取得
        ADC     R19, R2         ; 4ﾊﾞｲﾄ目の結果取得
```

5.58. ICALL – 間接ｻﾌﾞﾙｰﾁﾝ呼び出し (Indirect Call to Subroutine)

5.58.1. 説明

ﾚｼﾞｽﾀ ﾌｧｲﾙの(16ﾋﾞｯﾄ)Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されるｻﾌﾞﾙｰﾁﾝの間接呼び出し。Zﾎﾟｲﾝﾀ ﾚｼﾞｽは16ﾋﾞｯﾄ幅で、ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ空間の最下位64K語(128Kﾊﾞｲﾄ)内のｻﾌﾞﾙｰﾁﾝへの呼び出しを許します。ｽﾀｯｸ ﾎﾟｲﾝﾀ(SP)はICALL中に事後減少機構を使います。(CALL命令、EICALL命令、RCALL命令参照)

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作	注釈
PC ← Z PC(21～16) ← 0 (②のみ)	① 16ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは64K語(128Kﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間があります。 ② 22ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは4M語(8Mﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間があります。

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ
ICALL	なし	① PC ← Z	STACK ← PC + 1 SP ← SP – 2 (2ﾊﾞｲﾄ,16ﾋﾞｯﾄ)
		② PC(15～0) ← Z PC(21～16) ← 0	STACK ← PC + 1 SP ← SP – 3 (3ﾊﾞｲﾄ,22ﾋﾞｯﾄ)

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1

5.58.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　　1 (2ﾊﾞｲﾄ)

表5-58. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	16ﾋﾞｯﾄPC	3 (注)	2 (注)	2	3
	22ﾋﾞｯﾄPC	4 (注)	3 (注)	3	利用不可

注: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

例:

```
        ADD     ZL, R0          ; 分岐表にｵﾌｾｯﾄ(差分)を加算
        ICALL                   ; 対応処理実行(Z(R31:R30)が示すｻﾌﾞﾙｰﾁﾝ呼び出し)
```

5.59. IJMP - 間接無条件分岐 (Indirect Jump)

5.59.1. 説明

ﾚｼﾞｽﾀ ﾌｧｲﾙの(16ﾋﾞｯﾄ)Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されるｱﾄﾞﾚｽへの間接無条件分岐。Zﾎﾟｲﾝﾀ ﾚｼﾞｽは16ﾋﾞｯﾄ幅で、ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ空間の最下位64K語(128Kﾊﾞｲﾄ)内での分岐を許します。(JMP命令、EIJMP命令、RJMP命令参照)

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作	注釈
PC ← Z PC(21～16) ← 0 (②のみ)	① 16ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは64K語(128Kﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間があります。 ② 22ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは4M語(8Mﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間があります。

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ
IJMP	なし	① PC ← Z ② PC(15～0) ← Z PC(21～16) ← 0	影響なし

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1

5.59.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　　1 (2ﾊﾞｲﾄ)

表5-59. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	2

例:

```
        ADD     ZL, R0          ; 分岐表にｵﾌｾｯﾄ(差分)を加算
        IJMP                    ; (Z(R31:R30)が示す)対応処理へ分岐
```

5.60. IN – I/Oレジスタを汎用作業レジスタに取得 (Load an I/O Location to Register)

5.60.1. 説明

I/O空間からのデータをレジスタ ファイルの汎用作業レジスタ(Rd)に置きます。

動作

Rd ← I/O(A)

書式		オペランド	プログラム カウンタ
IN	Rd,A	d=0～31, A=0～63	PC ← PC + 1

機械語 (16ビット)

1	0	1	1	0	A5	A4	d4	d3	d2	d1	d0	A3	A2	A1	A0

5.60.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-60. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            IN      R25, $16        ; ポートBを読み込み (注:本注釈は$16がポートBのデバイスの場合です。)
            CPI     R25, 4          ; ポートB=4か検査
            BREQ    EXIT            ; ポートB=4(Z=1)で分岐
                    ～
EXIT:       NOP                     ; 処理なし(ポートB=4での分岐先)
```

5.61. INC – 汎用作業ﾚｼﾞｽﾀを増加(+1) (Increment)

5.61.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の内容に1を加算し、結果をRdﾚｼﾞｽﾀに配置。

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)はこの操作によって影響を及ぼされず、従って、倍精度演算の繰り返し計数器に使うことをINC命令に許します。

符号なし値での操作時、BREQとBRNEの条件分岐だけが矛盾なく実行することを期待できます。2の補数値での操作時、全ての符号付き条件分岐が利用可能です。

動作

Rd ← Rd + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
INC	Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	1	1

5.61.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	–

S – N XOR V
2の補数での符号。

V – R7 AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
2の補数での溢れ発生で設定(1)、その他で解除(0)。元のRdが$7Fの場合のみ2の補数での溢れが発生します。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-61. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        CLR     R17         ; 計数器(R17)を初期化($00)
LOOP:   ADD     R1, R2      ; R1にR2を加算
        INC     R17         ; 計数器進行(+1)
        CPI     R17, 10     ; 10回目か検査
        BRNE    LOOP        ; 10回まで継続
```

5.62. JMP - 絶対ｱﾄﾞﾚｽ指定による無条件分岐 (Jump)

5.62.1. 説明

4M語ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ全体への無条件分岐。RJMP命令もご覧ください。
この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

PC ← k

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ
JMP	k	0≦k＜4M	PC ← k	無変化

機械語 (32ﾋﾞｯﾄ)

1	0	0	1	0	1	0	k21	k20	k19	k18	k17	1	1	0	k16
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0

5.62.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 2 (4ﾊﾞｲﾄ)

表5-62. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	3	3	3	利用不可

例:

```
                MOV     R16, R0         ; R0をR16に複写
                JMP     FARJMP          ; 無条件分岐
                        ~
FARJMP:         NOP                     ; 処理なし(無条件分岐先)
```

5.63. LAC – 取得と解除 (Load And Clear)

5.63.1. 説明

間接指定でﾃﾞｰﾀ空間から汎用作業ﾚｼﾞｽﾀに1ﾊﾞｲﾄを取得し、汎用作業ﾚｼﾞｽﾀによって指定されるﾃﾞｰﾀ空間内のﾋﾞｯﾄを解除(0)。この命令は内部SRAMに対してだけ使うことができます。

ﾃﾞｰﾀ位置はﾚｼﾞｽﾀ ﾌｧｲﾙ内の(16ﾋﾞｯﾄ)Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されます。ﾒﾓﾘ ｱｸｾｽは現在の64Kﾊﾞｲﾄ ﾃﾞｰﾀ ｾｸﾞﾒﾝﾄに制限されます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽで他のﾃﾞｰﾀ ｾｸﾞﾒﾝﾄをｱｸｾｽするには、I/O領域のﾚｼﾞｽﾀでRAMPZが変更されなければなりません。

Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀはこの操作によって無変化のままです。この命令は特にSRAM内に格納された状態ﾋﾞｯﾄの解除(0)に適します。

動作

TEMP ← Rd, Rd ← DS(Z), DS(Z) ← DS(Z) AND ($FF – TEMP)

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
LAC	Z,Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	1	0

5.63.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-63. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	利用不可	2	利用不可	利用不可

5.64. LAS – 取得と設定 (Load And Set)

5.64.1. 説明

間接指定でﾃﾞｰﾀ空間から汎用作業ﾚｼﾞｽﾀに1ﾊﾞｲﾄを取得し、汎用作業ﾚｼﾞｽﾀによって指定されるﾃﾞｰﾀ空間内のﾋﾞｯﾄを設定(1)。この命令は内部SRAMに対してだけ使うことができます。

ﾃﾞｰﾀ位置はﾚｼﾞｽﾀ ﾌｧｲﾙ内の(16ﾋﾞｯﾄ)Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されます。ﾒﾓﾘ ｱｸｾｽは現在の64Kﾊﾞｲﾄ ﾃﾞｰﾀ ｾｸﾞﾒﾝﾄに制限されます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽで他のﾃﾞｰﾀ ｾｸﾞﾒﾝﾄをｱｸｾｽするには、I/O領域のﾚｼﾞｽﾀでRAMPZが変更されなければなりません。

Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀはこの操作によって無変化のままです。この命令は特にSRAM内に格納された状態ﾋﾞｯﾄの設定(1)に適します。

動作

TEMP ← Rd, Rd ← DS(Z), DS(Z) ← DS(Z) OR TEMP

書式	オﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
LAS Z,Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	0	1

5.64.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-64. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	利用不可	2	利用不可	利用不可

5.65. LAT – 取得と反転 (Load And Toggle)

5.65.1. 説明

間接指定でﾃﾞｰﾀ空間から汎用作業ﾚｼﾞｽﾀに1ﾊﾞｲﾄを取得し、汎用作業ﾚｼﾞｽﾀによって指定されるﾃﾞｰﾀ空間内のﾋﾞｯﾄの状態を切り替え(論理反転)。この命令は内部SRAMに対してだけ使うことができます。

ﾃﾞｰﾀ位置はﾚｼﾞｽﾀ ﾌｧｲﾙ内の(16ﾋﾞｯﾄ)Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されます。ﾒﾓﾘ ｱｸｾｽは現在の64Kﾊﾞｲﾄ ﾃﾞｰﾀ ｾｸﾞﾒﾝﾄに制限されます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽで他のﾃﾞｰﾀ ｾｸﾞﾒﾝﾄをｱｸｾｽするには、I/O領域のﾚｼﾞｽﾀでRAMPZが変更されなければなりません。

Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀはこの操作によって無変化のままです。この命令は特にSRAM内に格納された状態ﾋﾞｯﾄの変更に適します。

動作

TEMP ← Rd, Rd ← DS(Z), DS(Z) ← DS(Z) XOR TEMP

書式	オﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
LAT Z,Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	1	1

5.65.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-65. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	利用不可	2	利用不可	利用不可

5.66. LD – Xﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ取得 (Load Indirect from data space to Register using Index X)

5.66.1. 説明

間接指定でﾃﾞｰﾀ空間から1ﾊﾞｲﾄを汎用作業ﾚｼﾞｽﾀ(Rd)に取得します。ﾃﾞｰﾀ空間は通常、ﾚｼﾞｽﾀ ﾌｧｲﾙ、I/Oﾒﾓﾘ、SRAMから成り、ﾃﾞｰﾀ空間の詳細な定義についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

ﾃﾞｰﾀ位置はﾚｼﾞｽﾀ ﾌｧｲﾙ内のX(16ﾋﾞｯﾄ)ﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されます。ﾒﾓﾘ ｱｸｾｽは現在の64Kﾊﾞｲﾄ ﾃﾞｰﾀ ｾｸﾞﾒﾝﾄに制限されます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽで他のﾃﾞｰﾀ ｾｸﾞﾒﾝﾄをｱｸｾｽするにはI/O空間内のﾚｼﾞｽﾀのRAMPXが変更されなければなりません。

Xﾎﾟｲﾝﾀ ﾚｼﾞｽﾀは操作によって無変化のまま、または事後増加や事前減少の何れかにすることができます。これらの特徴は特に配列、表、Xﾎﾟｲﾝﾀ ﾚｼﾞｽﾀのｽﾀｯｸ ﾎﾟｲﾝﾀ使用でのｱｸｾｽに便利です。256ﾊﾞｲﾄ以下のﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽでXﾎﾟｲﾝﾀの下位ﾊﾞｲﾄだけが更新されることに注意してください。このようなﾃﾞﾊﾞｲｽについてはﾎﾟｲﾝﾀの上位ﾊﾞｲﾄはこの命令によって使われず、別の目的に使うことができます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間かﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘを持つﾃﾞﾊﾞｲｽではI/O空間のRAMPXﾚｼﾞｽﾀが更新され、このようなﾃﾞﾊﾞｲｽでは24ﾋﾞｯﾄ ｱﾄﾞﾚｽ全体に増加/減少が加えられます。

この命令の全ての変種が全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。

縮小されたｺｱのAVRrcではﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘがﾃﾞｰﾀ ﾒﾓﾘ空間に割り当てられるため、LPMと同じ動作を達成するのにLD命令を使うことができます。

注: 右の記述は結果が不定となります。

```
LD    R26,X+
LD    R27,X+
LD    R26,-X
LD    R27,-X
```

動作	注釈
① Rd ← DS(X)	;
② Rd ← DS(X), X ← X + 1	; 事後増加
③ X ← X – 1, Rd ← DS(X)	; 事前減少

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
① LD	Rd,X	d=0～31	PC ← PC + 1
② LD	Rd,X+	d=0～31	PC ← PC + 1
③ LD	Rd,-X	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	0	0	①
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	0	1	②
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	1	0	③

5.66.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2ﾊﾞｲﾄ)

表5-66. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	①	2 (注1)	2 (注1,3)	2 (注2)	1/2
	②	2 (注1)	2 (注1,3)	2 (注2)	2/3
	③	2 (注1)	3 (注1,3)	2 (注2)	2/3

注1: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

注2: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、NVMへのｱｸｾｽに対して有効ではありません。NVMをｱｸｾｽする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄでNVMCTRL章をご覧ください。

注3: LD命令がI/Oﾚｼﾞｽﾀをｱｸｾｽする場合、1周期減ります。

例:

```
        CLR     R27             ; Xﾚｼﾞｽﾀ上位を設定
        LDI     R26, $60        ; Xﾚｼﾞｽﾀ下位を設定($60)
        LD      R0, X+          ; ﾃﾞｰﾀ空間$0060の内容をR0に取得、ﾎﾟｲﾝﾀ進行
        LD      R1, X           ; ﾃﾞｰﾀ空間$0061の内容をR1に取得
        LDI     R26, $63        ; Xﾚｼﾞｽﾀ下位を設定($63)
        LD      R2, X           ; ﾃﾞｰﾀ空間$0063の内容をR2に取得
        LD      R3, -X          ; ﾎﾟｲﾝﾀ後退、ﾃﾞｰﾀ空間$0062の内容をR3に取得
```

5.67. LD (LDD) – Yレジスタ間接でのデータ取得 (Load Indirect from data space to Register using Index Y)

5.67.1. 説明

変位付き/なしの間接指定でデータ空間から1バイトを汎用作業レジスタ(Rd)に取得します。データ空間は通常、レジスタ ファイル、I/Oメモリ、SRAMから成り、データ空間の詳細な定義についてはデバイスのデータシートを参照してください。

データ位置はレジスタ ファイル内のY(16ビット)ポインタ レジスタによって指示されます。メモリ アクセスは現在の64Kバイト データ セグメントに制限されます。64Kバイトを超えるデータ空間を持つデバイスで他のデータ セグメントをアクセスするにはI/O空間内のレジスタのRAMPYが変更されなければなりません。

Yポインタ レジスタは操作によって無変化のまま、または事後増加や事前減少の何れかにすることができます。これらの特徴は特に配列、表、Yポインタ レジスタのスタック ポインタ使用でのアクセスに便利です。256バイト以下のデータ空間を持つデバイスでYポインタの下位バイトだけが更新されることに注意してください。このようなデバイスについてはポインタの上位バイトはこの命令によって使われず、別の目的に使うことができます。64Kバイトを超えるデータ空間かプログラム メモリを持つデバイスではI/O空間のRAMPYレジスタが更新され、このようなデバイスでは24ビット アドレス全体に増加/減少が加えられます。

この命令の全ての変種が全てのデバイスで利用可能な訳ではありません。

縮小されたコアのAVRrcではプログラム メモリがデータ メモリ空間に割り当てられるため、LPMと同じ動作を達成するのにLD命令を使うことができます。

注: 右の記述は結果が不定となります。

```
LD   R28,Y+
LD   R29,Y+
LD   R28,-Y
LD   R29,-Y
```

動作	注釈
① Rd ← DS(Y)	;
② Rd ← DS(Y), Y ← Y + 1	; 事後増加
③ Y ← Y – 1, Rd ← DS(Y)	; 事前減少
④ Rd ← DS(Y+q)	; 変位(q)付き

書式		オペランド	プログラム カウンタ
① LD	Rd,Y	d=0～31	PC ← PC + 1
② LD	Rd,Y+	d=0～31	PC ← PC + 1
③ LD	Rd,-Y	d=0～31	PC ← PC + 1
④ LDD	Rd,Y+q	d=0～31, q=0～63	PC ← PC + 1

機械語 (16ビット)

①	1	0	0	0	0	0	0	d4	d3	d2	d1	d0	1	0	0	0
②	1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	0	0	1
③	1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	0	1	0
④	1	0	q5	0	q4	q3	0	d4	d3	d2	d1	d0	1	q2	q1	q0

① 注: LDD Rd,Y+0命令と等価です。

5.67.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2バイト)

表5-67. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	①	2 (注1)	2 (注1,3)	2 (注2)	1/2
	②	2 (注1)	2 (注1,3)	2 (注2)	2/3
	③	2 (注1)	3 (注1,3)	2 (注2)	2/3
	④	2 (注1)	3 (注1,3)	2 (注2)	利用不可

注1: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

注2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、NVMへのアクセスに対して有効ではありません。NVMをアクセスする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なデバイスのデータシートでNVMCTRL章をご覧ください。

注3: LD命令がI/Oレジスタをアクセスする場合、1周期減ります。

例:

```
        CLR     R29             ; Yﾚｼﾞｽﾀ上位を設定
        LDI     R28, $60        ; Yﾚｼﾞｽﾀ下位を設定($60)
        LD      R0, Y+          ; ﾃﾞｰﾀ空間$0060の内容をR0に取得、ﾎﾟｲﾝﾀ進行
        LD      R1, Y           ; ﾃﾞｰﾀ空間$0061の内容をR1に取得
        LDI     R28, $63        ; Yﾚｼﾞｽﾀ下位を設定($63)
        LD      R2, Y           ; ﾃﾞｰﾀ空間$0063の内容をR2に取得
        LD      R3, -Y          ; ﾎﾟｲﾝﾀ後退、ﾃﾞｰﾀ空間$0062の内容をR3に取得
        LDD     R4, Y+2         ; ﾃﾞｰﾀ空間$0064の内容をR4に取得
```

5.68. LD (LDD) – Zレジスタ間接でのデータ取得 (Load Indirect from data space to Register using Index Z)

5.68.1. 説明

変位付き/なしの間接指定でデータ空間から1バイトを汎用作業レジスタ(Rd)に取得します。データ空間は通常、レジスタ ファイル、I/Oメモリ、SRAMから成り、データ空間の詳細な定義についてはデバイスのデータシートを参照してください。

データ位置はレジスタ ファイル内のZ(16ビット)ポインタ レジスタによって指示されます。メモリ アクセスは現在の64Kバイト データ セグメントに制限されます。64Kバイトを超えるデータ空間を持つデバイスで他のデータ セグメントをアクセスするにはI/O空間内のレジスタのRAMPZが変更されなければなりません。

Zポインタ レジスタは操作によって無変化のまま、または事後増加や事前減少の何れかにすることができます。これらの特徴は特に配列、表、Zポインタ レジスタのスタック ポインタ使用でのアクセスに便利です。けれども、Zポインタ レジスが間接サブルーチン呼び出し、間接分岐、表参照に使うことができるため、スタック ポインタ専用としてXまたはYポインタを使うことが多くの場合にもっと便利です。256バイト以下のデータ空間を持つデバイスでZポインタの下位バイトだけが更新されることに注意してください。このようなデバイスについてはポインタの上位バイトはこの命令によって使われず、別の目的に使うことができます。64Kバイトを超えるデータ空間かプログラム メモリを持つデバイスではI/O空間のRAMPZレジスタが更新され、このようなデバイスでは24ビット アドレス全体に増加/減少が加えられます。

この命令の全ての変種が全てのデバイスで利用可能な訳ではありません。

縮小コアのAVRrcではプログラム メモリがデータ メモリ空間に割り当てられるため、LPMと同じ動作を得るのにLD命令を使うことができます。

プログラム メモリでの表参照に対するZポインタ使用についてはLPM命令とELPM命令をご覧ください。

注: 右の記述は結果が不定となります。

```
LD   R30,Z+
LD   R31,Z+
LD   R30,-Z
LD   R31,-Z
```

動作	注釈
① Rd ← DS(Z)	;
② Rd ← DS(Z), Z ← Z + 1	; 事後増加
③ Z ← Z – 1, Rd ← DS(Z)	; 事前減少
④ Rd ← DS(Z+q)	; 変位(q)付き

書式		オペランド	プログラム カウンタ
① LD	Rd,Z	d=0～31	PC ← PC + 1
② LD	Rd,Z+	d=0～31	PC ← PC + 1
③ LD	Rd,-Z	d=0～31	PC ← PC + 1
④ LDD	Rd,Z+q	d=0～31, q=0～63	PC ← PC + 1

機械語 (16ビット)

1	0	0	0	0	0	0	d4	d3	d2	d1	d0	0	0	0	0
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	0	0	1
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	0	1	0
1	0	q5	0	q4	q3	0	d4	d3	d2	d1	d0	0	q2	q1	q0

① 注: LDD Rd,Z+0命令と等価です。
②
③
④

5.68.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2バイト)

表5-68. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	①	2 (注1)	2 (注1,3)	2 (注2)	1/2
	②	2 (注1)	2 (注1,3)	2 (注2)	2/3
	③	2 (注1)	3 (注1,3)	2 (注2)	2/3
	④	2 (注1)	3 (注1,3)	2 (注2)	利用不可

注1: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

注2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、NVMへのアクセスに対して有効ではありません。NVMをアクセスする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なデバイスのデータシートでNVMCTRL章をご覧ください。

注3: LD命令がI/Oレジスタをアクセスする場合、1周期減ります。

例:

```
        CLR     R31             ; Zﾚｼﾞｽﾀ上位を設定
        LDI     R30, $60        ; Zﾚｼﾞｽﾀ下位を設定($60)
        LD      R0, Z+          ; ﾃﾞｰﾀ空間$0060の内容をR0に取得、ﾎﾟｲﾝﾀ進行
        LD      R1, Z           ; ﾃﾞｰﾀ空間$0061の内容をR1に取得
        LDI     R30, $63        ; Zﾚｼﾞｽﾀ下位を設定($63)
        LD      R2, Z           ; ﾃﾞｰﾀ空間$0063の内容をR2に取得
        LD      R3, -Z          ; ﾎﾟｲﾝﾀ後退、ﾃﾞｰﾀ空間$0062の内容をR3に取得
        LDD     R4, Z+2         ; ﾃﾞｰﾀ空間$0064の内容をR4に取得
```

5.69. LDI - 即値ﾊﾞｲﾄ定数を汎用作業ﾚｼﾞｽﾀに取得 (Load Immediate)

5.69.1. 説明

ﾊﾞｲﾄ(8ﾋﾞｯﾄ)の即値定数を汎用作業ﾚｼﾞｽﾀ(R16～R31)に設定。

動作

Rd ← K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
LDI	Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	1	1	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0

5.69.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-2. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        CLR     R31             ; Zﾚｼﾞｽﾀ上位に$00を設定
        LDI     R30, $F0        ; Zﾚｼﾞｽﾀ下位に$F0を設定
        LPM                     ; ﾌﾟﾛｸﾞﾗﾑ空間$00F0の内容をR0に(定数)取得
```

5.70. LDS – データ空間直接指定で汎用作業レジスタに取得 (Load Direct from data space)

5.70.1. 説明

データ空間から汎用作業レジスタ(Rd)に1バイトを取得。データ空間は通常、レジスタ ファイル、I/Oメモリ、SRAMから成り、データ空間の詳細な定義についてはデバイスのデータシートを参照してください。

16ビット アドレスが供給されなければなりません。メモリ アクセスは現在の64Kバイト データ セグメントに制限されます。LDS命令は64Kバイトを超えるメモリをアクセスするのにRAMPDレジスタを使います。64Kバイトを超えるデータ空間を持つデバイスで他のデータ セグメントをアクセスするにはI/O空間内のレジスタのRAMPDが変更されなれければなりません。

この命令は全てのデバイスで利用可能な訳ではありません。「追補A」を参照してください。

動作

Rd ← DS(k)

書式		オペランド	プログラム カウンタ
LDS	Rd,(k)	d=0～31, k=0～65535	PC ← PC + 1

機械語 (32ビット)

1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	0	0	0
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0

5.70.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 2 (4バイト)

表5-70. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2 (注1)	3 (注1,3)	3 (注2)	利用不可

注1: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

注2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、NVMへのアクセスに対して有効ではありません。NVMをアクセスする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なデバイスのデータシートでNVMCTRL章をご覧ください。

注3: LD命令がI/Oレジスタをアクセスする場合、1周期減ります。

例:

```
        LDS     R2, $FF00       ; データ空間$FF00の内容をR2に取得
        ADD     R2, R1          ; R2とR1を加算
        STS     $FF00, R2       ; 書き戻し
```

5.71. LDS (AVRrc) - SRAMから直接指定で汎用作業ﾚｼﾞｽﾀに取得 (Load Direct from SRAM)

5.71.1. 説明

ﾃﾞｰﾀ空間から1ﾊﾞｲﾄを汎用作業ﾚｼﾞｽﾀに取得します。ﾃﾞｰﾀ空間は通常、ﾚｼﾞｽﾀ ﾌｧｲﾙ、I/Oﾒﾓﾘ、SRAMから成り、ﾃﾞｰﾀ空間の詳細な定義についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

7ﾋﾞｯﾄのｱﾄﾞﾚｽが供給されなければなりません。命令内で与えられるｱﾄﾞﾚｽは次のようにﾃﾞｰﾀ空間ｱﾄﾞﾚｽを符号化します。

ADDR7～0 = ($\overline{k4}$,k4,k6,k5,k3,k2,k1,k0)

ﾒﾓﾘ ｱｸｾｽは$0040～$00BFのｱﾄﾞﾚｽ範囲に制限されます。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

Rd ← DS(k)

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
LDS	Rd,(k)	d=16～31,$0040≦k≦$00BF	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	1	0	0	k6	k5	k4	d3	d2	d1	d0	k3	k2	k1	k0

注: R0～R15のﾚｼﾞｽﾀはR16～R31に割り当て直されます。

5.71.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-71. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	利用不可	利用不可	利用不可	2

例:

```
        LDS     R18, $0040      ; ﾃﾞｰﾀ空間$0040の内容をR18に取得
        ADD     R18, R17        ; R18とR17を加算
        STS     $0040, R18      ; 書き戻し
```

5.72. LPM – 間接によるﾌﾟﾛｸﾞﾗﾑ空間からの取得 (Load Program Memory)

5.72.1. 説明

Zﾚｼﾞｽﾀによって指示される1ﾊﾞｲﾄを転送先汎用作業ﾚｼﾞｽﾀ(Rd)に取得。この命令は100%空間効率的な定数初期化や定数ﾃﾞｰﾀ取得が特徴です。ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘが16ﾋﾞｯﾄ語で構成される一方でZﾎﾟｲﾝﾀはﾊﾞｲﾄ ｱﾄﾞﾚｽを指示します。従って、Zﾎﾟｲﾝﾀの最下位ﾋﾞｯﾄは下位ﾊﾞｲﾄ(ZLSb=0)または上位ﾊﾞｲﾄ(ZLSb=1)のどちらかを選びます。この命令はﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘの最初の32K語(64Kﾊﾞｲﾄ)をｱﾄﾞﾚｽ指定することができます。Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀは操作によって無変化のまま、または増加のどちらかにすることができます。

自己ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ能力を持つﾃﾞﾊﾞｲｽはﾋｭｰｽﾞ ﾋﾞｯﾄや施錠ﾋﾞｯﾄの値を読み出すLPM命令を使うことができます。詳細な記述についてはﾃﾞﾊﾞｲｽの文書を参照してください。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

注: 右に示す記述での実行結果は不定となります。

```
LPM   R30,Z+
LPM   R31,Z+
```

動作	注釈
① R0 ← PS(Z)	; R0暗黙指定
② Rd ← PS(Z)	; 無変化
③ Rd ← PS(Z), Z ← Z + 1	; 事後増加

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
① LPM		なし(R0暗黙指定)	PC ← PC + 1
② LPM	Rd,Z	d=0～31	PC ← PC + 1
③ LPM	Rd,Z+	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	1	1	0	0	1	0	0	0	①
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	0	0	②
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	0	1	③

5.72.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2ﾊﾞｲﾄ)

表5-72. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	3	3	3	利用不可

例:

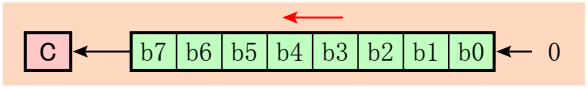
```
        LDI     ZH, BYTE2(TABLE<<1)     ; Zﾚｼﾞｽﾀ上位(R31)にﾊﾞｲﾄ単位ｱﾄﾞﾚｽ上位ﾊﾞｲﾄ値を設定
        LDI     ZL, BYTE1(TABLE<<1)     ; Zﾚｼﾞｽﾀ下位(R30)にﾊﾞｲﾄ単位ｱﾄﾞﾚｽ下位ﾊﾞｲﾄ値と下位ﾊﾞｲﾄ指定を設定
        ELPM    R16, Z                  ; TABLEの下位ﾊﾞｲﾄをR16に取得

;
TABLE:  .DW     $5876                   ; ZﾚｼﾞｽﾀのLSB=0の時に$76、LSB=1の時に$58がｱﾄﾞﾚｽ指定されます。
```

5.73. LSL – 汎用作業ﾚｼﾞｽﾀの論理左移動(ｼﾌﾄ) (Logical Shift Left)

5.73.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の全ﾋﾞｯﾄを1ﾋﾞｯﾄ左移動。ﾋﾞｯﾄ0は解除(0)され、(元の)ﾋﾞｯﾄ7はｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)に置かれます。この操作は符号付きと符号なしの値を2で効率的に乗算します。(ADD Rd,Rd命令と等価)

動作



書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
LSL	Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	0	1	1	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0

5.73.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H – Rd3

S – N XOR V
2の補数での符号。

V – N XOR C (移動後の)

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C – Rd7
移動前のRdの最上位ﾋﾞｯﾄが設定(1)ならば設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-73. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

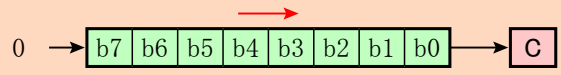
```
        ADD     R0, R4          ; R0とR4を加算
        LSL     R0              ; R0×2
```

5.74. LSR – 汎用作業ﾚｼﾞｽﾀの論理右移動(ｼﾌﾄ) (Logical Shift Right)

5.74.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の全ﾋﾞｯﾄを1ﾋﾞｯﾄ右移動。ﾋﾞｯﾄ7は解除(0)され、(元の)ﾋﾞｯﾄ0はｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)に置かれます。この操作は符号なし値を2で効率的に除算します。ｷｬﾘｰ ﾌﾗｸﾞ(C)は結果の丸めに使われます。

動作



書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
LSR Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	1	1	0

5.74.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	0	⇔	⇔

S – N XOR V
2の補数での符号。

V – N XOR C (移動後の)

N – 0
解除(0)。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C – Rd0
移動前のRdの最下位ﾋﾞｯﾄが設定(1)ならば設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-74. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R0, R4          ; R0とR4を加算
        LSR     R0              ; R0÷2
```

5.75. MOV - 汎用作業レジスタ間の複写 (Copy Register)

5.75.1. 説明

この命令は1つの汎用作業レジスタ(Rr)の複製を別の汎用作業レジスタ(Rd)に作ります。元レジスタRrは無変化のままです。転送先レジスタRdはRrの複製が置かれます。

動作

Rd ← Rr

書式		オペランド	プログラム カウンタ
MOV	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ビット)

0	0	1	0	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.75.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-75. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            MOV     R16,R0          ; R0をR16に複写
            CALL    CHECK           ; 検査処理
                    ⌇
CHECK:      CPI     R16,$11         ; R16=$11か検査
                    ⌇
            RET                     ; 呼び出し元へ復帰
```

5.76. MOVW – 汎用作業レジスタ対間の複写 (Copy Register Word)

5.76.1. 説明

この命令は1つの汎用作業レジスタ対(Rr+1:Rr)の複製を別の汎用作業レジスタ対(Rd+1:Rd)に作ります。元レジスタ対Rr+1:Rrが無変化のままな一方で転送先レジスタ対Rd+1:RdはRr+1:Rrの複製が置かれます。

この命令は全てのデバイスで利用可能な訳ではありません。「追補A」を参照してください。

動作

R[d+1]:Rd ← R[r+1]:Rr

書式		オペランド	プログラム カウンタ
MOVW	Rd,Rr	d={0,2,～,28,30}, r={0,2,～,28,30}	PC ← PC + 1

機械語 (16ビット)

0	0	0	0	0	0	0	1	d4	d3	d2	d1	r4	r3	r2	r1

5.76.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-76. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	利用不可

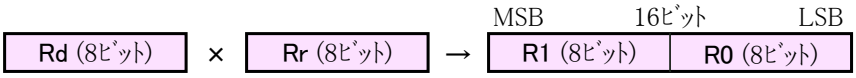
例:

```
            MOVW    R16, R0         ; R1:R0をR17:R16に複写
            CALL    CHECK           ; 検査処理
                    ⋮
CHECK:      CPI     R16, $11        ; R16=$11か検査
                    ⋮
            CPI     R17, $32        ; R17=$32か検査
                    ⋮
            RET                     ; 呼び出し元へ復帰
```

5.77. MUL – 符号なし乗算 (Multiply Unsigned)

5.77.1. 説明

この命令は8ﾋﾞｯﾄ×8ﾋﾞｯﾄ=16ﾋﾞｯﾄの符号なし乗算を実行します。



被乗数Rdと乗数Rrは符号なし数値を含む2つの汎用作業ﾚｼﾞｽﾀです。16ﾋﾞｯﾄ符号なし積はR1(上位)とR0(下位)に置かれます。被乗数または乗数がR0またはR1から選ばれた場合、乗算後に結果がそれらを上書きします。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

R1:R0 ← Rd × Rr　　(符号なし ← 符号なし × 符号なし)

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
MUL	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.77.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

Z – $\overline{R15}$ AND $\overline{R14}$ AND $\overline{R13}$ AND ・・・ $\overline{R1}$ AND $\overline{R0}$
結果が$0000で設定(1)、その他で解除(0)。

C – R15
結果のﾋﾞｯﾄ15=1で設定(1)、その他で解除(0)。

R(結果)は演算後のR1:R0と等しくなります。

命令語数　　1 (2ﾊﾞｲﾄ)

表5-77. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

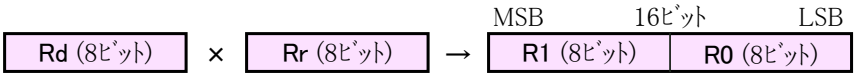
例:

```
        MUL     R5, R4          ; R5とR4の符号なし乗算
        MOVW    R4, R0          ; 結果をR5:R4に取得
```

5.78. MULS – 符号付き乗算 (Multiply Signed)

5.78.1. 説明

この命令は8ﾋﾞｯﾄ×8ﾋﾞｯﾄ=16ﾋﾞｯﾄの符号付き乗算を実行します。



被乗数Rdと乗数Rrは符号付き数値を含む2つの汎用作業ﾚｼﾞｽﾀです。16ﾋﾞｯﾄ符号付き積はR1(上位)とR0(下位)に置かれます。被乗数または乗数がR0またはR1から選ばれた場合、乗算後に結果がそれらを上書きします。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

R1:R0 ← Rd × Rr　　(符号付き ← 符号付き × 符号付き)

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
MULS　Rd,Rr	d=16～31, r=16～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	0	0	0	1	0	d3	d2	d1	d0	r3	r2	r1	r0

5.78.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

Z – $\overline{R15}$ AND $\overline{R14}$ AND $\overline{R13}$ AND ･･･ $\overline{R1}$ AND $\overline{R0}$
結果が$0000で設定(1)、その他で解除(0)。

C – R15
結果のﾋﾞｯﾄ15=1で設定(1)、その他で解除(0)。

R(結果)は演算後のR1:R0と等しくなります。

命令語数　　1 (2ﾊﾞｲﾄ)

表5-78. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

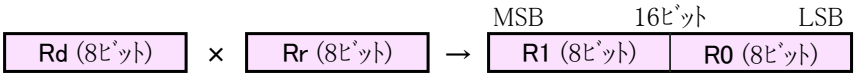
例:

```
        MULS    R21, R20        ; R21とR20の符号付き乗算
        MOVW    R20, R0         ; 結果をR21:R20に取得
```

5.79. MULSU – 符号付きと符号なしの乗算 (Multiply Signed with Unsigned)

5.79.1. 説明

この命令は符号付き数値と符号なし数値の8ﾋﾞｯﾄ×8ﾋﾞｯﾄ=16ﾋﾞｯﾄ乗算を実行します。



被乗数Rdと乗数Rrは2つの汎用作業ﾚｼﾞｽﾀです。被乗数Rdが符号付き数値で、乗数Rrが符号なし数値です。16ﾋﾞｯﾄ符号付き積はR1(上位)とR0(下位)に置かれます。被乗数または乗数がR0またはR1から選ばれた場合、乗算後に結果がそれらを上書きします。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

R1:R0 ← Rd × Rr　　(符号付き ← 符号付き × 符号なし)

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
MULSU Rd,Rr	d=16～23, r=16～23	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	0	0	0	1	1	0	d2	d1	d0	0	r2	r1	r0

5.79.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	⇔	⇔

Z – $\overline{R15}$ AND $\overline{R14}$ AND $\overline{R13}$ AND ・・・ $\overline{R1}$ AND $\overline{R0}$
結果が$0000で設定(1)、その他で解除(0)。

C – R15
結果のﾋﾞｯﾄ15=1で設定(1)、その他で解除(0)。

R(結果)は演算後のR1:R0と等しくなります。

命令語数　　1 (2ﾊﾞｲﾄ)

表5-79. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

例: 符号付き整数(16×16＝32ﾋﾞｯﾄ)　R19:R18×R17:R16＝R23:R22×R21:R20

```
        CLR     R2          ; R2を0に設定
        MULS    R23, R21    ; 符号付き非乗数上位×符号付き乗数上位
        MOVW    R18, R0     ; 3,4ﾊﾞｲﾄ目の結果を取得
        MUL     R22, R20    ; 符号なし非乗数下位×符号なし乗数下位
        MOVW    R16, R0     ; 1,2ﾊﾞｲﾄ目の結果を取得
        MULSU   R23, R20    ; 符号付き非乗数上位×符号なし乗数下位
        SBC     R19, R2     ; 3ﾊﾞｲﾄ目からのｷｬﾘｰ補正
        ADD     R17, R0     ; 2ﾊﾞｲﾄ目の結果取得
        ADC     R18, R1     ; 3ﾊﾞｲﾄ目の結果取得
        ADC     R19, R2     ; 4ﾊﾞｲﾄ目の結果取得
        MULSU   R21, R22    ; 符号なし非乗数下位×符号付き乗数上位
        SBC     R19, R2     ; 3ﾊﾞｲﾄ目からのｷｬﾘｰ補正
        ADD     R17, R0     ; 2ﾊﾞｲﾄ目の結果取得
        ADC     R18, R1     ; 3ﾊﾞｲﾄ目の結果取得
        ADC     R19, R2     ; 4ﾊﾞｲﾄ目の結果取得
```

5.80. NEG – 2の補数変換 (Two's Complement)

5.80.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の内容をそれの2の補数で置換。値$80は無変化のままです。

動作

Rd ← $00 – Rd

書式	オペランド	プログラム カウンタ
NEG Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	0	1

5.80.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H – R3 OR Rd3
ﾋﾞｯﾄ3からのｷｬﾘｰ発生で設定(1)、その他で解除(0)。

S – N XOR V
2の補数での符号。

V – R7 AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
暗黙の$00からの減算で2の補数での溢れ発生で設定(1)、その他で解除(0)。Rd=$80時のみｾｯﾄされます。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C – R7 OR R6 OR R5 OR R4 OR R3 OR R2 OR R1 OR R0
暗黙の$00からの減算による借入有りで設定(1)、その他で解除(0)。結果が$00時のみ解除されます。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-80. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            SUB     R11,R0          ; R11からR0を減算
            BRPL    POSITIVE        ; 結果が正で分岐
;
            NEG     R11             ; 結果が負で符号変換(絶対値取得)
POSITIVE:   NOP                     ; 処理なし(継続)
```

5.81. NOP – 無操作 (No Operation)

5.81.1. 説明

この命令は単一周期無操作を実行します。

動作

なし

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
NOP	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5.81.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-81. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        CLR     R16             ; R16に$00を設定
        SER     R17             ; R17に$FFを設定
        OUT     $18, R16        ; ﾎﾟｰﾄBに$00を出力 (注:本注釈は$18がﾎﾟｰﾄBのﾃﾞﾊﾞｲｽの場合です。)
        NOP                     ; 待機(無操作)
        OUT     $18, R17        ; ﾎﾟｰﾄBに$FFを出力
```

5.82. OR - 汎用作業ﾚｼﾞｽﾀ間の論理和を取得 (Logical OR)

5.82.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)と汎用作業ﾚｼﾞｽﾀ(Rr)の内容間の論理和(OR)を実行し、結果を転送先ﾚｼﾞｽﾀRdに配置。

動作

Rd ← Rd OR Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
OR	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	1	0	1	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.82.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S - N XOR V
2の補数での符号。

V - 0
解除(0)。

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-82. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            OR      R15, R16        ; R15とR16の論理和取得
            BST     R15, 6          ; ﾋﾞｯﾄ6をT(一時)ﾋﾞｯﾄに複写
            BRTS    ACTIVE          ; R15,R16何れかのﾋﾞｯﾄ6=1で分岐
                ≀
ACTIVE:     NOP                     ; 無操作(ﾋﾞｯﾄ6=1での分岐先)
```

5.83. ORI – 汎用作業ﾚｼﾞｽﾀと即値定数の論理和を取得 (Logical OR with Immediate)

5.83.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の内容と即値定数(K)間の論理和(OR)を実行し、結果を転送先ﾚｼﾞｽﾀRdに配置。

動作

Rd ← Rd OR K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
ORI	Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	1	1	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0

5.83.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

S – N XOR V
2の補数での符号。

V – 0
解除(0)。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-83. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ORI     R16, $F0        ; R16の上位ﾆﾌﾞﾙ(4ﾋﾞｯﾄ)を設定(1)
        ORI     R17, 1          ; R17のﾋﾞｯﾄ0を設定(1)
```

5.84. OUT – 汎用作業ﾚｼﾞｽﾀからI/Oﾚｼﾞｽﾀに設定 (Store Register to I/O Location)

5.84.1. 説明

ﾚｼﾞｽﾀ ファイルの汎用作業ﾚｼﾞｽﾀ(Rr)からのﾃﾞｰﾀをI/O空間に格納。

動作

I/O(A) ← Rr

書式	オﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
OUT A,Rr	r=0～31, A=0～63	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	1	1	1	A5	A4	r4	r3	r2	r1	r0	A3	A2	A1	A0

5.84.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-84. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        CLR     R16             ; R16に$00を設定
        SER     R17             ; R17に$FFを設定
        OUT     $18, R16        ; ﾎﾟｰﾄBに$00を出力 (注:本注釈は$18がﾎﾟｰﾄBのﾃﾞﾊﾞｲｽの場合です。)
        NOP                     ; 待機(無操作)
        OUT     $18, R17        ; ﾎﾟｰﾄBに$FFを出力
```

5.85. POP - スタックから汎用作業レジスタに取得 (Pop Register from Stack)

5.85.1. 説明

この命令はスタックから1バイトを汎用作業レジスタ(Rd)に取得します。スタック ポインタ(SP)はPOP前に1つ事前増加されます。
この命令は全てのデバイスで利用可能な訳ではありません。「追補A」を参照してください。

動作

Rd ← STACK

書式		オペランド	プログラム カウンタ	スタック ポインタ
POP	Rd	d=0～31	PC ← PC + 1	SP ← SP + 1

機械語 (16ビット)

1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	1	1

5.85.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-85. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2 (注)	2 (注)	2	3

注: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

例:

```
            CALL    ROUTINE         ; サブルーチン呼び出し
                    ～
ROUTINE:    PUSH    R14             ; R14を保存
            PUSH    R13             ; R13を保存
                    ～
            POP     R13             ; R13を復帰
            POP     R14             ; R14を復帰
            RET                     ; 呼び出し元へ復帰
```

5.86. PUSH – 汎用作業ﾚｼﾞｽﾀをｽﾀｯｸに格納 (Push Register on Stack)

5.86.1. 説明

この命令は汎用作業ﾚｼﾞｽﾀ(Rr)の内容をｽﾀｯｸ上に格納します。ｽﾀｯｸ ﾎﾟｲﾝﾀ(SP)はPUSP後に1つ事後減少されます。
この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

STACK ← Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ ﾎﾟｲﾝﾀ
PUSH	Rr	r=0～31	PC ← PC + 1	SP ← SP – 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	1	1

5.86.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-86. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2 (注)	1 (注)	1	1

注: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

例:

```
            CALL    ROUTINE         ; ｻﾌﾞﾙｰﾁﾝ呼び出し
                    ≀
ROUTINE:    PUSH    R14             ; R14を保存
            PUSH    R13             ; R13を保存
                    ≀
            POP     R13             ; R13を復帰
            POP     R14             ; R14を復帰
            RET                     ; 呼び出し元へ復帰
```

5.87. RCALL - PC相対ｻﾌﾞﾙｰﾁﾝ呼び出し (Relative Call to a Subroutine)

5.87.1. 説明

PC-2047～PC+2048(語内のｱﾄﾞﾚｽへの相対呼び出し。復帰(RCALL後の命令)ｱﾄﾞﾚｽがｽﾀｯｸ上に格納されます。CALL命令もご覧ください。4K語(8Kﾊﾞｲﾄ)を超えないﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘを持つAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗに対し、この命令は全てのｱﾄﾞﾚｽ位置からﾒﾓﾘ全体をｱﾄﾞﾚｽ指定することができます。ｽﾀｯｸ ﾎﾟｲﾝﾀ(SP)はRCALL中に事後減少機構を使います。

動作	注釈
PC ← PC + k + 1	① 16ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは64K語(128Kﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。 ② 22ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは4M語(8Mﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ
RCALL k	k=-2048～+2047	PC ← PC + k + 1	① STACK ← PC + 1 SP ← SP - 2 (2ﾊﾞｲﾄ,16ﾋﾞｯﾄ) ② STACK ← PC + 1 SP ← SP - 3 (3ﾊﾞｲﾄ,22ﾋﾞｯﾄ)

機械語 (16ﾋﾞｯﾄ)

1	1	0	1	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0

5.87.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数　　1 (2ﾊﾞｲﾄ)

表5-87. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	16ﾋﾞｯﾄPC	3 (注)	2 (注)	2	3
	22ﾋﾞｯﾄPC	4 (注)	3 (注)	3	利用不可

注: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

例:

```
            RCALL   ROUTINE         ; ｻﾌﾞﾙｰﾁﾝ呼び出し
                    ≀
ROUTINE:    PUSH    R16             ; R16をｽﾀｯｸに保存
                    ≀
            POP     R16             ; R16をｽﾀｯｸから復帰
            RET                     ; 呼び出し元へ復帰
```

5.88. RET – ｻﾌﾞﾙｰﾁﾝからの復帰 (Return from Subroutine)

5.88.1. 説明

ｻﾌﾞﾙｰﾁﾝからの復帰。復帰ｱﾄﾞﾚｽはｽﾀｯｸから取得されます。ｽﾀｯｸ ﾎﾟｲﾝﾀ(SP)はRET中に事前増加機構を使います。

動作	注釈
① PC(15～0) ← STACK	;16ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは64K語(128Kﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。
② PC(21～0) ← STACK	;22ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは4M語(8Mﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ
RET	なし	PC ← STACK	① SP ← SP + 2 (2ﾊﾞｲﾄ,16ﾋﾞｯﾄ) ② SP ← SP + 3 (3ﾊﾞｲﾄ,22ﾋﾞｯﾄ)

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0

5.88.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　　1 (2ﾊﾞｲﾄ)

表5-88. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	16ﾋﾞｯﾄPC	4 (注)	4 (注)	4	6
	22ﾋﾞｯﾄPC	5 (注)	5 (注)	5	利用不可

注: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

例:

```
            CALL    ROUTINE         ; ｻﾌﾞﾙｰﾁﾝ呼び出し
                    ~
ROUTINE:    PUSH    R16             ; R16をｽﾀｯｸに保存
                    ~
            POP     R16             ; R16をｽﾀｯｸから復帰
            RET                     ; 呼び出し元へ復帰
```

5.89. RETI – 割り込みからの復帰 (Return from Interrupt)

5.89.1. 説明

割り込みから復帰。復帰ｱﾄﾞﾚｽはｽﾀｯｸから取得し、ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の全体割り込み許可(I)ﾋﾞｯﾄが設定(1)されます。

注: ｽﾃｰﾀｽ ﾚｼﾞｽﾀは割り込みﾙｰﾁﾝ移行時に自動的に格納されず、割り込みﾙｰﾁﾝからの復帰時に自動的に回復されません。応用ﾌﾟﾛｸﾞﾗﾑがこれを処理しなければなりません。ｽﾀｯｸ ﾎﾟｲﾝﾀ(SP)はRETI中に事前増加機構を使います。

注: AVRxmとAVRxtのﾃﾞﾊﾞｲｽについては割り込み処理部の最後でRETI命令が実行される時に正しい割り込み段階に戻ることを保証する割り込み状態情報を割り込み制御器状態(CPUINT.STATUS)ﾚｼﾞｽﾀが含みます。RETIが実行される時に処理した割り込みに対応するﾌﾗｸﾞが解除(0)されます。

動作	注釈
① PC(15～0) ← STACK	;16ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは64K語(128Kﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。
② PC(21～0) ← STACK	;22ﾋﾞｯﾄ PCのﾃﾞﾊﾞｲｽは4M語(8Mﾊﾞｲﾄ)のﾌﾟﾛｸﾞﾗﾑ空間までです。

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ	ｽﾀｯｸ
RETI	なし	PC ← STACK	① SP ← SP + 2 (2ﾊﾞｲﾄ,16ﾋﾞｯﾄ ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ(PC)と最大128Kﾊﾞｲﾄのﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘを持つﾃﾞﾊﾞｲｽ) ② SP ← SP + 3 (3ﾊﾞｲﾄ,22ﾋﾞｯﾄ ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ(PC)と最大8Mﾊﾞｲﾄのﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘを持つﾃﾞﾊﾞｲｽ)

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0

5.89.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
1	–	–	–	–	–	–	–

I – 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-89. 実行周期数

名称		AVRe (注1)	AVRxm (注1)	AVRxt (注1)	AVRrc
周期数	16ﾋﾞｯﾄPC	4 (注2)	4 (注2)	4	6
	22ﾋﾞｯﾄPC	5 (注2)	5 (注2)	5	利用不可

注1: RETIはAVRe、AVRxm、AVRxtﾃﾞﾊﾞｲｽで違う様に動きます。AVRe系ﾃﾞﾊﾞｲｽでは一旦割り込みが起こると、ﾊｰﾄﾞｳｪｱによって全体割り込み許可(I)ﾋﾞｯﾄが解除(0)され、このﾋﾞｯﾄはRETIが実行される時に設定(1)されます。AVRxmとAVRxtﾃﾞﾊﾞｲｽでは、割り込み処理ﾙｰﾁﾝ(ISR)移行中にﾊｰﾄﾞｳｪｱによって解除(0)されないため、RETIはSREGの全体割り込み許可ﾋﾞｯﾄを変更しません。このﾋﾞｯﾄは必要とされる時にSEIとCLIの命令を使って変更されるべきです。

注2: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

例:

```
                ;
ENTINT:   PUSH  R16             ; R16をｽﾀｯｸに保存
          IN    R16, SREG       ; ｽﾃｰﾀｽ ﾚｼﾞｽﾀ値を取得
          PUSH  R16             ; ｽﾃｰﾀｽ ﾚｼﾞｽﾀ値を保存
                ;
EXITINT:  POP   R16             ; ｽﾃｰﾀｽ ﾚｼﾞｽﾀ値をｽﾀｯｸから取得
          OUT   SREG, R16       ; ｽﾃｰﾀｽ ﾚｼﾞｽﾀ値を復帰
          POP   R16             ; R16をｽﾀｯｸから復帰
          RETI                  ; 割り込みから復帰
```

5.90. RJMP - PC相対無条件分岐 (Relative Jump)

5.90.1. 説明

PC-2047～PC+2048(語)内のｱﾄﾞﾚｽへの相対無条件分岐。4K語(8Kﾊﾞｲﾄ)を超えないﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘを持つAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗに対し、この命令は全てのｱﾄﾞﾚｽ位置からﾒﾓﾘ全体をｱﾄﾞﾚｽ指定することができます。JMP命令もご覧ください。

動作

PC ← PC + k + 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
RJMP k	k=-2048～+2047	PC ← PC + k + 1

機械語 (16ﾋﾞｯﾄ)

1	1	0	0	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0

5.90.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-90. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	2

例:

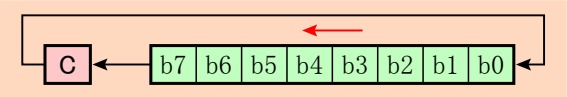
```
            CPI     R16, $42        ; R16＝$42か比較検査
            BRNE    ERROR           ; R16≠$42(Z=0)で分岐
            RJMP    DONE            ; R16＝$42で分岐
;
ERROR:      ADD     R16, R17        ; R16とR17を加算
            INC     R16             ; R16を+1
DONE:       NOP                     ; 無操作(RJMPの分岐先)
```

5.91. ROL - ｷｬﾘｰを含めた汎用作業ﾚｼﾞｽﾀの左回転 (Rotate Left trough Carry)

5.91.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の全ﾋﾞｯﾄを1ﾋﾞｯﾄ左移動。ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)がRdのﾋﾞｯﾄ0に移動されます。(元の)ﾋﾞｯﾄ7はｷｬﾘｰ ﾌﾗｸﾞ(C)に移動されます。LSL命令と組み合わせたこの操作は符号付きまたは符号なし倍精度値を2で効率的に乗算します。(ADC Rd,Rd命令と等価)

動作



書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
ROL	Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	1	1	1	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0

5.91.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H - Rd3

S - N XOR V
2の補数での符号。

V - N XOR C (移動後の)

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C - Rd7
移動前のRdの最上位ﾋﾞｯﾄが設定(1)ならば設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-91. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

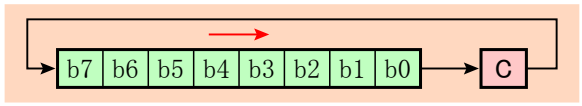
```
        LSL     R18             ; R19:R18×2
        ROL     R19             ;
```

5.92. ROR - キャリーを含めた汎用作業レジスタの右回転 (Rotate Right through Carry)

5.92.1. 説明

汎用作業レジスタ(Rd)の全ビットを1ビット右移動。ステータス レジスタ(SREG)のキャリー フラグ(C)がRdのビット7に移動されます。(元の)ビット0はキャリー フラグ(C)に移動されます。ASR命令と組み合わせたこの命令は符号付き倍精度値を2で効率的に除算します。LSR命令との組み合わせは符号なし倍精度値を2で効率的に除算します。キャリー フラグは結果を丸めるのに使うことができます。

動作



書式	オペランド	プログラム カウンタ
ROR Rd	d=0～31	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	1	1	1

5.92.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	⇔

S - N XOR V
2の補数での符号。

V - N XOR C (移動後の)

N - R7
結果の最上位ビットの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C - Rd0
移動前のRdの最下位ビットが設定(1)ならば設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2バイト)

表5-92. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        LSR     R19             ; R19:R18÷2(符号なし)
        ROR     R18             ;
        ASR     R17             ; R17:R16÷2(符号付き)
        ROR     R16             ;
```

5.93. SBC - キャリーを含めた汎用作業レジスタ間の減算 (Subtract with Carry)

5.93.1. 説明

2つの汎用作業レジスタ(Rd,Rr)で減算してステータス レジスタ(SREG)のキャリー フラグ(C)を減算し、結果を転送先レジスタRdに配置。

動作

Rd ← Rd - Rr - C

書式		オペランド	プログラム カウンタ
SBC	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ビット)

0	0	0	0	1	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.93.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H - $\overline{Rd3}$ AND Rr3 OR Rr3 AND R3 OR R3 AND $\overline{Rd3}$
ビット3からの借入有りで設定(1)、その他で解除(0)。

S - N XOR V
2の補数での符号。

V - Rd7 AND $\overline{Rr7}$ AND $\overline{R7}$ OR $\overline{Rd7}$ AND Rr7 AND R7
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N - R7
結果の最上位ビットの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$ AND Z
実行前がZ=1で結果が$00の場合に設定(1)、その他で解除(0)。

C - $\overline{Rd7}$ AND Rr7 OR Rr7 AND R7 OR R7 AND $\overline{Rd7}$
符号なしのRd＜(Rr+C)で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2バイト)

表5-93. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例: R3:R2 ← R3:R2 - R1:R0

```
        SUB     R2, R0          ; 下位バイト減算
        SBC     R3, R1          ; キャリーを含めて上位バイト減算
```

5.94. SBCI - キャリーを含めた汎用作業レジスタから即値定数の減算 (Subtract Immediate with Carry)

5.94.1. 説明

汎用作業レジスタ(Rd)から即値定数(K)を減算してステータス レジスタ(SREG)のキャリー フラグ(C)を減算し、結果を転送先レジスタRdに配置。

動作

Rd ← Rd - K - C

書式		オペランド	プログラム カウンタ
SBCI	Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ビット)

0	1	0	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0

5.94.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H - $\overline{Rd3}$ AND K3 OR K3 AND R3 OR R3 AND $\overline{Rd3}$
ビット3からの借入有りで設定(1)、その他で解除(0)。

S - N XOR V
2の補数での符号。

V - Rd7 AND $\overline{K7}$ AND $\overline{R7}$ OR $\overline{Rd7}$ AND K7 AND R7
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N - R7
結果の最上位ビットの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$ AND Z
実行前がZ=1で結果が$00の場合に設定(1)、その他で解除(0)。

C - $\overline{Rd7}$ AND K7 OR K7 AND R7 OR R7 AND $\overline{Rd7}$
符号なしのRd＜(K+C)で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2バイト)

表5-94. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例: R17:R16 ← R17:R16 - $4F23

```
SUBI    R16, $23        ; 下位バイト減算
SBCI    R17, $4F        ; キャリーを含めて上位バイト減算
```

5.95. SBI – I/Oレジスタのビット設定(1) (Set Bit in I/O Register)

5.95.1. 説明

指定したI/Oレジスタのビットを設定(1)。この命令はI/Oレジスタの下位32バイト(I/Oアドレス0～31)で動きます。

動作

I/O(A,b) ← 1

書式		オペランド	プログラム カウンタ
SBI	A,b	A=0～31, b=0～7	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	1	0	1	0	A4	A3	A2	A1	A0	b2	b1	b0

5.95.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2バイト)

表5-95. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	1	1	1

例: EEPROM読み出し (注:本例は$1C～$1EがEEPROM用I/Oレジスタのデバイスの場合です。)

```
        OUT     $1E, R0         ; EEPROMアドレス設定
        SBI     $1C, 0          ; 読み込み(EECRのEERE(ビット0)=1)
        IN      R1, $1D         ; R1にEEPROMデータを取得
```

5.96. SBIC - I/Oレジスタのビットが解除(0)でスキップ (Skip if Bit in I/O Register is Cleared)

5.96.1. 説明

この命令はI/Oレジスタの単一ビットを検査してそのビットが解除(0)されている場合に次の命令をスキップ(非実行に)します。この命令はI/Oレジスタの下位32バイト(I/Oアドレス 0～31)で動きます。

動作

I/O(A,b)=0なら、PC ← PC + 2または3, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ
SBIC	A,b	A=0～31, b=0～7	PC ← PC + 1　(条件不成立) PC ← PC + 2　(条件成立、次が1語命令) PC ← PC + 3　(条件成立、次が2語命令)

機械語 (16ビット)

1	0	0	1	1	0	0	1	A4	A3	A2	A1	A0	b2	b1	b0

5.96.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2バイト)

表5-96. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立 (スキップなし)	1	2	1	1
	条件成立、次が1語命令	2	3	2	2
	条件成立、次が2語命令	3	4	3	利用不可

例: EEPROM書き込み完了検査 (注:本例は$1CがEEPROM制御レジスタのデバイスの場合です。)

```
WAIT:   SBIC   $1C, 3        ; EEPROM書き込み完了(EECRのEEWEまたはEEPE=0)でスキップ
        RJMP   WAIT          ; EEWEまたはEEPE=0まで待機
               ≀
```

5.97. SBIS - I/Oレジスタのビットが設定(1)でスキップ (Skip if Bit in I/O Register is Set)

5.97.1. 説明

この命令はI/Oレジスタの単一ビットを検査してそのビットが設定(1)されている場合に次の命令をスキップ(非実行に)します。この命令はI/Oレジスタの下位32バイト(I/Oアドレス 0～31)で動きます。

動作

I/O(A,b)=1なら、PC ← PC + 2または3, さもなくば、PC ← PC + 1

書式		オペランド	プログラム カウンタ
SBIS	A,b	A=0～31, b=0～7	PC ← PC + 1　(条件不成立) PC ← PC + 2　(条件成立、次が1語命令) PC ← PC + 3　(条件成立、次が2語命令)

機械語 (16ビット)

1	0	0	1	1	0	1	1	A4	A3	A2	A1	A0	b2	b1	b0

5.97.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2バイト)

表5-97. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立 (スキップなし)	1	2	1	1
	条件成立、次が1語命令	2	3	2	2
	条件成立、次が2語命令	3	4	3	利用不可

例:

```
STWAIT:   SBIS    $10, 0          ; アドレス$0010のビット0=1でスキップ
          RJMP    STWAIT          ; アドレス$0010のビット0=1まで待機
                  ~
```

5.98. SBIW - 汎用作業ﾚｼﾞｽﾀ対から即値定数の減算 (Subtract Immediate from Word)

5.98.1. 説明

汎用作業ﾚｼﾞｽﾀ対(Rd+1:Rd)から即値定数(0～63)を減算し、結果をﾚｼﾞｽﾀ対Rd+1:Rdに配置。この命令は上位4つの汎用作業ﾚｼﾞｽﾀ対で動き、ﾎﾟｲﾝﾀ ﾚｼﾞｽﾀの操作に良く適合します。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

R[d+1]:Rd ← R[d+1]:Rd - K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SBIW	Rd,K	d={24,26,28,30}, K=0～63	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	1	1	K5	K4	d2	d1	K3	K2	K1	K0

5.98.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	⇔

S - N XOR V
2の補数での符号。

V - Rdh7 AND $\overline{R15}$
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N - R15
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R15}$ AND $\overline{R14}$ AND $\overline{R13}$ AND ･･･ $\overline{R1}$ AND $\overline{R0}$
結果が$0000で設定(1)、その他で解除(0)。

C - R15 AND $\overline{Rdh7}$
符号なしでRd＜Kの場合に設定(1)、その他で解除(0)。

R(結果)は演算後のR[d+1]:Rdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-98. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2	2	2	利用不可

例:

```
        SBIW    R24, 1          ; R25:R24-1
        SBIW    YL, 63          ; Yﾚｼﾞｽﾀ(R29:R28)-63
```

5.99. SBR - 汎用作業ﾚｼﾞｽﾀの(複数)ﾋﾞｯﾄ設定(1) (Set Bits in Register)

5.99.1. 説明

指定した汎用作業ﾚｼﾞｽﾀ(Rd)の(複数)ﾋﾞｯﾄを設定(1)。ﾚｼﾞｽﾀRdの内容と定数遮蔽K間で論理和(OR)を実行し、結果を転送先ﾚｼﾞｽﾀRdに配置。(ORI Rd,K命令と等価)

動作

Rd ← Rd OR K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SBR	Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	1	1	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0

5.99.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S - N XOR V
2の補数での符号。

V - 0
解除(0)。

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-99. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        SBR     R16, $F0        ; 上位ﾆﾌﾞﾙ(4ﾋﾞｯﾄ)を設定
        SBR     R18, 1          ; R18のﾋﾞｯﾄ0を設定(1)
```

5.100. SBRC - 汎用作業ﾚｼﾞｽﾀのﾋﾞｯﾄが解除(0)でｽｷｯﾌﾟ (Skip if Bit in Register is Cleared)

5.100.1. 説明

この命令は汎用作業ﾚｼﾞｽﾀの単一ﾋﾞｯﾄを調べ、そのﾋﾞｯﾄが解除(0)されている場合に次の命令をｽｷｯﾌﾟ(非実行に)します。

動作

Rr(b)=0なら、PC ← PC + 2または3, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SBRC	Rr,b	r=0～31, b=0～7	PC ← PC + 1　(条件不成立) PC ← PC + 2　(条件成立、次が1語命令) PC ← PC + 3　(条件成立、次が2語命令)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	1	1	0	r4	r3	r2	r1	r0	0	b2	b1	b0

5.100.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数　1 (2ﾊﾞｲﾄ)

表5-100. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立 (ｽｷｯﾌﾟなし)	1	1	1	1
	条件成立、次が1語命令	2	2	2	2
	条件成立、次が2語命令	3	3	3	利用不可

例:

```
        SUB     R0, R1          ; R0-R1
        SBRC    R0, 7           ; R0のﾋﾞｯﾄ7=0でｽｷｯﾌﾟ
        SUB     R0, R1          ; R0のﾋﾞｯﾄ7=1でR0-R1
      ~
```

5.101. SBRS - 汎用作業ﾚｼﾞｽﾀのﾋﾞｯﾄが設定(1)でｽｷｯﾌﾟ (Skip if Bit in Register is Set)

5.101.1. 説明

この命令は汎用作業ﾚｼﾞｽﾀの単一ﾋﾞｯﾄを調べ、そのﾋﾞｯﾄが設定(1)されている場合に次の命令をｽｷｯﾌﾟ(非実行に)します。

動作

Rr(b)=1なら、PC ← PC + 2または3, さもなくば、PC ← PC + 1

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SBRS	Rr,b	r=0～31, b=0～7	PC ← PC + 1　(条件不成立) PC ← PC + 2　(条件成立、次が1語命令) PC ← PC + 3　(条件成立、次が2語命令)

機械語 (16ﾋﾞｯﾄ)

1	1	1	1	1	1	1	r4	r3	r2	r1	r0	0	b2	b1	b0

5.101.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数　1 (2ﾊﾞｲﾄ)

表5-101. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	条件不成立 (ｽｷｯﾌﾟなし)	1	1	1	1
	条件成立、次が1語命令	2	2	2	2
	条件成立、次が2語命令	3	3	3	利用不可

例:

```
        SUB     R0, R1          ; R0-R1
        SBRS    R0, 7           ; R0のﾋﾞｯﾄ7=1でｽｷｯﾌﾟ
        SUB     R0, R1          ; R0のﾋﾞｯﾄ7=0でR0-R1
        ~
```

5.102. SEC - ｷｬﾘｰ ﾌﾗｸﾞを設定(1) (Set Carry Flag)

5.102.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｷｬﾘｰ ﾌﾗｸﾞ(C)を設定(1)。(BSET 0命令と等価)

動作

C ← 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SEC	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0

5.102.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	1

C - 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-102. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        SEC             ; ｷｬﾘｰ ﾌﾗｸﾞを設定
        ADD     R0, R0  ; R0×2+1
```

5.103. SEH - ハーフキャリー フラグを設定(1) (Set Half Carry Flag)

5.103.1. 説明

ステータス レジスタ(SREG)のハーフキャリー フラグ(H)を設定(1)。(BSET 5命令と等価)

動作

H ← 1

書式	オペランド	プログラム カウンタ
SEH	なし	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	1	0	0	0	1	0	1	1	0	0	0

5.103.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	1	-	-	-	-	-

H - 1
設定(1)。

命令語数 1 (2バイト)

表5-103. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
SEH                    ; ハーフキャリー フラグを設定
```

5.104. SEI - 全体割り込み許可ﾌﾗｸﾞを設定(1) (Set Global Interrupt Flag)

5.104.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の全体割り込み許可ﾌﾗｸﾞ(I)を設定(1)。SEIに後続する命令はどの保留中割り込みにも先立って実行されます。(BSET 7命令と等価)

動作

I ← 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SEI	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	1	1	1	1	0	0	0

5.104.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
1	-	-	-	-	-	-	-

I - 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-104. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
SEI             ; 全体割り込み許可
SLEEP           ; 休止へ移行、割り込み待ち
                ; 注:どの保留中割り込みもない状態で休止へ移行してください。
```

5.105. SEN - 負ﾌﾗｸﾞを設定(1) (Set Negative Flag)

5.105.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の負ﾌﾗｸﾞ(N)を設定(1)。(BSET 2命令と等価)

動作

N ← 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SEN	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0

5.105.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	1	-	-

N - 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-105. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        SEN                     ; 負ﾌﾗｸﾞを設定(1)
```

5.106. SER - 汎用作業ﾚｼﾞｽﾀを設定($FF) (Set all bits in Register)

5.106.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)に直接$FFを設定。(LDI Rd,$FF命令と等価)

動作

Rd ← $FF

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SER	Rd	d=16～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	1	1	0	1	1	1	1	d3	d2	d1	d0	1	1	1	1

5.106.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-106. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        CLR     R16             ; R16に$00を設定
        SER     R17             ; R17に$FFを設定
        OUT     $18, R16        ; ﾎﾟｰﾄBに$00を出力 (注:本注釈は$18がﾎﾟｰﾄBのﾃﾞﾊﾞｲｽの場合です。)
        NOP                     ; 保持時間待機(無操作)
        OUT     $18, R17        ; ﾎﾟｰﾄBに$FFを出力
```

5.107. SES – 符号ﾌﾗｸﾞを設定(1) (Set Signed Flag)

5.107.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の符号ﾌﾗｸﾞ(S)を設定(1)。(BSET 4命令と等価)

動作

S ← 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SES	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0

5.107.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	1	–	–	–	–

S – 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-107. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        SES                     ; 符号ﾌﾗｸﾞを設定
```

5.108. SET – Tﾋﾞｯﾄを設定(1) (Set T Flag)

5.108.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の一時ﾋﾞｯﾄ(T)を設定(1)。(BSET 6命令と等価)

動作

T ← 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SET	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	1	1	0	1	0	0	0

5.108.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	1	–	–	–	–	–	–

T – 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-108. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
SET                    ; Tﾋﾞｯﾄを設定
```

5.109. SEV - 2の補数溢れﾌﾗｸﾞを設定(1) (Set Overflow Flag)

5.109.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)の2の補数溢れﾌﾗｸﾞ(V)を設定(1)。(BSET 3命令と等価)

動作

V ← 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SEV	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0

5.109.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	1	-	-	-

V - 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-109. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        SEV                     ; 2の補数溢れﾌﾗｸﾞを設定
```

5.110. SEZ – ｾﾞﾛ ﾌﾗｸﾞを設定(1) (Set Zero Flag)

5.110.1. 説明

ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)のｾﾞﾛ ﾌﾗｸﾞ(Z)を設定(1)。(BSET 1命令と等価)

動作

Z ← 1

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SEZ	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0

5.110.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	1	–

Z – 1
設定(1)。

命令語数 1 (2ﾊﾞｲﾄ)

表5-110. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        ADD     R2, R3          ; R2とR3を加算
        SEZ                     ; ｾﾞﾛ ﾌﾗｸﾞを設定
```

5.111. SLEEP – 休止形態 (Sleep)

5.111.1. 説明

この命令は回路を(ﾃﾞﾊﾞｲｽに依存して)MCU制御ﾚｼﾞｽﾀ(MCUCR)や休止動作制御ﾚｼﾞｽﾀ(SMCR)などによって定義された休止動作形態に設定します。

動作

SLEEP命令使用の詳細はﾃﾞﾊﾞｲｽの資料を参照してください。

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SLEEP	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	1	0	0	0	1	0	0	0

5.111.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-111. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        MOV     R0, R11         ; R11をR0に複写
        LDI     R16, (1<<SE)    ; 休止許可値をR16に取得
        OUT     MCUCR, R16      ; 休止許可(MCUCRのSEﾋﾞｯﾄ=1)
        SLEEP                   ; 休止形態開始
```

5.112. SPM (AVRe) - 間接によるﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘの書き込み (Store Program Memori)

5.112.1. 説明

SPMはﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘのﾍﾟｰｼﾞ消去、(既に消去された)ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘのﾍﾟｰｼﾞ書き込み、ﾌﾞｰﾄ ﾛｰﾀﾞ施錠ﾋﾞｯﾄ設定に使うことができます。いくつかのﾃﾞﾊﾞｲｽで、ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘは1語毎に書くことができます。他のﾃﾞﾊﾞｲｽでは最初にﾍﾟｰｼﾞ一時緩衝部を満たした後でﾍﾟｰｼﾞ全体を同時に書くことができます。全ての場合で、ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘはﾍﾟｰｼﾞ毎に消去されなければなりません。ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ消去時、RAMPZﾚｼﾞｽﾀとZﾚｼﾞｽﾀがﾍﾟｰｼﾞ ｱﾄﾞﾚｽとして使われます。ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ書き込み時、RAMPZﾚｼﾞｽﾀとZﾚｼﾞｽﾀがﾍﾟｰｼﾞまたは語ｱﾄﾞﾚｽとして使われ、R1:R0ﾚｼﾞｽﾀ対がﾃﾞｰﾀ(注)として使われます。ｺｰﾄﾞ空間書き込み操作に対してﾌﾗｯｼｭ ﾒﾓﾘは語ｱｸｾｽされ、故にRAMPZﾚｼﾞｽﾀと連結されたZﾚｼﾞｽﾀの最下位ﾋﾞｯﾄは'0'に設定されるべきです。ﾌﾞｰﾄ ﾛｰﾀﾞ施錠ﾋﾞｯﾄ設定時、R1:R0ﾚｼﾞｽﾀ対がﾃﾞｰﾀとして使われます。SPM命令の使い方の詳細な記述についてはﾃﾞﾊﾞｲｽ文書を参照してください。この命令はﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ全体をｱﾄﾞﾚｽ指定することができます。

SPM命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

注: R1は命令上位ﾊﾞｲﾄ、R0は命令下位ﾊﾞｲﾄを決定します。

動作	注釈
① PS(RAMPZ:Z) ← $FFFF	; ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ ﾍﾟｰｼﾞ消去
② PS(RAMPZ:Z) ← R1:R0	; ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ ﾜｰﾄﾞ書き込み
③ PS(RAMPZ:Z) ← R1:R0	; ﾍﾟｰｼﾞ緩衝部に書き込み
④ PS(RAMPZ:Z) ← BUFFER	; ﾍﾟｰｼﾞ緩衝部からﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘへ書き込み
⑤ BLBITS ← R1:R0	; ﾌﾞｰﾄ ﾛｰﾀﾞ施錠ﾋﾞｯﾄ設定

書式	ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SPM	なし	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	1	1	1	1	0	1	0	0	0

5.112.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数　1 (2ﾊﾞｲﾄ)

表5-112. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	- (注)	利用不可	利用不可	利用不可

注: ﾃﾞﾊﾞｲｽのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ時間で変わります。

例: ﾍﾟｰｼﾞ書き込みﾃﾞﾊﾞｲｽの1ﾍﾟｰｼﾞ書き込み

```
; この例はﾍﾟｰｼﾞ書き込みを持つﾃﾞﾊﾞｲｽに対する1ﾍﾟｰｼﾞのSPM書き込みを示します。
; - このルーチンはRAMからﾌﾗｯｼｭ ﾒﾓﾘへ1ﾍﾟｰｼﾞのﾃﾞｰﾀを書きます。
;  RAM内の最初のﾃﾞｰﾀ位置はYﾎﾟｲﾝﾀによって指示されます。
;  ﾌﾗｯｼｭ ﾒﾓﾘの最初のﾃﾞｰﾀ位置はZﾎﾟｲﾝﾀによって指示されます。
; - 異常処理は含まれません。
; - このルーチンはﾌﾞｰﾄ空間の内側に置かれなければなりません。
;  (最低、SPMJｻﾌﾞﾙｰﾁﾝ)
; - 使用ﾚｼﾞｽﾀ: R0, R1, TMP1, TMP2, CNTL, CNTH, SPMC
;  (TMP1, TMP2, CNTL, CNTH, SPMCは使用者によって定義されなければなりません。)
;  ﾚｼﾞｽﾀの格納と復元はルーチン内に含まれません。
;  ﾚｼﾞｽﾀ使用量はｺｰﾄﾞ量を犠牲にすれば最適することができます。
; ﾍﾟｰｼﾞ内ﾃﾞｰﾀが256ﾊﾞｲﾄ以下の場合はｶｳﾝﾀ上位が不要になります。また関連する命令も変更になります。
; これらの部分を赤字で示します。
; このルーチンはﾌﾞｰﾄ領域に配置されなければなりません。(最低、SPMJｻﾌﾞﾙｰﾁﾝ)

        .EQU    PGSZB = PAGESIZE*2              ; PGSZBはﾍﾟｰｼﾞ内のﾊﾞｲﾄ数です。(PAGESIZEが語数)
        .ORG    SMALLBOOTSTART                  ;

                                                ; [ ﾍﾟｰｼﾞ消去 ]
WRPG:   LDI     SPMC,(1<<PGERS)+(1<<SPMEN)      ; ﾍﾟｰｼﾞ消去SPMCR値を取得
        CALL    SPMJ                            ; ﾍﾟｰｼﾞ消去
```

```
                                        ; [ RAMからﾌﾗｯｼｭ ﾍﾟｰｼﾞ緩衝部へ転送 ]
        LDI     CNTL,LOW(PGSZB)         ; ﾊﾞｲﾄ計数値初期化
        LDI     CNTH,HIGH(PGSZB)        ; (削除)
WLP:    LD      R0,Y+                   ; RAM上の下位ﾃﾞｰﾀを取得(位置進行)
        LD      R1,Y+                   ; RAM上の上位ﾃﾞｰﾀを取得(位置進行)
        LDI     SPMC,(1<<SPMEN)         ; ﾍﾟｰｼﾞ緩衝部書き込みSPMCR値を取得
        CALL    SPMJ                    ; 対応ﾜｰﾄﾞ(語)ﾃﾞｰﾀをﾍﾟｰｼﾞ緩衝部に設定
        ADIW    ZL,2                    ; ﾍﾟｰｼﾞ緩衝部位置進行
        SBIW    CNTL,2                  ; 計数値減数 (SUBI)
        BRNE    WLP                     ; 指定ﾊﾞｲﾄ数分継続
                                        ; [ ﾍﾟｰｼﾞ書き込み ]
        SUBI    ZL,LOW(PGSZB)           ; ﾍﾟｰｼﾞ緩衝部先頭に位置復帰
        SBCI    ZH,HIGH(PGSZB)          ; (削除)
        LDI     SPMC,(1<<PGWRT)+(1<<SPMEN)  ; ﾌﾗｯｼｭ書き込みSPMCR値を取得
        CALL    SPMJ                    ; ﾌﾗｯｼｭ ﾒﾓﾘ ﾍﾟｰｼﾞ書き込み
                                        ; [ 照合 ]
        LDI     CNTL,LOW(PGSZB)         ; ﾊﾞｲﾄ計数値初期化
        LDI     CNTH,HIGH(PGSZB)        ; (削除)
        SUBI    YL,LOW(PGSZB)           ; RAMﾃﾞｰﾀ先頭に位置復帰
        SBCI    YH,HIGH(PGSZB)          ;
RLP:    LPM     R0,Z+                   ; ﾌﾗｯｼｭ ﾒﾓﾘから1ﾊﾞｲﾄ取得(位置進行)
        LD      R1,Y+                   ; RAMから1ﾊﾞｲﾄ ﾃﾞｰﾀを取得(位置進行)
        CPSE    R0,R1                   ; 値一致でｽｷｯﾌﾟ
        JMP     ERROR                   ; 不一致で異常処理へ
;
        SBIW    CNTL,2                  ; 計数値を減数 (SUBI)
        BRNE    RLP                     ; 指定ﾊﾞｲﾄ数分継続
        RET                             ; 呼び出し元へ復帰

SPMJ:   IN      TMP2,SREG               ; 全体割り込み許可ﾌﾗｸﾞを保存
        CLI                             ; 全割り込みを禁止
WAIT:   IN      TMP1,SPMCR              ; SPM制御ﾚｼﾞｽﾀ値を取得
        SBRC    TMP1,SPMEN              ; 設定可能(直前のSPM完了)でｽｷｯﾌﾟ
        RJMP    WAIT                    ; 設定可まで待機
;
        OUT     SPMCR,SPMC              ; SPM動作指定
        SPM                             ; 対応SPM動作実行
        OUT     SREG,TMP2               ; 全体割り込み許可ﾌﾗｸﾞを復帰
        RET                             ; 呼び出し元へ復帰
```

5.113. SPM (AVRxm, AVRxt) – 間接によるﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘの書き込み (Store Program Memori)

5.113.1. 説明

SPMはﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘのﾍﾟｰｼﾞ消去、(既に消去された)ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘのﾍﾟｰｼﾞ書き込み、ﾌﾞｰﾄ ﾛｰﾀﾞ施錠ﾋﾞｯﾄ設定に使うことができます。いくつかのﾃﾞﾊﾞｲｽで、ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘは1語毎に書くことができます。他のﾃﾞﾊﾞｲｽでは最初にﾍﾟｰｼﾞ一時緩衝部を満たした後でﾍﾟｰｼﾞ全体を同時に書くことができます。全ての場合で、ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘはﾍﾟｰｼﾞ毎に消去されなければなりません。ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ消去時、RAMPZﾚｼﾞｽﾀとZﾚｼﾞｽﾀがﾍﾟｰｼﾞ ｱﾄﾞﾚｽとして使われます。ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ書き込み時、RAMPZﾚｼﾞｽﾀとZﾚｼﾞｽﾀがﾍﾟｰｼﾞまたは語ｱﾄﾞﾚｽとして使われ、R1:R0ﾚｼﾞｽﾀ対がﾃﾞｰﾀ(注)として使われます。ｺｰﾄﾞ空間書き込み操作に対してﾌﾗｯｼｭ ﾒﾓﾘは語ｱｸｾｽされ、故にRAMPZﾚｼﾞｽﾀと連結されたZﾚｼﾞｽﾀの最下位ﾋﾞｯﾄは'0'に設定されるべきです。

SPM命令の使い方の詳細な記述についてはﾃﾞﾊﾞｲｽ文書を参照してください。この命令はﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ全体をｱﾄﾞﾚｽ指定することができます。

SPM命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

注: R1は命令上位ﾊﾞｲﾄ、R0は命令下位ﾊﾞｲﾄを決定します。

動作		注釈
① PS(RAMPZ:Z) ← $FFFF		; ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ ﾍﾟｰｼﾞ消去
② PS(RAMPZ:Z) ← R1:R0		; ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ語書き込み (注1)
③ PS(RAMPZ:Z) ← R1:R0		; ﾍﾟｰｼﾞ緩衝部に書き込み (注2)
④ PS(RAMPZ:Z) ← BUFFER		; ﾍﾟｰｼﾞ緩衝部からﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘへ書き込み (注2)
⑤ PS(RAMPZ:Z) ← $FFFF,	Z ← Z + 2	; ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ ﾍﾟｰｼﾞ消去, Z事後増加
⑥ PS(RAMPZ:Z) ← R1:R0,	Z ← Z + 2	; ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ語書き込み, Z事後増加 (注1)
⑦ PS(RAMPZ:Z) ← R1:R0,	Z ← Z + 2	; ﾍﾟｰｼﾞ緩衝部設定, Z事後増加 (注2)
⑧ PS(RAMPZ:Z) ← BUFFER,	Z ← Z + 2	; ﾍﾟｰｼﾞ緩衝部からﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘへ書き込み, Z事後増加 (注2)

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
①～④ SPM		なし	PC ← PC + 1
⑤～⑧ SPM	Z+	なし	PC ← PC + 1

注1: 全てのﾃﾞﾊﾞｲｽがﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘに直接書くことができる訳ではありません。SPMの使い方の詳細な記述についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄをご覧ください。

注2: 全てのﾃﾞﾊﾞｲｽがﾍﾟｰｼﾞ緩衝部を持つ訳ではありません。SPMの使い方の詳細な記述についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄをご覧ください。

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	0	1	1	1	0	1	0	0	0
1	0	0	1	0	1	0	1	1	1	1	1	1	0	0	0

①～④ (注: 従来のSPM命令と同じ)

⑤～⑧

5.113.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2ﾊﾞｲﾄ)

表5-113. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	①～④	利用不可	– (注)	– (注)	利用不可
	⑤～⑧	利用不可	– (注)	– (注)	利用不可

注: ﾃﾞﾊﾞｲｽのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ時間で変わります。

5.114. ST – Xレジスタ間接でのデータ設定 (Store Indirect from Register to data space using Index X)

5.114.1. 説明

汎用作業レジスタ(Rr)から1バイトを間接指定でデータ空間に格納します。データ空間は通常、レジスタ ファイル、I/Oメモリ、SRAMから成り、データ空間の詳細な定義についてはデバイスのデータシートを参照してください。

データ位置はレジスタ ファイル内のX(16ビット)ポインタ レジスタによって指示されます。メモリ アクセスは現在の64Kバイト データ セグメントに制限されます。64Kバイトを超えるデータ空間を持つデバイスで他のデータ セグメントをアクセスするにはI/O空間内のレジスタのRAMPXが変更されなければなりません。

Xポインタ レジスタは操作によって無変化のまま、または事後増加や事前減少の何れかにすることができます。これらの特徴は特に配列、表、Xポインタ レジスタのスタック ポインタ使用でのアクセスに便利です。。256バイト以下のデータ空間を持つデバイスでXポインタの下位バイトだけが更新されることに注意してください。このようなデバイスについてはポインタの上位バイトはこの命令によって使われず、別の目的に使うことができます。64Kバイトを超えるデータ空間かプログラム メモリを持つデバイスではI/O空間のRAMPXレジスタが更新され、このようなデバイスでは24ビット アドレス全体に増加/減少が加えられます。

この命令の全ての変種が全てのデバイスで利用可能な訳ではありません。

注: 右の記述は結果が不定となります。

```
ST    X+,R26
ST    X+,R27
ST    -X,R26
ST    -X,R27
```

動作	注釈
① DS(X) ← Rr	;
② DS(X) ← Rr, X ← X + 1	; 事後増加
③ X ← X – 1, DS(X) ← Rr	; 事前減少

書式		オペランド	プログラム カウンタ
① ST	X,Rr	r=0～31	PC ← PC + 1
② ST	X+,Rr	r=0～31	PC ← PC + 1
③ ST	-X,Rr	r=0～31	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	0	0	①
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	0	1	②
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	1	0	③

5.114.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2バイト)

表5-114. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	①	2 (注1)	1 (注1)	1 (注2)	1
	②	2 (注1)	1 (注1)	1 (注2)	1
	③	2 (注1)	2 (注1)	1 (注2)	2

注1: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

注2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、NVMへのアクセスに対して有効ではありません。NVMをアクセスする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なデバイスのデータシートでNVMCTRL章をご覧ください。

例:

```
        CLR     R27         ; Xレジスタ上位を設定($00)
        LDI     R26, $60    ; Xレジスタ下位を設定($60)
        ST      X+, R0      ; R0の内容をデータ空間$0060に設定、ポインタ進行
        ST      X, R1       ; R1の内容をデータ空間$0061に設定
        LDI     R26, $63    ; Xレジスタ下位を設定($63)
        ST      X, R2       ; R2の内容をデータ空間$0063に設定
        ST      -X, R3      ; ポインタ後退、R3の内容をデータ空間$0062に設定
```

5.115. ST (STD) – Yレジスタ間接でのデータ設定 (Store Indirect from Register to data space using Index Y)

5.115.1. 説明

汎用作業レジスタ(Rr)から1バイトを変位付き/なしの間接指定でデータ空間に格納します。データ空間は通常、レジスタ ファイル、I/Oメモリ、SRAMから成り、データ空間の詳細な定義についてはデバイスのデータシートを参照してください。

データ位置はレジスタ ファイル内のY(16ビット)ポインタ レジスタによって指示されます。メモリ アクセスは現在の64Kバイト データ セグメントに制限されます。64Kバイトを超えるデータ空間を持つデバイスで他のデータ セグメントをアクセスするにはI/O空間内のレジスタのRAMPYが変更されなければなりません。

Yポインタ レジスタは操作によって無変化のまま、または事後増加や事前減少の何れかにすることができます。これらの特徴は特に配列、表、Yポインタ レジスタのスタック ポインタ使用でのアクセスに便利です。。256バイト以下のデータ空間を持つデバイスでXポインタの下位バイトだけが更新されることに注意してください。このようなデバイスについてはポインタの上位バイトはこの命令によって使われず、別の目的に使うことができます。64Kバイトを超えるデータ空間かプログラム メモリを持つデバイスではI/O空間のRAMPYレジスタが更新され、このようなデバイスでは24ビット アドレス全体に増加/減少が加えられます。

この命令の全ての変種が全てのデバイスで利用可能な訳ではありません。

注: 右の記述は結果が不定となります。

```
ST   Y+,R28
ST   Y+,R29
ST   -Y,R28
ST   -Y,R29
```

動作	注釈
① DS(Y) ← Rr	;
② DS(Y) ← Rr, Y ← Y + 1	; 事後増加
③ Y ← Y – 1, DS(Y) ← Rr	; 事前減少
④ DS(Y+q) ← Rr	; 変位(q)付き

書式		オペランド	プログラム カウンタ
① ST	Y,Rr	r=0～31	PC ← PC + 1
② ST	Y+,Rr	r=0～31	PC ← PC + 1
③ ST	-Y,Rr	r=0～31	PC ← PC + 1
④ STD	Y+q,Rr	r=0～31, q=0～63	PC ← PC + 1

機械語 (16ビット)

1	0	0	0	0	0	1	r4	r3	r2	r1	r0	1	0	0	0
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	0	0	1
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	0	1	0
1	0	q5	0	q4	q3	1	r4	r3	r2	r1	r0	1	q2	q1	q0

① 注: STD Y+0,Rr命令と等価です。
②
③
④

5.115.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2バイト)

表5-115. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	①	2 (注1)	1 (注1)	1 (注2)	1
	②	2 (注1)	1 (注1)	1 (注2)	1
	③	2 (注1)	2 (注1)	1 (注2)	2
	④	2 (注1)	2 (注1)	1 (注2)	利用不可

注1: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

注2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、NVMへのアクセスに対して有効ではありません。NVMをアクセスする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なデバイスのデータシートでNVMCTRL章をご覧ください。

例:

```
        CLR     R29             ; Yﾚｼﾞｽﾀ上位を設定
        LDI     R28, $60        ; Yﾚｼﾞｽﾀ下位を設定($60)
        ST      Y+, R0          ; R0の内容をﾃﾞｰﾀ空間$0060に設定、ﾎﾟｲﾝﾀ進行
        ST      Y, R1           ; R1の内容をﾃﾞｰﾀ空間$0061に設定
        LDI     R28, $63        ; Yﾚｼﾞｽﾀ下位を設定($63)
        ST      Y, R2           ; R2の内容をﾃﾞｰﾀ空間$0063に設定
        ST      -Y, R3          ; ﾎﾟｲﾝﾀ後退、R3の内容をﾃﾞｰﾀ空間$0062に設定
        STD     Y+2, R4         ; R4の内容をﾃﾞｰﾀ空間$0064に設定
```

5.116. ST (STD) – Zﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ設定 (Store Indirect from Register to data space using Index Z)

5.116.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rr)から1ﾊﾞｲﾄを変位付き/なしの間接指定でﾃﾞｰﾀ空間に格納します。ﾃﾞｰﾀ空間は通常、ﾚｼﾞｽﾀ ﾌｧｲﾙ、I/Oﾒﾓﾘ、SRAMから成り、ﾃﾞｰﾀ空間の詳細な定義についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

ﾃﾞｰﾀ位置はﾚｼﾞｽﾀ ﾌｧｲﾙ内のZ(16ﾋﾞｯﾄ)ﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されます。ﾒﾓﾘ ｱｸｾｽは現在の64Kﾊﾞｲﾄ ﾃﾞｰﾀ ｾｸﾞﾒﾝﾄに制限されます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽで他のﾃﾞｰﾀ ｾｸﾞﾒﾝﾄをｱｸｾｽするにはI/O空間内のﾚｼﾞｽﾀのRAMPZが変更されなければなりません。

Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀは操作によって無変化のまま、または事後増加や事前減少の何れかにすることができます。これらの特徴は特に配列、表、Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀのｽﾀｯｸ ﾎﾟｲﾝﾀ使用でのｱｸｾｽに便利です。けれども、Zﾎﾟｲﾝﾀ ﾚｼﾞｽが間接ｻﾌﾞﾙｰﾁﾝ呼び出し、間接分岐、表参照に使うことができるため、ｽﾀｯｸ ﾎﾟｲﾝﾀ専用としてXまたはYﾎﾟｲﾝﾀを使うことが多くの場合にもっと便利です。256ﾊﾞｲﾄ以下のﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽでZﾎﾟｲﾝﾀの下位ﾊﾞｲﾄだけが更新されることに注意してください。このようなﾃﾞﾊﾞｲｽについてはﾎﾟｲﾝﾀの上位ﾊﾞｲﾄはこの命令によって使われず、別の目的に使うことができます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間かﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘを持つﾃﾞﾊﾞｲｽではI/O空間のRAMPZﾚｼﾞｽﾀが更新され、このようなﾃﾞﾊﾞｲｽでは24ﾋﾞｯﾄ ｱﾄﾞﾚｽ全体に増加/減少が加えられます。

この命令の全ての変種が全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。

注: 右の記述は結果が不定となります。

```
ST    Z+,R30
ST    Z+,R31
ST    -Z,R30
ST    -Z,R31
```

動作	注釈
① DS(Z) ← Rr	;
② DS(Z) ← Rr, Z ← Z + 1	; 事後増加
③ Z ← Z – 1, DS(Z) ← Rr	; 事前減少
④ DS(Z+q) ← Rr	; 変位(q)付き

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
① ST	Z,Rr	r=0～31	PC ← PC + 1
② ST	Z+,Rr	r=0～31	PC ← PC + 1
③ ST	-Z,Rr	r=0～31	PC ← PC + 1
④ STD	Z+q,Rr	r=0～31, q=0～63	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	0	0	0	1	r4	r3	r2	r1	r0	0	0	0	0	① 注: STD Z+q,Rr命令と等価です。
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	0	0	0	1	②
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	0	0	1	0	③
1	0	q5	0	q4	q3	1	r4	r3	r2	r1	r0	0	q2	q1	q0	④

5.116.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数　1 (2ﾊﾞｲﾄ)

表5-116. 実行周期数

名称		AVRe	AVRxm	AVRxt	AVRrc
周期数	①	2 (注1)	1 (注1)	1 (注2)	1
	②	2 (注1)	1 (注1)	1 (注2)	1
	③	2 (注1)	2 (注1)	1 (注2)	2
	④	2 (注1)	2 (注1)	1 (注2)	利用不可

注1: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、外部RAMｱｸｾｽに対して有効ではありません。

注2: ﾃﾞｰﾀ ﾒﾓﾘ ｱｸｾｽに対するｸﾛｯｸ数は内部SRAMｱｸｾｽと仮定し、NVMへのｱｸｾｽに対して有効ではありません。NVMをｱｸｾｽする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄでNVMCTRL章をご覧ください。

例:

```
        CLR     R31             ; Zﾚｼﾞｽﾀ上位を設定
        LDI     R30, $60        ; Zﾚｼﾞｽﾀ下位を設定($60)
        ST      Z+, R0          ; R0の内容をﾃﾞｰﾀ空間$0060に設定、ﾎﾟｲﾝﾀ進行
        ST      Z, R1           ; R1の内容をﾃﾞｰﾀ空間$0061に設定
        LDI     R30, $63        ; Zﾚｼﾞｽﾀ下位を設定($63)
        ST      Z, R2           ; R2の内容をﾃﾞｰﾀ空間$0063に設定
        ST      -Z, R3          ; ﾎﾟｲﾝﾀ後退、R3の内容をﾃﾞｰﾀ空間$0062に設定
        STD     Z+2, R4         ; R4の内容をﾃﾞｰﾀ空間$0064に設定
```

5.117. STS – データ空間直接指定で汎用作業レジスタを設定 (Store Direct to data space)

5.117.1. 説明

汎用作業レジスタ(Rr)から1バイトをデータ空間に格納します。データ空間は通常、レジスタ ファイル、I/Oメモリ、SRAMから成り、データ空間の詳細な定義についてはデバイスのデータシートを参照してください。

16ビット アドレスが供給されなければなりません。メモリ アクセスは現在の64Kバイト データ セグメントに制限されます。STS命令は64Kバイトを超えるメモリをアクセスするのにRAMPDレジスタを使います。64Kバイトを超えるデータ空間を持つデバイスで他のデータ セグメントをアクセスするにはI/O空間内のレジスタのRAMPDが変更されなければなりません。

この命令は全てのデバイスで利用可能な訳ではありません。「追補A」を参照してください。

動作

DS(k) ← Rr

書式	オペランド	プログラム カウンタ
STS k,Rr	r=0～31, k=0～65535	PC ← PC + 1

機械語 (32ビット)

1	0	0	1	0	0	1	r4	r3	r2	r1	r0	0	0	0	0
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0

5.117.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 2 (4バイト)

表5-117. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	2 (注1)	2 (注1)	2 (注2)	利用不可

注1: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、外部RAMアクセスに対して有効ではありません。

注2: データ メモリ アクセスに対するクロック数は内部SRAMアクセスと仮定し、NVMへのアクセスに対して有効ではありません。NVMをアクセスする時に最低1つの余分な周期が追加されなければなりません。追加時間はNVM単位部実装に依存して変わります。より多くの情報については具体的なデバイスのデータシートでNVMCTRL章をご覧ください。

例:

```
        LDS     R2, $FF00       ; データ空間$FF00の内容をR2に取得
        ADD     R2, R1          ; R2とR1を加算
        STS     $FF00, R2       ; 書き戻し
```

5.118. STS (AVRrc) – SRAMへ直接指定で汎用作業ﾚｼﾞｽﾀを設定 (Store Direct to SRAM)

5.118.1. 説明

汎用作業ﾚｼﾞｽﾀから1ﾊﾞｲﾄをﾃﾞｰﾀ空間に格納します。ﾃﾞｰﾀ空間は通常、ﾚｼﾞｽﾀ ﾌｧｲﾙ、I/Oﾒﾓﾘ、SRAMから成り、ﾃﾞｰﾀ空間の詳細な定義についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

7ﾋﾞｯﾄのｱﾄﾞﾚｽが供給されなければなりません。命令内で与えられるｱﾄﾞﾚｽは次のようにﾃﾞｰﾀ空間ｱﾄﾞﾚｽを符号化します。

ADDR7～0 = (k4,k4,k6,k5,k3,k2,k1,k0)

ﾒﾓﾘ ｱｸｾｽは$0040～$00BFのｱﾄﾞﾚｽ範囲に制限されます。

この命令は全てのﾃﾞﾊﾞｲｽで利用可能な訳ではありません。「追補A」を参照してください。

動作

DS(k) ← Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
STS	k,Rr	r=16～31,$0040≦k≦$00BF	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	1	1	1	k6	k5	k4	r3	r2	r1	r0	k3	k2	k1	k0

注: R0～R15のﾚｼﾞｽﾀはR16～R31に割り当て直されます。

5.118.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

命令語数 1 (2ﾊﾞｲﾄ)

表5-118. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	利用不可	利用不可	利用不可	1

例:

```
        LDS     R18, $0040      ; ﾃﾞｰﾀ空間$0040の内容をR18に取得
        ADD     R18, R17        ; R18とR17を加算
        STS     $0040, R18      ; 書き戻し
```

5.119. SUB - 汎用作業ﾚｼﾞｽﾀ間の減算 (Subtract without Carry)

5.119.1. 説明

2つの汎用作業ﾚｼﾞｽﾀ(Rd,Rr)を減算し、結果を転送先ﾚｼﾞｽﾀRdに配置。

動作

Rd ← Rd - Rr

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SUB	Rd,Rr	d=0～31, r=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	0	1	1	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

5.119.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H - $\overline{Rd3}$ AND Rr3 OR Rr3 AND R3 OR R3 AND $\overline{Rd3}$
ﾋﾞｯﾄ3からの借入有りで設定(1)、その他で解除(0)。

S - N XOR V
2の補数での符号。

V - Rd7 AND $\overline{Rr7}$ AND $\overline{R7}$ OR $\overline{Rd7}$ AND Rr7 AND R7
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C - $\overline{Rd7}$ AND Rr7 OR Rr7 AND R7 OR R7 AND $\overline{Rd7}$
符号なしのRd＜Rrで設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-119. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            SUB     R4, R19         ; R4とR19を比較
            BRNE    NOMATCH         ; 不一致(Z=0)で分岐
                ﾚ
NOMATCH:    NOP                     ; 処理なし(不一致での分岐先)
```

5.120. SUBI - 汎用作業ﾚｼﾞｽﾀから即値定数の減算 (Subtract Immediate)

5.120.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)と即値定数(K)を減算し、結果を転送先ﾚｼﾞｽﾀRdに配置。この命令はR16～31で動き、X,Y,Zﾎﾟｲﾝﾀでの操作に非常に上手く適合します。

動作

Rd ← Rd - K

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SUBI	Rd,K	d=16～31, K=0～255	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	1	0	1	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0

5.120.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H - $\overline{Rd3}$ AND K3 OR K3 AND R3 OR R3 AND $\overline{Rd3}$
ﾋﾞｯﾄ3からの借入有りで設定(1)、その他で解除(0)。

S - N XOR V
2の補数での符号。

V - Rd7 AND $\overline{K7}$ AND $\overline{R7}$ OR $\overline{Rd7}$ AND K7 AND R7
2の補数での溢れ発生で設定(1)、その他で解除(0)。

N - R7
結果の最上位ﾋﾞｯﾄの複写値。

Z - $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

C - $\overline{Rd7}$ AND K7 OR K7 AND R7 OR R7 AND $\overline{Rd7}$
符号なしのRd＜Kで設定(1)、その他で解除(0)。

R(結果)は演算後のRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-120. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
            SUBI    R4, $11         ; R4 - $11
            BRNE    NOMATCH         ; R4≠$11(Z=0)で分岐
                ≀
NOMATCH:    NOP                     ; 処理なし(R4≠$11での分岐先)
```

(訳補) ADDI命令が存在しないため、本命令はSUBI Rd,Low(-K)として即値ﾊﾞｲﾄ加算の代用にも用いられます。その場合、ﾌﾗｸﾞの意味が本来の加算系命令とは異なりますので注意してください。

5.121. SWAP - 汎用作業ﾚｼﾞｽﾀの上下ﾆﾌﾞﾙ交換 (Swap Nibbles)

5.121.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)の上下ﾆﾌﾞﾙ(4ﾋﾞｯﾄ)を交換。

動作

Rd(7～4) ⇔ Rd(3～0)

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
SWAP	Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	1	0

5.121.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-121. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        INC     R1          ; R1の下位ﾆﾌﾞﾙを増加(+1)
        SWAP    R1          ; R1の上下ﾆﾌﾞﾙを交換
        INC     R1          ; R1の上位ﾆﾌﾞﾙを増加(+1)
        SWAP    R1          ; R1の上下ﾆﾌﾞﾙを交換(元位置復帰)
```

5.122. TST – 汎用作業ﾚｼﾞｽﾀのｾﾞﾛ、負検査 (Test for Zero or Minus)

5.122.1. 説明

汎用作業ﾚｼﾞｽﾀ(Rd)がｾﾞﾛ($00)または負か検査。ﾚｼﾞｽﾀとそれ自身間で論理積(AND)を実行。ﾚｼﾞｽﾀRdは無変化に留まります。(AND Rd,Rd命令と等価)

動作

Rd ← Rd AND Rd

書式	オﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
TST Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

0	0	1	0	0	0	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0

5.122.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

S – N XOR V
2の補数での符号。

V – 0
解除(0)。

N – R7
結果の最上位ﾋﾞｯﾄの複写値。

Z – $\overline{R7}$ AND $\overline{R6}$ AND $\overline{R5}$ AND $\overline{R4}$ AND $\overline{R3}$ AND $\overline{R2}$ AND $\overline{R1}$ AND $\overline{R0}$
結果が$00で設定(1)、その他で解除(0)。

R(結果)はRdと等しくなります。

命令語数 1 (2ﾊﾞｲﾄ)

表5-122. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
        TST     R0          ; R0検査
        BREQ    ZERO        ; R0=0で分岐
                ≀
ZERO:                       ; R0=0での分岐先
```

5.123. WDR - ウォチドッグ タイマ リセット (Watchdog Reset)

5.123.1. 説明

この命令はウォッチドッグ タイマをリセットします。この命令はウォッチドッグ前置分周器によって与えられた制限時間内に実行されなければなりません。ウォッチドッグ タイマ ハードウェア仕様をご覧ください。

動作

ウォッチドッグ タイマ再始動

書式	オペランド	プログラム カウンタ
WDR	なし	PC ← PC + 1

機械語 (16ビット)

1	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0

5.123.2. ステータス レジスタ(SREG)とブール式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2バイト)

表5-123. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	1	1	1	1

例:

```
WDR                    ; ウォッチドッグ タイマ リセット(再始動)
```

5.124. XCH - 交換 (Exchange)

5.124.1. 説明

汎用作業ﾚｼﾞｽﾀと間接指定のﾃﾞｰﾀ空間の間で1ﾊﾞｲﾄを交換。

ﾃﾞｰﾀ位置はﾚｼﾞｽﾀ ﾌｧｲﾙ内のZ(16ﾋﾞｯﾄ)ﾎﾟｲﾝﾀ ﾚｼﾞｽﾀによって指示されます。ﾒﾓﾘ ｱｸｾｽは現在の64Kﾊﾞｲﾄ ﾃﾞｰﾀ ｾｸﾞﾒﾝﾄに制限されます。64Kﾊﾞｲﾄを超えるﾃﾞｰﾀ空間を持つﾃﾞﾊﾞｲｽで他のﾃﾞｰﾀ ｾｸﾞﾒﾝﾄをｱｸｾｽするには、I/O領域のﾚｼﾞｽﾀでRAMPZが変更されなければなりません。

Zﾎﾟｲﾝﾀ ﾚｼﾞｽﾀはこの操作によって無変化のままです。この命令は特にSRAM内に格納された状態ﾋﾞｯﾄの読み書きに適します。

動作

DS(Z) ⇔ Rd

書式		ｵﾍﾟﾗﾝﾄﾞ	ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ
XCH	Z,Rd	d=0～31	PC ← PC + 1

機械語 (16ﾋﾞｯﾄ)

1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	0	0

5.124.2. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ(SREG)とﾌﾞｰﾙ式

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

命令語数 1 (2ﾊﾞｲﾄ)

表5-124. 実行周期数

名称	AVRe	AVRxm	AVRxt	AVRrc
周期数	利用不可	2	利用不可	利用不可

6. 追補A デバイス コア概要

6.1. コア説明

表は各種コア間で変わる全ての命令を一覧にし、それがコアに含まれるかを記します。命令が表にない場合、それは全てのコアに含まれます。

表6-1. コア説明

命令	AVR	AVRe	AVRe+	AVRxm	AVRxt	AVRrc
ADIW	○	○	○	○	○	×
BREAK	×	○	○	○	○	○
CALL	×	○	○	○	○	×
DES	×	×	×	○	×	×
EICALL	×	×	○	○	○ (注)	×
EIJMP	×	×	○	○	○ (注)	×
ELPM	×	×	○	○	○	×
FMUL	×	×	○	○	○	×
FMULS	×	×	○	○	○	×
FMULSU	×	×	○	○	○	×
JMP	×	○	○	○	○	×
LAC	×	×	×	○	×	×
LAS	×	×	×	○	×	×
LAT	×	×	×	○	×	×
LDD	○	○	○	○	○	×
LPM	○	○	○	○	○	×
LPM Rd,Z	×	○	○	○	○	×
LPM Rd,Z+	×	○	○	○	○	×
MOVW	×	○	○	○	○	×
MUL	×	×	○	○	○	×
MULS	×	×	○	○	○	×
MULSU	×	×	○	○	○	×
SBIW	○	○	○	○	○	×
SPM	×	○	○	○	○	×
SPM Z+	×	×	×	○	○	×
STD	○	○	○	○	○	×
XCH	×	×	×	○	×	×

注: EIJMPとEICALLの命令は128Kバイトよりも大きなメモリを持つデバイスに対して有効です。

6.2. デバイス表

6.2.1. megaAVR®デバイス

表6-2. megaAVR®デバイス

デバイス	コア	欠落命令
AT90CAN128	AVRe+	EIJMP, EICALL
AT90CAN32	AVRe+	ELPM, EIJMP, EICALL
AT90CAN64	AVRe+	
ATmega1280	AVRe+	
ATmega1281	AVRe+	
ATmega1284P	AVRe+	
ATmega1284RFR2	AVRe+	
ATmega1284	AVRe+	EIJMP, EICALL
ATmega128A	AVRe+	
ATmega128RFA1	AVRe+	
ATmega128RFR2	AVRe+	
ATmega128	AVRe+	
ATmega1608	AVRxt	ELPM, SPM, SPM Z+, EIJMP, EICALL
ATmega1609	AVRxt	
ATmega162	AVRe+	
ATmega164A	AVRe+	
ATmega164PA	AVRe+	
ATmega164P	AVRe+	
ATmega165A	AVRe+	
ATmega165PA	AVRe+	
ATmega165P	AVRe+	
ATmega168A	AVRe+	
ATmega168PA	AVRe+	
ATmega168PB	AVRe+	
ATmega168P	AVRe+	
ATmega168	AVRe+	ELPM, EIJMP, EICALL
ATmega169A	AVRe+	
ATmega169PA	AVRe+	
ATmega169P	AVRe+	
ATmega16A	AVRe+	
ATmega16HVA	AVRe+	
ATmega16HVB	AVRe+	
ATmega16HVB改訂B	AVRe+	
ATmega16M1	AVRe+	
ATmega16U2	AVRe+	
ATmega16U4	AVRe+	
ATmega16	AVRe+	
ATmega2560	AVRe+	
ATmega2561	AVRe+	
ATmega2564RFR2	AVRe+	
ATmega256RFR2	AVRe+	
ATmega3208	AVRxt	ELPM, SPM, SPM Z+, EIJMP, EICALL
ATmega3209	AVRxt	
ATmega324A	AVRe+	
ATmega324PA	AVRe+	ELPM, EIJMP, EICALL
ATmega324PB	AVRe+	

右表へ続く

左表から続く

デバイス	コア	欠落命令
ATmega324P	AVRe+	
ATmega3250A	AVRe+	
ATmega3250PA	AVRe+	
ATmega3250P	AVRe+	
ATmega3250	AVRe+	
ATmega325A	AVRe+	
ATmega325PA	AVRe+	
ATmega325P	AVRe+	
ATmega325	AVRe+	
ATmega328PB	AVRe+	
ATmega328P	AVRe+	
ATmega328	AVRe+	
ATmega3290A	AVRe+	
ATmega3290PA	AVRe+	
ATmega3290P	AVRe+	ELPM, EIJMP, EICALL
ATmega3290	AVRe+	
ATmega329A	AVRe+	
ATmega329PA	AVRe+	
ATmega329P	AVRe+	
ATmega329	AVRe+	
ATmega32A	AVRe+	
ATmega32C1	AVRe+	
ATmega32HVB	AVRe+	
ATmega32HVB改訂B	AVRe+	
ATmega32M1	AVRe+	
ATmega32U2	AVRe+	
ATmega32U4	AVRe+	
ATmega32	AVRe+	
ATmega406	AVRe+	
ATmega4808	AVRxt	ELPM, SPM, SPM Z+, EIJMP, EICALL
ATmega4809	AVRxt	
ATmega48A	AVRe+	
ATmega48PA	AVRe+	CALL, JMP, ELPM, EIJMP, EICALL
ATmega48PB	AVRe+	
ATmega48P	AVRe+	
ATmega48	AVRe+	
ATmega640	AVRe+	
ATmega644A	AVRe+	
ATmega644PA	AVRe+	
ATmega644P	AVRe+	
ATmega644RFR2	AVRe+	ELPM, EIJMP, EICALL
ATmega644	AVRe+	
ATmega6450A	AVRe+	
ATmega6450P	AVRe+	
ATmega6450	AVRe+	
ATmega645A	AVRe+	

次頁へ続く

前頁から続く

デバイス	コア	欠落命令
ATmega645P	AVRe+	
ATmega645	AVRe+	
ATmega6490A	AVRe+	
ATmega6490P	AVRe+	
ATmega6490	AVRe+	
ATmega649A	AVRe+	
ATmega649P	AVRe+	ELPM, EIJMP, EICALL
ATmega649	AVRe+	
ATmega64A	AVRe+	
ATmega64C1	AVRe+	
ATmega64HVE2	AVRe+	
ATmega64M1	AVRe+	
ATmega64RFR2	AVRe+	
ATmega64	AVRe+	
ATmega808	AVRxt	CALL, JMP, ELPM, SPM, SPM Z+, EIJMP, EICALL
ATmega809	AVRxt	
ATmega8515	AVRe+	BREAK, CALL, JMP, ELPM, EIJMP, EICALL
ATmega8535	AVRe+	
ATmega88A	AVRe+	
ATmega88PA	AVRe+	CALL, JMP, ELPM, EIJMP, EICALL
ATmega88PB	AVRe+	
ATmega88P	AVRe+	
ATmega88	AVRe+	
ATmega8A	AVRe+	BREAK, CALL, JMP, ELPM, EIJMP, EICALL
ATmega8	AVRe+	
ATmega8HVA	AVRe+	CALL, JMP, ELPM, EIJMP, EICALL
ATmega8U2	AVRe+	
AT90PWM1	AVRe+	
AT90PWM216	AVRe+	ELPM, EIJMP, EICALL
AT90PWM316	AVRe+	
AT90PWM2B	AVRe+	
AT90PWM3B	AVRe+	CALL, JMP, ELPM, EIJMP, EICALL
AT90PWM81	AVRe+	
AT90PWM161	AVRe+	
AT90USB82	AVRe+	
AT90USB162	AVRe+	
AT90USB646	AVRe+	ELPM, EIJMP, EICALL
AT90USB647	AVRe+	
AT90USB1286	AVRe+	EIJMP, EICALL
AT90USB1287	AVRe+	

6.2.2. tinyAVR®デバイス

表6-3. tinyAVR®デバイス

デバイス	コア	欠落命令
ATtiny10	AVRrc	BREAK
ATtiny11	AVR	BREAK, LPM Z+ ADIW, SBIW, IJMP, ICALL, LD X, LD Y, LD -Z, LD Z+, LD
ATtiny12	AVR	
ATtiny15	AVR	
ATtiny13	AVRe	CALL, JMP, ELPM
ATtiny13A	AVRe	
ATtiny102	AVRrc	
ATtiny104	AVRrc	
ATtiny1604	AVRxt	
ATtiny1606	AVRxt	
ATtiny1607	AVRxt	
ATtiny1614	AVRxt	ELPM, EIJMP, EICALL, SPM, SPM Z+
ATtiny1616	AVRxt	
ATtiny1617	AVRxt	
ATtiny1624	AVRxt	
ATtiny1626	AVRxt	
ATtiny1627	AVRxt	
ATtiny1634	AVRe	
ATtiny167	AVRe	
ATtiny20	AVRrc	BREAK
ATtiny202	AVRxt	
ATtiny204	AVRxt	CALL, JMP, ELPM, SPM, SPM Z+, EIJMP, EICALL
ATtiny212	AVRxt	
ATtiny214	AVRxt	
ATtiny2313	AVRe	
ATtiny2313A	AVRe	
ATtiny24	AVRe	CALL, JMP, ELPM
ATtiny24A	AVRe	
ATtiny25	AVRe	
ATtiny26	AVR	BREAK
ATtiny261	AVRe	CALL, JMP, ELPM
ATtiny261A	AVRe	
ATtiny3216	AVRxt	
ATtiny3217	AVRxt	CALL, JMP, ELPM, SPM, SPM Z+, EIJMP, EICALL
ATtiny3224	AVRxt	
ATtiny3226	AVRxt	
ATtiny3227	AVRxt	
ATtiny40	AVRrc	
ATtiny402	AVRxt	
ATtiny404	AVRxt	
ATtiny406	AVRxt	
ATtiny412	AVRxt	
ATtiny414	AVRxt	CALL, JMP, ELPM, SPM, SPM Z+, EIJMP, EICALL
ATtiny416	AVRxt	
ATtiny417	AVRxt	
ATtiny424	AVRxt	
ATtiny426	AVRxt	
ATtiny427	AVRxt	

右表へ続く

左表から続く

デバイス	コア	欠落命令
ATtiny4313	AVRe	
ATtiny43U	AVRe	
ATtiny44	AVRe	
ATtiny441	AVRe	
ATtiny441A	AVRe	CALL, JMP, ELPM
ATtiny45	AVRe	
ATtiny461	AVRe	
ATtiny461A	AVRe	
ATtiny48	AVRe	
ATtiny4	AVRrc	BREAK
ATtiny5	AVRrc	
ATtiny804	AVRxt	
ATtiny806	AVRxt	
ATtiny807	AVRxt	
ATtiny814	AVRxt	CALL, JMP, ELPM, SPM, SPM Z+, EIJMP, EICALL
ATtiny816	AVRxt	
ATtiny817	AVRxt	
ATtiny824	AVRxt	
ATtiny826	AVRxt	
ATtiny827	AVRxt	
ATtiny828	AVRe	
ATtiny84	AVRe	
ATtiny84A	AVRe	
ATtiny841	AVRe	
ATtiny85	AVRe	CALL, JMP, ELPM
ATtiny861	AVRe	
ATtiny861A	AVRe	
ATtiny87	AVRe	
ATtiny88	AVRe	
ATtiny9	AVRrc	BREAK

6.2.3. AVR-Dxデバイス

表6-4. AVR-Dxデバイス

デバイス	コア	欠落命令
AVR32DA28	AVRxt	
AVR32DA32	AVRxt	
AVR32DA48	AVRxt	
AVR64DA28	AVRxt	ELPM, EIJMP, EICALL
AVR64DA32	AVRxt	
AVR64DA48	AVRxt	
AVR64DA64	AVRxt	
AVR128DA28	AVRxt	
AVR128DA32	AVRxt	EIJMP, EICALL
AVR128DA48	AVRxt	
AVR128DA64	AVRxt	
AVR32DB28	AVRxt	
AVR32DB32	AVRxt	
AVR32DB48	AVRxt	
AVR64DB28	AVRxt	ELPM, EIJMP, EICALL
AVR64DB32	AVRxt	
AVR64DB48	AVRxt	
AVR64DB64	AVRxt	
AVR128DB28	AVRxt	
AVR128DB32	AVRxt	EIJMP, EICALL
AVR128DB48	AVRxt	
AVR128DB64	AVRxt	
AVR32DD28	AVRxt	
AVR32DD32	AVRxt	
AVR32DD48	AVRxt	
AVR64DD28	AVRxt	ELPM, EIJMP, EICALL
AVR64DD32	AVRxt	
AVR64DD48	AVRxt	
AVR64DD64	AVRxt	
AVR128DD28	AVRxt	
AVR128DD32	AVRxt	EIJMP, EICALL
AVR128DD48	AVRxt	
AVR128DD64	AVRxt	
AVR16DU14	AVRxt	
AVR16DU20	AVRxt	ELPM, EIJMP, EICALL
AVR16DU28	AVRxt	
AVR16DU32	AVRxt	
AVR32EA28	AVRxt	
AVR32EA32	AVRxt	
AVR32EA48	AVRxt	ELPM, EIJMP, EICALL
AVR64EA28	AVRxt	
AVR64EA32	AVRxt	
AVR64EA48	AVRxt	
AVR16EB14	AVRxt	
AVR16EB20	AVRxt	
AVR16EB28	AVRxt	ELPM, EIJMP, EICALL
AVR16EB32	AVRxt	
AVR32EB14	AVRxt	

右表へ続く

左表から続く

デバイス	コア	欠落命令
AVR32EB20	AVRxt	
AVR32EB28	AVRxt	ELPM, EIJMP, EICALL
AVR32EB32	AVRxt	
AVR32SD20	AVRxt	
AVR32SD28	AVRxt	
AVR32SD32	AVRxt	EIJMP, EICALL
AVR64SD28	AVRxt	
AVR64SD32	AVRxt	
AVR64SD48	AVRxt	

6.2.4. XMEGA®デバイス

表6-5. XMEGA®デバイス

デバイス	コア	欠落命令
ATxmega16A4	AVRxm	LAC, LAT, LAS, XCH, ELPM, EIJMP, EICALL
ATxmega32A4	AVRxm	
ATxmega64A1	AVRxm	LAC, LAT, LAS, XCH, EIJMP, EICALL
ATxmega64A3	AVRxm	
ATxmega128A1	AVRxm	
ATxmega128A3	AVRxm	
ATxmega192A3	AVRxm	LAC, LAT, LAS, XCH
ATxmega256A3	AVRxm	
ATxmega256A3B	AVRxm	
ATxmega16A4U	AVRxm	ELPM, EIJMP, EICALL
ATxmega32A4U	AVRxm	
ATxmega64A1U	AVRxm	
ATxmega64A3U	AVRxm	EIJMP, EICALL
ATxmega64A4U	AVRxm	
ATxmega128A1U	AVRxm	
ATxmega128A3U	AVRxm	
ATxmega128A4U	AVRxm	
ATxmega192A3U	AVRxm	
ATxmega256A3U	AVRxm	
ATxmega256A3BU	AVRxm	
ATxmega64B1	AVRxm	EIJMP, EICALL
ATxmega64B3	AVRxm	
ATxmega128B1	AVRxm	
ATxmega128B3	AVRxm	
ATxmega16C4	AVRxm	
ATxmega32C3	AVRxm	ELPM, EIJMP, EICALL
ATxmega32C4	AVRxm	
ATxmega64C3	AVRxm	EIJMP, EICALL
ATxmega128C3	AVRxm	
ATxmega192C3	AVRxm	
ATxmega256C3	AVRxm	
ATxmega384C3	AVRxm	
ATxmega16D4	AVRxm	LAC, LAT, LAS, XCH, ELPM, EIJMP, EICALL
ATxmega32D3	AVRxm	
ATxmega32D4	AVRxm	
ATxmega64D3	AVRxm	LAC, LAT, LAS, XCH, EIJMP, EICALL
ATxmega64D4	AVRxm	
ATxmega128D3	AVRxm	
ATxmega128D4	AVRxm	
ATxmega192D3	AVRxm	LAC, LAT, LAS, XCH
ATxmega256D3	AVRxm	
ATxmega384D3	AVRxm	
ATxmega8E5	AVRxm	
ATxmega16E5	AVRxm	ELPM, EIJMP, EICALL
ATxmega32E5	AVRxm	

6.2.5. 車載デバイス

表6-6. 車載デバイス

デバイス	コア	欠落命令
ATA5272	AVRe	CALL, JMP, ELPM
ATA5505	AVRe	
ATA5700M322	AVRe+	
ATA5720M322	AVRe+	
ATA5781	AVRe+	
ATA5782	AVRe+	
ATA5783	AVRe+	ELPM, EIJMP, EICALL
ATA5787	AVRe+	
ATA5790	AVRe+	
ATA5790N	AVRe+	
ATA5791	AVRe+	
ATA5795	AVRe+	CALL, JMP, ELPM, EIJMP, EICALL
ATA5831	AVRe+	
ATA5832	AVRe+	ELPM, EIJMP, EICALL
ATA5833	AVRe+	
ATA5835	AVRe+	
ATA6285	AVRe+	
ATA6286	AVRe+	CALL, JMP, ELPM, EIJMP, EICALL
ATA6612C	AVRe+	
ATA6613C	AVRe+	ELPM, EIJMP, EICALL
ATA6614Q	AVRe+	
ATA6616C	AVRe	CALL, JMP, ELPM
ATA6617C	AVRe	
ATA664251	AVRe	
ATA8210	AVRe+	
ATA8215	AVRe+	ELPM, EIJMP, EICALL
ATA8510	AVRe+	
ATA8515	AVRe+	
ATtiny416auto	AVRxt	CALL, JMP, ELPM, SPM, SPM Z+, EIJMP, EICALL

A. 機械語命令の構造

ｱｾﾝﾌﾞﾘ言語に於けるﾆｰﾓﾆｯｸ定義は本来の機械語命令に加え、使用上の便宜から以下の派生ﾆｰﾓﾆｯｸが含まれています。

① BRBC b,k命令のbの値に対して各々、BRID k (BRBC 7,k), BRTC k (BRBC 6,k), BRHC k (BRBC 5,k), BRGE k (BRBC 4,k), BRVC k (BRBC 3,k), BRPL k (BRBC 2,k), BRNE k (BRBC 1,k), BRSH k (BRBC 0,k), BRCC k (BRBC 0,k)の9種類の派生ﾆｰﾓﾆｯｸが定義されています。

② 同様にBRBS b,k命令のbの値に対して各々、BRIE k (BRBS 7,k), BRTS k (BRBS 6,k), BRHS k (BRBS 5,k), BRLT k (BRBS 4,k), BRVS k (BRBS 3,k), BRMI k (BRBS 2,k), BREQ k (BRBS 1,k), BRLO k (BRBS 0,k), BRCS k (BRBS 0,k)の9種類の派生ﾆｰﾓﾆｯｸが定義されています。

③ 同様にBCLR s命令のsの値に対して各々、CLI (BCLR 7), CLT (BCLR 6), CLH (BCLR 5), CLS (BCLR 4), CLV (BCLR 3), CLN (BCLR 2), CLZ (BCLR 1), CLC (BCLR 0)の8種類の派生ﾆｰﾓﾆｯｸが定義されています。

④ 同様にBSET s命令のsの値に対して各々、SEI (BSET 7), SET (BSET 6), SEH (BSET 5), SES (BSET 4), SEV (BSET 3), SEN (BSET 2), SEZ (BSET 1), SEC (BSET 0)の8種類の派生ﾆｰﾓﾆｯｸが定義されています。

⑤ LDI Rd,K命令のKの値に対してK=$FF専用の派生ﾆｰﾓﾆｯｸ、SER Rd (LDI Rd,$FF)が定義されています。

⑥ ANDI Rd,K命令のKの値に対して全ﾋﾞｯﾄ論理反転値となる専用の派生ﾆｰﾓﾆｯｸ、CBR Rd,K (ANDI Rd,$\overline{K}$)が定義されています。

⑦ ORI Rd,K命令に対して別名の派生ﾆｰﾓﾆｯｸ、SBR Rd,K (ORI Rd,K)が定義されています。

⑧ EOR Rd,Rr命令に対して別名の派生ﾆｰﾓﾆｯｸ、CLR Rd (EOR Rd,Rd)が定義されています。

⑨ AND Rd,Rr命令に対して別名の派生ﾆｰﾓﾆｯｸ、TST Rd (AND Rd,Rd)が定義されています。

⑩ ADC Rd,Rr命令に対して別名の派生ﾆｰﾓﾆｯｸ、ROL Rd (ADC Rd,Rd)が定義されています。

⑪ ADD Rd,Rr命令に対して別名の派生ﾆｰﾓﾆｯｸ、LSL Rd (ADD Rd,Rd)が定義されています。

⑫ STD Y+q,RrとSTD Z+q,Rr命令のqに対してq=0専用の派生ﾆｰﾓﾆｯｸ、ST Y,Rr (STD Y+0,Rr)とST Z,Rr (STD Z+0,Rr)が定義されています。

⑬ LDD Rd,Y+qとLDD Rd,Z+q命令のqに対してq=0専用の派生ﾆｰﾓﾆｯｸ、LD Rd,Y (LDD Rd,Y+0)とLD Rd,Z (LDD Rd,Z+0)が定義されています。

上記⑬同様の理由から、LPMとELPM命令は各々LPM RdとELPM Rdの機械語命令で実現可能ですが、これは行われておりません。LPMやELPMだけを実装するﾃﾞﾊﾞｲｽが存在するため、実装する命令復号回路の関係で、これらが独立した機械語命令として存在するものと予測されます。

上記から機械語命令数はｱｾﾝﾌﾞﾘ言語命令数の140から45減って95種類になります。

xmega系列では暗号化/解読命令DESと、SPM命令のｵﾍﾟﾗﾝﾄﾞにZ+が追加されました。これらを含めると93種類になります。

複数語(ﾜｰﾄﾞ)命令は(本資料の)最上位語がﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ ｱﾄﾞﾚｽの下位側(ﾋﾞｯｸ ｴﾝﾃﾞｨｱﾝ)になります。これは現在定義されている複数語命令が2語命令のみで、これら全ての命令がｱﾄﾞﾚｽを含み、このｱﾄﾞﾚｽの上下関係を命令語順に反映させた結果と思われます。一般的にAVRではﾚｼﾞｽﾀ対の上下順のように使用者が充分に意識する部分は上位が上位ｱﾄﾞﾚｽ(ﾘﾄﾙ ｴﾝﾃﾞｨｱﾝ)ですが、あまり意識しない部分(ｽﾀｯｸ上のPC値など)はﾋﾞｯｸ ｴﾝﾃﾞｨｱﾝですので、これらの順番には充分な注意が必要です。

命令の復号は基本的に上位ﾋﾞｯﾄ側からの組み合わせで分類されていますが、転送元と転送先のﾋﾞｯﾄ位置を固定化して考えられているため、一部でそれらの空きﾋﾞｯﾄ位置が命令の復号に使われることによって、必ずしも上位側からの分類でない部分もあります。これは「ｺｰﾄﾞ順機械語命令一覧」を参考にしてください。

ｵﾍﾟﾗﾝﾄﾞ部の種類は基本形が17種ですが、これらには転送元と転送先の逆関係も含まれます。これらは「ｵﾍﾟﾗﾝﾄﾞ種別一覧」を参照してください。

図A-1. 代表的な機械語形式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	x	x	x	x	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0

標準AVR8ｺｱとXMEGAｺｱは機械語符号的にXMEGAｺｱで新設のDES命令が追加されているだけです。これに対してAVRrcｺｱはﾌﾗｯｼｭ ﾒﾓﾘ領域をﾃﾞｰﾀ(SRAM)空間に割り当てるためにLPM,ELPM命令が削除されてます。また、LDS,STS命令の動作が変更になり、これらの命令はﾃﾞｰﾀ(SRAM)空間上の$0040～$00BFの128ﾊﾞｲﾄだけがｱｸｾｽでき、これに伴ってこれらの命令語長が2語から1語に変更され、機械語符号的には標準AVR8ｺｱに於ける変位付き間接ｱﾄﾞﾚｽ指定(LDD/STD Y/Z+q)に割り当てられています。このためにAVRrcｺｱでは変位付き間接ｱﾄﾞﾚｽ指定が削除されています。

A-1. ｺｰﾄﾞ順機械語一覧 (1/5)

	1	1	1	1	1	x	x	x	x	x	x	x	1	x	x	x	未定義
	1	1	1	1	1	1	1	r4	r3	r2	r1	r0	0	b2	b1	b0	SBRS Rr,b
	1	1	1	1	1	1	0	r4	r3	r2	r1	r0	0	b2	b1	b0	SBRC Rr,b
	1	1	1	1	1	0	1	d4	d3	d2	d1	d0	0	b2	b1	b0	BST Rd,b
	1	1	1	1	1	0	0	d4	d3	d2	d1	d0	0	b2	b1	b0	BLD Rd,b
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	s2	s1	s0	BRBC s,k
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	1	1	BRID k (BRBC 7,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	1	0	BRTC k (BRBC 6,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	0	1	BRHC k (BRBC 5,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	1	0	0	BRGE k (BRBC 4,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	1	1	BRVC k (BRBC 3,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	1	0	BRPL k (BRBC 2,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	0	1	BRNE k (BRBC 1,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	0	0	BRSH k (BRBC 0,k)
	1	1	1	1	0	1	k6	k5	k4	k3	k2	k1	k0	0	0	0	BRCC k (BRBC 0,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	s2	s1	s0	BRBS s,k
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	1	1	BRIE k (BRBS 7,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	1	0	BRTS k (BRBS 6,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	0	1	BRHS k (BRBS 5,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	1	0	0	BRLT k (BRBS 4,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	1	1	BRVS k (BRBS 3,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	1	0	BRMI k (BRBS 2,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	0	1	BREQ k (BRBS 1,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	0	0	BRLO k (BRBS 0,k)
	1	1	1	1	0	0	k6	k5	k4	k3	k2	k1	k0	0	0	0	BRCS k (BRBS 0,k)
	1	1	1	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	LDI Rd,K
	1	1	1	0	1	1	1	1	d3	d2	d1	d0	1	1	1	1	SER Rd (LDI Rd,$FF)
	1	1	0	1	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0	RCALL k
	1	1	0	0	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0	RJMP k
	1	0	1	1	1	A5	A4	r4	r3	r2	r1	r0	A3	A2	A1	A0	OUT A,Rr
	1	0	1	1	0	A5	A4	d4	d3	d2	d1	d0	A3	A2	A1	A0	IN Rd,A
	1	0	q5	0	q4	q3	1	r4	r3	r2	r1	r0	1	q2	q1	q0	STD Y+q,Rr (注: AVRrcはLDS/STS k,Rr)
	1	0	0	0	0	0	1	r4	r3	r2	r1	r0	1	0	0	0	ST Y,Rr (STD Y+0,Rr)
	1	0	q5	0	q4	q3	1	r4	r3	r2	r1	r0	0	q2	q1	q0	STD Z+q,Rr (注: AVRrcはLDS/STS k,Rr)
	1	0	0	0	0	0	1	r4	r3	r2	r1	r0	0	0	0	0	ST Z,Rr (STD Z+0,Rr)
	1	0	q5	0	q4	q3	0	d4	d3	d2	d1	d0	1	q2	q1	q0	LDD Rd,Y+q (注: AVRrcはLDS/STS k,Rr)
	1	0	0	0	0	0	0	d4	d3	d2	d1	d0	1	0	0	0	LD Rd,Y (LDD Rd,Y+0)
	1	0	q5	0	q4	q3	0	d4	d3	d2	d1	d0	0	q2	q1	q0	LDD Rd,Z+q (注: AVRrcはLDS/STS k,Rr)
	1	0	0	0	0	0	0	d4	d3	d2	d1	d0	0	0	0	0	LD Rd,Z (LDD Rd,Z+0)

ｺｰﾄﾞ順機械語一覧 (2/5)

MUL Rd,Rr	1	0	0	1	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0
SBIS A,b	1	0	0	1	1	0	1	1	A4	A3	A2	A1	A0	b2	b1	b0
SBI A,b	1	0	0	1	1	0	1	0	A4	A3	A2	A1	A0	b2	b1	b0
SBIC A,b	1	0	0	1	1	0	0	1	A4	A3	A2	A1	A0	b2	b1	b0
CBI A,b	1	0	0	1	1	0	0	0	A4	A3	A2	A1	A0	b2	b1	b0
SBIW Rd,K (注: Rd=d2,1×2+24)	1	0	0	1	0	1	1	1	K5	K4	d2	d1	K3	K2	K1	K0
ADIW Rd,K (注: Rd=d2,1×2+24)	1	0	0	1	0	1	1	0	K5	K4	d2	d1	K3	K2	K1	K0
CALL k	1	0	0	1	0	1	0	k21	k20	k19	k18	k17	1	1	1	k16
	k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0
JMP k	1	0	0	1	0	1	0	k21	k20	k19	k18	k17	1	1	0	k16
	k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0
未定義	1	0	0	1	0	1	0	1	x	x	x	x	1	0	1	1
DES	1	0	0	1	0	1	0	0	K3	K2	K1	K0	1	0	1	1
DEC Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	1	0	1	0
未定義	1	0	0	1	0	1	0	x	1	x	x	x	1	0	0	1
未定義	1	0	0	1	0	1	0	x	x	1	x	x	1	0	0	1
未定義	1	0	0	1	0	1	0	x	x	x	1	x	1	0	0	1
EICALL	1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1
ICALL	1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1
SPM Z+	1	0	0	1	0	1	0	1	1	1	1	1	1	0	0	0
SPM	1	0	0	1	0	1	0	1	1	1	1	0	1	0	0	0
ELPM	1	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0
LPM	1	0	0	1	0	1	0	1	1	1	0	0	1	0	0	0
未定義	1	0	0	1	0	1	0	1	1	0	1	1	1	0	0	0
WDR	1	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0
BREAK	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0
SLEEP	1	0	0	1	0	1	0	1	1	0	0	0	1	0	0	0
未定義	1	0	0	1	0	1	0	1	0	1	x	x	1	0	0	0
未定義	1	0	0	1	0	1	0	1	0	x	1	x	1	0	0	0
RETI	1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0
RET	1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0
ROR Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	1	1	1
LSR Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	1	1	0
ASR Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	1	0	1
未定義	1	0	0	1	0	1	0	x	x	x	x	x	0	1	0	0
INC Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	1	1
SWAP Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	1	0
NEG Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	0	1
COM Rd	1	0	0	1	0	1	0	d4	d3	d2	d1	d0	0	0	0	0

ｺｰﾄﾞ順機械語一覧 (3/5)

1	0	0	1	0	1	0	0	x	x	x	x	1	1	x	x	未定義
1	0	0	1	0	1	0	0	x	x	x	x	1	0	1	x	未定義
1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	EIJMP
1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	IJMP
1	0	0	1	0	1	0	0	1	s2	s1	s0	1	0	0	0	BCLR s
1	0	0	1	0	1	0	0	1	1	1	1	1	0	0	0	CLI (BCLR 7)
1	0	0	1	0	1	0	0	1	1	1	0	1	0	0	0	CLT (BCLR 6)
1	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0	CLH (BCLR 5)
1	0	0	1	0	1	0	0	1	1	0	0	1	0	0	0	CLS (BCLR 4)
1	0	0	1	0	1	0	0	1	0	1	1	1	0	0	0	CLV (BCLR 3)
1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	0	CLN (BCLR 2)
1	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0	CLZ (BCLR 1)
1	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	CLC (BCLR 0)
1	0	0	1	0	1	0	0	0	s2	s1	s0	1	0	0	0	BSET s
1	0	0	1	0	1	0	0	0	1	1	1	1	0	0	0	SEI (BSET 7)
1	0	0	1	0	1	0	0	0	1	1	0	1	0	0	0	SET (BSET 6)
1	0	0	1	0	1	0	0	0	1	0	1	1	0	0	0	SEH (BSET 5)
1	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	SES (BSET 4)
1	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0	SEV (BSET 3)
1	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	SEN (BSET 2)
1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	SEZ (BSET 1)
1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	SEC (BSET 0)
1	0	0	1	0	1	0	0	x	x	x	x	0	x	x	x	未定義
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	1	1	PUSH Rr
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	1	0	ST -X,Rr
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	0	1	ST X+,Rr
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	1	0	0	ST X,Rr
1	0	0	1	0	0	1	x	x	x	x	x	1	0	1	1	未定義
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	0	1	0	ST -Y,Rr
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	1	0	0	1	ST Y+,Rr
1	0	0	1	0	0	1	x	x	x	x	x	1	0	0	0	未定義
1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	1	1	LAT Z,Rd
1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	1	0	LAC Z,Rd
1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	0	1	LAS Z,Rd
1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	1	0	0	XCH Z,Rd
1	0	0	1	0	0	1	x	x	x	x	x	0	0	1	1	未定義
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	0	0	1	0	ST -Z,Rr
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	0	0	0	1	ST Z+,Rr
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	0	0	0	0	STS k,Rr
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0	

ｺｰﾄﾞ順機械語一覧 (4/5)

1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	1	1	POP Rd
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	1	0	LD Rd,−X
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	0	1	LD Rd,X+
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	1	0	0	LD Rd,X
1	0	0	1	0	0	0	x	x	x	x	x	1	0	1	1	未定義
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	0	1	0	LD Rd,−Y
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	1	0	0	1	LD Rd,Y+
1	0	0	1	0	0	0	x	x	x	x	x	1	0	0	0	未定義
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	1	1	ELPM Rd,Z+
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	1	0	ELPM Rd,Z
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	0	1	LPM Rd,Z+
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	1	0	0	LPM Rd,Z
1	0	0	1	0	0	0	x	x	x	x	x	0	0	1	1	未定義
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	0	1	0	LD Rd,−Z
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	0	0	1	LD Rd,Z+
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	0	0	0	LDS Rd,k
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0	
1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	LD(D),ST(D)で定義済み(1/5参照)
0	1	1	1	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	ANDI Rd,K
0	1	1	1	$\overline{K7}$	$\overline{K6}$	$\overline{K5}$	$\overline{K4}$	d3	d2	d1	d0	$\overline{K3}$	$\overline{K2}$	$\overline{K1}$	$\overline{K0}$	CBR Rd,K (ANDI Rd,$\overline{K}$)
0	1	1	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	ORI Rd,K
0	1	1	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	SBR Rd,K (ORI Rd,K)
0	1	0	1	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	SUBI Rd,K
0	1	0	0	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	SBCI Rd,K
0	0	1	1	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	CPI Rd,K
0	0	1	0	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	MOV Rd,Rr
0	0	1	0	1	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	OR Rd,Rr
0	0	1	0	0	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	EOR Rd,Rr
0	0	1	0	0	1	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0	CLR Rd (EOR Rd,Rd)
0	0	1	0	0	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	AND Rd,Rr
0	0	1	0	0	0	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0	TST Rd (AND Rd,Rd)
0	0	0	1	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	ADC Rd,Rr
0	0	0	1	1	1	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0	ROL Rd (ADC Rd,Rd)
0	0	0	1	1	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	SUB Rd,Rr
0	0	0	1	0	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	CP Rd,Rr
0	0	0	1	0	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	CPSE Rd,Rr
0	0	0	0	1	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	ADD Rd,Rr
0	0	0	0	1	1	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0	LSL Rd (ADD Rd,Rd)
0	0	0	0	1	0	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	SBC Rd,Rr
0	0	0	0	0	1	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	CPC Rd,Rr

ｺｰﾄﾞ順機械語一覧 (5/5)

0	0	0	0	0	0	1	1	1	d2	d1	d0	1	r2	r1	r0	FMULSU Rd,Rr (注: Rd=d2～0+16, Rr=r2～0+16)
0	0	0	0	0	0	1	1	1	d2	d1	d0	0	r2	r1	r0	FMULS Rd,Rr (注: Rd=d2～0+16, Rr=r2～0+16)
0	0	0	0	0	0	1	1	0	d2	d1	d0	1	r2	r1	r0	FMUL Rd,Rr (注: Rd=d2～0+16, Rr=r2～0+16)
0	0	0	0	0	0	1	1	0	d2	d1	d0	0	r2	r1	r0	MULSU Rd,Rr (注: Rd=d2～0+16, Rr=r2～0+16)
0	0	0	0	0	0	1	0	d3	d2	d1	d0	r3	r2	r1	r0	MULS Rd,Rr (注: Rd=d2～0+16, Rr=r2～0+16)
0	0	0	0	0	0	0	1	d4	d3	d2	d1	r4	r3	r2	r1	MOVW Rd,Rr (注: Rd=d4～0×2, Rr=r4～0×2)
0	0	0	0	0	0	0	0	1	x	x	x	x	x	x	x	未定義
0	0	0	0	0	0	0	0	x	1	x	x	x	x	x	x	未定義
0	0	0	0	0	0	0	0	x	x	1	x	x	x	x	x	未定義
0	0	0	0	0	0	0	0	x	x	x	1	x	x	x	x	未定義
0	0	0	0	0	0	0	0	x	x	x	x	1	x	x	x	未定義
0	0	0	0	0	0	0	0	x	x	x	x	x	1	x	x	未定義
0	0	0	0	0	0	0	0	x	x	x	x	x	x	1	x	未定義
0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	1	未定義
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NOP

注: 以下はAVRrc専用で、本来のLDD Rd,Y+q、LDD Rd,Z+q、STD Y+q,Rr、STD Z+q,Rr機械語符号に代わります。AVRrcのﾚｼﾞｽﾀﾌｧｲﾙはR16～R31の16個なので、d3～d0はその値+16でR16～R31を示します。k6～k0は0～127で128位置を指示しますが、これはｱﾄﾞﾚｽ$0040～$00BFの128位置です。この位置の符号化は単純な変位ではなく、以下のように符号化されます。

ADDR7～0 = ($\overline{k4}$,k4,k6,k5,k3,k2,k1,k0)

1	0	1	0	1	k6	k5	k4	r3	r2	r1	r0	k3	k2	k1	k0	STS k,Rr
1	0	1	0	0	k6	k5	k4	d3	d2	d1	d0	k3	k2	k1	k0	LDS Rd,k

A-2. ｵﾍﾟﾗﾝﾄﾞ種別一覧 (1/2)

機械語																ｵﾍﾟﾗﾝﾄﾞ表記	該当ﾆｰﾓﾆｯｸ
x	0	x	x	x	x	r4	d4	d3	d2	d1	d0	r3	r2	r1	r0	Rd,Rr	ADC, ADD, AND, CP, CPC, CPSE, EOR, MOV, MUL, OR, SBC, SUB
0	0	x	x	x	x	d4	d4	d3	d2	d1	d0	d3	d2	d1	d0	Rd	CLR (EOR Rd,Rd), LSL (ADD Rd,Rd), ROL (ADC Rd,Rd), TST (AND Rd,Rd)
x	x	x	x	K7	K6	K5	K4	d3	d2	d1	d0	K3	K2	K1	K0	Rd,K	ANDI, CPI, LDI, ORI, SBCI, SBR (ORI Rd,K), SUBI
0	1	1	1	$\overline{K7}$	$\overline{K6}$	$\overline{K5}$	$\overline{K4}$	d3	d2	d1	d0	$\overline{K3}$	$\overline{K2}$	$\overline{K1}$	$\overline{K0}$	Rd,K	CBR (ANDI Rd,$\overline{K}$)
1	1	1	0	1	1	1	1	d3	d2	d1	d0	1	1	1	1	Rd	SER (LDI Rd,$FF)
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	x	x	x	x	Rr	PUSH, ST(X/X+/−X/Y+/−Y/Z+/−Z)
1	0	0	1	0	x	x	d4	d3	d2	d1	d0	x	x	x	x	Rd	ASR, COM, DEC, ELPM(Rd), INC, LPM(Rd), LSR, NEG, POP, ROR, SWAP, LD(X/X+/−X/Y+/−Y/Z+/−Z)
1	0	q5	0	q4	q3	1	r4	r3	r2	r1	r0	1	q2	q1	q0	Y/Z+q,Rr	STD (Y/Z)
1	0	0	0	0	0	1	r4	r3	r2	r1	r0	1	0	0	0	Y/Z,Rr	ST (Y/Z) (STD Y/Z+0)
1	0	q5	0	q4	q3	0	d4	d3	d2	d1	d0	1	q2	q1	q0	Rd,Y/Z+q	LDD (Y/Z)
1	0	0	0	0	0	0	d4	d3	d2	d1	d0	1	0	0	0	Rd,Y/Z	LD (Y/Z) (LDD Y/Z+0)
1	0	0	1	0	0	1	d4	d3	d2	d1	d0	0	x	x	x	Z,Rd	XCH, LAC, LAS, LAT
1	0	1	1	1	A5	A4	r4	r3	r2	r1	r0	A3	A2	A1	A0	A,Rr	OUT
1	0	1	1	0	A5	A4	d4	d3	d2	d1	d0	A3	A2	A1	A0	Rd,A	IN
1	0	0	1	0	0	1	r4	r3	r2	r1	r0	0	0	0	0	k,Rr	STS
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0		
1	0	0	1	0	0	0	d4	d3	d2	d1	d0	0	0	0	0	Rd,k	LDS
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0		
0	0	0	0	0	0	x	x	d3	d2	d1	d0	r3	r2	r1	r0	Rd,Rr	MOVW, MULS
1	1	1	1	1	1	x	r4	r3	r2	r1	r0	0	b2	b1	b0	Rr,b	SBRC, SBRS
1	1	1	1	1	0	x	d4	d3	d2	d1	d0	0	b2	b1	b0	Rd,b	BLD, BST
1	0	0	1	0	1	0	0	x	s2	s1	s0	1	0	0	0	s	BCLR, BSET
1	0	0	1	0	1	0	0	x	1	1	1	1	0	0	0	−	CLI (BCLR 7), SEI (BSET 7)
1	0	0	1	0	1	0	0	x	1	1	0	1	0	0	0	−	CLT (BCLR 6), SET (BSET 6)
1	0	0	1	0	1	0	0	x	1	0	1	1	0	0	0	−	CLH (BCLR 5), SEH (BSET 5)
1	0	0	1	0	1	0	0	x	1	0	0	1	0	0	0	−	CLS (BCLR 4), SES (BSET 4)
1	0	0	1	0	1	0	0	x	0	1	1	1	0	0	0	−	CLV (BCLR 3), SEV (BSET 3)
1	0	0	1	0	1	0	0	x	0	1	0	1	0	0	0	−	CLN (BCLR 2), SEN (BSET 2)
1	0	0	1	0	1	0	0	x	0	0	1	1	0	0	0	−	CLZ (BCLR 1), SEZ (BSET 1)
1	0	0	1	0	1	0	0	x	0	0	0	1	0	0	0	−	CLC (BCLR 0), SEC (BSET 0)
0	0	0	0	0	0	1	1	x	d2	d1	d0	x	r2	r1	r0	Rd,Rr	FMUL, FMULS, FMULSU, MULSU
1	0	0	1	1	0	x	x	A4	A3	A2	A1	A0	b2	b1	b0	A,b	CBI, SBI, SBIC, SBIS

ｵﾍﾟﾗﾝﾄﾞ種別一覧 (2/2)

機械語																ｵﾍﾟﾗﾝﾄﾞ表記	該当ﾆｰﾓﾆｯｸ
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	S2	S1	S0	s,k	BRBC, BRBS
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	1	1	1	k	BRID (BRBC 7,k), BRIE (BRBS 7,k)
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	1	1	0	k	BRTC (BRBC 6,k), BRTS (BRBS 6,k)
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	1	0	1	k	BRHC (BRBC 5,k), BRHS (BRBS 5,k)
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	1	0	0	k	BRGE (BRBC 4,k), BRLT (BRBS 4,k)
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	0	1	1	k	BRVC (BRBC 3,k), BRVS (BRBS 3,k)
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	0	1	0	k	BRPL (BRBC 2,k), BRMI (BRBS 2,k)
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	0	0	1	k	BRNE (BRBC 1,k), BREQ (BRBS 1,k)
1	1	1	1	0	x	k6	k5	k4	k3	k2	k1	k0	0	0	0	k	BRCC, BRSH (BRBC 0,k), BRCS, BRLO (BRBS 0,k)
1	0	0	1	0	1	0	0	K3	K2	K1	K0	1	0	1	1	K	DES
1	0	0	1	0	1	1	x	K5	K4	d2	d1	K3	K2	K1	K0	Rd,K	ADIW, SBIW
1	1	0	x	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0	k	RCALL, RJMP
1	0	0	1	0	1	0	k21	k20	k19	k18	k17	1	1	x	k16	k	CALL, JMP
k15	k14	k13	k12	k11	k10	k9	k8	k7	k6	k5	k4	k3	k2	k1	k0		
1	0	1	0	1	k6	k5	k4	r3	r2	r1	r0	k3	k2	k1	k0	k,Rr	STS (注: AVRrc専用)
1	0	1	0	0	k6	k5	k4	d3	d2	d1	d0	k3	k2	k1	k0	Rd,k	LDS (注: AVRrc専用)

A-3. ｵﾍﾟﾗﾝﾄﾞなし機械語命令一覧

1	0	0	1	0	1	0	1	1	1	1	1	1	0	0	0	SPM Z+ (注: 機械語的にｵﾍﾟﾗﾝﾄﾞ ﾋﾞｯﾄなし)
1	0	0	1	0	1	0	1	1	1	1	0	1	0	0	0	SPM
1	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	ELPM
1	0	0	1	0	1	0	1	1	1	0	0	1	0	0	0	LPM
1	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	WDR
1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0	BREAK
1	0	0	1	0	1	0	1	1	0	0	0	1	0	0	0	SLEEP
1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1	EICALL
1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	RETI
1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	EIJMP
1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	ICALL
1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0	RET
1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	IJMP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NOP

注: 派生ﾆｰﾓﾆｯｸ、CLxとSExを除く。

7. 改訂履歴

本項内の頁番号参照が本資料への参照であることに注意してください。本項内の改訂参照は資料改訂への参照です。

7.1. 改訂0856F – 2008年5月

1. この改訂はAVR命令一式0856E-AVR-11/05に基きます。
   AVR命令一式0856E-AVR-11/05に対比して行われた変更:
   - DESとSPM #2とで"**4. 全命令一式要約**"を更新
   - XMEGAｸﾛｯｸ周期と命令説明とでAVR命令一式を更新

7.2. 改訂0856G – 2008年7月

1. 「**改訂履歴**」挿入。
2. ④削除によってSTに関する"XMEGA周期数"を更新
3. "SPM #2"ｵﾍﾟｺｰﾄﾞ更新

7.3. 改訂0856H – 2009年4月

1. 「**命令一式要約**」を更新
   tinyAVRrc(ATtiny10等)を含めるようにｸﾛｯｸ周期数を更新
2. tinyAVRrc(ATtiny10等)適合用に項を更新
   「**CBI – I/Oﾚｼﾞｽﾀのﾋﾞｯﾄ解除(0)**」項
   「**LD – Xﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ取得**」項
   「**LD (LDD) – Yﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ取得**」項
   「**LD (LDD) – Zﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ取得**」項
   「**RCALL – PC相対ｻﾌﾞﾙｰﾁﾝ呼び出し**」項
   「**SBI – I/Oﾚｼﾞｽﾀのﾋﾞｯﾄ設定(1)**」項
   「**ST – Xﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ設定**」項
   「**ST (STD) – Yﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ設定**」項
   「**ST (STD) – Zﾚｼﾞｽﾀ間接でのﾃﾞｰﾀ設定**」項
3. tinyAVRrc(ATtiny10等)適合用に項を追加
   「**LDS(1語) – SRAMから直接指定で汎用ﾚｼﾞｽﾀに取得**」項
   「**STS(1語) – SRAMへ直接指定で汎用ﾚｼﾞｽﾀを設定**」項

7.4. 改訂0856I – 2010年7月

1. 「**命令一式要約**」をLAC,LAS,LATとXCHの新命令で更新
   「**LAC – 取得と解除**」項
   「**LAS – 取得と設定**」項
   「**LAT – 取得と反転**」項
   「**XCH – 交換**」項
2. 縮小されたtinyAVRｺｱを含めるようにｸﾛｯｸ周期数欄を更新
   (縮小tinyAVRｺｱによってATtiny置換(**訳注**:本書はAVRrcとして表記))

7.5. 改訂0856J – 2014年2月

1. 「**条件分岐要約**」項を修正
2. 「**5.55.1. 説明**」で最初の図を修正
3. 「**5.113.1. 説明**」で「**例**」内の"TBD"を削除
4. 「**LAC – 取得と解除**」項でLAC操作を修正
5. 新雛形適用

7.6. 改訂0856K – 2016年5月

1. 「**RETI – 割り込みからの復帰**」に注を追加

7.7. 改訂0856L – 2016年11月

文書の完全な再吟味。新文書雛形。

7.8. DS40002198A – 2020年5月

1. Microchip形式に変換してAtmel文書番号0856を置換

2. 文書全体の完全な再吟味
3. 第2.章の全ての図を更新
4. 「**条件分岐要約**」項を削除 (**訳注**:本書では命令一式概要内に移動)
5. 全命令に対して周期数と動作を更新
6. 全ての命令は今やAVRｺｱの全ての変種に対して全ての周期数を一覧にする表を持ちます。
7. ﾃﾞﾊﾞｲｽに対してどの命令が有効かを一覧にする追補Aを追加

7.9. DS40002198B - 2021年2月

1. 編集上の変更、ｺｰﾄﾞ例は美化が必要でした。
2. avr-gccとxc8でのｲﾝﾗｲﾝでｺﾝﾊﾟｲﾙできない構文を持つ2つのﾚｼﾞｽﾀ(ﾚｼﾞｽﾀ対)に影響を及ぼす命令を修正
3. 上付き線を否定として定義
4. いくつかの簡素化と共に条件分要約表を再導入(**訳注**:本書では**4.**章内に内包)
5. AVR® DA系ｺｱ概要をAVRxmからAVRxtに修正
6. AVR® DB系とAVR® DD系ｺｱ概要を追加

7.10. DS40002198C - 2024年11月

1. 命令一式要約でAVRxmに対して読み/変更/書きを明確化
2. AVRxmとAVRxrのｺｱに対するRETI命令を明確化
3. 128Kﾊﾞｲﾄよりも大きなﾒﾓﾘを持つAVRxtﾃﾞﾊﾞｲｽに対してEIJMPとEICALLの命令が有効との注をｺｱ説明に追加
4. ATtiny11、12、15の欠落命令としてLPMを削除することで誤植を修正
5. ｺｱ説明項にAVR® DU、AVR® EA、AVR® EB、AVR® SD系を追加

Microchip情報

商標

“Microchip”の名称とﾛｺﾞ、“M”のﾛｺﾞ、それと他の名称、ﾛｺﾞ、商標は米国や他の国に於けるMicrochip Technology Incorporatedまたはその系列会社や子会社の登録または未登録の商標です(“Microchip商標”)。Microchip商標に関する情報はhttps://www.microchip.com/en-us/about/legalinformation/microchip-trademarksで見つけることができます。

法的通知

この刊行物と契約での情報は設計、試験、応用とのMicrochip製品の統合を含め、Microchip製品でだけ使えます。他の何れの方法でのこの情報の使用はこれらの条件に違反します。ﾃﾞﾊﾞｲｽ応用などに関する情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。追加支援については最寄りのMicrochip営業所にお問い合わせ頂くか、www.microchip.com/en-us/support/design-help/client-support-servicesで追加支援を得てください。

この情報はMicrochipによって「現状そのまま」で提供されます。Microchipは非侵害、商品性、特定目的に対する適合性の何れの黙示的保証やその条件、品質、性能に関する保証を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。

如何なる場合においても、Microchipは情報またはその使用に関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、情報またはその使用に関連する全ての請求に対するMicrochipの全責任は、もしあれば、情報のためにMicrochipへ直接支払った料金を超えないものとします。

生命維持や安全応用でのMicrochipﾃﾞﾊﾞｲｽの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

Microchipﾃﾞﾊﾞｲｽ ｺｰﾄﾞ保護機能

Microchip製品での以下のｺｰﾄﾞ保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipﾃﾞｰﾀｼｰﾄに含まれる仕様に合致します。
- Microchipは動作仕様内で意図した方法と通常条件下で使われる時に、その製品系統が安全であると考えます。
- Microchipはその知的所有権を尊重し、積極的に保護します。Microchip製品のｺｰﾄﾞ保護機能を侵害する試みは固く禁じられ、ﾃﾞｼﾞﾀﾙ ﾐﾚﾆｱﾑ著作権法に違反するかもしれません。
- Microchipや他のどの半導体製造業者もそれのｺｰﾄﾞの安全を保証することはできません。ｺｰﾄﾞ保護は製品が”破ることができない”ことを当社が保証すると言うことを意味しません。ｺｰﾄﾞ保護は常に進化しています。Microchipは当社製品のｺｰﾄﾞ保護機能を継続的に改善することを約束します。

本手引書はMicrochipのAVR®命令一式手引書(DS40002198C-2024年11月)の翻訳日本語版です。日本語での表現が省略、意訳されている部分もあります。必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。「付録」部分は独自に追加したものです。原書表記で4章の内容を5章に含めたため、本書の4章以降の章番号が原書に対して-1されています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。