

nX-U16/100 コア
インストラクションマニュアル
CMOS 16 ビットマイクロコントローラ

正式 2 版 発行日 2012 年 7 月 31 日

ご注意

本資料の一部または全部をラピスセミコンダクタの許可なく、転載・複写することを堅くお断りします。本資料の記載内容は改良などのため予告なく変更することがあります。本資料に記載されている内容は製品のご紹介資料です。ご使用にあたりましては、別途仕様書を必ずご請求のうえ、ご確認ください。本資料に記載されております応用回路例やその定数などの情報につきましては、本製品の標準的な動作や使い方を説明するものです。したがいまして、量産設計をされる場合には、外部諸条件を考慮していただきまますようお願いいたします。本資料に記載されております情報は、正確を期すため慎重に作成したものですが、万が一、当該情報の誤り・誤植に起因する損害がお客様に生じた場合においても、ラピスセミコンダクタはその責任を負うものではありません。本資料に記載されております技術情報は、製品の代表的動作および応用回路例などを示したものであり、ラピスセミコンダクタまたは他社の知的財産権その他のあらゆる権利について明示的にも黙示的にも、その実施または利用を許諾するものではありません。上記技術情報の使用に起因して紛争が発生した場合、ラピスセミコンダクタはその責任を負うものではありません。本資料に掲載されております製品は、一般的な電子機器(AV機器、OA機器、通信機器、家電製品、アミューズメント機器など)への使用を意図しています。本資料に掲載されております製品は、「耐放射線設計」はなされておりません。ラピスセミコンダクタは常に品質・信頼性の向上に取り組んでおりますが、種々の要因で故障することもあります。ラピスセミコンダクタ製品が故障した際、その影響により人身事故、火災損害等が起こらないようご使用機器でのディレーティング、冗長設計、延焼防止、フェイルセーフ等の安全確保をお願いします。定格を超えたご使用や使用上の注意書が守られていない場合、いかなる責任もラピスセミコンダクタは負うものではありません。極めて高度な信頼性が要求され、その製品の故障や誤動作が直接人命を脅かしかかるいは人体に危害を及ぼすおそれのある機器・装置・システム(医療機器、輸送機器、航空宇宙機、原子力制御、燃料制御、各種安全装置など)へのご使用を意図して設計・製造されたものではありません。上記特定用途に使用された場合、いかなる責任もラピスセミコンダクタは負うものではありません。上記特定用途への使用を検討される際は、事前にローム営業窓口までご相談願います。本資料に記載されております製品および技術のうち「外国為替及び外国貿易法」に該当する製品または技術を輸出する場合、または国外に提供する場合には、同法に基づく許可が必要です。

Copyright 2011 - 2012 LAPI Semiconductor Co., Ltd.

ラピスセミコンダクタ株式会社

〒193-8550 東京都八王子市東浅川町 550-1

<http://www.lapis-semi.com/jp/>

はじめに

このマニュアルは、ラピスセミコンダクタオリジナル 16 ビット1チップマイクロコントローラの CPU コアとして用いられる nX-U16/100 コアの命令セットについて述べています。本文中は nX-U16/100 コアを略して U16 コア、または U16 と表現する場合もございますのでご了承ください。

このマニュアルでは、nX-U16/100 コアの基本アーキテクチャを前提として解説を行っています。お使いになる機種によっては実際に使用できるメモリ容量に制限がある場合があります。制限事項につきましては、各機種のユーザーズマニュアルをご参照ください。

nX-U16/100 をコアとする製品群に関連するマニュアルとして、本書のほかに以下のマニュアルがあります。あわせてお読みください。

■MACU8 アセンブラー・パッケージ ユーザーズマニュアル

リロケータブルアセンブラー、リンカ、ライブラリアン、オブジェクトコンバータ
の操作方法の説明およびアセンブリ言語仕様の説明

■CCU8 ユーザーズマニュアル

コンパイラの操作方法の説明

■Dr.610xxx ユーザーズマニュアル

エミュレータ Dr.610xxx の操作方法説明

■DTU8 デバッガユーザーズマニュアル

デバッガ DTU8 の操作方法説明

このマニュアルは3つの章と付録より構成されます。

各章の概要は次のとおりです。

第1章 アーキテクチャ

nX-U16/100 の基本アーキテクチャについて説明します。レジスタやプログラムで扱う CPU 資源を解説した後、特徴的な機能、制限およびプログラミングに関する留意点について述べます。第2章および第3章を理解するために必要とされる基本事項について記述しています。

第2章 アドレッシングモード

レジスタやメモリに対するアクセスを指定する記述方法およびその詳細な動作について説明します。

第3章 命令の詳細

命令の機能とその詳細な動作および命令コードについて解説します。命令の解説はアルファベット順に並べてあります。

付録 命令一覧

nX-U16/100 コア命令の、オペランド表記と命令コードを、機能別に列挙しています。

また、本マニュアルでは説明をわかりやすくするために、次のような表現方法を用いています。

■ 値の表現

数値の終わりに“H”が付加されている場合、その値は 16 進数であることを示します。たとえば 1000H と記述した場合は 16 進数の 1000(10 進では 4096)を表します。

■ 単位の表現

数値の単位が“バイト”で表現されている場合、8 ビットデータであることを意味します。

数値の単位が“ワード”と表現されている場合、16 ビットデータであることを意味します。

数値の単位が“ダブルワード”と表現されている場合、32 ビットデータであることを意味します。

数値の単位が“クワッドワード”と表現されている場合、64 ビットデータであることを意味します。

■ 範囲の表現

A-B という表現は、A および B を含む値範囲を表します。同様の目的で A-B を減算と混同しないと思われる個所で用いることがあります。

目次

1 アーキテクチャ	1-1
1.1 概要	1-1
1.1.1 特長	1-1
1.2 CPU資源とプログラミングモデル	1-2
1.2.1 レジスタ	1-3
1.2.1.1 汎用レジスタ	1-4
1.2.1.2 ベースポインタおよびフレームポインタ	1-4
1.2.2 コントロールレジスタ	1-5
1.2.2.1 プログラム・ステータスワード(PSW)	1-5
1.2.2.2 プログラムカウンタ(PC)	1-7
1.2.2.3 コードセグメントレジスタ(CSR)	1-7
1.2.2.4 リンクレジスタ(LR, ELR1, ELR2, ELR3)	1-8
1.2.2.5 CSR退避レジスタ(LCSR, ECSR1, ECSR2,ECSR3)	1-9
1.2.2.6 PSW退避レジスタ(EPSW1、EPSW2、EPSW3)	1-9
1.2.2.7 スタックポインタ(SP)	1-10
1.2.2.8 EAレジスタ(EA)	1-10
1.2.2.9 ARレジスタ(AR)	1-10
1.2.2.10 データセグメントレジスタ(DSR)	1-11
1.3 メモリ空間	1-12
1.3.1 プログラム・メモリ空間	1-12
1.3.2 ベクタテーブル領域	1-13
1.3.2.1 リセットベクタ領域	1-13
1.3.2.2 割込みベクタ領域	1-14
1.3.2.3 ベクタテーブルの記述方法	1-15
1.3.3 プログラムメモリ領域	1-16
1.3.4 DSRプリフィックスコードについて	1-16
1.3.5 データメモリ空間	1-17
1.3.5.1 データ型	1-18
1.3.5.2 アドレス割り付け	1-19
1.3.5.3 ワードバウンダリ	1-19
1.3.5.4 ROMウインドウ機能	1-20
1.3.6 メモリモデル	1-20
1.3.7 割込み動作について	1-22
1.3.7.1 割込み成立の条件	1-22
1.3.7.2 ノンマスカブル割込み(NMI)	1-23
1.3.7.3 マスカブル割込み(MI)	1-24
1.3.7.4 ソフトウェア割込み(SWI)	1-25
1.4 例外レベルと退避レジスタの取り扱いについて	1-26
1.5 ノンマスカブル割込みに関する注意	1-32

目次

1.6	割込み禁止状態について.....	1-33
1.7	スタックの変化について.....	1-34
2	アドレッシングモード	2-1
2.1	アドレッシングモード	2-1
2.2	レジスタアドレッシング	2-1
2.3	メモリアドレッシング	2-2
2.3.1	レジスタ間接アドレッシング	2-3
2.3.2	ダイレクトアドレッシング	2-6
2.4	即値アドレッシング	2-7
2.5	プログラムメモリアドレッシング	2-8
3	命令の詳細	3-1
3.1	概要	3-1
3.2	nX-U16/100コアの命令セット機能別分類表.....	3-2
3.3	命令実行時間について	3-11
3.4	各命令の説明	3-23
ADD	ER _n , ER _m	3-24
ADD	ER _n , #imm7.....	3-25
ADD	R _n , obj.....	3-26
ADD	SP , #signed8.....	3-27
ADDC	R _n , obj.....	3-28
AND	R _n , obj.....	3-29
B	Cadr.....	3-30
B	ER _n	3-31
Bcond	Radr.....	3-32
BL	Cadr.....	3-34
BL	ER _n	3-35
BRK	3-36
CMP	ER _n , ER _m	3-37
CMP	R _n , obj.....	3-38
CMPC	R _n , obj.....	3-39
CPLC	3-40
DAA	R _n	3-41
DAS	R _n	3-42
DEC	[EA].....	3-43
DI	3-44
DIV	ER _n , R _m	3-45
EI	3-46

EXTBW ER n	3-47
INC [EA].....	3-48
L ER n , <i>obj</i>	3-49
L QR n , <i>obj</i>	3-51
L R n , <i>obj</i>	3-52
L XR n , <i>obj</i>	3-54
LEA <i>obj</i>	3-55
MOV CER n , <i>obj</i>	3-56
MOV CQR n , <i>obj</i>	3-57
MOV CR n , <i>obj</i>	3-58
MOV CR n , R m	3-59
MOV CXR n , <i>obj</i>	3-60
MOV ECSR , R m	3-61
MOV ELR , ER m	3-62
MOV EPSW , R m	3-63
MOV ER n , ELR	3-64
MOV ER n , ER m	3-65
MOV ER n , #imm7.....	3-66
MOV ER n , SP	3-67
MOV <i>obj</i> , CER m	3-68
MOV <i>obj</i> , CQR m	3-69
MOV <i>obj</i> , CR m	3-70
MOV <i>obj</i> , CXR m	3-71
MOV PSW , <i>obj</i>	3-72
MOV R n , CR m	3-73
MOV R n , ECSR.....	3-74
MOV R n , EPSW	3-75
MOV R n , PSW	3-76
MOV R n , <i>obj</i>	3-77
MOV SP , ER m	3-78
MUL ER n ,R m	3-79
NEG R n	3-80
NOP	3-81
OR R n , <i>obj</i>	3-82
POP レジスタリスト.....	3-83
POP <i>obj</i>	3-85
PUSH レジスタリスト	3-86
PUSH <i>obj</i>	3-88
RB Dbitadr.....	3-89
RB R n . bit_offset.....	3-90
RC	3-91
RT	3-92

RTI.....	3-93
SB <i>Dbitadr</i>	3-94
SB <i>Rn</i> . <i>bit_offset</i>	3-95
SC.....	3-96
SLL <i>Rn</i> , <i>obj</i>	3-97
SLLC <i>Rn</i> , <i>obj</i>	3-98
SRA <i>Rn</i> , <i>obj</i>	3-99
SRL <i>Rn</i> , <i>obj</i>	3-100
SRLC <i>Rn</i> , <i>obj</i>	3-101
ST <i>ERn</i> , <i>obj</i>	3-102
ST <i>QRn</i> , <i>obj</i>	3-104
ST <i>Rn</i> , <i>obj</i>	3-105
ST <i>XRn</i> , <i>obj</i>	3-107
SUB <i>Rn</i> , <i>Rm</i>	3-108
SUBC <i>Rn</i> , <i>Rm</i>	3-109
SWI # <i>snum</i>	3-110
TB <i>Dbitadr</i>	3-111
TB <i>Rn</i> . <i>bit_offset</i>	3-112
XOR <i>Rn</i> , <i>obj</i>	3-113

4 付録

4-1

4.1 インストラクション表	4-1
----------------------	-----

1アーキテクチャ

1.1概要

1.1.1特長

nX-U16/100コアは以下のような特長を持っています。

- 豊富な命令セット
 - 転送, 算術演算, 比較, 論理演算, ビット操作, ビット論理演算, ジャンプ, 条件ジャンプ, コール・リターンスタック操作, 算術シフト
- 豊富なアドレッシングモード
 - レジスタアドレッシング
 - レジスタ間接アドレッシング
 - スタックポインタアドレッシング
 - コントロールレジスタアドレッシング
 - EA レジスタ間接アドレッシング
 - 汎用レジスタ間接アドレッシング
 - ダイレクトアドレッシング
 - レジスタ間接ビットアドレッシング
 - ダイレクトビットアドレッシング
- メモリ空間
 - プログラムメモリ空間(ROM)
 - 最大 32K ワード(0000H—FFFFH)* 16 セグメント
 - データメモリ空間(RAM)
 - 最大 64K バイト(0000H—FFFFH)* 256 セグメント
- 割込み
 - エミュレータ専用割込み
 - ノンマスカブル割込み
 - マスカブル割込み
 - ソフトウェア割込み

1.2 CPU 資源とプログラミングモデル

U16 は、最大 1M バイトのプログラムメモリ空間と、最大 16M バイトのデータメモリ空間を持っています。全メモリ空間は、64K バイト単位の物理セグメントで区切られており、物理セグメント 0 の空間(0:0000H-0:FFFFH)と、物理セグメントが 1 以上の空間(1:0000-FF:FFFF)では、メモリ構成がつぎのように異なります。

セグメント 0 の空間は、32K ワードのプログラムメモリ領域と、64K バイトのデータメモリ領域が独立して存在し、それぞれプログラムカウンタ(PC)とアドレスレジスタ(AR)を介してその内容を指定します。ただし、ARにより指定された領域が ROM ウィンドウ領域であった場合は、AR の内容でプログラムメモリ領域を指定します。

セグメント 1 以上の空間は、アドレッシング対象が 64K バイトを超える空間であり、同一空間上にプログラムメモリ領域とデータメモリ領域がアサインされています。

セグメント 1 以上の空間のプログラムメモリ領域は、コードセグメントレジスタ(CSR)を上位 4 ビット、PC を下位 16 ビットとする 20 ビット(CSR:PC)でその内容を指定します。

セグメント 1 以上の空間のデータメモリ領域は、データセグメントレジスタ(DSR)を上位 8 ビット、AR の内容を下位 16 ビットとする 24 ビットアドレス(DSR:AR)で指定します。

U16 のメモリ空間の概念図を以下に示します。

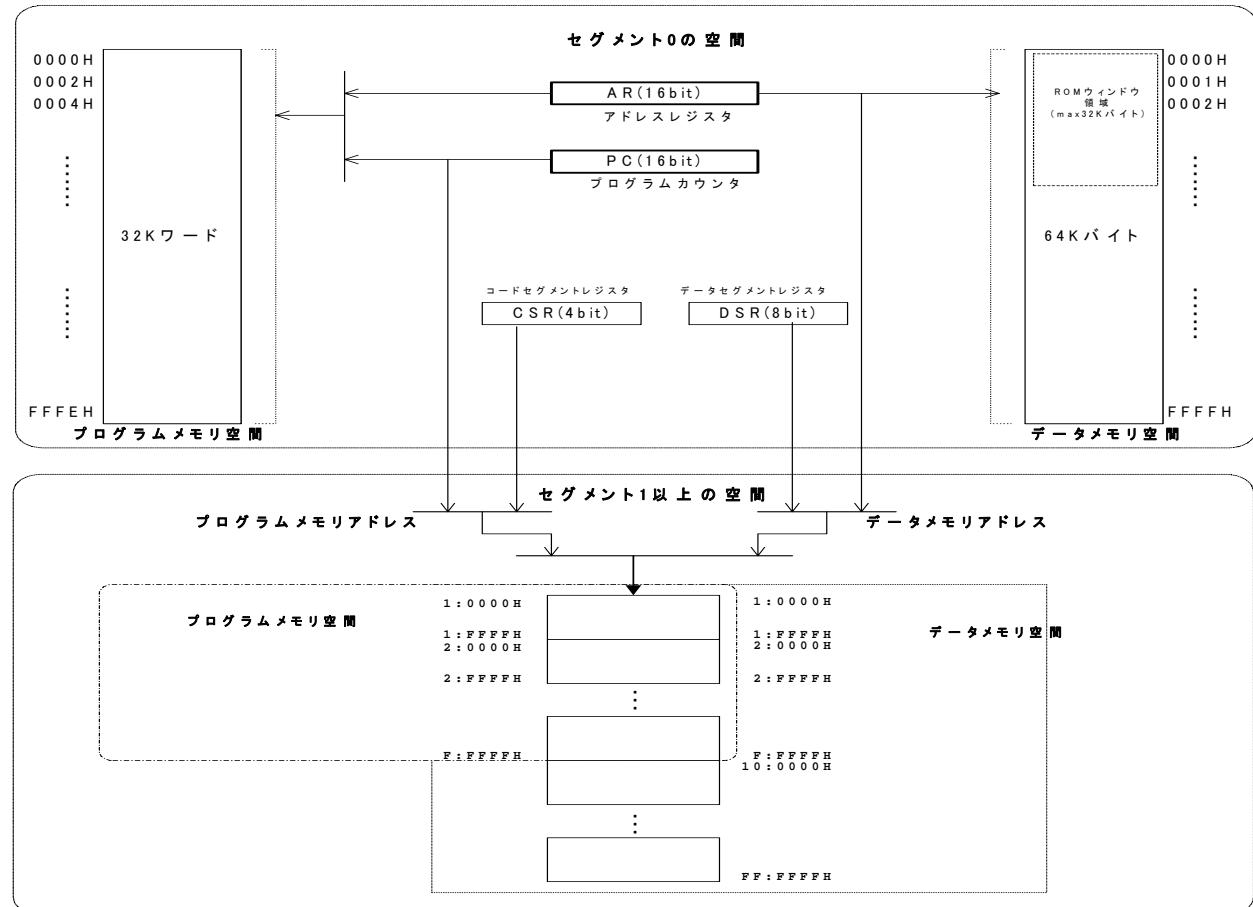


図 1-1:U16 のメモリ空間

1.2.1 レジスタ

U16 では汎用レジスタを中心とした処理方式を採用します。レジスタ構成は、その特徴によって、図 1-2 のように汎用レジスタ、コントロールレジスタに分けることができます。

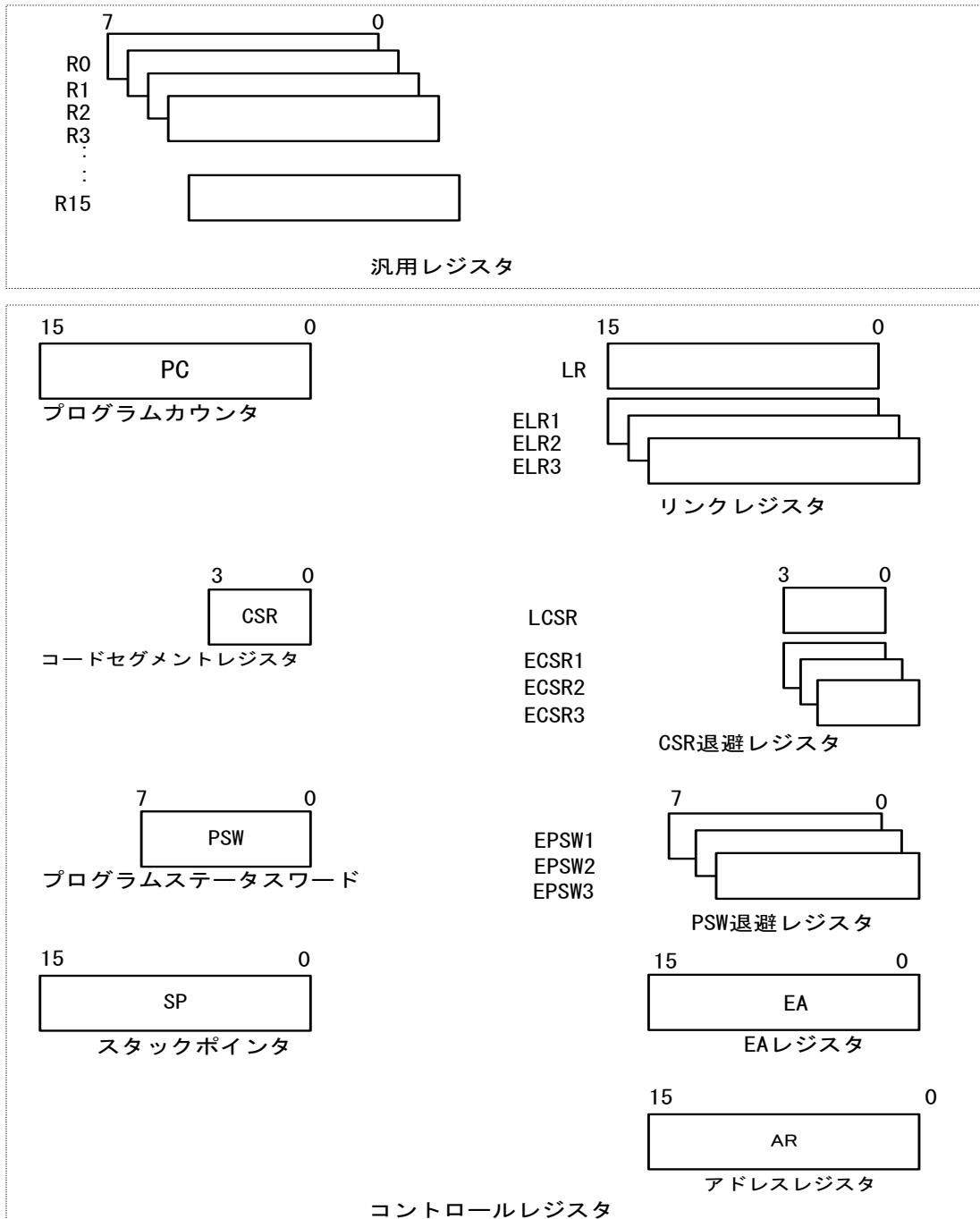


図 1-2: レジスタ構成

1.2.1.1 汎用レジスタ

演算の中心となる 16 本のバイト型レジスタです。

レジスタはバイト長ですが、アドレッシングモードの使い分けにより、連続するレジスタを結合して、8 本のワード型レジスタ(ER_n)、4 本のダブルワード型レジスタ(XR_n)、または 2 本のクワッドワード型レジスタ(QR_n)としてアクセスすることができます。割込み発生時のデータの退避はソフトウェアでおこないます。具体的には、割込みサブルーチンの先頭で PUSH 命令を使用して任意のレジスタを退避します。また POP 命令を使用して退避したデータを復帰します。



図 1-3: 汎用レジスタ

■例 汎用レジスタの使用

```

MOV  R0 , #7      ; バイト型
L    ER0 ,[EA+]   ; ワード型
L    XR0 ,[EA]    ; ダブルワード型
ST   QR0 ,[EA]    ; クワッドワード型
SB   R3.2        ; ビット型

```

1.2.1.2 ベースポインタおよびフレームポインタ

C コンパイラ使用時に、グローバルポインタとして、ER12 がベースポインタ(BP)、ER14 がフレームポインタ(FP)として使用されます。BP および FP は、汎用レジスタが使用できるアドレッシング以外に、専用のアドレッシングでアクセスすることができます。詳細は第 2 章:アドレッシングモードの項を参照して下さい。

1.2.2コントロールレジスタ

プログラムの流れを制御し、現在の状態を保持するレジスタ群です。コントロールレジスタは 18 個あり、各々専用の機能を有しています。これらのレジスタが保持するのは、一般にプログラムコンテキストと呼ばれる情報群の核をなすものです。

1.2.2.1プログラム・ステータスワード(PSW)

PSW	7	6	5	4	3	2	1	0
	C	Z	S	OV	MIE	HC	ELEVEL	

命令の実行結果の状態が格納される 8 ビットのレジスタです。プログラムの状態を保持または指定するフラグおよびフィールドから構成されます。PSW の内容は、割込み時に PSW 退避レジスタ(EPSW)に自動退避されます。EPSW に退避された PSW の値は RTI 命令の実行により PSW に復帰します。

PSW は、CPU の演算状態を保持する 5 つのフラグと、割込み制御に関する 1 つのフラグ、および現在の割込みのレベルを示す 2 ビットのフィールド(ELEVEL)から構成され、プログラムにより値を任意に設定することができます。リセット時には 0 になります。各フラグおよびフィールドの動作を以下に述べます。

- ビット 7: キャリフラグ(C)
算術演算命令、シフト命令、および比較命令の実行結果、ビット 7 あるいはビット 0 からのキャリの発生、およびビット 7 へのボローの発生があると 1 になり、なければ 0 になります。
また、SC/RC/CPLC 命令により、直接セット・リセット・反転、および条件分岐時命令によってテストすることができます。
- ビット 6: ゼロフラグ(Z)
演算の結果が 0 であることを示します。
演算命令および転送命令の実行結果が 0 の場合 1 になり、それ以外の時に 0 になります。
また条件分岐命令によってテストすることができます。
- ビット 5: サインフラグ(S)
演算結果が負の数であることを示します。算術、比較演算及び論理演算命令実行の結果、実行結果の符号ビットが 1 の場合に 1 になり、そうでない場合 0 になります。
- ビット 4: オーバーフロー・フラグ(OV)
符号付き演算におけるキャリまたはボローを保持します。算術演算および比較演算などにおいて、2 の補数で表現される範囲を越えた場合に 1 になり、そうでない場合 0 になります。

- ビット 3:マスティンタラプトイネーブルフラグ(MIE)

マスカブル割込み全体の許可／禁止を制御します。1 の場合に割込み許可になり、0 の場合割込み禁止になります。マスカブル割込みが成立すると、本フラグは割込み移行サイクル中に 0 になります。なお、EI/DI 命令により MIE をセット／リセットすることができます。

- ビット 2:ハーフキャリフラグ(HC)

10 進演算を実現するために用います。算術演算命令、比較命令実行の結果、ビット 3 またはビット 11 からのキャリまたはボローがあると 1 になり、そうでない場合 0 になります。

- ビット 1-0:割込みレベル(ELEVEL)

現在実行中の割込みレベルを保持するフィールドです。

割込みレベルは、割込みの優先度を示す 0-3 の整数で、割込みの種類別にその値が定められています。割込みの種類と割込みレベルについての関係は、“1.3.7 割込み動作について”を参照して下さい。

割込みレベルは大きいほど優先順位が高くなります。

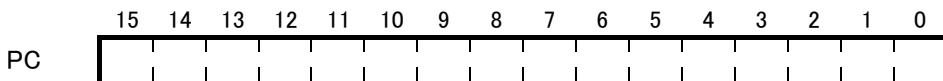
実行中に何らかの割込み要求があると、U16 は要求があった割込みのレベルと、現在の ELEVEL の値を比較し、要求された割込みのレベルが ELEVEL よりも等しいか大きい場合に割込みが発生します。

割込みの種類と優先順位については、“1.3.7 割込み動作について”を参照して下さい。

1.2.2.1.1 命令と PSW のフラグの変化

PSW のフラグ変化を伴う命令と、その実行により変化するフラグについては、第 3 章 “3.2 nX-U16/100 コアの命令セット機能別分類表”、または第 4 章のインストラクション表を参照してください。

1.2.2.2 プログラムカウンタ(PC)



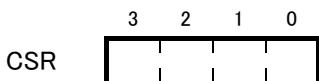
PC は、次に実行するプログラムコードのアドレスを保持する 16 ビットのカウンタです。PC はプログラムメモリからのプログラムコードのフェッチ直後にカウントアップされ、この繰り返しがプログラム実行の流れを作ります。分岐命令では新しいプログラムコードのアドレスが設定されます。

プログラムコードはワード境界に配置されるため、PC の更新は +2 づつ行われ、LSB は常に 0 が入ります。

リセット直後の PC は、リセット要因に対応するベクタ値となります。割込み時は、実行再開アドレスが割込みレベルに対応する 1 つのリンクレジスタに自動退避されます。

退避されたこの値は、RTI 命令実行により PC に復帰します。

1.2.2.3 コードセグメントレジスタ(CSR)



CSR は現在実行しているプログラムコードが属するセグメントを保持するための 4 ビットのレジスタです。セグメントは 0-15 まで指定することができます。

ひとつのセグメントには 0-FFFFH のセグメント内オフセットアドレスが割り振られており、PC でオフセットアドレスを指定します。

すなわち全プログラムメモリ空間は、CSR を上位 4 ビット、PC を下位 16 ビットとする 20 ビット(CSR:PC)で指定されます。

アドレッシング対象を決定するためのアドレス計算は 16 ビットのオフセットアドレスで行われ、この際生じるオーバフローやアンダフローは無視されます。したがってこれにより CSR が変化することはありません。同様に PC のオーバフローにより CSR は更新されません。よって CSR 書き換え手段無しにプログラム実行が、セグメント境界をまたいで進行することはありません。

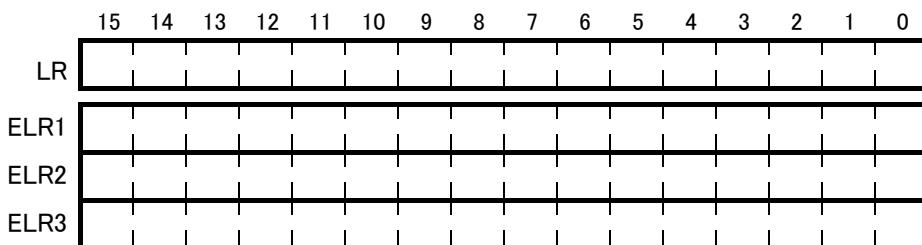
CSR は、次の場合にのみ書き換わります。

- 割込み発生時 0 になります。
- リセット時 0 になります。
- B Cadr 命令 命令中に指定された CSR が書き込まれます。
- BL Cadr 命令 命令中に指定された CSR が書き込まれます。
- RTI 命令 PSW 中の ELEVEL で指示する ECSR の、いずれかの内容が復帰します。
- RT 命令 LCSR の内容が復帰します。
- POP PC 命令 スタックメモリの内容が復帰します。

割込みが許可されると、現在の CSR の値は、PSW の ELEVEL フィールドが指示する 1 つの CSR 退避レジスタ(ECSR1,ECSR2,ESCR3 のいずれか)に自動退避されます。

CSR のリセット直後の値は 0 となります。

1.2.2.4 リンクレジスタ(LR , ELR1 , ELR2 , ELR3)



リンクレジスタ(LR , ELR1-3)は PC を退避するための 16 ビット×4 本のレジスタで、サブルーチン用(LR)と、例外処理用(ELR1-3)の 2 種類があります。いずれのレジスタにも LSB には常に0が入ります。

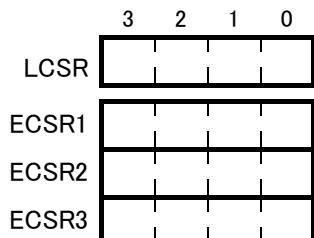
LRはBL命令によりサブルーチンコールがあった場合に選択され、BL命令実行中にサブルーチン終了後の戻り番地が自動退避されます。LR の内容は、RT 命令によって PC に復帰します。上位ルーチンへの復帰は、アプリケーションプログラムの実行状態により、RT 命令を使用する方法と、POP 命令を使用する方法する場合があります。詳細は“1.4 例外レベルと退避レジスタの取り扱いについて”を参照して下さい。

ELR1-3 には、割込みが発生した場合に、割込みから復帰する PC が退避されます。ELR1-3 のどのレジスタに退避されるかは、割込みレベルによって決定します。具体的には、割込みの種類に対応する割込みレベル値を指すとする、いずれかひとつのレジスタが選択されることになります。割込みの種類と割込みのレベルについての関係は、“1.3.7 割込み動作について”を参照して下さい。プログラム内でELEVELが変化すると、それが指示する ELR も変化します。したがって PSW を書き換える場合は、現在の ELEVEL と ELR の関係に注意を払う必要があります。

■ 注意 ■

ELR3 はエミュレータおよびデバッガが専用に取り扱うレジスタのため、アプリケーションプログラム中に操作しないで下さい。アプリケーションプログラム中に ELR3 を操作する命令を実行した場合、プログラムの動作は予想できませんのでご注意ください。

1.2.2.5CSR 退避レジスタ(LCSR, ECSR1, ECSR2,ECSR3)



CSR 退避レジスタは、CSR を退避するための 4 ビット×4 本のレジスタで、サブルーチン用(LCSR)と、例外処理用(ECSR1、ECSR2、ECSR3)の 3 種類があります。

LCSR は BL 命令によりサブルーチンコールがあった場合に選択され、BL 命令実行中に戻りセグメント値が自動退避されます。LCSR の内容は、RT 命令によって CSR に復帰します。

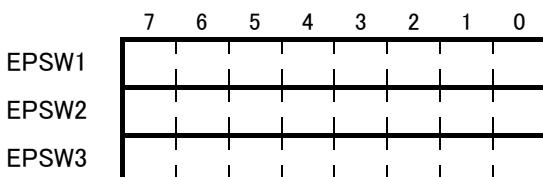
サブルーチンからメイン関数への復帰は、アプリケーションプログラムの実行状態により、RT 命令を使用する方法と、POP 命令を使用する場合があります。詳細は“1.4 例外レベルと退避レジスタの取り扱いについて”を参照して下さい。

ECSR1、ECSR2、および ECSR3 には、割込みが発生した場合にいずれか1つのレジスタが選択され、そこに割込みから復帰するセグメントが退避されます。どのレジスタが選択されるかは割込みの種類によって決定します。具体的には、割込みの種類に対応する割込みレベル値を指數とする、いずれかひとつのレジスタが選択されることになります。割込みの種類と割込みのレベルについての関係は“1.3.7 割込み動作について”を参照して下さい。プログラム内で ELEVEL が変化すると、それが指示する ECSR も変化します。したがって PSW を書き換える場合は、現在の ELEVEL と ECSR の関係に注意を払う必要があります。

■ 注意 ■

ECSR3 はエミュレータおよびデバッガが専用に取り扱うレジスタのため、アプリケーションプログラムに操作しないで下さい。です。アプリケーションプログラム中に ECSR3 を操作する命令を実行した場合、プログラムの動作は予想できませんのでご注意ください。

1.2.2.6PSW 退避レジスタ(EPSW1、EPSW2、EPSW3)

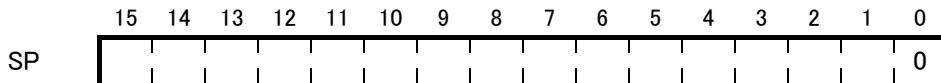


EPSW1、EPSW2、および EPSW3 は、割込み発生時に PSW を退避するための 8 ビットのレジスタです。割込みが許可されると、移行サイクル中に、PSW の ELEVEL フィールドの値を指數とするひとつのレジスタが選択され、PSW が自動退避されます。この値は RTI 命令によって PSW に復帰します。

■ 注意 ■

EPSW3 はエミュレータおよびデバッガが専用に取り扱うレジスタのため、アプリケーションプログラム中で操作しないで下さい。アプリケーションプログラム中に EPSW3 を操作する命令を実行した場合、プログラムの動作は予想できませんのでご注意ください。

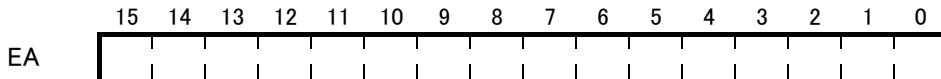
1.2.2.7 スタックポインタ(SP)



SP は、PUSH/POP 命令でレジスタの復帰/退避するための、スタックの先頭アドレスを保持する 16 ビットのレジスタです。ビット 0 は 0 固定です。

SP を介したスタックへのデータの退避・復帰は常にワード単位で行います。ワード型のデータをスタックに退避する場合、ハードウェアは SP を 2 デクリメントした後に、スタック上にデータを書き込みます。復帰する場合は、現在のスタック上のワード型データを読み出した後、SP を 2 インクリメントします。SP は独立したレジスタとして存在し、専用の命令でアクセスすることができます。リセット直後には、プログラムメモリの 0000H 番地の内容を下位バイト、0001H 番地の内容を上位バイトとする値が SP にセットされます。

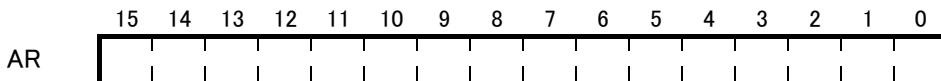
1.2.2.8 EA レジスタ(EA)



EA はメモリアドレスを保持する 16 ビットのレジスタです。メモリにアクセスする命令のいくつかは、EA を間接的に使用して、目的とするアドレスを指定することができます。

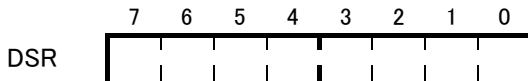
セグメント 0 の空間では、EA の内容そのものがメモリアドレスとなります。セグメント 1 以上の空間では、後述するデータセグメントレジスタ(DSR)を上位 8 ビット、EA を下位 16 ビットとする 24 ビットアドレス (DSR:EA) で、U16 のセグメント 1 以上の空間の全領域を指定することができます。EA は、専用のロード命令でプログラム内で値を任意に書き換えることができます。また、PUSH/POP 命令によって現在の値をスタック上に待避/復帰することができます。

1.2.2.9 AR レジスタ(AR)



AR は、メモリアドレスを一時的に保持する 16 ビットのレジスタで、メモリアクセス命令実行中に使用されます。なお、AR は、U16 コアのみが使用可能であり、ユーザプログラム内で取り扱えません。

1.2.2.10 データセグメントレジスタ(DSR)



DSR はデータセグメントを保持するための 8 ビットのレジスタで、セグメント 1 以上の空間のデータセグメントを指定します。セグメントは 0-255 まで指定することができます。

ひとつのデータセグメントには 0-0FFFFH のセグメント内オフセットアドレスが割り振られており、AR でオフセットアドレスを指定します。

すなわち全データメモリ空間は、DSR を上位 8 ビット、EA または AR を下位 16 ビットとする 24 ビットで指定することになります。

メモリアクセス命令中で、データセグメントの値が直接指定された場合は、命令実行中に DSR の内容は更新され、アドレッシング対象は指定されたセグメント値のデータメモリとなります。オペランドとして”DSR”が指定された場合は、アドレッシング対象は現在の DSR が指すセグメント内のデータメモリとなります。

DSR が省略された場合は、現在の DSR の値に関わらず、物理セグメント 0 空間のデータメモリがアドレッシング対象となります。

以下にメモリアクセス命令の実行例を示します。

L	R0	,5:1234H	;	DSR を 5 に更新した後、セグメント 1 以上の空間の 5:1234H の内容を、R0 にロード。
;				
LEA		55AAH		
ST	R0	,3:[EA+]	;	DSR を 3 に更新した後、セグメント 1 以上の空間の 3:55AAH に、R0 の内容をストア。 EA をインクリメント
;				
ST	R1	,3:[EA+]	;	DSR を 3 に更新した後、セグメント 1 以上の空間の 3:55ABH に、R1 の内容をストア。 EA をインクリメント
;				
ST	R2	,3:[EA+]	;	DSR を 3 に更新した後、セグメント 1 以上の空間の 3:55ACH に、R1 の内容をストア。 EA をインクリメント
;				
L	R0	,5:1234H	;	DSR を 5 に更新した後、セグメント 1 以上の空間の 5:1234H の内容を、R0 にロード。
L	R1	,1234H	;	セグメント 0 の空間のデータメモリ 1234H の内容を、R1 にロード
L	R2	,01235H	;	DSR を 0 に更新した後、セグメント 0 の空間の 1235H の内容を、R2 にロード
;				
LEA		AA55H		
L	R5,DSR: [EA+]		;	現在の DSR が指すセグメントの、AA55H の内容を、R5 にロード EA をインクリメント
L	R6,DSR:[EA+]		;	現在の DSR が指すセグメントの、AA56H の内容を、R6 にロード EA をインクリメント

DSR の内容は、リセット直後は 0 になります。

1.3メモリ空間

U16 の全メモリ空間は、64K バイトを 1 つの物理セグメントとして、最大 256 個の物理セグメントで構成されます。

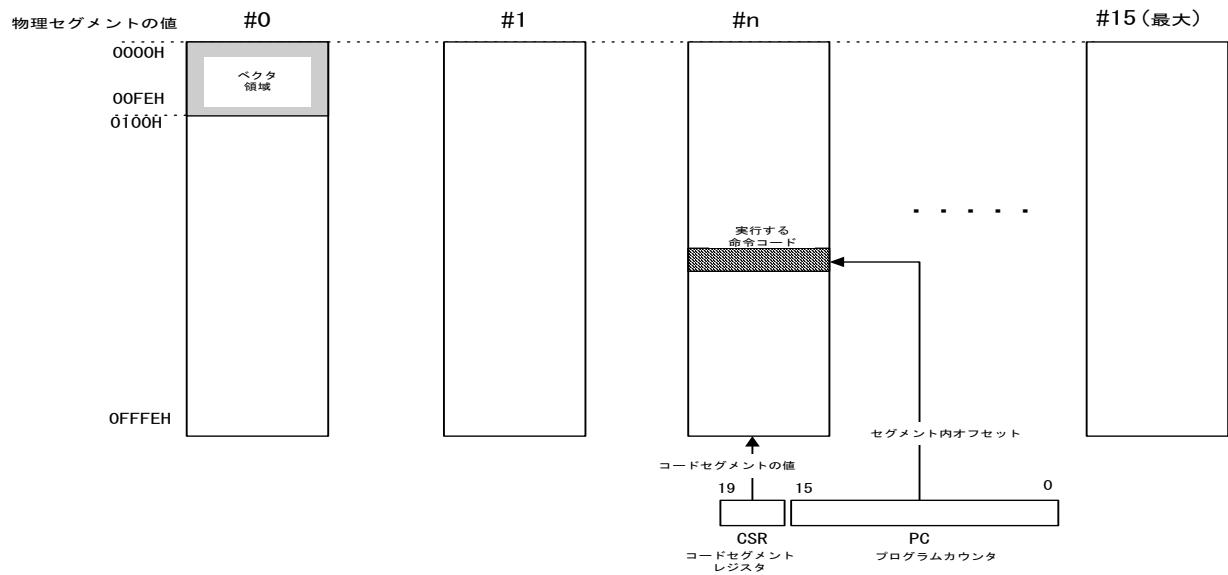
物理セグメント内にはプログラムメモリ空間と、データメモリ空間が存在し、その容量は、プログラムメモリ空間は最大 1M バイト(0:0000H-F:FFFFH)、データメモリ空間は最大 16M バイト(0:0000H-FF:FFFFH)です。それぞれの空間の構造は、物理セグメント 0(0:0000H-0:FFFFH)と 1 以上(1:0000H-FF:FFFFH)で異なっています。ここではプログラムメモリ空間とデータメモリ空間の構造について述べます。

1.3.1プログラム・メモリ空間

U16 のプログラム・メモリ空間は 32K ワードのセグメント 16 個で構成され、全体で 1M バイトの容量を持ちます。プログラム・メモリは、実行される命令コード(プログラム・コード)あるいは読み出し専用データ(テーブル・データ)を配置します。

実行中のプログラムコードは、CSR を上位 4 ビット、PC を下位 16 ビットとする 20 ビット(CSR:PC)で指定されます。CSR により選択されるセグメントをコード・セグメントと呼びます。命令実行での PC のインクリメントや、相対ジャンプによる PC に対するディスプレースメントの加減で生じるオーバフロー やアンダフローは無視され、これにより CSR が変化することはありません。また、ROM ウィンドウ機能を用いると、セグメント 0 空間の指定された領域に対して RAM アドレッシングが使用できます。

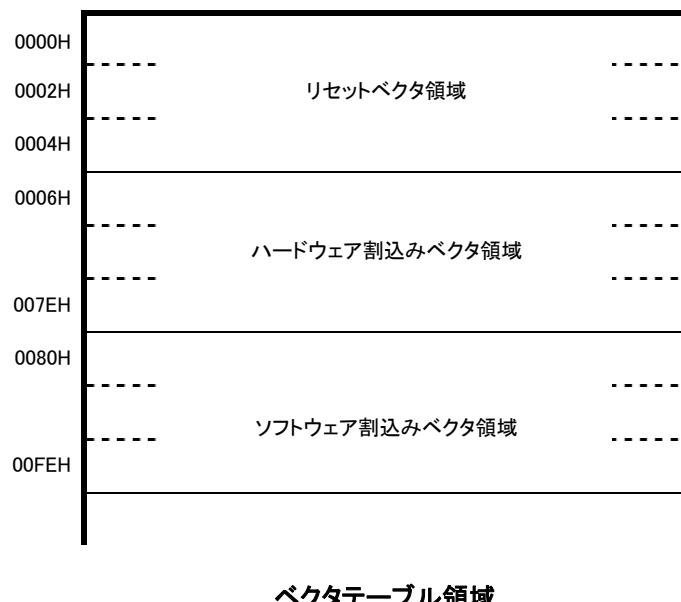
ひとつのセグメントには、0 から OFFFEH のセグメント内オフセット・アドレスが割り振られています。アドレッシング対象を決定するためのアドレス計算は、16 ビットのオフセットアドレスで行われ、この際生じるオーバフロー やアンダフローは無視されます。以下にプログラム・メモリ空間の概要を示します。



1.3.2 ベクターテーブル領域

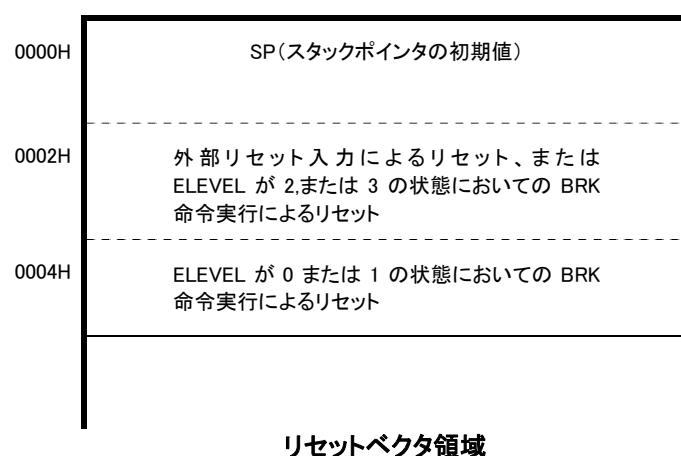
プログラムメモリ空間の 0:0H–0:0FEH は、リセットおよび割込み時に用いられる処理プログラムのエントリアドレス(ベクタ)を格納するベクターテーブル領域です。

各ベクタは、偶数番地から配置されるワード型のデータです。処理プログラムに移行する時、CSR の値はハードウェアにより 0 にリセットされます。従って処理プログラムのエントリアドレスはセグメント 0 にのみ存在します。



1.3.2.1 リセットベクタ領域

ベクターテーブルの最初の 3 本は、リセット要因に対応したリセットベクタが割り当てられています。ベクタのアドレスとリセット要因は次の通りです。

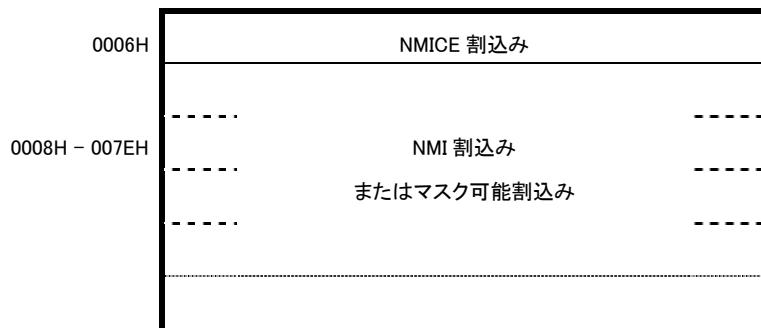


1.3.2.2割込みベクタ領域

1.3.2.2.1ハードウェア割込み領域

割込み要因は、機種の持つ周辺機能により異なります。本コアには2つのマスク不可(NMICE、NMI)割込みベクタと、マスク可能割込みベクタ(MI)が割り付けられます。

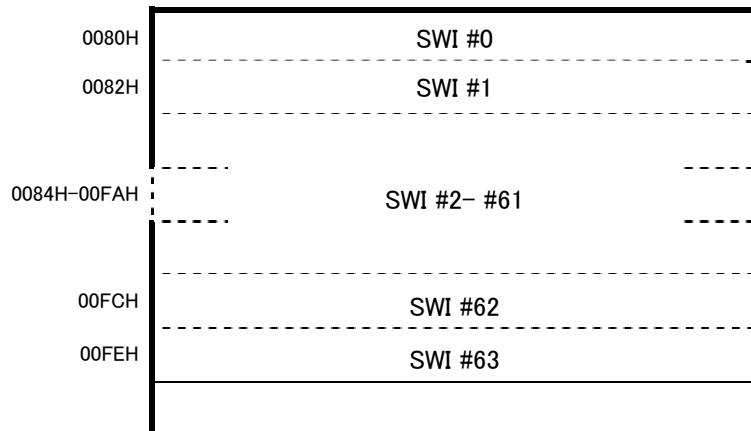
MI割込みベクタは最大59本割り付けることができます。



ハードウェア割込み領域

1.3.2.2.2ソフトウェア割込み領域

SWI命令のためのベクタ領域です。ベクタのアドレスと、それに対応するSWI命令は次の通りです。



ソフトウェア割込み領域

1.3.2.3 ベクターテーブルの記述方法

アセンブラーでは、dw 疑似命令のオペランドに処理プログラムのエントリアドレスを表すラベルを記述します。ベクターテーブル領域定義のプログラム記述例を次に示します。なお、リセット入力によるリセット以外のベクタ領域を、ベクタとして使用しない場合、通常のプログラムコードを記述して構いません。

```

;-----  

;reset vector table  

;-----  

;  

cseg at 0000h  

    dw      spinit          ;スタックポインタの初期アドレス  

    dw      start            ;pc の初期値  

    dw      brk              ;brk 命令によるベクタアドレス  

;  

org    0008h  

    dw      nmi_entry;ノンマスカブル割込み  

    dw      Int1_entry       ;マスカブル割込み#0  

    dw      Int2_entry       ;マスカブル割込み#1  

    :  

    :  

    :  

;  

;software interrupts  

;-----  

;  

cseg at 0080h  

swi_0:  

    dw      sw0_entry        ;ソフトウェア割込み#1  

swi_1:  

    dw      sw1_entry        ;ソフトウェア割込み#2  

    :  

    :  

    :  

;  

;start of main procedure  

;-----  

start:           ;プログラムの先頭  

    :  

    :  

    :

```

1.3.3 プログラムメモリ領域

セグメント0の空間のプログラム・メモリ領域と、セグメント1以上の空間のプログラム・メモリを使用する場合のプログラミング上の論理的な違いはありません。リンクなどを適切に用いて、対象の機種に実装されている内部プログラム・メモリの領域や、外部プログラムメモリ領域にメモリを実装して下さい。

1.3.4 DSR プリフィックスコードについて

物理セグメント1以上のデータメモリ空間のアクセスは、DSR プリフィックスコードを実行することで実現することができます。DSR プリフィックスコードと、それに対応する動作は以下のとおりです。

命令フォーマット	動作
1110_0011_iiii_iiii	DSR に、iiii_iiii で示される 8bit 即値をライトする。
1001_0000_dddd_1111	DSR に、dddd で指定される汎用レジスタ値をライトする。
1111_1110_1001_1111	現在の DSR の値を有効にする。

DSR プリフィックスコードは、単独命令として実行するとプログラマの意図しない動作を行う可能性があるため、アセンブラーでは常にメモリアクセス命令と組み合わせて使用します。アセンブラーでの記述方法については、第2章 2.3“メモリアドレッシング”的項を参照してください。

DSR プリフィックスコードは、その直後に配置した1命令だけに有効です。すなわち、DSR プリフィックスコードが直前にはないメモリアクセス命令は、現在の DSR の値にかかわらず物理セグメント0がアクセス対象となります。

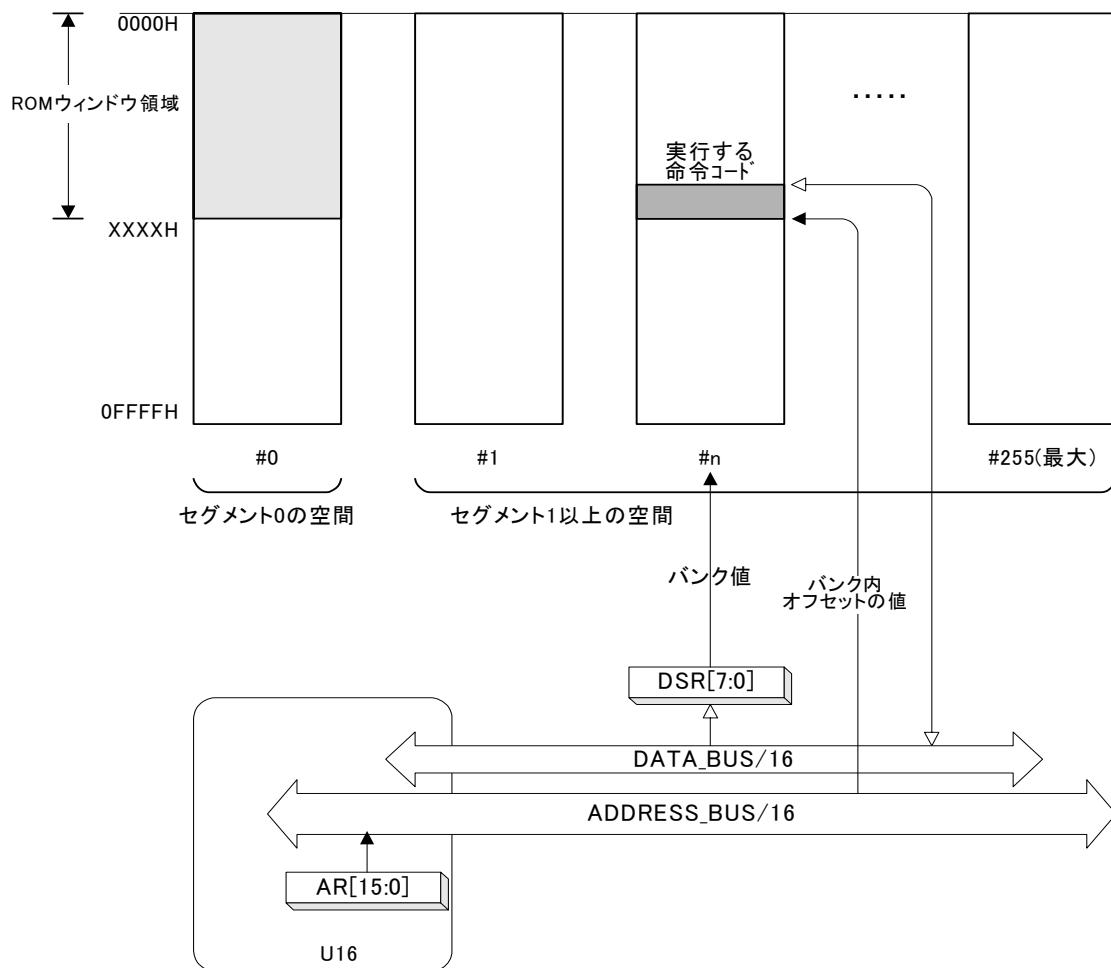
また、DSR プリフィックスコードとその直後の命令の間は、全ての割込みが受け付けられない割込み禁止状態となります。“1.5 割込み禁止状態について”を参照してください。

1.3.5 データメモリ空間

U16 のデータメモリ空間は 64K バイトのセグメント 256 個で構成され、全体で 16M バイトの容量を持ちます。データメモリは、通常読み書き可能なデータを配置します。

データメモリの内容は、DSR を上位 8 ビット、AR を下位 16 ビットとする 24 ビット(DSR:AR)アドレスで指定されます。DSR により選択されるセグメントをデータセグメントと呼びます。U16 のセグメント 0 の空間は、ROM ウィンドウ領域とデータ領域で構成されます。ROM ウィンドウ領域は、ROM 領域のデータを RAM のアドレッシングを用いてアクセスするために開いた窓の存在する領域です。内部データ・メモリのマッピングされていない領域に対し窓が開き、この窓を通して同じアドレスのテーブル・データを読み出すことができます。

データメモリ空間の概要を以下に示します。



1.3.5.1 データ型

ここでは、U16 の命令で使用可能なデータ型について述べます。

符号無しバイト型

バイト型の命令で操作できるデータ型です。0-255 の値範囲を持ちます。この型に対する算術演算で、0-255 の範囲からオーバフローやアンダフローが生じた場合にはキャリ(C)が 1 になります。結果は 256 でモジュロを取った値になります。この型に対するビット演算は、ビット毎に行われます。1 バイトのデータの各ビットには、MSB をビット 7、LSB をビット 0 とするビット位置を表す番号が付けられています。

符号付きバイト型

バイト型の命令で操作できるデータ型です。最上位ビットを符号ビットと見なした 2 の補数表現で、-128 - +127 の値範囲を持ちます。この型に対する算術演算で、-128 - +127 の値範囲からのオーバフローやアンダフローが生じた場合には、オーバフロー・フラグ(OV)が 1 になります。

符号無しワード型

ワード型の命令で操作できるデータ型です。0-65535 の値範囲を持ちます。メモリ上ではビット 7-0 の下位バイトが下位アドレスに、ビット 15-8 の上位バイトが上位アドレスに割り付けられます。データ・メモリ空間ではワードバウンダリのために下位バイトの置かれる下位アドレスは常に偶数アドレスとなります。この型に対する算術演算で、0-65535 の値範囲からのオーバフローやアンダフローが生じた場合には、キャリ(C)が 1 になり結果は 65536 でモジュロをとった値になります。この型に対する論理演算はビット毎に行われます。1 ワードのデータの各ビットには MSB をビット 15、LSB をビット 0 とするビット位置を表す番号が付けられています。

符号付きワード型

ワード型の命令で操作できるデータ型です。最上位ビットを符号ビットとみなした 2 の補数表現で、-32768 - +32767 の値範囲を持ちます。

メモリ上では、ビット 7-0 の下位バイトが下位アドレスに、ビット 15-8 の上位バイトが上位アドレスに割り付けられます。データメモリ空間では、ワードバウンダリのために下位バイトに置かれる下位アドレスは常に偶数アドレスとなります。この型に対する算術演算で、-32768 - +32767 の値範囲からのオーバフローやアンダフローが生じた場合には、オーバフロー・フラグ(OV)が 1 になります。

ビット型

ビット操作命令でアクセスされるデータ型です。0と1のいずれかの値を持ちます。ビット型のレジスタとメモリ上の全てのビットでこのデータ型を表現できます。バイト型のレジスタや、メモリへのアドレッシングを基に0-7のビット位置指定を加えたオペランド記述で指定します。対象の1ビットに対して、転送、ビットテスト＆ジャンプなどができます。

1.3.5.2 アドレス割り付け

メモリ上のアドレス割り付けは、バイト単位で行われています。

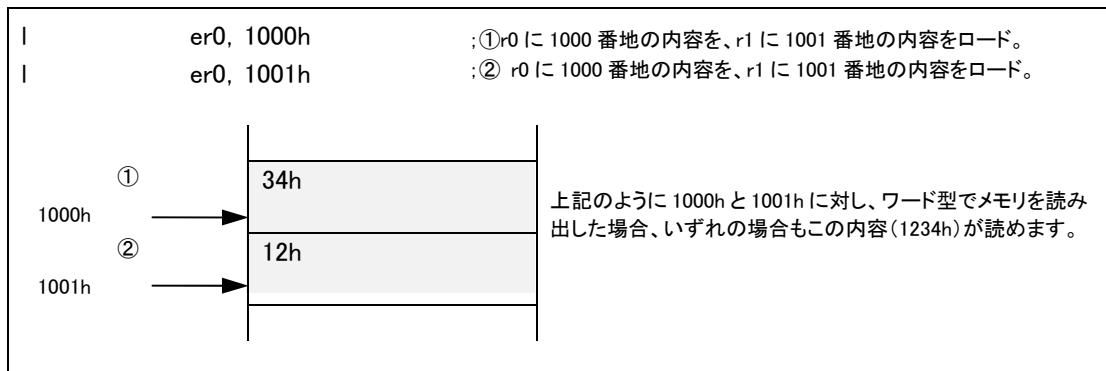
バイトアドレスは、メモリ上の各バイトに対して、個別のアドレスを割り当てたものです。64Kバイト空間に最下位アドレス0より最上位アドレス0FFFFHまでの65536個のアドレスを割り付けています。

U16には、プログラム・メモリ空間とデータ・メモリ空間の独立した2つの空間があり、それぞれの空間にバイトアドレスが割り付けられています。

1.3.5.3 ワードバウンダリ

U16のデータメモリ空間上には、ワードバウンダリ(ワードアライメント)が存在します。ワードバウンダリとは、奇数番地に対するワード型/ダブルワード型/クワッドワード型のメモリアクセスで生じる仕組みのことです。U16コアは、奇数番地に対してワード型/ダブルワード型/クワッドワード型でアクセスした場合、アドレスの最下位ビットは0とみなされ、1番地前の偶数番地から始まるデータをアクセスします。この場合アドレスエラーは発生しません。従って、データメモリ空間上のワード長以上のデータはワードバウンダリに従って配置しなければなりません。なお、ダブルワードバウンダリ、クワッドワードバウンダリは存在しません。従ってダブルワード型/クワッドワード型でメモリをアクセスした場合もワードバウンダリ以外の制限はありません。

プログラム・メモリ空間についてもワードバウンダリが存在します。また、ROMウインドウ越しにアクセスするプログラムメモリ空間にもワードバウンダリは存在します。



メモリ空間上のワードバウンダリ

1.3.5.4 ROM ウィンドウ機能

ROM ウィンドウとは、セグメント 0 の空間の、メモリが割り付けられていない領域に対してプログラムメモリ領域の内容を参照する機能です。U16 ではプログラムメモリ空間上のデータに対しアクセスするための特別な命令は存在しません。ROM ウィンドウを開いた場合、そこに見えるコードメモリに対するアクセスは、メモリに対する読み出し命令を用います。

ROM ウィンドウ領域を参照した場合の命令サイクル数は、同じ命令でデータメモリを参照したときよりも増加します。マシンサイクルについての詳細は、第 3 章“3.3 命令実行時間について”を参照してください。

ROM ウィンドウ領域に対するアクセスは読み出しのみ可能です。ROM ウィンドウ領域に対する書き込み動作の結果は保証されません。

1.3.6 メモリモデル

U16 には、ハードウェア・メモリ・モデルという考え方があり、アクセス可能な最大プログラムメモリ容量を、64K バイト(32K ワード)と 1M バイト(512K ワード)の 2 つから選択することができます。メモリモデルの選択の方法については、ターゲットチップのユーザーズマニュアルを参照してください。

メモリモデルの呼称と、それに対応するメモリモデルの状態を以下に記します。

メモリモデルの呼称	メモリの範囲	CSR CSR 退避レジスタ
SMALL	プログラムメモリ… 0H–FFFFH データメモリ … 0H–FF:FFFFH	—
LARGE	プログラムメモリ… 0H–F:FFFFH データメモリ … 0H–FF:FFFFH	有効

メモリモデルによって以下の項目が変化します。

- アクセスできるプログラムメモリの容量
- サブルーチン呼び出しおよびこれに対応する RT 命令の動作
- 割込みおよびこれに対応する RTI 命令の動作
- メモリへの復帰・待避命令の動作

これらの具体的な違いを以下の表に示します。

	SMALL	LARGE
アクセスできるプログラムの容量	64K バイト (0H – 0FFFFH)	1M バイト (0H – F:FFFFH)
サブルーチン呼び出しで待避される資源	PC CSR	PC CSR
RT 命令で復帰する資源	PC CSR	PC CSR
割込みで待避される資源	PC PSW CSR	PC PSW CSR
RTI 命令で復帰する資源	PC PSW CSR	PC PSW CSR
PUSH 命令で LR/ELR を指定したときにスタックメモリへ待避される資源	LR	LR LCSR
	ELR	ELR ECSR
POP 命令で LR/PC を指定した場合にスタックメモリから復帰される資源	LR	LR LCSR
	PC	PC CSR

1.3.7 割込み動作について

1.3.7.1 割込み成立の条件

割込み条件は、nmice, nmi, mi のいずれかの割込み要求が入力されている状態で、その割込みレベルが現在実行中の割込みレベルよりも等しいか高いときに成立します。割込みレベルとは、割込みの優先度を示す 0~3 の整数で、割込みの種類別にその値が次のように定められています。

各割込み要因に対応する割込みレベルを以下に示します。

割込みの種類	割込みレベル
エミュレータ専用割込み *1	3
ノンマスカブル割込み	2
ソフトウェア割込み	1
マスカブル割込み	1

*1 : エミュレータ専用割込みのため、通常のアプリケーションでは使用できません。

割込みレベルが 0 の場合は、割込みが入っていない状態を表します。

割込みレベル値が大きいほど、優先順位の高い割込みとなります。

割込みレベルは、割込み移行サイクル中に PSW の ELEVEL フィールドにセットされます。

ハードウェアは、発生した割込みのレベルと ELEVEL の値を比較し、発生した割込みのレベルが等しいか大きい場合に、割込み処理に移行します。以下に割込みベクターテーブルアドレスと、ハードウェアの動作の対応を示します。

割込みベクタ テーブルアドレス	備考
0000H	SP(スタックポインタ)の初期値として参照されます。
0002H	ハードウェアリセット入力時、または ELEVEL が 2 以上の状態で BRK 命令を実行した場合のリセットベクタアドレスとして参照されます。
0004H	ELEVEL の値が 1 以下の状態で BRK 命令を実行した場合のベクタアドレスとして参照されます。
0006H	エミュレータ専用割込みの処理に移行します。
0008H ~ 007EH	ノンマスカブル割込み、またはマスカブル割込みの処理に移行します
0080H ~ 00FEH	ソフトウェア割込みの処理に移行します。

各割込みの動作の詳細を以下に示します。

1.3.7.2 ノンマスカブル割込み(NMI)

ノンマスカブル割込みはいっさいのマスクができない割込みです。CPU は、NMI 割込み要求を検出すると直ちに NMI 割込み処理に移行します。NMI 割込み実行中に NMI 割込みを検出した場合も、CPU は NMI 割込みに移行します。例外として CPU の実行状態が以下の場合、NMI はマスクされます。

- ・リセット後(ハードウェアリセット入力、または ELEVEL が 3 の状態で BRK 命令の実行)最初の命令の実行を終了するまでの間。
- ・割込み移行サイクルと、割込みルーチンの先頭にある命令の間。
- ・DSR プリフィックスコードと次の命令の間。

NMI 割込みが発生すると、

- ① PC を ELR2 へ転送。
- ② CSR を ECSR2 へ転送。
- ③ PSW を EPSW2 へ転送。
- ④ PSW の ELEVEL フィールドへ 2 をセット。
- ⑤ CSR を 0 にリセット。
- ⑥ プログラムカウンタ(PC)に、ベクタテーブルに書かれている値をロード。
- ⑦ 割込み先頭アドレス番地に対し、割込みの受け付け禁止。

などの一連のハードウェア処理を行い、NMI 割込み処理の最初の命令を実行します。これらのハードウェア処理に必要なサイクル数は3サイクルですが、[EA+]アドレッシングを持つ命令の直後に NMI が発生すると、CPU 内部で1サイクルのウェイトが挿入されたあと、ハードウェア処理が開始されます。詳細は、“3.3 命令実行時間について”を参照して下さい。

NMI ルーチンからの復帰方法は、NMI 割込みルーチンのプログラム構成により変化します。詳細は、“1.4 例外レベルと退避レジスタの取り扱いについて”を参照して下さい。

1.3.7.3マスカブル割込み(MI)

マスカブル割込みは、内蔵されている周辺ハードウェアや、外部入力端子による様々な要因により発生します。PSW 内の MIE ビットが“1”的状態で割込み条件が成立すると MI 割込みが発生します。ただし、CPU の実行状態が以下の場合、MI はマスクされます。

- ・リセット後(ハードウェアリセット入力、または ELEVEL が 3 の状態で BRK 命令の実行)最初の命令の実行を終了するまでの間。
- ・割込み移行サイクルと、割込みルーチンの先頭にある命令の間。
- ・DSR プリフィックスコードと次の命令の間。
- ・ELEVEL の値が 2 以上の状態の間。

MI 割込みが発生すると、

- ① PC を ELR1 へ転送。
- ② CSR を ECSR1 へ転送。
- ③ PSW を EPSW1 へ転送。
- ④ PSW の ELEVEL フィールドへ 1 をセット。
- ⑤ MIE クリア。
- ⑥ CSR を 0 にリセット。
- ⑦ プログラムカウンタ(PC)に、ベクタテーブルに書かれている値をロード。
- ⑧ 割込み先頭アドレス番地に対し、割込みの受付禁止。

などの一連のハードウェア処理を行い、MI 割込み処理の最初の命令を実行します。これらのハードウェア処理に必要なサイクル数は3サイクルですが、[EA+]アドレッシングを持つ命令の直後に MI が発生すると、CPU 内部で1サイクルのウェイトが挿入されたあと、ハードウェア処理が開始されます。詳細は、“3.3 命令実行時間について”を参照して下さい。

MI ルーチンから復帰方法は、MI 割込みルーチンのプログラム構成により変化します。詳細は後述の“1.4 例外レベルと退避レジスタの取り扱いについて”を参照して下さい。

1.3.7.4 ソフトウェア割込み(SWI)

ソフトウェア割込みはアプリケーションプログラム内で任意に発生させるものです。プログラム実行中に SWI 命令を実行すると SWI 割込みが発生します。ソフトウェア割込みは、割込み番号を SWI 命令のオペランド中で指定します。SWI 割込みが発生すると、

- ① PC を ELR1 へ転送。
- ② CSR を ECSR1 へ転送。
- ③ PSW を EPSW1 へ転送。
- ④ PSW の ELEVEL フィールドへ 1 をセット。
- ⑤ MIE クリア。
- ⑥ CSR を 0 にリセット。
- ⑦ プログラムカウンタ(PC)に、ベクターテーブルに書かれている値をロード。
- ⑧ 割込み先頭アドレス番地に対し、割込みの受け付け禁止。

などの一連のハードウェア処理を行い、SWI 割込み処理の最初の命令を実行します。これらのハードウェア処理に必要なサイクル数は3サイクルですが、[EA+]アドレッシングを持つ命令の直後に SWI を実行すると、CPU 内部で1サイクルのウェイトが挿入されたあと、ハードウェア処理が開始されます。詳細は、“3.3 命令実行時間について”を参照して下さい。

割込みからの復帰方法は“1.4 例外レベルと退避レジスタの取り扱いについて”を参照して下さい。

1.4例外レベルと退避レジスタの取り扱いについて

U16 は、サブルーチンコール時と割込み発生時の戻り番地保持のため、PC 退避レジスタおよび CSR 退避用レジスタを備えています。また割込みエントリ時のプログラム実行状態保持のため、PSW 退避レジスタを備えています。各レジスタの名称と、用途を以下に記します。

PC 退避用レジスタ

名称	用途
LR	BL 命令によるサブルーチンコール時に、戻り PC 値を退避。
ELR1	マスカブル割込み発生時、あるいは SWI 命令実行時に、戻り PC 値を退避。
ELR2	ノンマスカブル割込み発生時に、戻り PC 値を退避。
ELR3	エミュレータ専用割込み発生時に、戻り PC 値を退避。

CSR 退避用レジスタ

名称	用途
LCSR	BL 命令によるサブルーチンコール時、あるいは SWI 命令実行時に、戻り CSR の値を退避。
ECSR1	マスカブル割込み発生時、あるいは SWI 命令実行時に、戻りセグメント値を退避。
ECSR2	ノンマスカブル割込み発生時に、戻りセグメント値を退避。
ECSR3	エミュレータ専用割込み発生時に、戻りセグメント値を退避。

PSW 退避用レジスタ

名称	用途
EPSW1	マスカブル割込み発生時、あるいは SWI 命令実行時に、割込み発生直前の PSW を退避。
EPSW2	ノンマスカブル割込み発生時に、割込み発生直前の PSW を退避。
EPSW3	エミュレータ専用割込み発生時に、割込み発生直前の PSW を退避。

LR,LCSR は、BL 命令実行によるサブルーチンコールおよび RT 命令によるサブルーチンからの復帰時にアクセスされます。

ELR,ECSR,および EPSW は、PSW の ELEVEL の値を指数とするいずれかひとつのレジスタが選択され、割込み発生時および RTI 命令実行時にアクセスされます。

退避レジスタは各要因に対して 1 つ備えてあります。通常、サブルーチンから復帰する場合は RT 命令を、割込みから復帰する場合は RTI 命令を使用しますが、サブルーチンがネストする場合あるいは多重割込みを許可する場合は、退避レジスタの内容が上書きされてしまうため、RT 命令および RTI 命令は使用できません。このような場合は、割込みルーチンまたはサブルーチンの先頭に PUSH 命令を置き、退避レジスタの内容をスタックメモリに退避します。そして RT 命令ではなく POP 命令を使用してサブルーチンおよび割込みルーチンから復帰します。

このように、アプリケーションプログラムにおける上記レジスタ群の取り扱い方法は、U16 の実行状態により異なりますので、アプリケーションプログラム設計時に注意を払う必要があります。次ページ以降に、U16 の実行状態とアプリケーションプログラムでの対応について記します。

U16 の実行状態を、ELEVEL の値ごとに分け、それぞれの状態下でサブルーチンを呼び出す場合と呼び出さない場合、および、多重割込みを許可する場合と禁止する場合に細分化した場合の、プログラミング時の注意事項を記します。

状態 A: 割込みが実行されていない状態

ELEVEL が 0 の実行状態で、退避レジスタとして LR,LCSR が選択されます。

サブルーチンを呼び出すか呼び出さないかで、サブルーチンの開始直後の処理と終了処理が、以下のように異なります。

A-1: サブルーチンを呼び出さない場合

- サブルーチン開始直後の処理
注意すべき取り扱いはありません。
- サブルーチン終了処理

RT 命令を指定し、PC に LR レジスタの内容を復帰させます。

記述例: 状態 A-1

```
Sub_A-1:          ; サブルーチン開始
:
:
RT               ; PC を LR より復帰
; サブルーチン終了
```

A-2: サブルーチンを呼び出す場合

- サブルーチン開始直後の処理

“PUSH LR”命令を指定し、戻り番地をスタックへ退避します。

- サブルーチン終了処理

RT 命令の代わりに“POP PC”を指定し、PC にスタックの内容を復帰させます。

記述例: 状態 A-2

```
Sub_A-2:          ; サブルーチン開始
PUSH LR          ; LR をスタックに退避
:
:
BL   Sub_1       ; サブルーチン Sub_1 呼び出し
:
POP PC          ; PC をスタックより復帰
; サブルーチン終了
```

```
Sub_1:          ; サブルーチン開始
:
:
RT              ; PC を LR より復帰
; サブルーチン終了
```

状態 B:マスカブル割込み実行中

ELEVEL が1の実行状態で、退避レジスタとして ELR1,EPSW1,ECSR1 が選択されます。

サブルーチンを呼び出す場合と呼び出さない場合、および、多重割込みを禁止する場合と許可する場合で、以下のように処理が異なります。

B-1:割込みルーチン内でサブルーチンを呼び出さない場合

B-1-1:多重割込みを禁止する場合

- 割込みルーチン実行開始直後の処理

注意すべき事項は特にありません。

- 割込みルーチン実行終了時の処理

RTI 命令を配置し、PC に ELR1 レジスタの内容を、PSW に EPSW1 レジスタの内容を復帰させます。

記述例:状態 B-1-1

```
Intrpt_B-1-1:          ;割込みルーチン開始
:
:
RTI                  ;PC を ELR より復帰
;PSW を EPSW より復帰
;割込みルーチン終了
```

B-1-2:多重割込みを許可する場合

- 割込みルーチン実行開始直後の処理

“PUSH ELR, EPSW”を指定し、割込みの戻り番地と PSW の状態をスタックに退避します。

- 割込みルーチン実行終了時の処理

RTI命令の代わりに“POP PSW, PC”を指定し、PC と PSW にスタックの内容を復帰させます。

記述例:状態 B-1-2

```
Intrpt_B-1-2:          ;割込みルーチン開始
PUSH ELR,EPSW         ;先頭で ELR,EPSW を退避
:
:
EI                   ;割込み許可
:
POP PSW,PC          ;PC をスタックより復帰
;PSW をスタックより復帰
;割込みルーチン終了
```

B-2: 割込みルーチン内でサブルーチンを呼び出す場合

B-2-1: 多重割込みを禁止する場合

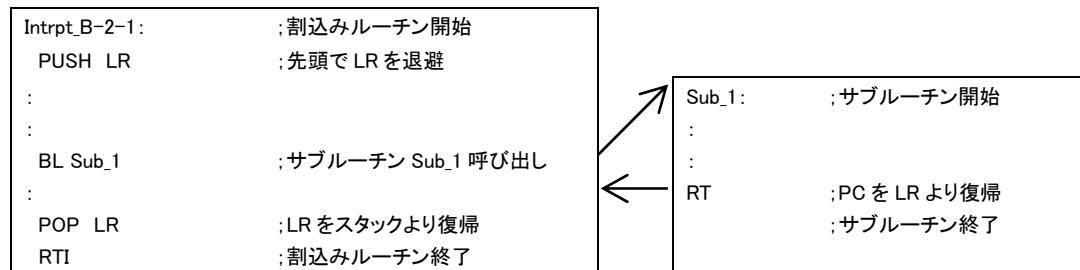
- ・ 割込みルーチン実行開始直後の処理

“PUSH LR”命令を指定し、サブルーチンの戻り番地をスタックに退避します。

- ・ 割込みルーチン実行終了時の処理

RTI 命令の直前に“POP LR”を指定し、サブルーチンの戻り番地を LR に復帰させた後、割込みから復帰します。

記述例: 状態 B-2-1



B-2-2: 多重割込みを許可する場合

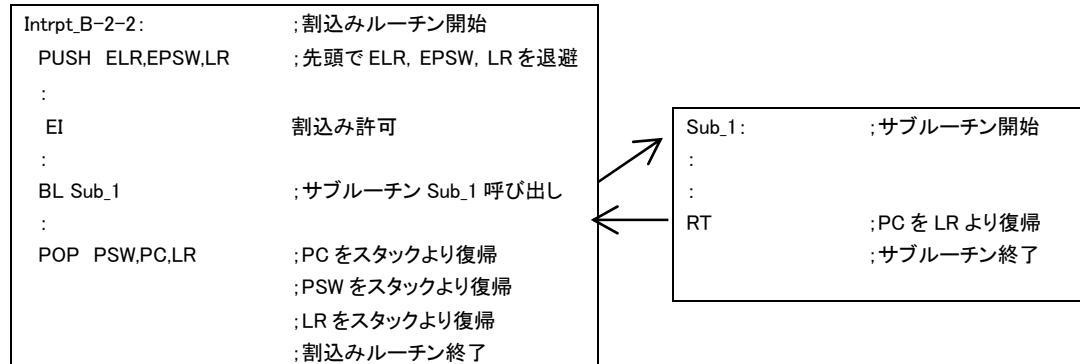
- ・ 割込みルーチン実行開始直後の処理

“PUSH ELR、EPSW、LR”を指定し、割込みの戻り番地、サブルーチンの戻り番地および EPSW1 の状態をスタックに退避します。

- ・ 割込みルーチン実行終了時の処理

RTI 命令の代わりに“POP PSW、PC、LR”を指定し、割込みの戻り番地の退避データは PC へ、EPSW1 の退避データは PSW へ、LR の退避データは LR に復帰させます。

記述例: 状態 B-2-2



状態 C::ノンマスカブル割込み実行中

ELEVEL が2の状態で、退避レジスタとして ELR2,EPSW2,ECSR2 が選択されます。

サブルーチンを呼び出す場合と呼び出さない場合で、以下のように処理が異なります。

C-1: 割込みルーチン内でサブルーチンを呼び出さない場合

- ・ 割込みルーチンの実行開始直後の処理
“PUSH ELR, EPSW”を指定し、割込みの戻り番地と PSW の状態を、スタックに退避します。
- ・ 割込みルーチンの実行終了処理
“POP PSW,PC”を指定し、PC と PSW にスタックの内容を復帰させます。

記述例: 状態 C-1

```
Intrpt_C-1:          ;割込みルーチン開始
    PUSH ELR,EPSW      ;先頭で ELR, EPSW を退避
    :
    :
    POP PSW,PC         ;PC をスタックより復帰
                        ;PSW をスタックより復帰
                        ;LR をスタックより復帰
                        ;割込みルーチン終了
```

C-2: 割込みルーチン内でサブルーチンを呼び出す場合

- ・ 割込みルーチンの開始直後の処理
“PUSH ELR, EPSW, LR”を指定し、割込みの戻り番地、サブルーチンの戻り番地および EPSW の状態をスタックに退避します。
- ・ 割込みルーチンの終了処理
“POP PSW, PC, LR”を指定し、割込みの戻り番地の退避データは PC へ、EPSW の退避データは PSW へ、LR の退避データは LR に復帰させます。

記述例: 状態 C-2

```
Intrpt_C-2:          ;開始
    PUSH ELR,EPSW,LR    ;先頭で ELR, EPSW, LR を退避
    :
    :
    BL Sub_1            ;サブルーチン Sub_1 呼び出し
    :
    POP PSW,PC,LR       ;PC をスタックより復帰
                        ;PSW をスタックより復帰
                        ;LR をスタックより復帰
                        ;終了
```

```
Sub_1:           ;
    :
    RT              ;PC を LR より復帰
                    ;サブルーチン終了
```

このようにアプリケーションプログラムにおける上記レジスタ群の取り扱い方法は、割込み発生時、およびサブルーチンを呼び出し時の CPU の実行状態により異なりますので、アプリケーションプログラム設計時に注意を払う必要があります。

1.5ノンマスカブル割込みに関する注意

本項では、C 言語でプログラム開発する際の、ノンマスカブル割込みに関する注意点を記します。

ノンマスカブル割込み要求は禁止できないため、プログラム中でノンマスカブル割込みの多重割込み対策を実施しなければ、ノンマスカブル割込み実行中に他のノンマスカブル割込みに割り込まれ、プログラムが割込みルーチンから復帰できなくなる可能性があります。

対策として、ターゲットチップがノンマスカブル割込みを使用する場合は、INTERRUPT プラグマで category を 2 に設定してください。このようにすることで、多重割込みが発生しても問題なく動作するよう、割込みルーチンの入口、出口に ELR 等の退避・復帰コードが生成されます。

```
static void int_WDTINT(void);

#pragma interrupt int_WDTINT 0x08 2

static void int_WDTINT(void)

{
    /* 処理*/
}
```

上記の C プログラムに対し、コンパイラが出力するアセンブリコードは以下のようになります。

```
_int_WDTINT :
    push elr, epsw
    ...
    pop psw, pc
```

※割込みルーチン内にて関数呼び出しを伴う場合には、上記 ELR、EPSW の他に、LR および呼び出される関数によって変更される可能性のあるレジスタ(EA, R0~R3)に対する退避・復帰コードが生成されます

1.6 割込み禁止状態について

U16 には、割込み条件が成立していても割込みを一切受け付けない動作状態があります。これを割込み禁止状態と呼びます。

割込み禁止状態と、その状態における割込みの取り扱いは次のようになります。

状態 1. 割込み移行サイクルと、割込みルーチンの先頭にある命令の間

ここで割込み条件が成立した場合、すでに許可されている割込みと対応する割込みルーチンの先頭にある命令の実行直後に割込みが発生します。

状態 2. DSR プリフィックスコードと次の命令の間

ここで割込み条件が成立した場合、DSR プリフィックスコードと、次の命令実行直後に割込みが発生します。

DSR プリフィックスコードについては、“1.3.4 DSR プリフィックスコードについて”を参照してください。

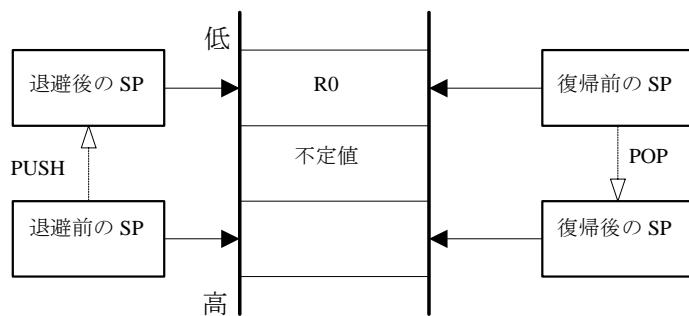
DSR プリフィックスコードを連続して配置すると割込み禁止状態は解除されますが、プログラマの意図しない CPU 動作となる可能性があるため、プログラム中でこのような記載をしないようにしてください。

1.7 スタックの変化について

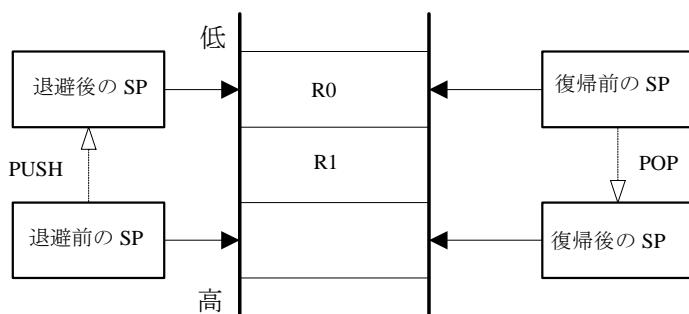
ここでは PUSH 命令および POP 命令実行時のスタックの変化をまとめます。各々の命令の詳細については、第3章をご覧下さい。命令実行時スタックポインタは常に偶数バイト単位で変化します。奇数バイトのデータを退避する場合、レジスタの退避が行われる前に、1サイクルのダミーサイクルが挿入されます。ダミーサイクル実行中は1バイトの不定値がメモリへの書き込まれます。奇数バイトのデータを復帰する場合には、レジスタ復帰の後、SP のみが1インクリメントされるダミーサイクルが挿入されます。ダミーサイクル実行中にスタックポインタの指す内容は復帰されません。また LR および ELR を選択した場合は、その動作がメモリモデルの影響を受けます。

以下に PUSH・POP 命令実行時のスタック動作例を関係を示します。

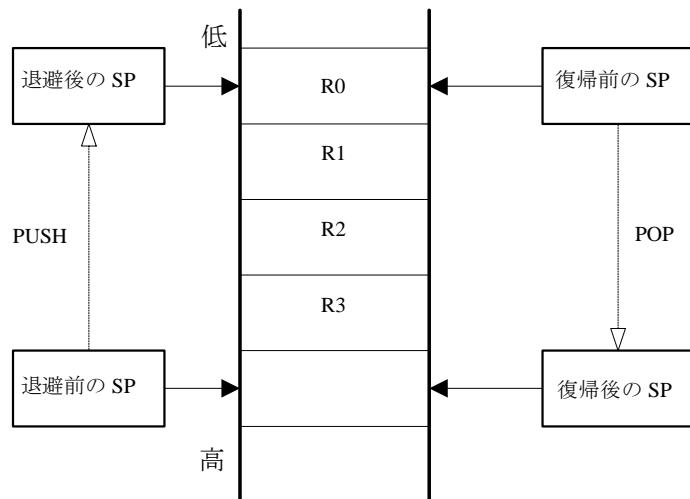
PUSH R0 / POP R0 の動作



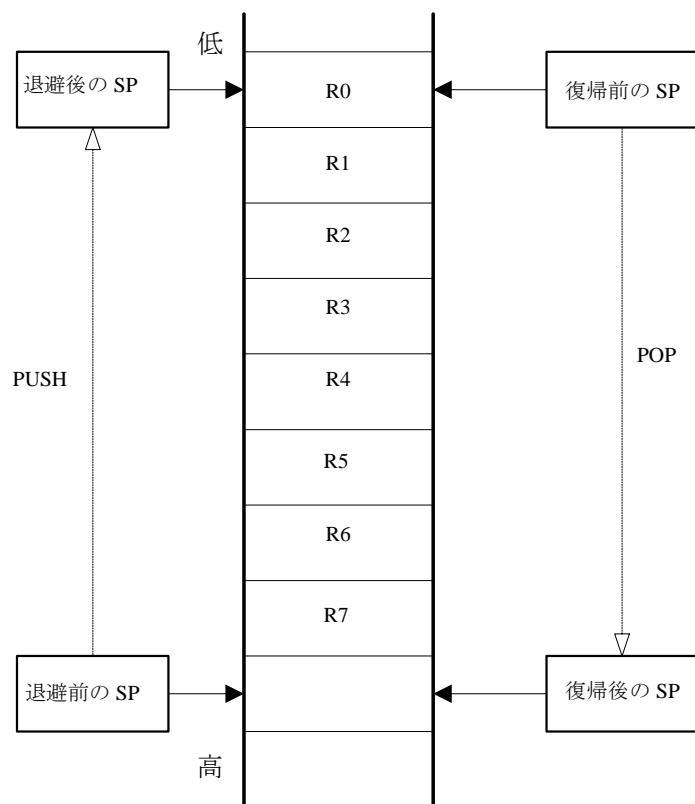
PUSH ER0 / POP ER0 の動作



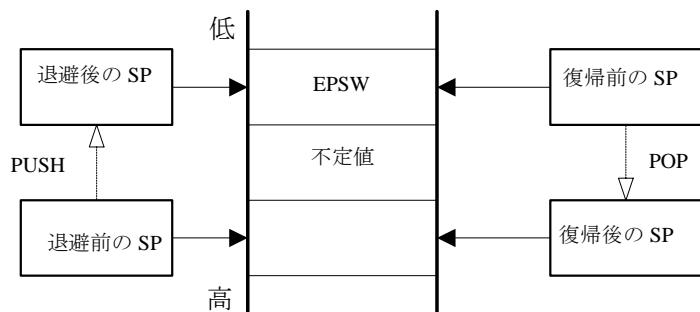
PUSH XR0 / POP XR0 の動作



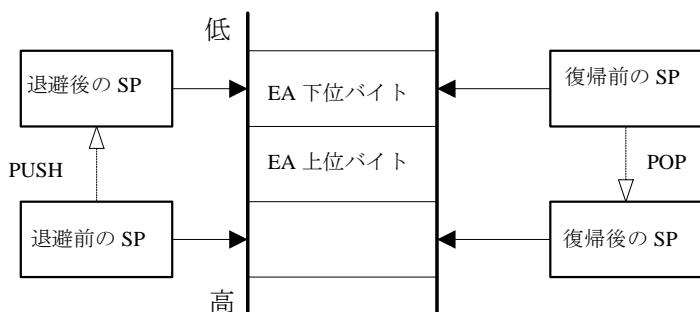
PUSH QR0 / POP QR0 の動作



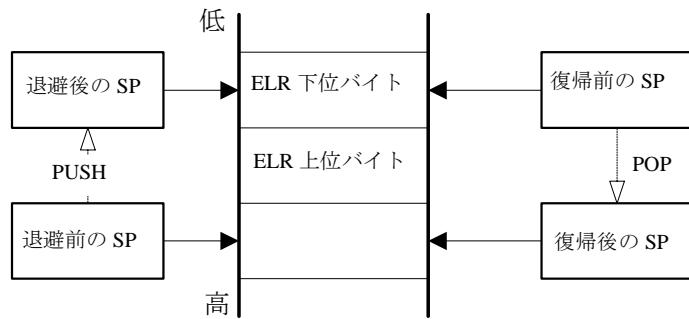
PUSH EPSW / POP PSW の動作



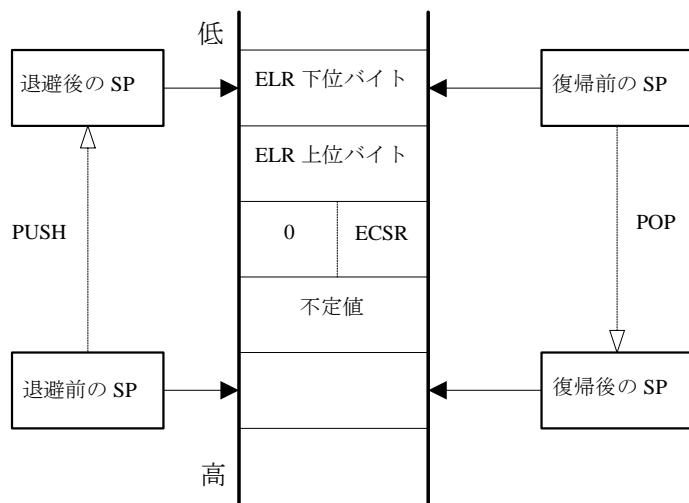
PUSH EA / POP EA の動作



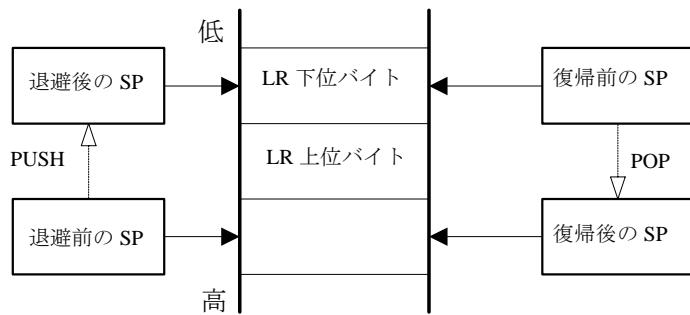
PUSH ELR / POP PC の動作(スモールモデル)



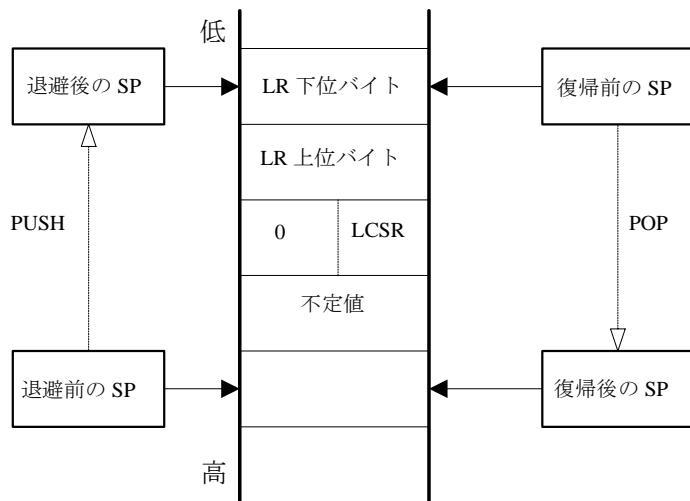
PUSH ELR / POP PC の動作(ラージモデル)



PUSH LR / POP LR の動作(スマールモデル)



PUSH LR / POP LR の動作(ラージモデル)



2アドレスシングモード

2.1アドレッシングモード

U16 のアドレッシングモードは、U16 内部のレジスタとコプロセッサレジスタに対応するレジスタアドレッシング、データメモリ空間および ROM ウィンドウで設定された領域に対応するメモリアドレッシング、プログラムメモリ空間に対してのみ対応するプログラムメモリアドレッシング、および命令自身の中にオペランドを指定する即値アドレッシングの 4 種類があります。

2.2レジスタアドレッシング

指定したレジスタそのものがアクセスの対象となります。レジスタアドレッシングの記述は、以下のとおりです。

アドレッシング記述	機能
Rn	バイト型の汎用レジスタ (Rn) がアクセスの対象となります。
ERn	ワード型の汎用レジスタ (ERn) がアクセスの対象となります。インストラクション表の記述で、オペランドに ERn が記されている場合、ER12 の代用として BP, ER14 の代用として FP の記述が可能です。
XRn	ダブルワード型の汎用レジスタ (XRn) がアクセスの対象となります。
QRn	クワッドワード型の汎用レジスタ (QRn) がアクセスの対象となります。
CRn	バイト型のコプロセッサレジスタ (CRn) がアクセスの対象となります。
CERn	ワード型のコプロセッサレジスタ (CERn) がアクセスの対象となります。
CXRn	ダブルワード型のコプロセッサレジスタ (CXRn) がアクセスの対象となります。
CQRn	クワッドワード型のコプロセッサレジスタ (CQRn) がアクセスの対象となります。
PC	プログラムカウンタがアクセスの対象となります。
LR	リンクレジスタがアクセスの対象となります。
EA	EA レジスタがアクセスの対象となります。
SP	スタックポインタがアクセスの対象となります。
PSW	プログラムステータスワードがアクセスの対象となります。
ELR	例外処理用のリンクレジスタがアクセスの対象となります。
ECSR	CSR 退避レジスタがアクセスの対象となります。
EPSW	PSW 退避レジスタがアクセスの対象となります。
Rn.bit_offset	汎用レジスタ Rn の bit_offset で示されるビット位置のデータがアクセスの対象となります。

2.3メモリアドレッシング

メモリアドレッシングは、データメモリ空間上のメモリをアクセスの対象とするものです。物理セグメント#0以外のメモリをアクセスする場合は、DSRプリフィックスコードを使用して、アクセスすべき物理セグメントのアドレスを設定します。しかしDSRプリフィックスコードは、単独命令として取り扱うとプログラマの意図しない動作を行う可能性があるため、アセンブラーでは、常にメモリアクセス命令と組み合わせて使用します。DSRプリフィックスコードと、オペランド記述は以下のように対応しています。

DSR プリフィックスコード	ハードウェアの動作	オペランド記述
1110_0011_iiii_iiii	DSRに、iiii_iiiiで示される8bit即値 <i>pseg_addr</i> ：または FARをライトする。	
1001_0000_dddd_1111	DSRに、ddddで指定される汎用レジスタの内容をライトする。	
1111_1110_1001_1111	現在のDSRの値を有効にする。	DSR :

DSRプリフィックスコードとメモリアクセス命令を組み合わせた場合の記述例を以下に示します。

オペランドの記述例	命令の動作
L R0, <u>1:2345H</u>	DSR ← 1 R0 ← [2345H]
L R0, <u>R1:[ER2]</u>	DSR ← R1 R0 ← [ER2]
ST R1, <u>DSR:[EA]</u>	[(DSR<<16) EA] ← R1

上の表の下線部分が、実際のDSRプリフィックスコードに対応する記述となります。

DSRプリフィックスコードを使用しない場合は、アクセスの対象となるメモリの物理セグメントは#0となります。

2.3.1 レジスタ間接アドレッシング

レジスタの内容をアドレスとするデータメモリ空間上のメモリが、アクセスの対象となります。レジスタ間接アドレッシングの記述は以下のとおりです。

アドレッシング記述	機能																							
[EA]	<p>EA レジスタの内容をアドレスとする物理セグメント#0 のデータメモリ空間メモリがアクセスの対象となります。</p> <p>実効アドレス計算方法</p> <p>実効アドレス</p>																							
pseg_addr:[EA]	物理セグメント指定付きのアドレッシングです。物理セグメント #pseg_addr のデータメモリ空間がアクセスの対象となります。																							
DSR:[EA]	物理セグメント指定付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。																							
Rd:[EA]	物理セグメント指定付きのアドレッシングです。汎用レジスタ Rd によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。																							
[EA+]	<p>EA レジスタの内容をアドレスとする物理セグメント#0 のデータメモリ空間上のメモリが、アクセスの対象となります。命令実行終了後、EA の内容はオペランドサイズに応じて更新されますが、EA の内容とオペランドサイズにより、更新される値が以下のように異なります。</p> <table border="1"> <thead> <tr> <th>オペランドサイズ</th> <th>EA の内容</th> <th>加算される値</th> </tr> </thead> <tbody> <tr> <td rowspan="2">バイト型</td> <td>偶数</td> <td>1</td> </tr> <tr> <td>奇数</td> <td>1</td> </tr> <tr> <td rowspan="2">ワード型</td> <td>偶数</td> <td>2</td> </tr> <tr> <td>奇数</td> <td>1</td> </tr> <tr> <td rowspan="2">ダブルワード型</td> <td>偶数</td> <td>4</td> </tr> <tr> <td>奇数</td> <td>3</td> </tr> <tr> <td rowspan="2">クワッドワード型</td> <td>偶数</td> <td>8</td> </tr> <tr> <td>奇数</td> <td>7</td> </tr> </tbody> </table>	オペランドサイズ	EA の内容	加算される値	バイト型	偶数	1	奇数	1	ワード型	偶数	2	奇数	1	ダブルワード型	偶数	4	奇数	3	クワッドワード型	偶数	8	奇数	7
オペランドサイズ	EA の内容	加算される値																						
バイト型	偶数	1																						
	奇数	1																						
ワード型	偶数	2																						
	奇数	1																						
ダブルワード型	偶数	4																						
	奇数	3																						
クワッドワード型	偶数	8																						
	奇数	7																						
	<p>実効アドレス計算方法</p> <p>実効アドレス</p> <p>メモリアクセス後、インクリメントされる</p>																							

第2章 命令の詳細

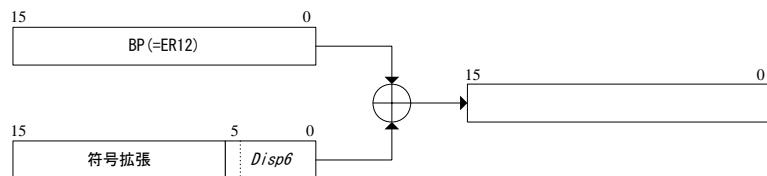
アドレッシングモード

アドレッシング記述	機能				
<i>pseg_addr:[EA+]</i>	物理セグメント指定付きのアドレッシングです。物理セグメント # <i>pseg_addr</i> のデータメモリ空間が指定されます。				
<i>DSR:[EA+]</i>	物理セグメント指定付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間が指定されます。				
<i>Rd:[EA+]</i>	物理セグメント指定付きのアドレッシングです。汎用レジスタ <i>Rd</i> によって示される物理セグメントのデータメモリ空間が指定されます。				
<i>[ERn]</i>	ワード型汎用レジスタ <i>ERn</i> の内容をアドレスとする物理セグメント#0 のデータメモリ空間上のメモリが、アクセスの対象となります。[ER12]の代用として[BP], [ER14]の代用として[FP]を記述することも可能です。				
	<table style="width: 100%;"><thead><tr><th style="text-align: center;">実効アドレス計算方法</th><th style="text-align: center;">実効アドレス</th></tr></thead><tbody><tr><td style="text-align: center;"></td><td style="text-align: center;"></td></tr></tbody></table>	実効アドレス計算方法	実効アドレス		
実効アドレス計算方法	実効アドレス				
<i>pseg_addr:[ERn]</i>	物理セグメント指定付きのアドレッシングです。物理セグメント # <i>pseg_addr</i> のデータメモリ空間がアクセスの対象となります。				
<i>DSR:[ERn]</i>	物理セグメント指定付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。				
<i>Rd:[ERn]</i>	物理セグメント指定付きのアドレッシングです。汎用レジスタ <i>Rd</i> によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。				
<i>Disp16[ERn]</i>	<i>Disp16+ERn</i> をアドレスとするデータメモリ空間上のメモリが、アクセスの対象となります。				
	<table style="width: 100%;"><thead><tr><th style="text-align: center;">実効アドレス計算方法</th><th style="text-align: center;">実効アドレス</th></tr></thead><tbody><tr><td style="text-align: center;"></td><td style="text-align: center;"></td></tr></tbody></table>	実効アドレス計算方法	実効アドレス		
実効アドレス計算方法	実効アドレス				
<i>pseg_addr:Disp16[ERn]</i>	物理セグメント指定付きのアドレッシングです。物理セグメント # <i>pseg_addr</i> のデータメモリ空間がアクセスの対象となります。				
<i>DSR:Disp16[ERn]</i>	物理セグメント指定付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。				
<i>Rd:Disp16[ERn]</i>	物理セグメント指定付きのアドレッシングです。汎用レジスタ <i>Rd</i> によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。				

アドレッシング記述 機能

Disp6[BP] *Disp6+BP* をアドレスとするデータメモリ空間上のメモリがアクセスの対象となります。加算に際して、*Disp6* は符号拡張されます。DSR プリフィックスがない場合、物理セグメント#0 に限定したアドレッシングになります。

実効アドレス計算方法 実効アドレス



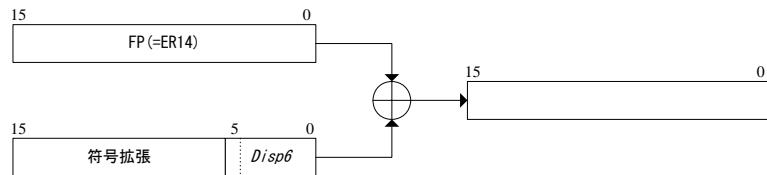
pseg_addr:Disp6[BP] 物理セグメント指定付きのアドレッシングです。物理セグメント #*pseg_addr* のデータメモリ空間がアクセスの対象となります

DSR:Disp6[BP] 物理セグメント指定付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。

Rd:Disp6[BP] 物理セグメント指定付きのアドレッシングです。汎用レジスタ *Rd* によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。

Disp6[FP] *Disp6+FP* をアドレスとするデータメモリ空間上のメモリがアクセスの対象となります。加算に際して、*Disp6* は符号拡張されます。DSR プリフィックスがない場合、物理セグメント#0 に限定したアドレッシングになります。

実効アドレス計算方法 実効アドレス



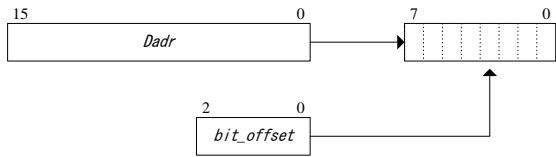
pseg_addr:Disp6[FP] 物理セグメント指定付きのアドレッシングです。物理セグメント #*pseg_addr* のデータメモリ空間がアクセスの対象となります。

DSR:Disp6[FP] 物理セグメント指定付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。

Rd:Disp6[FP] 物理セグメント指定付きのアドレッシングです。汎用レジスタ *Rd* によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。

2.3.2 ダイレクトアドレッシング

記述した値をアドレスとするデータメモリ空間上のメモリをアクセスの対象とします。ダイレクトアドレッシングの記述は以下のとおりです。

アドレッシング記述	機能
<i>Dadr</i>	記述した値をバイトアドレスとするデータメモリ空間のメモリがアクセスの対象となります。 実効アドレス 
<i>pseg_addr:Dadr</i>	物理セグメント指定付きのアドレッシングです。物理セグメント <i>pseg_addr</i> のデータメモリ空間がアクセスの対象となります。
<i>DSR:Dadr</i>	物理セグメント付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間がアクセスの対象となります。
<i>Rd:Dadr</i>	物理セグメント付きのアドレッシングです。汎用レジスタ <i>Rd</i> によって示される物理セグメントのデータメモリ空間が指定されます。
<i>Dbitadr</i>	記述した値をビットアドレスとするデータメモリ空間のメモリがアクセスの対象となります。記述形式は <i>Dadr.bit_offset</i> なります。 <i>Dadr</i> は対象メモリのアドレス式を表し、 <i>bit_offset</i> はメモリ内のビット位置を表します。 実効アドレス計算方法  実効アドレス
<i>pseg_addr:Dbitadr</i>	物理セグメント付きのアドレッシングです。物理セグメント <i>pseg_addr</i> のデータメモリ空間がアクセスの対象となります。
<i>DSR:Dbitadr</i>	物理セグメント付きのアドレッシングです。DSR によって示される物理セグメントのデータメモリ空間が指定されます。
<i>Rd:Dbitadr</i>	物理セグメント付きのアドレッシングです。汎用レジスタ <i>Rd</i> によって示される物理セグメントのデータメモリ空間が指定されます。

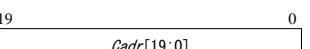
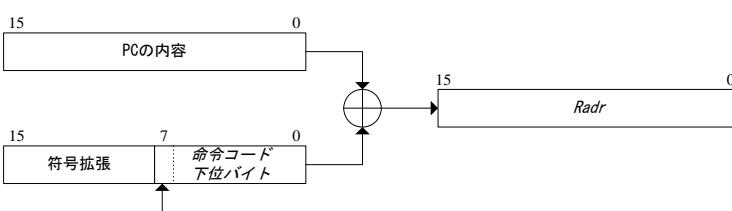
2.4 即値アドレッシング

記述した値そのものが対象となります。即値アドレッシングの記述は、以下のとおりです。

アドレッシング記述	機能
#imm8	記述した値は、8ビット即値として扱われます。
#signed8	記述した値は、符号付きの8ビット即値として扱われます。 ADD SP, #imm8 を記述した場合に、imm8 は signed8 として扱われます。 値の範囲は、 $-128 \leq \text{signed}8 \leq +127$ となります。
#unsigned8	記述した値は、符号なしの8ビット即値として扱われます。 MOV PSW, #imm8 を記述した場合に、imm8 は unsigned8 として扱われます。 値の範囲は、 $0 \leq \text{unsigned}8 \leq \text{OFFH}$ となります。
#width	記述した値は、シフト幅として扱われます。 値の範囲は、 $0 \leq \text{width} \leq 7$ となります。
#snum	記述した値は、SWI命令のベクタ番号として扱われます。 値の範囲は、 $0 \leq \text{snum} \leq 63$ となります。
#imm7	記述した値は、符号付きの7ビット即値として扱われます。 値の範囲は、 $-64 \leq \text{imm7} \leq +63$ となります。

2.5 プログラムメモリアドレッシング

プログラムメモリ空間上のメモリがアクセスの対象となります。プログラムメモリアドレッシングの記述は、以下のとおりです。

アドレッシング記述	機能
<i>Cadr</i>	B, BL の分岐先アドレスとなります。 <i>Cadr</i> は物理セグメントアドレスを含みますので、異なる物理セグメントアドレスへの分岐が可能です。
	実効アドレス
	
<i>Radr</i>	条件分岐、または最適化分岐擬似命令の分岐先アドレスとなります。分岐先アドレスは、同一物理セグメント内に限定されます。
	実効アドレス計算方法
	実効アドレス
	
<i>ERn</i>	ワード型汎用レジスタ <i>ERn</i> の内容が分岐先アドレスとなります。 B, BL で用いられます。 分岐先アドレスは、同一物理セグメント内に限定されます。
	実効アドレス
	

3 命令の詳細

この章では nX-U16/100 コアの各命令の機能について詳細に説明します。

3.1概要

nX-U16/100 コアの命令には、オペランドを記述しない命令と、第 1 オペランドまたは第 2 オペランドを持つ命令があります。

[オペランド表記]

オペランド表記は

<デスティネーション>、<ソース>

の順番で表記します。

オペランドには、nX-U16/100 コアのアドレッシングモードに対応する記述を表記することができます。アドレッシング記述に関する詳細は、第 2 章を参照してください。

[記号表記]

命令の機能をわかりやすく表現するため、以下の記号を使用しています。

記号	意味
←	データの代入方向
+、または⊕	加算
-	減算
*	乗算
/	除算
>>	右シフト
<<	左シフト
=	等号
!=	不等号
&	論理積
	論理和
^	排他的論理和
~	反転論理

第3章 命令の詳細 インストラクションセット

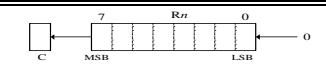
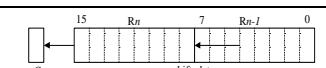
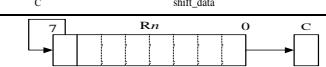
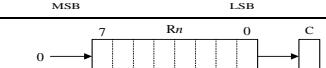
3.2nX-U16/100コアの命令セット機能別分類表

各命令の詳細な動作については、3.4 各命令の説明 を参照して下さい。

演算命令

ニーモニック	第1オペランド	第2オペランド	C	Z	S	OV	MIE	HC	機能
ADD	Rn	Rm	*	*	*	*	*	*	加算命令 Rn ← Rn+obj
		#imm8	*	*	*	*	*	*	
MOV	Rn	Rm	*	*					転送命令 Rn ← obj
		#imm8	*	*					
ADDC	Rn	Rm	*	*	*	*	*	*	キャリ付き加算命令 Rn ← Rn+obj+c
		#imm8	*	*	*	*	*	*	
CMP	Rn	Rm	*	*	*	*	*	*	比較命令 Rn-obj
		#imm8	*	*	*	*	*	*	
CMPC	Rn	Rm	*	*	*	*	*	*	キャリ付き比較命令 Rn-obj-c
		#imm8	*	*	*	*	*	*	
AND	Rn	Rm	*	*					論理積命令 Rn ← Rn&obj
		#imm8	*	*					
OR	Rn	Rm	*	*					論理和命令 Rn ← Rn obj
		#imm8	*	*					
XOR	Rn	Rm	*	*					排他的論理和命令 Rn ← Rn ^ obj
		#imm8	*	*					
SUB	Rn	Rm	*	*	*	*	*	*	減算命令 Rn ← Rn-Rm
SUBC	Rn	Rm	*	*	*	*	*	*	キャリ付き減算命令 Rn ← Rn-Rm-c
MOV	ERn	ERm	*	*					転送命令 ERn ← obj
		#imm7	*	*					
ADD	ERn	ERm	*	*	*	*	*	*	加算命令 ERn ← ERn+obj
		#imm7	*	*	*	*	*	*	
CMP	ERn	ERm	*	*	*	*	*	*	比較命令 ERn-ERm

シフト命令

ニーモニック	第1オペランド	第2オペランド	C	Z	S	OV	MIE	HC	機能
SLL	Rn	Rm	*						パイト型左シフト命令
		#width	*						
SLLC	Rn	Rm	*						論理左シフト継続命令
		#width	*						
SRA	Rn	Rm	*						算術右シフト命令
		#width	*						
SRL	Rn	Rm	*						論理右シフト命令
		#width	*						
SRLC	Rn	Rm	*						論理右シフト継続命令
		#width	*						

ロード/ストア命令

ニーモニック	第1オペランド	第2オペランド	C	Z	S	OV	MIE	HC	機能
L	Rn	[EA] <i>pseg_addr</i> :[EA] DSR:[EA] Rd:[EA]	*	*					バ'ト型転送命令 Rn←[EA]
		[EA+] <i>pseg_addr</i> :[EA+] DSR:[EA+] Rd:[EA+]	*	*					バ'ト型転送命令 Rn←[EA] EA←EA+1
		[ERm] <i>pseg_addr</i> :[ERm] DSR:[ERm] Rd:[ERm]	*	*					バ'ト型転送命令 Rn←[ERm]
		<i>Disp16</i> [ERm] <i>pseg_addr</i> : <i>Disp16</i> [ERm] DSR: <i>Disp16</i> [ERm] Rd: <i>Disp16</i> [ERm]	*	*					バ'ト型転送命令 Rn← Disp16[ERm]
		<i>Disp0</i> [BP] <i>pseg_addr</i> : <i>Disp0</i> [BP] DSR: <i>Disp0</i> [BP] Rd: <i>Disp0</i> [BP]	*	*					バ'ト型転送命令 Rn← Disp0[BP]
		<i>Disp0</i> [FP] <i>pseg_addr</i> : <i>Disp0</i> [FP] DSR: <i>Disp0</i> [FP] Rd: <i>Disp0</i> [FP]	*	*					バ'ト型転送命令 Rn← Disp0[FP]
		<i>Dadr</i> <i>pseg_addr</i> : <i>Dadr</i> DSR: <i>Dadr</i> Rd: <i>Dadr</i>	*	*					バ'ト型転送命令 Rn← Dadr
ERn	[EA] <i>pseg_addr</i> :[EA] DSR:[EA] Rd:[EA]	*	*						ワード型転送命令 ERn←[EA]
		[EA+] <i>pseg_addr</i> :[EA+] DSR:[EA+] Rd:[EA+]	*	*					ワード型転送命令 ERn←[EA] EA←EA+1
		[ERm] <i>pseg_addr</i> :[ERm] DSR:[ERm] Rd:[ERm]	*	*					ワード型転送命令 ERn←[ERm]
		<i>Disp16</i> [ERm] <i>pseg_addr</i> : <i>Disp16</i> [ERm] DSR: <i>Disp16</i> [ERm] Rd: <i>Disp16</i> [ERm]	*	*					ワード型転送命令 ERn← Disp16[ERm]
		<i>Disp0</i> [BP] <i>pseg_addr</i> : <i>Disp0</i> [BP] DSR: <i>Disp0</i> [BP] Rd: <i>Disp0</i> [BP]	*	*					ワード型転送命令 ERn← Disp0[BP]
		<i>Disp0</i> [FP] <i>pseg_addr</i> : <i>Disp0</i> [FP] DSR: <i>Disp0</i> [FP] Rd: <i>Disp0</i> [FP]	*	*					ワード型転送命令 ERn← Disp0[FP]
		<i>Dadr</i> <i>pseg_addr</i> : <i>Dadr</i> DSR: <i>Dadr</i> Rd: <i>Dadr</i>	*	*					ワード型転送命令 ERn← Dadr

(次頁につづく)

第3章 命令の詳細 インストラクションセット

ロード/ストア命令(続き)

ニーモニック	第1 オペランド	第2 オペランド	C	Z	S	OV	MIE	HC	機能
L	XRn	[EA]	*	*					ダブルワード型転送命令
		pseg_addr:[EA]	*	*					XRn←[EA]
		DSR:[EA]	*	*					
		Rd[EA]	*	*					
	[EA+]		*	*					ダブルワード型転送命令
		pseg_addr:[EA+]	*	*					XRn←[EA]
		DSR:[EA+]	*	*					EA ← EA+1
		Rd[EA+]	*	*					
QRn	[EA]		*	*					クロッドワード型転送命令
		pseg_addr:[EA]	*	*					QRn←[EA]
		DSR:[EA]	*	*					
		Rd[EA]	*	*					
	[EA+]		*	*					クロッドワード型転送命令
		pseg_addr:[EA+]	*	*					QRn←[EA]
		DSR:[EA+]	*	*					EA ← EA+1
		Rd[EA+]	*	*					

ニーモニック	第1 オペランド	第2 オペランド	C	Z	S	OV	MIE	HC	機能
ST	Rn	[EA]							バイト型転送命令
		pseg_addr:[EA]							[EA] ← Rn
		DSR:[EA]							
		Rd[EA]							
	[EA+]								バイト型転送命令
		pseg_addr:[EA+]							[EA] ← Rn
		DSR:[EA+]							EA ← EA+1
		Rd[EA+]							
	[ERn]								バイト型転送命令
		pseg_addr:[ERn]							[ERn]← Rn
		DSR:[ERn]							
		Rd[ERn]							
	Disp16[ERn]								バイト型転送命令
		Disp16[ERn]							Disp16[ERn]← Rn
		pseg_addr:Disp16[ERn]							
		DSR:Disp16[ERn]							
	Rd:Disp16[ERn]								バイト型転送命令
		Disp16[BP]							Disp16[BP]← Rn
		pseg_addr:Disp16[BP]							
		DSR:Disp16[BP]							
	Rd:Disp16[BP]								バイト型転送命令
		Disp16[FP]							Disp16[FP]← Rn
		pseg_addr:Disp16[FP]							
		DSR:Disp16[FP]							
	Rd:Disp16[FP]								バイト型転送命令
		Dadr							Dadr← Rn
		pseg_addr:Dadr							
		DSR:Dadr							
	Rd:Dadr								Rd:Dadr
		[EA]							ワード型転送命令
		pseg_addr:[EA]							[EA] ← ERn
		DSR:[EA]							
	Rd[EA]								Rd[EA]
		[EA+]							ワード型転送命令
		pseg_addr:[EA+]							[EA] ← ERn
		DSR:[EA+]							EA ← EA+1
	Rd[EA+]								Rd[EA+]

ロード/ストア命令(続き)

ニーモニック	第1 オペランド	第2 オペランド	C Z S OV MIE HC	機能
ST	ERn	[ERm] pseg_addr:[ERm] DSR:[ERm] Rd:[ERm]		ワード型転送命令 [ERm] ← ERn
		<u>Disp16[ERm]</u> pseg_addr: Disp16[ERm] DSR: Disp16[ERm] Rd: Disp16[ERm]		ワード型転送命令 Disp16[ERm] ← ERn
		<u>Disp6[BP]</u> pseg_addr: Disp6[BP] DSR: Disp6[BP] Rd: Disp6[BP]		ワード型転送命令 Disp6[BP] ← ERn
		<u>Disp6[FP]</u> pseg_addr: Disp6[FP] DSR: Disp6[FP] Rd: Disp6[FP]		ワード型転送命令 Disp6[FP] ← ERn
		<u>Dadr</u> pseg_addr: Dadr DSR: Dadr Rd: Dadr		ワード型転送命令 [Dadr] ← ERn
XRn	[EA] pseg_addr:[EA] DSR:[EA] Rd:[EA]			ダブルワード型転送命令 [EA] ← XRn
	[EA+] pseg_addr:[EA+] DSR:[EA+] Rd:[EA+]			ダブルワード型転送命令 [EA] ← XRn EA ← EA+1
QRn	[EA] pseg_addr:[EA] DSR:[EA] Rd:[EA]			クワッドワード型転送命令 [EA] ← QRn
	[EA+] pseg_addr:[EA+] DSR:[EA+] Rd:[EA+]			クワッドワード型転送命令 [EA] ← QRn EA ← EA+1

第3章 命令の詳細 インストラクションセット

コントロールレジスタアクセス命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
ADD	SP	#signed8		加算命令 •ELEVEL が 0 SP←SP+signed8
MOV	ECSR	Rm		転送命令 LCSR ← Rm •ELEVEL が 0 以外 ECSR[ELEVEL] ← Rm
ELR	ERm			転送命令 •ELEVEL が 0 LR ← ERm •ELEVEL が 0 以外 ELR[ELEVEL] ← ERm
EPSW	Rm			転送命令 •ELEVEL が 0 以外 EPSW[ELEVEL] ← Rm
ERn	ELR			転送命令 •ELEVEL が 0 ERn ← LR •ELEVEL が 0 以外 ERn ← ELR[ELEVEL]
	SP			転送命令 ERn←SP
PSW	Rm	* * * * *	転送命令	PSW←Rm
	#unsigned8	* * * * *	転送命令	PSW←unsigned8
				•ELEVEL が 0 Rn ← LCSR
Rn	ECSR			転送命令 •ELEVEL が 0 以外 Rn ← ECSR[ELEVEL]
	EPSW			転送命令 Rn ← EPSW[ELEVEL]
	PSW			転送命令 Rn←PSW
SP	ERm			転送命令 SP←ERm

PUSH/POP 命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
PUSH	ERn			汎用レジスタ退避命令 SP ← SP-n スタック領域 ← 汎用レジスタ
	Rn			
	QRn			
	XRn			
	register_list			コントロールレジスタ退避命令 SP ← SP-n スタック領域 ← レジスタ群
POP	ERn			汎用レジスタ復帰命令 汎用レジスタ ← スタック領域 SP←SP+n
	Rn			
	QRn			
	XRn			
	register_list	*	*	コントロールレジスタ復帰命令 レジスタ群 ← スタック領域 * 1 SP←SP+n

*1 *** register_list として PSW が指定されたときのみ PSW が影響を受ける

コプロセッサ転送

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
MOV	CR _n	R _m		バイト型転送命令 CR _n ← R _m
	CR _n	[EA]		バイト型転送命令 CR _n ← [EA]
		pseg_addr:[EA]		
		DSR:[EA]		
		Rd[EA]		
		[EA+]		バイト型転送命令 CR _n ← [EA+]
		pseg_addr:[EA+]		EA ← EA+1
		DSR:[EA+]		
		Rd[EA+]		
CER _n	[EA]			ワード型転送命令 CER _n ← [EA]
		pseg_addr:[EA]		
		DSR:[EA]		
		Rd[EA]		
		[EA+]		ワード型転送命令 CER _n ← [EA+]
		pseg_addr:[EA+]		EA ← EA+1
		DSR:[EA+]		
		Rd[EA+]		
CXR _n	[EA]			ダブルワード型転送命令 CXR _n ← [EA]
		pseg_addr:[EA]		
		DSR:[EA]		
		Rd[EA]		
		[EA+]		ダブルワード型転送命令 CXR _n ← [EA+]
		pseg_addr:[EA+]		EA ← EA+1
		DSR:[EA+]		
		Rd[EA+]		
CQR _n	[EA]			クワッドワード型連続転送命令 CQR _n ← [EA]
		pseg_addr:[EA]		
		DSR:[EA]		
		Rd[EA]		
		[EA+]		クワッドワード型連続転送命令 CQR _n ← [EA+]
		pseg_addr:[EA+]		EA ← EA+1
		DSR:[EA+]		
		Rd[EA+]		

(次頁に続く)

第3章 命令の詳細 インストラクションセット

コプロセッサ転送(前頁からのつづき)

ニーモニック	第1 オペランド	第2 オペランド	C Z S OV MIE HC	機能
MOV	Rn	CRm		バイト型転送命令 $Rn \leftarrow CRm$
	[EA]	CRm		バイト型転送命令 $[EA] \leftarrow CRm$
	pseg_addr:[EA]			
	DSR:[EA]			
	Rd[EA]			
	[EA+]	CRm		バイト型転送命令 $[EA] \leftarrow CRm$
	pseg_addr:[EA+]			
	DSR:[EA+]			
	Rd[EA+]			
	[EA]	CERm		ワード型転送命令 $[EA] \leftarrow CERm$
	pseg_addr:[EA]			
	DSR:[EA]			
	Rd[EA]			
	[EA+]	CERm		ワード型転送命令 $[EA] \leftarrow CERm$
	pseg_addr:[EA+]			
	DSR:[EA+]			
	Rd[EA+]			
	[EA]	CXRm		ダブルワード型転送命令 $[EA] \leftarrow CXRm$
	pseg_addr:[EA]			
	DSR:[EA]			
	Rd[EA]			
	[EA+]	CXRm		ダブルワード型転送命令 $[EA] \leftarrow CXRm$
	pseg_addr:[EA+]			
	DSR:[EA+]			
	Rd[EA+]			
	[EA]	CQRm		クロッドワード型連続転送 $[EA] \leftarrow CQRm$ 命令
	pseg_addr:[EA]			
	DSR:[EA]			
	Rd[EA]			
	[EA+]	CQRm		クロッドワード型連続転送 $[EA] \leftarrow CQRm$ 命令 $EA \leftarrow EA+1$
	pseg_addr:[EA+]			
	DSR:[EA+]			
	Rd[EA+]			

EA レジスタ転送命令

ニーモニック	第1 オペランド	第2 オペランド	C Z S OV MIE HC	機能
LEA	[ERn]			EAへの転送命令 $EA \leftarrow ERn$
	Disp16[ERm]			$EA \leftarrow Disp16 + ERm$
	Dadr			$EA \leftarrow Dadr$

ALU 命令

ニーモニック	第1 オペランド	第2 オペランド	C Z S OV MIE HC	機能
DAA	Rn		* * *	* バイト型 10進加算補正命令
DAS	Rn		* * *	* バイト型 10進減算補正命令
NEG	Rn		* * * *	* 符号反転命令 $Rn \leftarrow 0 - Rn$

ビットアクセス命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
SB	Rn.bit_offset		*	セットビット命令 $z \leftarrow \sim Rn.bit_offset$ $Rn.bit_offset \leftarrow 1$
	Dbitadr		*	
	pseg_addr: Dbitadr		*	$z \leftarrow \sim [Dbitadr]$
	DSR: Dbitadr		*	$[Dbitadr] \leftarrow 1$
	Rd.Dbitadr		*	
RB	Rn.bit_offset		*	リセットビット命令 $z \leftarrow \sim Rn.bit_offset$ $Rn.bit_offset \leftarrow 0$
	Dbitadr		*	
	pseg_addr: Dbitadr		*	$z \leftarrow \sim [Dbitadr]$
	DSR: Dbitadr		*	$[Dbitadr] \leftarrow 0$
	Rd.Dbitadr		*	
TB	Rn.bit_offset		*	テストビット命令 $z \leftarrow \sim Rn.bit_offset$
	Dbitadr		*	
	pseg_addr: Dbitadr		*	
	DSR: Dbitadr		*	$z \leftarrow \sim [Dbitadr]$
	Rd.Dbitadr		*	

PSWアクセス命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
EI			*	割込み許可命令 $MIE \leftarrow 1$
DI			*	割込みマスク命令 $MIE \leftarrow 0$
SC			*	セットキャリ命令 $C \leftarrow 1$
RC			*	リセットキャリ命令 $C \leftarrow 0$
CPLC			*	キャリ反転命令 $C \leftarrow \sim C$

条件相対分岐命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
Bcond	Radr			条件分岐命令 if cond? Radr: PC+2
BC	cond	Radr		

符号拡張命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
EXTBW	ERn		*	符号拡張命令 $ERn \leftarrow (\text{符号拡張})Rn$

第3章 命令の詳細 インストラクションセット

ソフトウェア割込み命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
SWI	# <i>snum</i>		*	ソフトウェア割込み命令 address←(<i>snum</i> <<1), PC←Vector-table(address)
BRK				ブレーク命令 •ELEVEL が2以上 System reset •ELEVEL が1以下 PC←(Vector-table 0004H)

分岐命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
B	<i>Cadr</i>			分岐命令 CSR ← <i>Cadr</i> [19:16] PC ← <i>Cadr</i> [15:0]
		ER <i>n</i>		PC ← ER <i>n</i>
BL	<i>Cadr</i>			分岐命令 LR ← 次の命令の先頭アドレス LCSR ← CSR CSR ← <i>Cadr</i> [19:16] PC ← <i>Cadr</i> [15:0]
		ER <i>n</i>		LR ← 次の命令の先頭アドレス LCSR ← CSR PC ← ER <i>n</i>

乗除算命令

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
MUL	ER <i>n</i>	R <i>m</i>	*	乗算命令 ER <i>n</i> ← R <i>n</i> * R <i>m</i>
DIV	ER <i>n</i>	R <i>m</i>	* *	除算命令 ER <i>n</i> ← ER <i>n</i> / R <i>m</i> , R <i>m</i> ←ER <i>n</i> mod R <i>m</i>

その他

ニーモニック	第1オペランド	第2オペランド	C Z S OV MIE HC	機能
INC	[EA]		* * *	* メモリ加算 [EA] ← [EA] + 1
	<i>pseg_addr</i> [EA]		* * *	*
	DSR:[EA]		* * *	*
	Rd:[EA]		* * *	*
DEC	[EA]		* * *	* メモリ減算 [EA] ← [EA] - 1
	<i>pseg_addr</i> [EA]		* * *	*
	DSR:[EA]		* * *	*
	Rd:[EA]		* * *	*
RT				サブルーチンからの復帰命令 CSR ← LCSR PC ← LR
RTI			* * * * *	* 割込みからの復帰命令 CSR ← ECSR[ELEVEL] PC ← ELR [ELEVEL] PSW ← EPSW[ELEVEL]
NOP				

3.3命令実行時間について

本項では、nX-U16/100 コアの命令実行時間を示します。

クロック周波数への依存を避けるため、命令実行時間をクロック・サイクルで示します。

また、メモリのリードライトサイクルタイムはともに1サイクルであると仮定しています。それよりも長いメモリサイクルがあると待ち状態が発生するため、それを命令実行時間の合計に加える必要があります。

命令はそのマシンサイクルを単位として最低3マシンサイクルで実行されますが、nX-U16/100 コアは命令フェッチ、命令デコード、命令実行および演算の書き込みの3つのステートを並列処理するため、最も効率的に命令処理を行った場合、本来のマシンサイクルよりも早く命令実行を終了することができます。

このように命令がもっとも効率的に処理された場合の実行サイクル値を最小実行サイクルと呼びます。

命令実行中は、並列処理であるがゆえに、各ステート間で資源の競合が発生する場合があります。この場合 nX-U16/100 コアは、最低1マシンサイクルのウェイトサイクルを命令実行パイプラインに挿入し、一方のステートの処理開始を遅らせることで資源の競合を防ぎます。

以下にウェイトサイクルが挿入される条件をまとめます。

条件1. RomWindows 領域のアクセスを行うと、 $n \times m$ (n : アクセス回数 m :1 アクセスあたりのウェイト数)

のウェイトサイクルが挿入されます。

条件2. [EA+] のアドレッシングを持つ命令の直後に配置すると、CPU 内部バスが競合する命令があります。この場合、1サイクルのウェイトサイクルが付加されます。

条件3. [EA+] のアドレッシングを持つ命令の直後に割込みが発生すると、CPU 内部バスが競合します。この場合、1サイクルのウェイトサイクルが付加された後、割込み処理が開始されます。割込みについては、"1.3.7 割込み動作について"を参照して下さい。

すなわち、ある命令の全体の実行サイクルは、

(最小実行サイクル数+バスの競合によるウェイト数+データアクセス時のウェイト数)
となります。

次ページ以降に、nX-U16/100 コアの全命令と、それぞれが上記状態にある場合に付加されるサイクル数を記します。表中空白の欄は、その命令実行中には資源が競合する状態が存在しないか、メモリアクセスを行わないことを意味します。

第3章 命令の詳細 インストラクションセット

ニーモニック	第1オペランド	第2オペランド	最小実行サイクル	RomWindowアクセス	[EA+]アドレッシングの影響
ADD	ER _n	ER _m	1		
		#imm7	1		
ADD	R _n	R _m	1		
		#imm8	1		
	SP	#signed8	1		
ADDC	R _n	R _m	1		
		#imm8	1		
AND	R _n	R _m	1		
		#imm8	1		
B	Cadr		2		1
	ER _n		2		1
Bcond	Radr		1 / 3 _{(*)1}		1
BL	Cadr		2		1
	ER _n		2		1
BRK			7		1
CMP	ER _n	ER _m	1		
	R _n	R _m	1		
		#imm8	1		
CMPC	R _n	R _m	1		
		#imm8	1		
CPLC			1		
DAA	R _n		1		
DAS	R _n		1		
DEC	[EA]		2		1
	pseg_addr:[EA]				
	DSR:[EA]		3		
	Rd:[EA]				
DI			3		
DIV	ER _n	R _m	17		
EI			1		
EXTBW	ER _n		1		
INC	[EA]		2		1
	pseg_addr:[EA]				
	DSR:[EA]		3		
	Rd:[EA]				

(*)1 … (分岐条件非成立時 / 条件成立時)

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+]アドレッシング の影響
L	ER _n	[EA]	1	1	
		pseg_addr:[EA]			
		DSR:[EA]	2	1	
		Rd[EA]			
		[EA+]	1	1	
		pseg_addr:[EA+]			
		DSR:[EA+]	2	1	
		Rd[EA+]			
		[ER _m]	1	1	1
		pseg_addr:[ER _m]			
		DSR:[ER _m]	2	1	
		Rd[ER _m]			
		Disp16[ER _m]	2	1	1
		pseg_addr: Disp16[ER _m]			
		DSR: Disp16[ER _m]	3	1	
		Rd. Disp16[ER _m]			
		Disp6[BP]	2	1	1
		pseg_addr: Disp6[BP]			
		DSR: Disp6[BP]	3	1	
		Rd. Disp6[BP]			
		Disp6[FP]	2	1	1
		pseg_addr: Disp6[FP]			
		DSR: Disp6[FP]	3	1	
		Rd. Disp6[FP]			
		Dadr	2	1	1
		pseg_addr: Dadr			
		DSR: Dadr	3	1	
		Rd. Dadr			
QR _n		[EA]	4	4	
		pseg_addr:[EA]			
		DSR:[EA]	5	4	
		Rd[EA]			
		[EA+]	4	4	
		pseg_addr:[EA+]			
		DSR:[EA+]	5	4	
		Rd[EA+]			

第3章 命令の詳細 インストラクションセット

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+]アドレッシン グの影響
L	Rn	[EA]	1	1	
		<i>pseg_addr.[EA]</i>			
		DSR:[EA]	2	1	
		Rd[EA]			
		[EA+]	1	1	
		<i>pseg_addr.[EA+]</i>			
		DSR:[EA+]	2	1	
		Rd[EA+]			
		[ERm]	1	1	1
		<i>pseg_addr.[ERm]</i>			
		DSR:[ERm]	2	1	
		Rd[ERm]			
		<i>Disp16[ERm]</i>	2	1	1
		<i>pseg_addr. Disp16[ERm]</i>			
		DSR: <i>Disp16[ERm]</i>	3	1	
		Rd. <i>Disp16[ERm]</i>			
		<i>Disp6[BP]</i>	2	1	1
		<i>pseg_addr. Disp6[BP]</i>			
		DSR: <i>Disp6[BP]</i>	3	1	
		Rd. <i>Disp6[BP]</i>			
		<i>Disp6[FP]</i>	2	1	1
		<i>pseg_addr. Disp6[FP]</i>			
		DSR: <i>Disp6[FP]</i>	3	1	
		Rd. <i>Disp6[FP]</i>			
		<i>Dadr</i>	2	1	1
		<i>pseg_addr. Dadr</i>			
		DSR: <i>Dadr</i>	3	1	
		Rd. <i>Dadr</i>			
XRn		[EA]	2	2	
		<i>pseg_addr.[EA]</i>			
		DSR:[EA]	3	2	
		Rd[EA]			
		[EA+]	2	2	
		<i>pseg_addr.[EA+]</i>			
		DSR:[EA+]	3	2	
		Rd[EA+]			

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+]アドレッシン グの影響
LEA	[ERm]		1		
	Disp16[ERm]		2		
	Dadr		2		
MOV	CERn	[EA]	1	1	1
		pseg_addr:[EA]			
		DSR:[EA]	2	1	
		Rd[EA]			
		[EA+]	1	1	1
		pseg_addr:[EA+]			
		DSR:[EA+]	2	1	
		Rd[EA+]			
		[EA]	4	4	1
CQRn	CQRn	pseg_addr:[EA]			
		DSR:[EA]	5	4	
		Rd[EA]			
		[EA+]	4	4	1
		pseg_addr:[EA+]			
		DSR:[EA+]	5	4	
		Rd[EA+]			
		[EA]	1	1	1
		pseg_addr:[EA]			
CRn	CRn	DSR:[EA]	2	1	
		Rd[EA]			
		[EA+]	1	1	1
		pseg_addr:[EA+]			
		DSR:[EA+]	2	1	
		Rd[EA+]			
		Rm	1		
		[EA]	2	2	1
		pseg_addr:[EA]			
CXRn	CXRn	DSR:[EA]	3	2	
		Rd[EA]			
		[EA+]	2	2	1
		pseg_addr:[EA+]			
		DSR:[EA+]	3	2	
		Rd[EA+]			
		Rm	1		
		[ERm]	1		
		Rm	1		
ERn	ERn	ELR	1		
		ERm	1		
		#imm7	1		
		SP	1		

第3章 命令の詳細 インストラクションセット

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+]アドレッシン グの影響
MOV	[EA]	CER m	1	1	1
	<i>pseg_addr:[EA]</i>				
	DSR:[EA]	CER m	2	1	
	Rd[EA]				
	[EA+]	CER m	1	1	1
	<i>pseg_addr:[EA+]</i>				
	DSR:[EA+]	CER m	2	1	
	Rd[EA+]				
	[EA]	CQR m	4	4	1
	<i>pseg_addr:[EA]</i>				
	DSR:[EA]	CQR m	5	4	
	Rd[EA]				
	[EA+]	CQR m	4	4	1
	<i>pseg_addr:[EA+]</i>				
	DSR:[EA+]	CQR m	5	4	
	Rd[EA+]				
	[EA]	CR m	1	1	1
	<i>pseg_addr:[EA]</i>				
	DSR:[EA]	CR m	2	1	
	Rd[EA]				
	[EA+]	CR m	1	1	1
	<i>pseg_addr:[EA+]</i>				
	DSR:[EA+]	CR m	2	1	
	Rd[EA+]				
	[EA]	CXR m	2	2	1
	<i>pseg_addr:[EA]</i>				
	DSR:[EA]	CXR m	3	2	
	Rd[EA]				
	[EA+]	CXR m	2	2	1
	<i>pseg_addr:[EA+]</i>				
	DSR:[EA+]	CXR m	3	2	
	Rd[EA+]				
	PSW	R m	1		
		#unsigned8	1		
	R n	CR m	1		
		ECSR	1		
		EPSW	1		
		PSW	1		
		R m	1		
		#imm8	1		
	SP	ER m	1		1

第3章 命令の詳細 インストラクションセット

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+]アドレッシ ングの影響
MUL	ERn	Rm	9		
NEG	Rn		1		
NOP			1		
OR	Rn	Rm	1		
		#imm8	1		
POP	EA		2		1
	EA,LR		3 / 4 <small>(*1)</small>		1
	EA,PC		5 / 6 <small>(*1)</small>		1
	EA,PC,LR		6 / 8 <small>(*1)</small>		1
	EA,PC,PSW		6 / 7 <small>(*1)</small>		1
	EA,PC,PSW,LR		7 / 9 <small>(*1)</small>		1
	EA,PSW		3		1
	EA,PSW,LR		4 / 5 <small>(*1)</small>		1
	LR		1 / 2 <small>(*1)</small>		1
	LR,PSW		2 / 3 <small>(*1)</small>		1
	PC		3 / 4 <small>(*1)</small>		1
	PC,LR		4 / 6 <small>(*1)</small>		1
	PC,PSW		4 / 5 <small>(*1)</small>		1
	PC,PSW,LR		5 / 7 <small>(*1)</small>		1
	PSW		1		1
	ERn		1		1
	QRn		4		1
	Rn		1		1
	XRn		2		1

(*1) ... (メモリモデルが SMALL の時のサイクル/ LARGE の時のサイクル)

第3章 命令の詳細 インストラクションセット

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+] アドレッシ ングの影響
PUSH	EA		1		1
	ELR		1 / 2 (*1)		1
	EA,ELR		2 / 3 (*1)		1
	EPSW		1		1
	EPSW,EA		2		1
	EPSW,ELR		2 / 3 (*1)		1
	EPSW,ELR,EA		3 / 4 (*1)		1
	LR		1 / 2 (*1)		1
	LR,EA		2 / 3 (*1)		1
	LR,ELR		2 / 4 (*1)		1
	LR,EA,ELR		3 / 5 (*1)		1
	LR,EPSW		2 / 3 (*1)		1
	LR,EPSW,EA		3 / 4 (*1)		1
	LR,EPSW,ELR		3 / 5 (*1)		1
	LR,ELR,EPSW,EA		4 / 6 (*1)		1
	ERn		1		1
	QRn		4		1
	Rn		1		1
	XRn		2		1
RB	<i>Dbitadr</i>		2		1
	<i>pseg_addr. Dbitadr</i>				
	DSR: <i>Dbitadr</i>		3		
	<i>Rd. Dbitadr</i>				
	<i>Rn. bit_offset</i>		1		
RC			1		
RT			2		1
RTI			2		1
SB	<i>Dbitadr</i>		2		1
	<i>pseg_addr. Dbitadr</i>				
	DSR: <i>Dbitadr</i>		3		
	<i>Rd. Dbitadr</i>				
	<i>Rn. bit_offset</i>		1		
SC			1		

(*1) … (メモリモデルが SMALL の時のサイクル/ LARGE の時のサイクル)

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+]アドレッシ ングの影響
SLL	Rn	Rm	1		1
		#width	1		1
SLLC	Rn	Rm	1		1
		#width	1		1
SRA	Rn	Rm	1		1
		#width	1		1
SRL	Rn	Rm	1		1
		#width	1		1
SRLC	Rn	Rm	1		1
		#width	1		1
ST	ERn	[EA]	1		
		pseg_addr:[EA]			
		DSR:[EA]	2		
		Rd[EA]			
		[EA+]	1		
		pseg_addr:[EA+]			
		DSR:[EA+]	2		
		Rd[EA+]			
		[ERm]	1		1
		pseg_addr:[ERm]			
		DSR:[ERm]	2		
		Rd[ERm]			
		Disp16[ERm]	2		1
		pseg_addr: Disp16[ERm]			
		DSR: Disp16[ERm]	3		
		Rd: Disp16[ERm]			
		Disp6[BP]	2		1
		pseg_addr: Disp6[BP]			
		DSR: Disp6[BP]	3		
		Rd: Disp6[BP]			
		Disp6[FP]	2		1
		pseg_addr: Disp6[FP]			
		DSR: Disp6[FP]	3		
		Rd: Disp6[FP]			
		Dadr	2		1
		pseg_addr: Dadr			
		DSR: Dadr	3		
		Rd: Dadr			

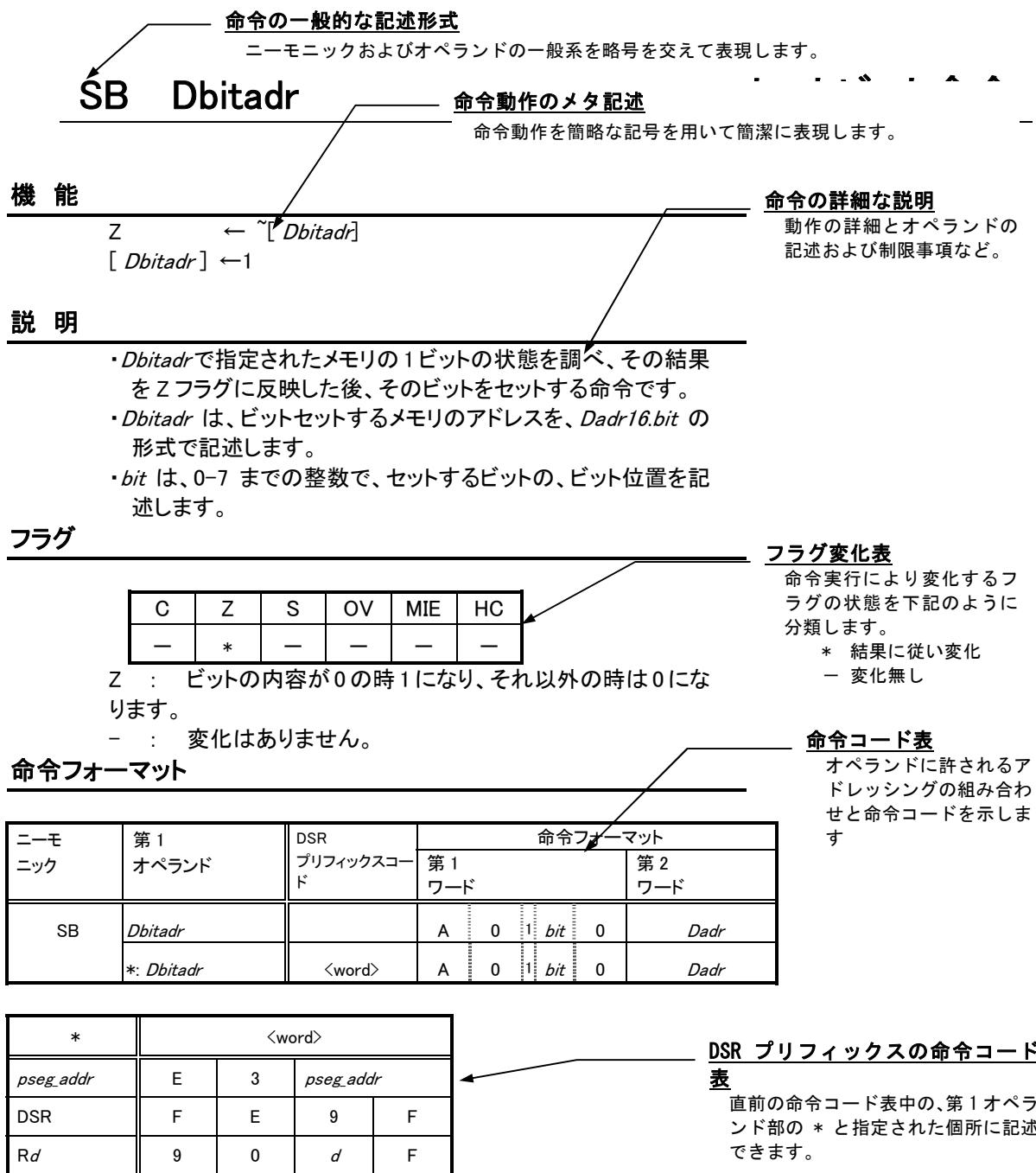
第3章 命令の詳細 インストラクションセット

第3章 命令の詳細 インストラクションセット

ニーモニック	第1 オペランド	第2 オペランド	最小実行 サイクル	RomWindow アクセス	[EA+]アドレッシン グの影響
SUB	Rn	Rm	1		
SUBC	Rn	Rm	1		
SWI	#snum		3		1
TB	<i>Dbitadr</i>		2	1	1
	<i>pseg_addr. Dbitadr</i>				
	DSR: <i>Dbitadr</i>		3		1
	<i>Rd. Dbitadr</i>				
	<i>Rn. bit_offset</i>		1		
XOR	Rn	Rm	1		
		#imm8	1		

3.4 各命令の説明

次ページより各命令の詳細について説明します。命令はアルファベット順に並べられており、ひとつの命令を1ページあるいは2ページで説明しています。命令フォーマットに記載されていないコードを、命令として実行した場合のCPU動作は保証できませんのでプログラマはご注意ください。以下に命令の説明の見方の例を示します。



ADD ER_n, ER_m

加算命令

機能

ER_n ← ER_n + ER_m

説明

• ER_nレジスタの内容に、ER_mの内容を加算し、結果を ER_nレジスタに格納します。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	—	*

- C : 演算の結果、ビット 15 よりキャリが発生したときに 1 になり、それ以外の時は 0 になります。
Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
S : 実行の結果の最上位ビットがセットされます。
OV : オーバフローが生じた場合 1 になり、それ以外の時は 0 になります。
HC : ビット 11 にキャリあるいはボローが生じた時 1 になります。それ以外の時は 0 になります。
— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット		
			第1ワード	第2ワード	
ADD	ER _n	ER _m	F n m	6	

ADD ER_n, #imm7

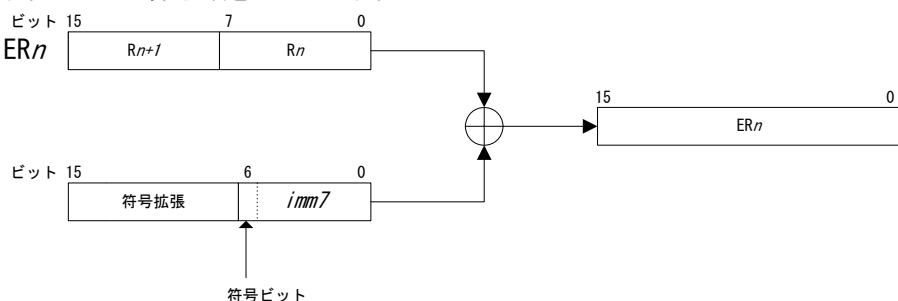
加算命令

機能

$$ER_n \leftarrow ER_n + (\text{signed})imm7$$

説明

- ER_nレジスタの内容に、imm7を符号拡張して加算し、結果を ER_nレジスタに格納します。以下に計算手順を図示します。



フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	—	*

- C : 演算の結果、ビット 15 よりキャリが発生したときに 1 になり、それ以外の時は 0 になります。
 Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
 S : 実行の結果の最上位ビットがセットされます。
 OV : オーバフローが生じた場合 1 になり、それ以外の時は 0 になります。
 HC : ビット 11 にキャリあるいはボローが生じた時 1 になります。それ以外の時は 0 になります。
 — : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード		第2ワード	
ADD	ER _n	#imm7	E	n	1	imm7

ADD R_n, obj

加算命令

機能

$R_n \leftarrow R_n + obj$

説明

- R_nレジスタの内容に、objの内容を加算し、結果を R_nレジスタに格納します。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	—	*

- C : 演算の結果、ビット 7 よりキャリが発生したときに 1 になり、それ以外の時は 0 になります。
 Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
 S : 実行の結果の最上位ビットがセットされます。
 OV : オーバフローがセットされた時 1 になり、それ以外の時は 0 になります。
 HC : ビット 3 にキャリあるいはボローが生じた時 1 になります。それ以外の時は 0 になります。
 — : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
ADD	R _n	R _m	8	n	m	1
		#imm8	1	n	imm8	

ADD SP , #*signed8*

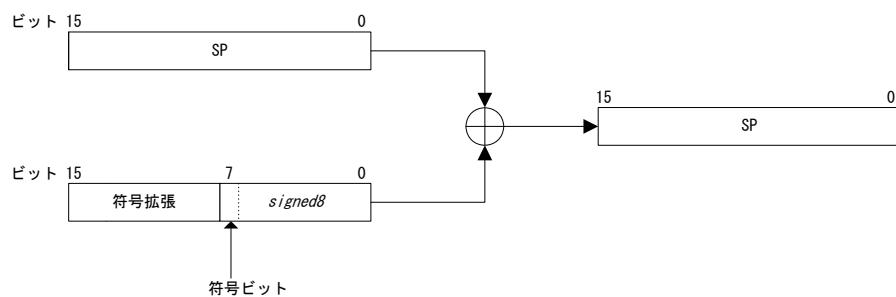
加算命令

機能

$SP \leftarrow SP + signed8$

説明

- ・スタックポインタの内容に、*signed8* で示された即値を加算し、結果を SP に格納します。
- ・*signed8* はビット 7 を符号とする、-128 ~ +127 で示される範囲の整数値となります。
計算手順を以下に図示します。



フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット	
			第1 ワード	第2 ワード
ADD	SP	# <i>signed8</i>	E 1	<i>signed8</i>

ADDC R_n, obj

キャリ付き加算命令

機能

$R_n \leftarrow R_n + obj + C$

説明

- R_n レジスタの内容に、obj の内容と C フラグの内容を加算し、結果を R_n レジスタに格納します。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	—	*

- C : 演算の結果、ビット 7 よりキャリが発生した時に 1 になり、それ以外の時は 0 になります。
 Z : 実行前の Z の内容が 0 の時は、実行結果に関わらず 0 になります。
 実行前の Z の内容が 1 の時は、実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
 S : 実行の結果の最上位ビットがセットされます。
 OV : オーバフローがセットされた時 1 になり、それ以外の時に 0 になります。
 HC : ビット 3 にキャリ、あるいはボローが生じた時 1 になります。それ以外の時は 0 になります。
 — : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード	第2ワード		
ADDC	R _n	R _m	8 n m 6			
		#imm8	6 n imm8			

AND R_n, obj

論理積命令

機能

R_n ← R_n & obj

説明

・R_nレジスタの内容と obj の内容の論理積をとり、結果を R_nレジスタに格納します。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。

S : 実行の結果の最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
AND	R _n	R _m	8	n	m	2
		#imm8	2	n	imm8	

B *Cadr*

ダイレクトジャンプ命令

機能

CSR $\leftarrow Cadr[19:16]$
PC $\leftarrow Cadr[15:0]$

説明

- ・プログラムメモリ空間の任意のアドレスにジャンプします。*Cadr* には分岐先アドレスを指定します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	命令フォーマット								
		第1 ワード			第2 ワード					
B	<i>Cadr</i>	F	⋮	<i>Cadr[19:16]</i>	⋮	0	⋮	0	⋮	<i>Cadr[15:0]</i>

第3章 命令の詳細 インストラクションセット

B ER_n

インダイレクトジャンプ命令

機能

PC ← ER_n

説明

- ER_nレジスタの内容を飛び先アドレスとする、同一セグメント内の任意のアドレスにジャンプします。
- ER_nは、ワード長のレジスタの内容です。ER_nレジスタにはこの命令に先立って飛び先アドレスを設定しておく必要があります。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	命令フォーマット				
		第1 ワード				
B	ER _n	F	0	<i>n</i>		2

Bcond Radr

条件分岐命令

BC cond, Radr

機能

if(*cond*= 真)then PC \leftarrow *Radr*
ただし、(NextPC -128word) \leq *Radr* \leq (NextPC +127word)
NextPC：次の命令の先頭アドレス

説明

- ・*cond* で指定される条件が真であれば、*Radr* で指定されるアドレスへジャンプする命令です。
- ・条件は、PSW に残されているフラグの状態を示します。従って、この命令に先立って、比較命令などの PSW に結果を残す命令が行われることを前提とし、この命令でその結果の評価を行います。
- ・*cond* をオペランドとして記述する方法と、二モニック文字列の一部として記述する方法があります。

例)

```
CMP R0,#12H
BEQ LABEL ; condを、二モニック文字列の一部として記述
CMP R0,#56H
BC NC,LABEL ; condをオペランドとして記述
:
:
```

LABEL:

フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

第3章 命令の詳細 インストラクションセット

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット		
Bcond	Radr		C	Condition	(Radr -NextPC)>>1
BC	cond	Radr			

Condition

命令記述			意味	フラグ条件
Bcond	BC cond	Condition		
BGE	BC GE	0000	符号無し \geq	C=0
BNC	BC NC			
BLT	BC LT	0001	符号無し $<$	C=1
BCY	BC CY			
BGT	BC GT	0010	符号無し $>$	(C=0)&&(Z=0)
BLE	BC LE	0011	符号無し \leq	(Z=1) ((C=1))
BGES	BC GES	0100	符号あり \geq	(OV^S)=0
BLTS	BC LTS	0101	符号あり $<$	(OV^S)=1
BGTS	BC GTS	0110	符号あり $>$	((OV^S) Z) = 0
BLES	BC LES	0111	符号あり \leq	((OV^S) Z) = 1
BNE	BC NE	1000	\neq	Z=0
BNZ	BC NZ			
BEQ	BC EQ	1001	=	Z=1
BZ	BC ZF			
BNV	BC NV	1010	オーバフロー無し	OV=0
BOV	BC OV	1011	オーバフローあり	OV=1
BPS	BC PS	1100	正	S=0
BNS	BC NS	1101	負	S=1
BAL	BC AL	1110	無条件	

BL *Cadr*

ブランチアンドリンク命令

機能

LR	← 次の命令の先頭アドレス
LCSR	← CSR
CSR	← <i>Cadr</i> [19:16]
PC	← <i>Cadr</i> [15:0]

説明

- サブルーチン用リンクレジスタ(LR)に次の PC の値を、サブルーチン用 CSR 退避レジスタ(LCSR)に現在の CSR の値をストアした後、オペランド内で指定された任意のセグメントの任意のアドレスにジャンプします。*Cadr*には飛び先アドレスを記述します。
- この命令はサブルーチンの読み出しを行うために使われます。そのサブルーチンから復帰する場合は RT 命令を使用します。
- サブルーチンがネストする場合は、上位サブルーチンの先頭に PUSH 命令を配置し、現在の LR および LCSR の内容をメモリに退避する必要があります。この場合、サブルーチンから復帰する場合には、POP 命令を使用します。
- 割込みルーチン内で本命令を使用する場合は、割込みの先頭番地で PUSH 命令を配置し、現在の LR、および LCSR の内容を、スタックメモリに退避する必要があります。この場合、サブルーチンから復帰する場合には、POP 命令を使用します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

		命令フォーマット					
二一モニック	第1オペランド	第1ワード			第2ワード		
BL	<i>Cadr</i>	F	<i>Cadr</i> [19:16]	0	1		<i>Cadr</i> [15:0]

BL ER_n

ブランチアンドリンク
命令

機能

PC	← ER _n
LR	← 次の命令の先頭アドレス
LCSR	← CSR

説明

- サブルーチン用リンクレジスタ(LR)に次の PC の値をストアし、ER_n レジスタの内容を飛び先とする同一セグメント内の任意のアドレスにジャンプする命令です。
- この命令はサブルーチンの読み出しを行うために使われます。そのサブルーチンから復帰する場合は RT 命令を使用します。
- サブルーチンがネストする場合は、上位サブルーチンの先頭に PUSH 命令を配置し、現在の LR および LCSR の内容をメモリに退避する必要があります。この場合、サブルーチンから復帰する場合には、POP 命令を使用します。
- 割込みルーチン内で本命令を使用する場合は、割込みの先頭番地で PUSH 命令を配置し、現在の LR、および LCSR の内容を、STACKメモリに退避する必要があります。この場合、サブルーチンから復帰する場合には、POP 命令を使用します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

		命令フォーマット						
ニーモ	第1	第1ワード						
ニック	オペランド	F	...	0	...	n	...	3
BL	ER _n							

BRK

ブレーク命令
(ソフトウェアリセット)

機能

- ・ ELEVEL の値が 2 以上の場合
 システムリセット
- ・ ELEVEL の値が 1 以下の場合
 ELR2 ← 次の命令の先頭アドレス
 ECSR2 ← CSR
 EPSW2 ← PSW
 ELEVEL ← 2
 PC ← (Vector table 0004H)

説明

- ・ソフトウェアでシステムをリセットする命令です。
- ・ELEVEL が 2 以上の状態でこの命令が実行された場合、CPU は以下の手順でシステムリセット処理を行います。
 - ① CPU 内部のレジスタを全て初期化。
 - ② コード空間上の 0 番地から始まるワードデータを SP に転送
 - ③ コード空間上の 2 番地から始まるワードデータを PC に転送。
- ・ELEVEL が 1 以下の状態でこの命令が実行された場合は、NMI 割込み相当の動作を行った後、コード空間上の割込みベクターテーブルの 4 番地からのワードデータを PC に転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第 1 オペランド	命令フォーマット					
		第 1 ワード					
BRK		F		F		F	

CMP ER n , ER m

比較命令

機能

ER n - ER m

説明

- ER n レジスタの内容と、ER m の内容を比較します。
- 実際には ER n の内容から ER m の内容を減算し、その結果で各フラグを設定します。この結果は条件分岐などによって参照できます。
- ER n の内容は変化しません。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	-	*

- C : 演算の結果、ビット 15 にボローが発生したとき 1 になり、それ以外の時は 0 になります。
Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
S : 実行の結果の最上位ビットがセットされます。
OV : オーバフローがセットされた時 1 になり、それ以外の時は 0 になります。
HC : ビット 11 にキャリヤーアリ或者是ボローが生じた時 1 になります。それ以外の時は 0 になります。
- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード	第2 ワード		
CMP	ER n	ER m	F	n	m	7

CMP R_n, obj

比較命令

機能

R_n - obj

説明

- R_nレジスタの内容と、objの内容を比較します。
- 実際には R_n の内容から obj の内容を減算し、その結果で各フラグを設定します。この結果は条件分岐などによって参照できます。
- R_n の内容は変化しません。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	-	*

- C : 演算の結果、ビット 7 にボローが発生したとき 1 になり、それ以外の時は 0 になります。
 Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
 S : 実行の結果の最上位ビットがセットされます。
 OV : オーバフローがセットされた時 1 になり、それ以外の時は 0 になります。
 HC : ビット 3 にキャリヤあるいはボローが発生した時 1 になります。それ以外の時は 0 になります。
 - : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード	第2 ワード		
CMP	R _n	R _m	8	n	m	7
		#imm8	7	n	imm8	

CMPC R_n, obj

キャリ付き比較命令

機能

R_n - obj ← C

説明

- ・R_nレジスタの内容と、objの内容をキャリ付きで比較します。
- ・実際には R_n の内容から (obj + キャリ) の内容を減算し、その結果で各フラグを設定します。この結果は条件分岐などによって参照できます。
- ・R_n の内容は変化しません。
- ・この命令を CMP 命令の後に続けて実行することで、多バイトデータの比較が可能になります。

例) CMP R0, R4
 CMPC R1, R5 (ER0 と ER4 の比較完了)

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	-	*

- C : 演算の結果、ビット 7 にボローが発生したとき 1 になり、それ以外の時は 0 になります。
Z : 実行前の Z の内容が 0 の時は、実行結果に関わらず 0 になります。
実行前の Z の内容が 1 の時は、実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
S : 実行の結果の最上位ビットがセットされます。
OV : オーバフローがセットされた時 1 になり、それ以外の時は 0 になります。
HC : ビット 3 にキャリあるいはボローが生じた時 1 になります。それ以外の時は 0 になります。
- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット				
			第1 ワード			第2 ワード	
CMPC	R _n	R _m	8	n	m	5	
		#imm8	5	n	imm8		

CPLC

キャリ反転命令

機能

$C \leftarrow \sim C$

説明

・C フラグの内容を反転します。

フラグ

C	Z	S	OV	MIE	HC
*	-	-	-	-	-

C : 反転した内容がセットされます。

- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード	第2 ワード		
CPLC			F	E	C	F

DAA R_n

バイト型 10進加算補正命令

機能

$R_n \leftarrow (10\text{進補正})R_n$

説明

- R_n の内容を 2桁のBCD(2進化10進)に変換します。実際には直前の R_n 、C、および HC の値を参照し、下表に従って R_n の内容を補正します。 x はドントケアであることを示します。

C	$R_n[7:4]$ の値	HC	$R_n[3:0]$ の値	加算値	補正後の C
0	0-9	0	0-9	00	0
0	0-8	0	A-F	06	0
0	0-9	1	x	06	0
0	A-F	0	0-9	60	1
0	9-F	0	A-F	66	1
0	A-F	1	x	66	1
1	x	0	0-9	60	1
1	x	0	A-F	66	1
1	x	1	x	66	1

- 実行に先立って ADD R_{n,obj}により 2進加算命令が実行され、その結果が R_nと PSW に残っている必要があります。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	-	-	*

- C : 命令実行前のキャリが1の場合は1になります。
そうでない場合は、10進補正の結果、100の位への桁上げがあれば1になります、なければ0になります。
- Z : 実行結果が0の時1になり、それ以外の時は0になります。
- S : 実行の結果の最上位ビットがセットされます。
- HC : ビット3にキャリあるいはボローが生じた時1になり、それ以外の時は0になります。
- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
DAA	R _n		8	n	1	F

DAS R_n

バイト型 10進減算補正命令

機能

$R_n \leftarrow (10\text{進補正})R_n$

説明

- R_n の内容を 2 桁の BCD(2 進化 10 進)に変換します。実際には直前の R_n 、C、および HC の値を参照し、下表に従って R_n の内容を補正します。x はドントケアであることを示します。

C	$R_n[7:4]$ の値	HC	$R_n[3:0]$ の値	減算値
0	0-9	0	0-9	00
0	0-9	0	A-F	06
0	0-9	1	x	06
0	A-F	0	0-9	60
0	A-F	1	x	66
0	A-F	0	A-F	66
1	x	0	0-9	60
1	x	1	x	66
1	x	0	A-F	66

- 実行に先立って SUB R_{n,obj} により 2 進減算命令が実行され、その結果が R_n と PSW に残っている必要があります。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	-	-	*

- C : 命令実行前のキャリガ1の場合は1になります。
そうでない場合、10進減算補正の結果 100 の位へのボローがあれば 1 になり、なければ 0 なります。
- Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 なります。
- S : 実行の結果の最上位ビットがセットされます。
- HC : ビット 3 にキャリあるいはボローが生じた時 1 になります。それ以外の時は 0 なります。
- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
DAS	R _n		8	n	3	F

DEC [EA]

デクリメント命令
(EA間接)

機能

[EA] ← [EA] -1

説明

- EAレジスタが指すメモリの内容を-1します。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	*	—	*

Z : 実行結果が0の時1になり、それ以外の時は0になります。

S : 実行の結果の最上位ビットがセットされます。

OV : オーバフローがセットされた時1になり、それ以外の時は0になります。

HC : ビット3にキャリヤーアリ或者是ボローが生じた時1になります。それ以外の時は0になります。

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR プリフィックスコード	命令フォーマット			
				第1ワード		第2ワード	
DEC	[EA]			F	E	3	F
	*:[EA]		<word>	F	E	3	F

*	<word>		
pseg_addr	E	3	pseg_addr
DSR	F	E	9 F
Rd	9	0	d F

DI

割込みマスク命令

機能

MIE ← 0

説明

・MIE を0にして、マスカブル割込みを禁止する命令です。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	*	—

MIE : 0になります。

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード	第2 ワード		
DI			E	B	F	7

DIV ER_n, R_m

除算命令

機能

$$\begin{aligned}ER_n &\leftarrow ER_n / R_m \\R_m &\leftarrow ER_n \bmod R_m\end{aligned}$$

説明

- ワード型レジスタ ER_n と バイト型レジスタ R_m の除算を行い、商 16ビットと剰余 8ビットを得る命令です。被除数はワード型レジスタ ER_n で、除数はバイト型レジスタ R_m です。演算の結果、ER_n に商が、R_m に剰余が入ります。
- 0 で除算を行った場合は、キャリがセットされ、ER_n、R_m の値は不定となります。

フラグ

C	Z	S	OV	MIE	HC
*	*	-	-	-	-

- C : 0 による除算を行った場合 1、それ以外で 0 になります。
Z : 商が 0 で 1、それ以外で 0 になります。
- : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード	第2ワード		
DIV	ER _n	R _m	F n m 9			

EI

割込み許可命令

機能

MIE ← 1

説明

- ・MIE を1にし、マスカブル割込みを許可する命令です。
- ・MIE は、この命令の実行サイクルを含めて3サイクル後に1になります。従って MIE が0 の状態で EI 命令を実行しても、その直後の2サイクルはマスカブル割込みの禁止状態が継続しますので、アプリケーションプログラム設計時に注意を払う必要があります。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	*	—

MIE : 1になります。

— : 変化はありません。

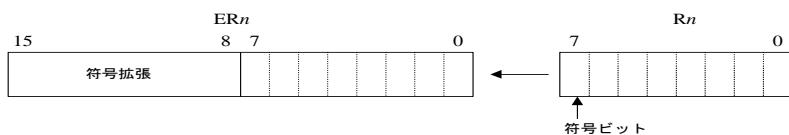
命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
EI			E	D	0	8

EXTBW ER_n

符号拡張

機能



説明

- R_n レジスタの内容を符号付きで16ビットに拡張し、結果を ER_n レジスタに反映します。
- 動作としては、 R_{n+1} の内容を R_n のビット7で埋めます。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : R_n の内容が 0 の時 1 になり、それ以外の時は 0 になります。

S : R_n のビット7の内容がセットされます。

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード	第2ワード		
EXTBW	ER _n		8	$n+1$	n	F

INC [EA]

インクリメント命令
(EA 間接)

機能

$[EA] \leftarrow [EA] +1$

説明

EA レジスタが指すメモリの内容を+1 します。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	*	—	*

Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。

S : 実行の結果の最上位ビットがセットされます。

OV : オーバフローがセットされた時 1 になり、それ以外の時は 0 になります。

HC : ビット 3 にキャリあるいはボローが生じた時 1 になります。それ以外の時は 0 になります。

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	DSR プリフィックスコード	命令フォーマット			
				第1 ワード	第2 ワード		
INC	[EA]			F	E	2	F
	*:[EA]		<word>	F	E	2	F

*	<word>		
pseg_addr	E	3	pseg_addr
DSR	F	E	9 F
Rd	9	0	d F

L ERn, obj

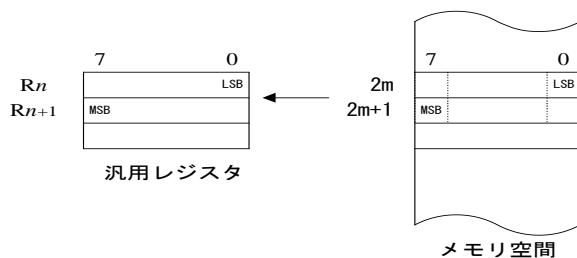
ワード型転送命令

機能

$ERn \leftarrow obj$

説明

- ERnレジスタに、objをアドレスとするメモリの内容をワード長で転送します。



フラグ

C	Z	S	OV	MIE	HC
-	*	*	-	-	-

Z : ERn が 0 の時 1 になり、それ以外の時は 0 になります。

S : 最上位ビットがセットされます。

- : 変化はありません。

命令フォーマット

次項に示します。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR プリフィックスコード	命令フォーマット			
				第1ワード		第2ワード	
L	ER_n	[EA]		9	n	3	2
		$*:[EA]$	$\langle word \rangle$	9	n	3	2
		[EA+]		9	n	5	2
		$*:[EA+]$	$\langle word \rangle$	9	n	5	2
		$[ER_m]$		9	n	m	2
		$*:[ER_m]$	$\langle word \rangle$	9	n	m	2
		$Disp16[ER_m]$		A	n	m	8
		$*:Disp16[ER_m]$	$\langle word \rangle$	A	n	m	8
		$Disp6[BP]$		B	n	0:0	$Disp6$
		$*:Disp6[BP]$	$\langle word \rangle$	B	n	0:0	$Disp6$
		$Disp6[FP]$		B	n	0:1	$Disp6$
		$*:Disp6[FP]$	$\langle word \rangle$	B	n	0:1	$Disp6$
		$Dadr$		9	n	1	2
		$*.Dadr$	$\langle word \rangle$	9	n	1	2

*	$\langle word \rangle$			
$pseg_addr$	E	3	$pseg_addr$	
DSR	F	E	9	F
Rd	9	0	d	F

L QR_n,obj

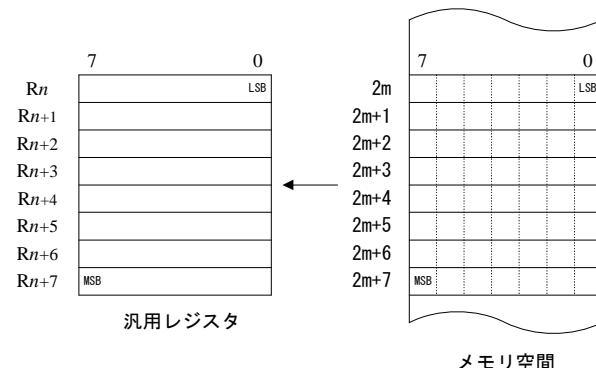
クワッドワード型
連続転送命令

機能

QR_n ← obj

説明

- QR_n レジスタに、obj で指定されるメモリデータをクワッドワード長で転送します。



フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : QR_n が 0 の時 1 になり、それ以外の時は 0 になります。

S : 最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

二一モニック	第1オペランド	第2オペランド	DSR プリフィックスコード	命令フォーマット			
				第1ワード			第2ワード
L	QR _n	[EA]		9	n	3	6
		*:[EA]	<word>	9	n	3	6
		[EA+]		9	n	5	6
		*:[EA+]	<word>	9	n	5	6

*	<word>			
pseg_addr	E	3	pseg_addr	
DSR	F	E	9	F
Rd	9	0	d	F

L Rn, obj

バイト型転送命令

機能

Rn \leftarrow obj

説明

・Rnレジスタに、objで指定されるメモリデータをバイト長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : Rnが0の時1になり、それ以外の時は0になります。

S : 最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

次頁に示します。

第3章 命令の詳細 インストラクションセット

ニーモ ニック	第1 オペランド	第2 オペランド	DSR プリフィックスコー ド	命令フォーマット			
				第1 ワード		第2 ワード	
L	Rn	[EA]		9	n	3	0
		*:[EA]	<word>	9	n	3	0
		[EA+]		9	n	5	0
		*:[EA+]	<word>	9	n	5	0
		[ERm]		9	n	m	0
		*:[ERm]	<word>	9	n	m	0
		Disp16[ERm]		9	n	m	8
		*:Disp16[ERm]	<word>	9	n	m	8
		Disp6[BP]		D	n	0:0	Disp6
		*:Disp6[BP]	<word>	D	n	0:0	Disp6
		Disp6[FP]		D	n	0:1	Disp6
		*:Disp6[FP]	<word>	D	n	0:1	Disp6
		Dadr		9	n	1	0
		* Dadr	<word>	9	n	1	0

*	<word>			
pseg_addr	E	3	pseg_addr	
DSR	F	E	9	F
Rd	9	0	d	F

L XR_n,obj

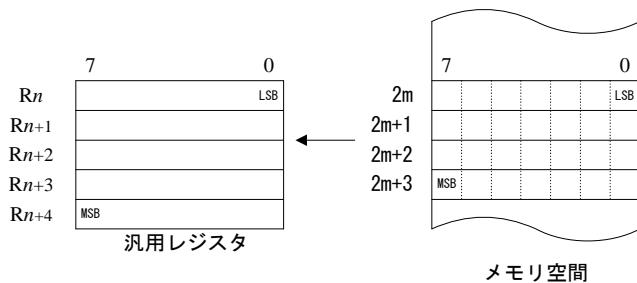
ダブルワード型
転送命令

機能

$XR_n \leftarrow obj$

説明

- XR_nレジスタに、objで示されるメモリデータをダブルワード長で転送します。



フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : XR_nが0の時1になり、それ以外の時は0になります。

S : 最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	DSR ブリフィックスコード	命令フォーマット	
				第1 ワード	第2 ワード
L	XR _n	[EA]		9	n 3 4
		*:[EA]	<word>	9	n 3 4
		[EA+]		9	n 5 4
		*:[EA+]	<word>	9	n 5 4

*	<word>		
pseg_addr	E	3	pseg_addr
DSR	F	E	9 F
Rd	9	0	d F

LEA *obj*

EAへの転送命令

機能

$EA \leftarrow obj$

説明

- EAレジスタに、*obj*で示されたワード値をセットします。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

—：変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード		第2ワード	
LEA	[ER <i>m</i>]		F	0	<i>m</i>	A
	<i>Dadr</i>		F	0	0	C
	<i>Disp16</i> [ER <i>m</i>]		F	0	<i>m</i>	B
						<i>Disp16</i>

MOV CER n , obj

コプロセッサ転送命令

機能

$CERn \leftarrow obj$

説明

- obj で示される内容を先頭とするメモリの内容を、 $CERn$ で示されるコプロセッサにワード長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR ブリフィックスコード	命令フォーマット	
				第1ワード	第2ワード
MOV	CER n	[EA]		F n 2 D	
		*: [EA]	<word>	F n 2 D	
		[EA+]		F n 3 D	
		*:[EA+]	<word>	F n 3 D	

*	<word>		
pseg_addr	E	3	pseg_addr
DSR	F	E	9 F
Rd	9	0	d F

MOV CQR_n, obj

コプロセッサ転送命令

機能

CQR_n ← obj

説明

- obj で示される内容を先頭とするメモリの内容を、CQR_n で示されるコプロセッサにクロップドワード長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR プリフィックスコード	命令フォーマット	
				第1ワード	第2ワード
MOV	CQR _n	[EA]		F n 6 D	
		*:[EA]	<word>	F n 6 D	
		[EA+]		F n 7 D	
		*:[EA+]	<word>	F n 7 D	

*	<word>			
pseg_addr	E	3	pseg_addr	
DSR	F	E	9	F
Rd	9	0	d	F

MOV CR n , obj

コプロセッサ転送命令

機能

$CRn \leftarrow obj$

説明

- objで示されるメモリの内容を、CR n で示されるコプロセッサにバイト長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	DSR プリフィックスコード	命令フォーマット				
				第1 ワード	第2 ワード			
MOV	CR n	[EA]		F	n	0	D	
		*:[EA]	<word>	F	n	0	D	
		[EA+]		F	n	1	D	
		*:[EA+]	<word>	F	n	1	D	

*	<word>		
pseg_addr	E	3	pseg_addr
DSR	F	E	9
Rd	9	0	d

MOV CR n , R m

コプロセッサ転送命令

機能

CR $n \leftarrow Rm$

説明

CR n で示されるコプロセッサに、R m の結果を転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット								
			第1 ワード	第2 ワード							
MOV	CR n	R m	A	⋮	n	⋮	m	⋮	E	⋮	

MOV CXR*n*, *obj*

コプロセッサ転送命令

機能

$CXRn \leftarrow obj$

説明

- *obj* で示されるメモリの内容を、*CXRn* で示されるコプロセッサにダブルワード長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR ブリフィックスコード	命令フォーマット	
				第1ワード	第2ワード
MOV	CXR <i>n</i>	[EA]		F n 4 D	
		*:[EA]	<word>	F n 4 D	
		[EA+]		F n 5 D	
		*:[EA+]	<word>	F n 5 D	

*	<word>		
<i>pseg_addr</i>	E	3	<i>pseg_addr</i>
DSR	F	E	9 F
R <i>d</i>	9	0	<i>d</i> F

MOV ECSR , R_m

転送命令

機能

- ELEVEL が0の時
 $LCSR \leftarrow R_m$
- ELEVEL が0以外の時
 $ECSR[ELEVEL] \leftarrow R_m$

説明

- R_m の内容を、ELEVEL が0の場合は LCSR に、0以外の場合は ECSR1-3 のいずれかに転送します。
- ELEVEL が0以外の場合に対象となる ECSR は、現在の ELEVEL の値を指す、いずれかひとつのレジスタです。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
MOV	ECSR	R _m	A	0	<i>m</i>	F

MOV ELR , ER_m

転送命令

機能

- ELEVEL が0の時
 $LR \leftarrow ER_m$
- ELEVEL が0以外の時
 $ELR[ELEVEL] \leftarrow ER_m$

説明

- ER_m の内容を、ELEVEL が0の場合は LR に転送し、0以外の場合は ELR1-3 のいずれかのレジスタに転送します。
- ELEVEL が0以外の場合に対象となる ELR は、現在の ELEVEL の値を指数とする、いずれかひとつのレジスタです。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
MOV	ELR	ER _m	A	⋮ m	⋮ 0	⋮ D

MOV EPSW , R_m

転送命令

機能

- ELEVEL が0以外の場合
 $\text{EPSW[ELEVEL]} \leftarrow \text{R}_m$

説明

- EPSW レジスタに、R_mレジスタの内容を転送します。
- 対象となる EPSW は、現在の ELEVEL の値を指数とする、いずれかひとつのレジスタです。ELEVEL が0の場合はレジスタへの書き込みを行わず、PC を次の命令の先頭へ進めるだけの動作となります。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
MOV	EPSW	R _m	A	0	<i>m</i>	C

MOV ER n , ELR

転送命令

機能

- ELEVEL が0の時
 $ERn \leftarrow LR$
- ELEVEL が0以外の時
 $ERn \leftarrow ELR[ELEVEL]$

説明

- ER n レジスタに、ELEVEL が0の場合は LR の内容を転送し、0以外の場合は ELR1-3 のいずれかの内容を転送します。
- ELEVEL が0以外の場合、対象となる ELR は、現在の ELEVEL の値を指数とする、いずれかひとつのレジスタです。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
MOV	ER n	ELR	A	n	0	5

MOV ER n , ER m

転送命令

機能

ER n \leftarrow ER m

説明

・ER n レジスタに、ER m レジスタの内容を転送します。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : ER n が 0 の時 1 になり、それ以外の時は 0 になります。

S : 実行の結果の最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

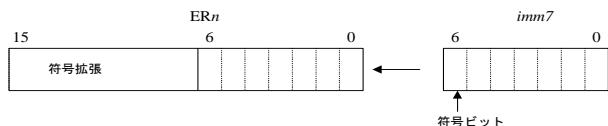
ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード	第2 ワード		
MOV	ER n	ER m	F	n	m	5

MOV ER_n, #imm7

転送命令

機能

$ER_n \leftarrow (\text{sign-extends})imm7$



説明

- ER_n レジスタに、imm7 で示される内容を符号付きで転送します。具体的には、imm7 を R_n レジスタのビット 0-6 にライトし、imm7 のビット 6 の値で、R_n レジスタのビット 7 と、R_{n+1} レジスタの全ビットを埋める動作となります。

実行例

```
MOV R0,#07Fh
MOV R1,#0h
MOV ERO,#-64 ; 実行の結果 1 が拡張され、R0=0C0h, R1=0FFH となります。
```

```
MOV R0,#03Fh
MOV R1,#0FFh
MOV ERO,#3Fh ; 実行の結果 0 が拡張され、R0=03fh, R1=0H となります。
```

フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : ER_n が 0 の時 1 になり、それ以外の時は 0 になります。

S : 実行の結果の最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット		
			第1 ワード	第2 ワード	
MOV	ER _n	#imm7	E	n	0 imm7

MOV ER n , SP

転送命令

機能

ER n ← SP

説明

- ・スタックポインタの内容を、ER n で示されるレジスタに転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード	第2ワード		
MOV	ER n	SP	A	⋮	n	⋮
			1	⋮	A	⋮

MOV *obj*, CER*m*

コプロセッサ転送命令

機能

(WORD) *obj* ← CER*m*

説明

- ・CER*m*で指定されるコプロセッサの内容を、現在のEAレジスタの内容を先頭アドレスとするメモリに、ワード長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR プリフィックスコード	命令フォーマット			
				第1ワード	第2ワード		
MOV	[EA]	CER <i>m</i>		F m A D			
	*:[EA]	CER <i>m</i>	<word>	F m A D			
	[EA+]	CER <i>m</i>		F m B D			
	*:[EA+]	CER <i>m</i>	<word>	F m B D			

*	<word>		
<i>pseg_addr</i>	E	3	<i>pseg_addr</i>
DSR	F	E	9 F
Rd	9	0	d F

MOV *obj*, CQR*m*

コプロセッサ転送命令

機能

(QWORD) *obj* ← CQR*m*

説明

- CQR*m* で指定されるコプロセッサの内容を、EA レジスタの内容を先頭アドレスとするメモリに、クワッドワード長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR プリフィックスコード	命令フォーマット	
				第1ワード	第2ワード
MOV	[EA]	CQR <i>m</i>		F m E D	
	*: [EA]	CQR <i>m</i>	<word>	F m E D	
	[EA+]	CQR <i>m</i>		F m F D	
	*: [EA+]	CQR <i>m</i>	<word>	F m F D	

*	<word>		
<i>pseg_addr</i>	E	3	<i>pseg_addr</i>
DSR	F E		9 F
R <i>d</i>	9 0		<i>d</i> F

MOV *obj*, CR*m*

コプロセッサ転送命令

機能

(BYTE) *obj* ← CR*m*

説明

- CR*m* で指定されるコプロセッサの内容を、EA レジスタの内容をアドレスとするメモリに、バイト長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	DSR プリフィックスコード	命令フォーマット	
				第1 ワード	第2 ワード
MOV	[EA]	CR <i>m</i>		F m 8	D
	*: [EA]	CR <i>m</i>	<word>	F m 8	D
	[EA+]	CR <i>m</i>		F m 9	D
	*: [EA+]	CR <i>m</i>	<word>	F m 9	D

*	<word>		
<i>pseg_addr</i>	E	3	<i>pseg_addr</i>
DSR	F	E	9 F
Rd	9	0	<i>d</i> F

MOV *obj*, CXR*m*

コプロセッサ転送命令

機能

(DOUBLE WORD) *obj* ← CXR*m*

説明

- CXR*m* で指定されるコプロセッサの内容を、EA レジスタの内容を先頭アドレスとするメモリに、ダブルワード長で転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	DSR プリフィックスコード	命令フォーマット	
				第1ワード	第2ワード
MOV	[EA]	CXR <i>m</i>		F m C D	
	*: [EA]	CXR <i>m</i>	<word>	F m C D	
	[EA+]	CXR <i>m</i>		F m D D	
	*: [EA+]	CXR <i>m</i>	<word>	F m D D	

*	<word>			
<i>pseg_addr</i>	E	3	<i>pseg_addr</i>	
DSR	F	E	9	F
Rd	9	0	d	F

MOV PSW , *obj*

転送命令

機能

$\text{PSW} \leftarrow \text{obj}$

説明

- PSW に、*obj*で示されたバイト値の内容を転送します。
- 本命令を使用して ELEVEL の値を書き換える場合、直後に NOP 命令を配置してください。

記述例

```
MOV PSW , #5Ah
NOP
RTI
```

NOP 命令を挿入しない場合、次に配置した命令が本命令で変化する前の ELEVEL を参照して実行されます。この動作がアプリケーションプログラムが誤動作する要因となる場合があります。

- MIE の値を1から0に書き換える場合、本命令を使用せず、DI 命令を使用してください。本命令を使用した場合、実行終了後1サイクルの間、MIE が1の状態が継続します。これによりプログラマの意図せぬ割込みがアサートされる可能性があることが、アプリケーションプログラムが誤動作する要因となる場合があります。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	*	*

* : 対応するビットの値に変化します。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
MOV	PSW	# <i>unsigned8</i>	E	9	<i>unsigned8</i>	
		R <i>m</i>	A	0	<i>m</i>	B

MOV R_n, CR_m

コプロセッサ転送命令

機能

R_n ← CR_m

説明

- ・R_nレジスタに、コプロセッサの内容を1バイト転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット		
			第1ワード	第2ワード	
MOV	R _n	CR _m	A n m	6	

MOV R_n, ECSR

転送命令

機能

- ・ELEVEL が0の時
 $R_n \leftarrow LCSR$
- ・ELEVEL が0以外の時
 $R_n \leftarrow ECSR[ELEVEL]$

説明

- ・R_n レジスタに、ELEVEL が0の場合は LCSR の内容を転送し、0以外の場合は ECSR1-3 のいずれかの内容を転送します。
- ・ELEVEL が0以外の場合、対象となる ECSR は、現在の ELEVEL の値を指数とするレジスタです。

フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
MOV	R _n	ECSR	A	n	0	7

MOV R_n, EPSW

転送命令

機能

- ELEVEL が0以外の時
 $R_n \leftarrow EPSW[ELEVEL]$

説明

- R_nレジスタに、EPSW の内容を転送します。
- 対象となる EPSW は、現在の ELEVEL の値を指すレジスタです。ELEVEL が0の場合は R_nレジスタへの書き込みを行わず、PC を次の命令の先頭へ進めるだけの動作となります。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
MOV	R _n	EPSW	A	n	0	4

MOV R_n, PSW

転送命令

機能

R_n ← PSW

説明

- ・R_nレジスタに、PSW の内容を転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット		
			第1ワード	第2ワード	
MOV	R _n	PSW	A n 0 3		

MOV Rn , obj

転送命令

機能

$Rn \leftarrow obj$

説明

- Rn レジスタに、 obj の内容を転送します。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : Rn が 0 の時 1 になり、それ以外の時は 0 になります。

S : 実行の結果の最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード		第2ワード	
MOV	Rn	Rm	8	n	m	0
		#imm8	0	n	imm8	

MOV SP , ER m

転送命令

機能

SP ← ER m

説明

・ER m で示されるレジスタの内容をスタックポインタに転送します。

フラグ

C	Z	S	OV	MIE	HC
—	—	—	—	—	—

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード		第2ワード	
MOV	SP	ER m	A	1	m	A

MUL ER_n,R_m

乗算命令

機能

ER_n \leftarrow R_n * R_m (_nは偶数)

説明

- ・バイト型レジスタ R_n と R_m の乗算を行い、積 16 ビットを得る命令です。被乗数は R_n の内容、乗数は R_m です。演算の結果、ER_n に積が入ります。

フラグ

C	Z	S	OV	MIE	HC
-	*	-	-	-	-

Z : 積がゼロの時 1 になり、それ以外の時は 0 になります。

- : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード	第2ワード		
MUL	ER _n	R _m	F	n	m	4

NEG R_n

符号反転命令

機能

R_n ← 0-R_n

説明

・バイト型レジスタ R_n の 2 の補数を取る命令です。結果は R_n に戻されます。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	-	*

- C : 演算の結果、ビット 7 よりキャリが発生した時に 1 になり、それ以外の時は 0 になります。
Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。
S : 実行の結果の最上位ビットがセットされます。
OV : オーバフローがセットされた時 1 になり、それ以外の時は 0 になります。
HC : ビット 3 にキャリあるいはボローが生じた時 1 になります。それ以外の時は 0 になります。
- : 変化はありません。

命令フォーマット

二モ ニック	第 1 オペランド	第 2 オペランド	命令フォーマット			
			第 1 ワード		第 2 ワード	
NEG	R _n		8	n	5	F

NOP

ノーオペレーション命令

機能

No operation

説明

- ・プログラムカウンタを次に進めます。

フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット	
			第1ワード	第2ワード
NOP			F E 8 F	

OR Rn , obj

論理和命令

機能

$Rn \leftarrow Rn | obj$

説明

- Rnレジスタの内容と objで示される内容の論理 ORをとり、結果を Rnレジスタに格納します。

フラグ

C	Z	S	OV	MIE	HC
—	*	*	—	—	—

Z : 実行結果が 0 の時に 1 にセットされ、それ以外の時は 0 になります。

S : 実行の結果の最上位ビットがセットされます。

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード		第2ワード	
OR	Rn	Rm	8	n	m	3
		#imm8	3	n	imm8	

POP レジスタリスト

コントロールレジスタ
復帰命令

機能

レジスタ群 \leftarrow (SP)

SP \leftarrow SP + n

説明

- ・スタックメモリに退避しているコントロールレジスタ群をレジスタリストで指定されたレジスタに復帰した後、スタックポインタ(SP)を指定したレジスタの個数に対応するバイト数だけインクリメントします。各々のレジスタが指定された時の実際のスタックの動作については、“1.6 スタックの変化について”を参照してください。
- ・レジスタリストは、次のいずれかです。
 - ①プログラムステータスワード(PSW)
 - ②サブルーチン用 PC 退避レジスタ(LR)
 - ③プログラムカウンタ(PC)
 - ④EA レジスタ
- ・レジスタリストは、全て記述する必要はありません。ただし少なくとも一つのレジスタリストを記述する必要があります。
- ・オペランドのレジスタ記述は順不同ですが、復帰順は固定です。この命令を用いた場合のレジスタ復帰順は
EA \rightarrow LR \rightarrow PSW \rightarrow PC となります。
- ・この命令実行中は、メモリ空間のワードバウンダリが発生しません。従って SP が奇数の場合も、そのアドレスから始まるデータが操作の対象になります。
- ・通常、サブルーチンから復帰する場合は RTI 命令を、割込みから復帰する場合は RTI 命令を使用しますが、サブルーチンがネストする場合、あるいは多重割込みが発生する可能性がある場合は、PUSH 命令で現在の退避レジスタの内容をスタックメモリに退避する必要があります。この場合 POP 命令を使用して、退避レジスタをコントロールレジスタに復帰させます。詳細は、“1.4 例外レベルと退避レジスタの取り扱いについて”を参照して下さい。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	*	*

* : PSW の復帰が指定された場合は、対応するフラグが変化します。
PSW の復帰が指定されていない場合は変化ありません。

命令フォーマット

ニーモ ニック	第1 オペランド	命令フォーマット			
		第1 ワード			
POP	EA	F	1	8	E
	PC	F	2	8	E
	EA, PC	F	3	8	E
	PSW	F	4	8	E
	EA, PSW	F	5	8	E
	PC, PSW	F	6	8	E
	EA, PC, PSW	F	7	8	E
	LR	F	8	8	E
	EA, LR	F	9	8	E
	PC, LR	F	A	8	E
	EA, PC, LR	F	B	8	E
	LR, PSW	F	C	8	E
	EA, PSW, LR	F	D	8	E
	PC, PSW, LR	F	E	8	E
	EA, PC, PSW, LR	F	F	8	E

POP *obj*

汎用レジスタ
復帰命令

機能

レジスタ群 \leftarrow (SP)

SP \leftarrow SP + n

説明

- ・スタックポインタの示すシステムスタックの内容を汎用レジスタに復帰したあと、SP を *obj* で指定したレジスタに対応するバイト数だけインクリメントします。各々のレジスタが指定された時の実際のスタックの動作については、“1.6 スタックの変化について”を参照してください。
- ・*obj* には、復帰するレジスタの型を指定します。
- ・スタックポインタは偶数バイト単位で変化します。Rn レジスタが指定された場合は、レジスタの復帰動作後、SPのみが1インクリメントされるダミーサイクルが1サイクル分挿入されます。ダミーサイクル実行中はレジスタの復帰は行われません。
- ・この命令実行中は、メモリ空間のワードバウンダリが発生しません。従って SP が奇数の場合も、そのアドレスから始まるデータが操作の対象になります。

フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	命令フォーマット				
		第1 ワード				
POP	Rn	F	n	0:0:0:0	E	
	ERn	F	n	0:0:0:1	E	
	XRn	F	n	0:0:1:0	E	
	QRn	F	n	0:0:1:1	E	

PUSH レジスリスト

コントロールレジスタ
退避命令

機能

SP ← SP-n
(SP) ← コントロールレジスタ

説明

- ・スタックポインタを レジスリストで指定されたコントロールレジスタに対応するバイト数だけデクリメントした後、コントロールレジスタ群の内容をスタックポインタの示すシステムstackに退避する命令です。各々のレジスタが指定された時の実際のstackの動作については、“1.6 スタックの変化について”を参照してください。
- ・レジスリストは、次のいずれかです。
 - ① PSW 退避レジスタ(EPSW)
 - ② 割込み用リンクレジスタ(ELR)
 - ③ サブルーチン用リンクレジスタ(LR)
 - ④ EA レジスタ(EA)
- ・レジスリストは、全て記述する必要はありません。ただし少なくとも一つのレジスリストを記述する必要があります。
- ・オペランドのレジスタ記述は順不同ですが、退避順は固定です。この命令を用いた場合のレジスタ退避順は
 - ELR → EPSW → LR → EAとなります。
- ・この命令実行中は、メモリ空間のワードバウンダリが発生しません。従って SP が奇数の場合も、そのアドレスから始まるデータが操作の対象になります。
- ・通常、サブルーチンから復帰する場合は RT 命令を、割込みから復帰する場合は RTI 命令を使用しますが、サブルーチンがネストする場合、あるいは多重割込みが発生する可能性がある場合は、PUSH 命令で現在の退避レジスタの内容をstackメモリに退避する必要があります。詳細は、“1.4 例外レベルと退避レジスタの取り扱いについて”を参照して下さい。

フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

第3章 命令の詳細 インストラクションセット

命令フォーマット

ニーモ ニック	第1 オペランド	命令フォーマット			
		第1 ワード			
PUSH	EA	F	1	C	E
	ELR	F	2	C	E
	EA, ELR	F	3	C	E
	EPSW	F	4	C	E
	EPSW, EA	F	5	C	E
	EPSW, ELR	F	6	C	E
	EPSW, ELR, EA	F	7	C	E
	LR	F	8	C	E
	LR, EA	F	9	C	E
	LR, ELR	F	A	C	E
	LR, EA, ELR	F	B	C	E
	LR, EPSW	F	C	C	E
	LR, EPSW, EA	F	D	C	E
	LR, EPSW, ELR	F	E	C	E
	LR, EPSW, ELR, EA	F	F	C	E

PUSH *obj*

汎用レジスタ
退避命令

機能

SP \leftarrow SP - n
(SP) \leftarrow 汎用レジスタ

説明

- ・スタックポインタを *obj* で指定したレジスタに対応するバイト数だけデクリメントした後、汎用レジスタをスタックポインタの示すシステムスタックに退避します。*obj* には、退避するレジスタの型を指定します。
- ・スタックポインタは常に偶数バイト単位で変化します。R_n レジスタが指定された場合は、レジスタの退避動作の前に、SP のみが1デクリメントされるダミーサイクルが1サイクル分挿入されます。ダミーサイクル実行中はレジスタの退避は行われません。各々のレジスタが指定された時の実際のスタックの動作については、”1.6 スタックの変化について”を参照してください。
- ・この命令実行中は、メモリ空間のワードバウンダリが発生しません。従って SP が奇数の場合も、そのアドレスから始まるデータが操作の対象になります。

フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

命令フォーマット

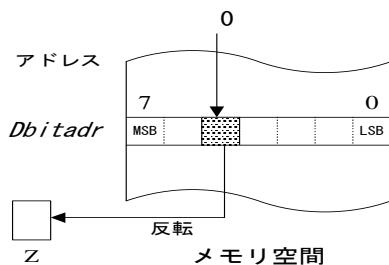
ニーモ ニック	第1 オペランド	命令フォーマット					
		第1 ワード					
PUSH	R _n	F	n	0100	E		
	ER _n	F	n	0101	E		
	XR _n	F	n	0110	E		
	QR _n	F	n	0111	E		

RB *Dbitadr*

リセットビット命令

機能

$$\begin{aligned} Z &\leftarrow \sim[Dbitadr] \\ [Dbitadr] &\leftarrow 0 \end{aligned}$$



説明

- 指定したメモリの1ビットの状態を読み出し、その結果をZフラグに反映した後、そのビットをリセットする命令です。
- Dbitadr*には、ビットリセットするメモリのアドレスを *Dadr.bit* の形式で記述します。
- bit*は、0-7までの整数で、リセットするビットの、ビット位置を記述します。

フラグ

C	Z	S	OV	MIE	HC
—	*	—	—	—	—

Z : ビットの内容が0の時1になり、それ以外の時は0になります。

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	DSR プリフィックスコード	命令フォーマット		
			第1ワード	第2ワード	
RB	<i>Dbitadr</i>		A 0 1 <i>bit</i> 2		<i>Dadr</i>
	*: <i>Dbitadr</i>	<word>	A 0 1 <i>bit</i> 2		<i>Dadr</i>

*	<word>			
<i>pseg_addr</i>	E	3	<i>pseg_addr</i>	
DSR	F	E	9	F
Rd	9	0	<i>d</i>	F

RB Rn . bit_offset

リセットビット命令

機能

$Z \leftarrow \sim Rn[bit_offset]$
 $Rn[bit_offset] \leftarrow 0$

説明

- ・指定したレジスタの1ビットの状態を読み出し、その結果をZフラグに反映した後、そのビットをリセットする命令です。
- ・*bit_offset*には、リセットするビットの、ビット位置を記述します。

フラグ

C	Z	S	OV	MIE	HC
—	*	—	—	—	—

Z : ビットの内容が0の時1になり、それ以外の時は0になります。
— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット	
			第1 ワード	第2 ワード
RB	Rn.bit_offset		A n 0 bit 2	

RC

リセットキャリ命令

機能

C ← 0

説明

- C フラグをリセットします。

フラグ

C	Z	S	OV	MIE	HC
*	—	—	—	—	—

C : 0になります。
— : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
RC			E	B	7	F

RT

サブルーチンからの復帰命令

機能

CSR ← LCSR
PC ← LR

説明

- BL 命令より呼び出されたサブルーチンからの復帰命令です。
- CSR にはサブルーチン用セグメントレジスタ(LCSR)の内容が復帰され、次の命令は復帰したコードセグメントからセグメント処理を行います。
- PC にはサブルーチン用リンクレジスタ(LR)の内容が復帰され、次の命令は復帰した PC から処理を行います。

フラグ

C	Z	S	OV	MIE	HC
-	-	-	-	-	-

- : 変化はありません。

命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード	第2 ワード		
RT			F	E	1	F

RTI

割込みからの
復帰命令

機能

```
CSR ← ECSR[ELEVEL]
PC  ← ELR [ELEVEL]
PSW ← EPSW[ELEVEL]
```

説明

- ・割込みルーチンから復帰します。
- ・EPSW は例外処理用 PSW 退避レジスタです。現在の ELEVEL の値より、EPSW1(割込み用)、EPSW2(NMI 用)のいずれかが選択されて PSW に復帰されます。
- ・ELR は例外処理用リンクレジスタです。現在の ELEVEL の値より、ELR1(割込み用)、ELR2(例外処理用)のいずれかが選択されて PC に復帰されます。

フラグ

C	Z	S	OV	MIE	HC
*	*	*	*	*	*

* : EPSW の内容が書き込まれます
— : 変化はありません。

命令フォーマット

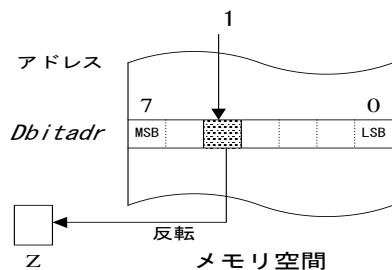
ニーモ ニック	第 1 オペランド	第 2 オペランド	命令フォーマット			
			第 1 ワード	第 2 ワード		
RTI			F	E	0	F

SB *Dbitadr*

セットビット命令

機能

$Z \leftarrow \sim [Dbitadr]$
 $[Dbitadr] \leftarrow 1$



説明

- *Dbitadr*で指定されたメモリの1ビットの状態を調べ、その結果をZフラグに反映した後、そのビットをセットする命令です。
- *Dbitadr*は、ビットセットするメモリのアドレスを、*Dadr16.bit*の形式で記述します。
- *bit*は、0-7までの整数で、セットするビットの、ビット位置を記述します。

フラグ

C	Z	S	OV	MIE	HC
—	*	—	—	—	—

Z : ビットの内容が0の時1になり、それ以外の時は0になります。

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	DSR プリフィックスコード	命令フォーマット			
			第1ワード		第2ワード	
SB	<i>Dbitadr</i>		A 0 1 <i>bit</i> 0		<i>Dadr</i>	
	*: <i>Dbitadr</i>	<word>	A 0 1 <i>bit</i> 0		<i>Dadr</i>	

*	<word>			
<i>pseg_addr</i>	E	3	<i>pseg_addr</i>	
DSR	F	E	9	F
Rd	9	0	<i>d</i>	F

SB Rn . bit_offset

セットビット命令

機能

$Z \leftarrow \sim Rn[bit_offset]$
 $Rn[bit_offset] \leftarrow 1$

説明

- ・指定したレジスタの1ビットの状態を調べ、その結果をZフラグに反映した後、そのビットをセットする命令です。
- ・*bit_offset*には、セットするビットの、ビット位置を記述します。

フラグ

C	Z	S	OV	MIE	HC
—	*	—	—	—	—

Z : ビットの内容が0の時1になり、それ以外の時は0になります。

— : 変化はありません。

命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット	
			第1ワード	第2ワード
SB	Rn.bit_offset		A n 0 bit 0	

SC

セットキャリ命令

機能

C ← 1

説明

- ・キャリフラグを 1 にします。

フラグ

C	Z	S	OV	MIE	HC
*	—	—	—	—	—

C : 1 になります。

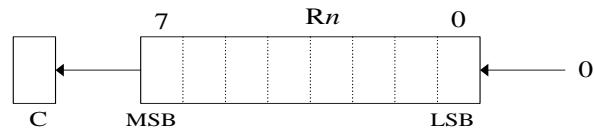
命令フォーマット

ニーモ ニック	第1 オペランド	第2 オペランド	命令フォーマット			
			第1 ワード		第2 ワード	
SC			E	D	8	0

SLL Rn , obj

論理左シフト命令

機能



説明

- ・Rnの内容を第2オペランドで指定されたビット数だけ左に論理シフトし、最後にシフトアウトした値をCに格納します。
- ・シフト幅は0-7の範囲です。第2オペランドにRmが指定された場合は、Rmのビット2-0のデータがシフト幅として読み込まれ、ビット7-3のデータは無視されます。シフト幅として0を指定した場合はNOP命令と同じ動作となります。
- ・LSBには、シフト幅の数だけ0がシフトされて入ります。
- ・本命令の直前にSLLC命令を配置することにより、多バイトデータのシフト演算が可能となります。

フラグ

C	Z	S	OV	MIE	HC
*	-	-	-	-	-

C : シフト動作で、最後にキャリアウトしたデータがセットされます。
- : 変化はありません。

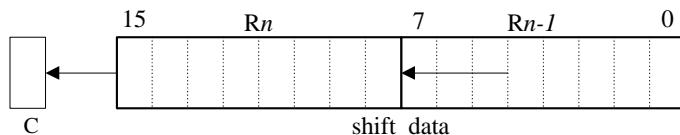
命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット					
			第1ワード				第2ワード	
SLL	Rn	Rm	8	n	m	A		
		#width	9	n	0 width	A		

SLLC R_n, obj

論理左シフト継続命令

機能



説明

- R_n と、 R_{n-1} レジスタを、同時に左に最大 7 ビットシフトし、演算結果を R_n レジスタに格納します。
- シフトデータは、 R_n レジスタを上位バイト、 R_{n-1} レジスタを下位バイトとするワード型データです。R0 を上位バイトとして選択した場合、R15 が下位バイトとなります。演算終了後は上位バイトが R_n レジスタに格納されます。
- obj* の内容はシフトするビット幅で、命令に先立って 0-7 の値が格納されている必要があります。第 2 オペランドに R_m が指定された場合、 R_m のビット 2-0 のデータがシフト幅として読み込まれ、ビット 7-3 のデータは無視します。シフト幅として 0 を指定した場合、NOP 命令と同じ動作となります。
- 本命令を記述した直後に SLL 命令を配置することで、多バイトデータのシフト演算が可能となります。

例) ダブルワード長データの左シフト

SLLC R3, R5

SLLC R2, R5

SLLC R1, R5

SLL R0, R5 (XR0 のシフト演算完了)

フラグ

C	Z	S	OV	MIE	HC
*	-	-	-	-	-

C : シフト動作で、最後にキャリアウトしたデータがセットされます。

- : 変化はありません。

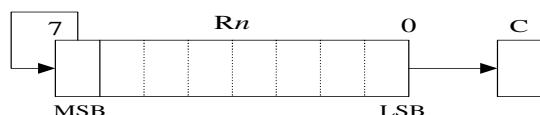
命令フォーマット

ニーモ ニック	第 1 オペランド	第 2 オペランド	命令フォーマット			
			第 1 ワード			第 2 ワード
SLLC	R _n	R _m	8	n	m	B
		#width	9	n	0 width	B

SRA R_n, obj

算術右シフト命令

機能



説明

- ・R_nレジスタの内容を、第2オペランドで指定されたビット数だけ右算術シフトします。
- ・シフト幅は0-7の範囲です。第2オペランドにR_mが指定された場合、R_mのビット2-0のデータがシフト幅として読み込まれ、ビット7-3のデータは無視されます。シフト幅として0を指定した場合、NOP命令と同じ動作となります。
- ・R_nの最上位ビットは変化しません。

フラグ

C	Z	S	OV	MIE	HC
*	-	-	-	-	-

C : 最後にビット0よりアウトした値が設定されます。
- : 変化はありません。

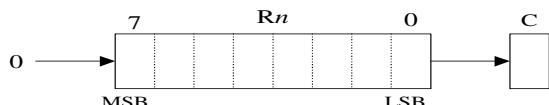
命令フォーマット

ニーモニック	第1オペランド	第2オペランド	命令フォーマット			
			第1ワード			第2ワード
SRA	R _n	R _m	8	⋮ n ⋮ m ⋮	E	
		#width	9	⋮ n ⋮ 0 ⋮ width ⋮	E	

SRL R_n, obj

論理右シフト命令

機能



説明

- R_nレジスタの内容を、第2オペランドで指定されたビット数だけ論理右シフトし、最後にシフトアウトされたビットを C に格納します。
- シフト幅は 0-7 の範囲です。第2オペランドに R_m が指定された場合、R_m のビット 2-0 のデータがシフト幅として読み込まれ、ビット 7-3 のデータは無視されます。シフト幅として 0 を指定した場合、NOP 命令と同じ動作となります。
- MSB にはシフト幅の値だけ 0 がシフトして入ります。
- 本命令の直前に SRLC 命令を配置することにより、多バイトデータのシフト演算が可能となります。

フラグ

C	Z	S	OV	MIE	HC
*	-	-	-	-	-

C : 最後にビット 0 よりアウトした値が設定されます。
- : 変化はありません。

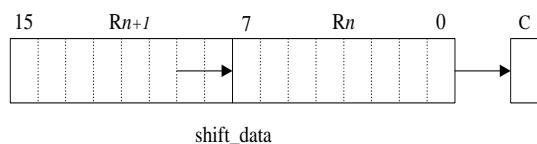
命令フォーマット

			命令フォーマット			
			第1ワード			第2ワード
ニーモ ニック	第1 オペランド	第2 オペランド	8	n	m	C
SRL	R _n	R _m	9	n	0	C
		#width			width	

SRLC R_n, obj

論理右シフト継続命令

機能



説明

- ・R_nと、R_{n+1}レジスタを、同時に右に最大7ビットシフトし、演算結果をR_nレジスタに格納します。
 - ・シフトデータはR_nレジスタを下位バイト、R_{n+1}レジスタを上位バイトとするワード型データです。演算終了後の下位バイトがR_nレジスタに格納されます。R15を下位バイトとして選択した場合、R0が上位バイトとなります。演算終了後は下位バイトがR_nレジスタに格納されます。
 - ・objの内容はシフトするビット幅で、命令に先立って0-7の値が格納されている必要があります。第2オペランドにR_mが指定された場合、R_mのビット2-0のデータがシフト幅として読み込まれ、ビット7-3のデータは無視されます。シフト幅として0を指定した場合、NOP命令と同じ動作となります。
 - ・本命令を記述した直後にSRL命令を配置することで、多バイトデータのシフト演算が可能となります。
- 例) ダブルワード長データの右シフト
- ```

SRLC R0, R5
SRLC R1, R5
SRLC R2, R5
SRL R3, R5 (XR0 のシフト演算完了)

```

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| * | - | - | -  | -   | -  |

C : 最後にビット0よりアウトした値が設定されます。  
- : 変化はありません。

### 命令フォーマット

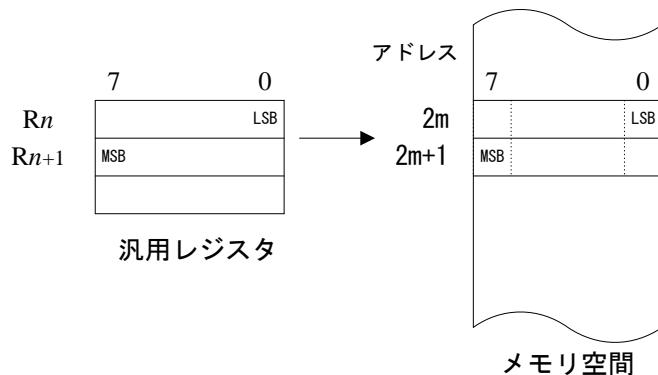
| ニーモ<br>ニック | 第1<br>オペランド    | 第2<br>オペランド    | 命令フォーマット  |     |   |           |       |
|------------|----------------|----------------|-----------|-----|---|-----------|-------|
|            |                |                | 第1<br>ワード |     |   | 第2<br>ワード |       |
| SRLC       | R <sub>n</sub> | R <sub>m</sub> | 8         | ... | n | ...       | m     |
|            |                | #width         | 9         | ... | n | 0         | width |

## ST ER*n*, *obj*

ワード型転送命令

### 機能

*obj*  $\leftarrow$  ER*n*



### 説明

- ER*n* レジスタの内容を、*obj* で示される内容をアドレスとするメモリに、ワード型でストアします。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| — | — | — | —  | —   | —  |

— : 変化はありません。

### 命令フォーマット

次ページに示します。

### 第3章 命令の詳細 インストラクションセット

---

| ニーモ<br>ニック | 第1<br>オペランド | 第2<br>オペランド       | DSR<br>プリフィックスコー<br>ド | 命令フォーマット  |     |           |       |
|------------|-------------|-------------------|-----------------------|-----------|-----|-----------|-------|
|            |             |                   |                       | 第1<br>ワード |     | 第2<br>ワード |       |
| ST         | ER $n$      | [EA]              |                       | 9         | $n$ | 3         | 3     |
|            |             | *:[EA]            | <word>                | 9         | $n$ | 3         | 3     |
|            |             | [EA+]             |                       | 9         | $n$ | 5         | 3     |
|            |             | *:[EA+]           | <word>                | 9         | $n$ | 5         | 3     |
|            |             | [ER $m$ ]         |                       | 9         | $n$ | $m$       | 3     |
|            |             | *:[ER $m$ ]       | <word>                | 9         | $n$ | $m$       | 3     |
|            |             | Disp16[ER $m$ ]   |                       | A         | $n$ | $m$       | 9     |
|            |             | *:Disp16[ER $m$ ] | <word>                | A         | $n$ | $m$       | 9     |
|            |             | Disp6[BP]         |                       | B         | $n$ | 1:0       | Disp6 |
|            |             | *:Disp6[BP]       | <word>                | B         | $n$ | 1:0       | Disp6 |
|            |             | Disp6[FP]         |                       | B         | $n$ | 1:1       | Disp6 |
|            |             | *:Disp6[FP]       | <word>                | B         | $n$ | 1:1       | Disp6 |
|            |             | Dadr              |                       | 9         | $n$ | 1         | 3     |
|            |             | *:Dadr            | <word>                | 9         | $n$ | 1         | 3     |
|            |             |                   |                       |           |     |           | Dadr  |

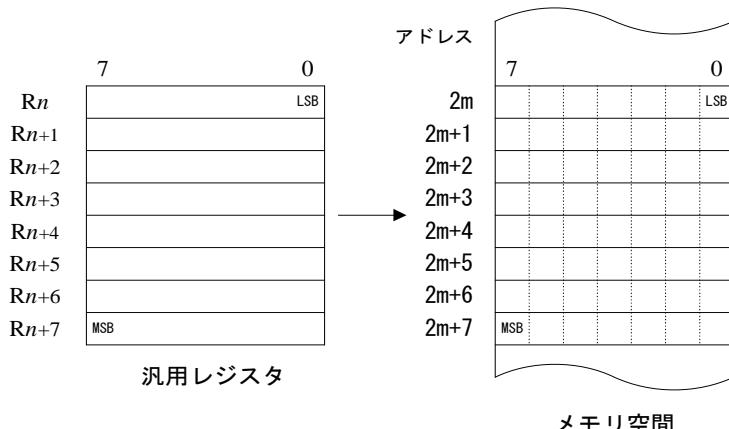
| *         | <word> |   |           |   |
|-----------|--------|---|-----------|---|
| pseg_addr | E      | 3 | pseg_addr |   |
| DSR       | F      | E | 9         | F |
| Rd        | 9      | 0 | d         | F |

## ST QR<sub>n</sub>, obj

ダブルワード型  
連続転送命令

### 機能

*obj* ← QR<sub>n</sub>



### 説明

- QR<sub>n</sub> レジスタの内容を、obj で指定されるメモリに、クワッドワード型でストアします。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| — | — | — | —  | —   | —  |

— : 変化はありません。

### 命令フォーマット

| ニーモ<br>ニック | 第1<br>オペランド     | 第2<br>オペランド | DSR<br>ブリフィックスコード | 命令フォーマット  |           |
|------------|-----------------|-------------|-------------------|-----------|-----------|
|            |                 |             |                   | 第1<br>ワード | 第2<br>ワード |
| ST         | QR <sub>n</sub> | [EA]        |                   | 9 n 3     | 7         |
|            |                 | *: [EA]     | <word>            | 9 n 3     | 7         |
|            |                 | [EA+]       |                   | 9 n 5     | 7         |
|            |                 | *:[EA+]     | <word>            | 9 n 5     | 7         |

|           |        |   |           |
|-----------|--------|---|-----------|
| *         | <word> |   |           |
| pseg_addr | E      | 3 | pseg_addr |
| DSR       | F      | E | 9 F       |
| Rd        | 9      | 0 | d F       |

## ST R<sub>n</sub>, obj

バイト型転送命令

### 機能

obj ← R<sub>n</sub>

### 説明

- R<sub>n</sub> レジスタの内容を、obj で示される値をアドレスとするメモリに、バイト型でストアします。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| — | — | — | —  | —   | —  |

— : 変化はありません。

### 命令フォーマット

次頁に示します。

| ニーモ<br>ニック | 第1<br>オペランド | 第2<br>オペランド      | DSR<br>プリフィックスコー<br>ド | 命令フォーマット  |   |           |         |
|------------|-------------|------------------|-----------------------|-----------|---|-----------|---------|
|            |             |                  |                       | 第1<br>ワード |   | 第2<br>ワード |         |
| ST         | $Rn$        | [EA]             |                       | 9         | n | 3         | 1       |
|            |             | *:[EA]           | <word>                | 9         | n | 3         | 1       |
|            |             | [EA+]            |                       | 9         | n | 5         | 1       |
|            |             | *:[EA+]          | <word>                | 9         | n | 5         | 1       |
|            |             | [ERm]            |                       | 9         | n | m         | 1       |
|            |             | *:[ERm]          | <word>                | 9         | n | m         | 1       |
|            |             | $Disp16[ERm]$    |                       | 9         | n | m         | 9       |
|            |             | *: $Disp16[ERm]$ | <word>                | 9         | n | m         | 9       |
|            |             | $Disp6[BP]$      |                       | D         | n | 0         | $Disp6$ |
|            |             | *: $Disp6[BP]$   | <word>                | D         | n | 0         | $Disp6$ |
|            |             | $Disp6[FP]$      |                       | D         | n |           | $Disp6$ |
|            |             | *: $Disp6[FP]$   | <word>                | D         | n | 1         | $Disp6$ |
|            |             | $Dadr$           |                       | 9         | n | 1         | 1       |
|            |             | *: $Dadr$        | <word>                | 9         | n | 1         | 1       |
|            |             |                  |                       |           |   |           | $Dadr$  |

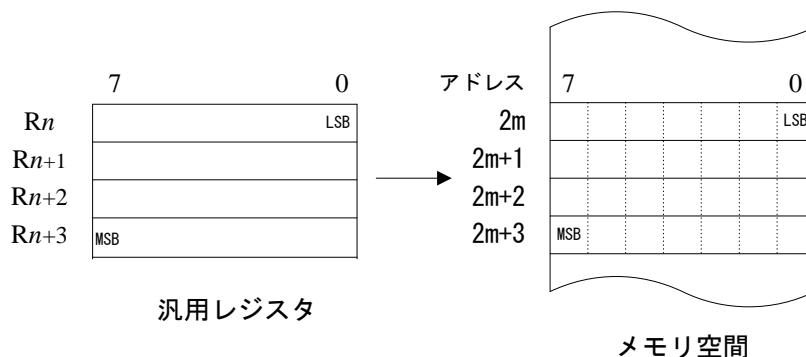
|              |        |   |              |
|--------------|--------|---|--------------|
| *            | <word> |   |              |
| $pseg\_addr$ | E      | 3 | $pseg\_addr$ |
| DSR          | F      | E | 9            |
| $Rd$         | 9      | 0 | $d$          |

## ST $XRn, obj$

ダブルワード型転送命令

### 機能

$obj \leftarrow XRn$



### 説明

- $XRn$  レジスタの内容を、 $obj$  で指定されるメモリにダブルワード型でストアします。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| — | — | — | —  | —   | —  |

— : 変化はありません。

### 命令フォーマット

| ニーモ<br>ニック | 第1<br>オペランド | 第2<br>オペランド | DSR<br>プリフィックスコー<br>ド | 命令フォーマット  |           |   |   |
|------------|-------------|-------------|-----------------------|-----------|-----------|---|---|
|            |             |             |                       | 第1<br>ワード | 第2<br>ワード |   |   |
| ST         | $XRn$       | [EA]        |                       | 9         | $n$       | 3 | 5 |
|            |             | *:[EA]      | <word>                | 9         | $n$       | 3 | 5 |
|            |             | [EA+]       |                       | 9         | $n$       | 5 | 5 |
|            |             | *:[EA+]     | <word>                | 9         | $n$       | 5 | 5 |

| *            | <word> |   |              |   |
|--------------|--------|---|--------------|---|
| $pseg\_addr$ | E      | 3 | $pseg\_addr$ |   |
| DSR          | F      | E | 9            | F |
| Rd           | 9      | 0 | d            | F |

## SUB R<sub>n</sub>, R<sub>m</sub>

減算命令

### 機能

R<sub>n</sub> ← R<sub>n</sub> - R<sub>m</sub>

### 説明

- R<sub>n</sub> レジスタの内容から、R<sub>m</sub> レジスタの内容を減算し、結果を R<sub>n</sub> レジスタに格納します。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| * | * | * | *  | -   | *  |

- C : 演算の結果、ビット 7 にボローが発生した時に 1 になり、それ以外の時は 0 になります。  
Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。  
S : 実行の結果のビット 7 最上位ビットがセットされます。  
OV : オーバフローがセットされた時 1 になり、それ以外の時は 0 になります。  
HC : ビット 3 にキャリあるいはボローが生じた時 1 になり、それ以外の時は 0 になります。  
- : 変化はありません。

### 命令フォーマット

| ニーモ<br>ニック | 第1<br>オペランド    | 第2<br>オペランド    | 命令フォーマット  |           |   |   |
|------------|----------------|----------------|-----------|-----------|---|---|
|            |                |                | 第1<br>ワード | 第2<br>ワード |   |   |
| SUB        | R <sub>n</sub> | R <sub>m</sub> | 8         | n         | m | 8 |

## SUBC R<sub>n</sub>, R<sub>m</sub>

キャリ付き減算命令

### 機能

R<sub>n</sub> ← R<sub>n</sub> - R<sub>m</sub> - C

### 説明

・R<sub>n</sub>レジスタの内容から、R<sub>m</sub>レジスタの内容とキャリを減算します。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| * | * | * | *  | -   | *  |

C : 演算の結果、ビット7にボローが発生した時に1になり、それ以外の時は0になります。

Z : 実行前のZの内容が0の時は、実行結果に関わらず0になります。  
実行前のZの内容が1の時は、実行結果が0の時1になり、それ以外の時は0になります

S : 実行の結果のビット7最上位ビットがセットされます。

OV : オーバフローがセットされた時1になり、それ以外の時は0になります。

HC : ビット3にキャリあるいはボローが生じた時1になり、それ以外の時は0になります。

- : 変化はありません。

### 命令フォーマット

| ニーモ<br>ニック | 第1<br>オペランド    | 第2<br>オペランド    | 命令フォーマット  |           |   |   |
|------------|----------------|----------------|-----------|-----------|---|---|
|            |                |                | 第1<br>ワード | 第2<br>ワード |   |   |
| SUBC       | R <sub>n</sub> | R <sub>m</sub> | 8         | n         | m | 9 |

## SWI #*snum*

---

ソフトウェア割込み

### 機能

---

```
EPSW1 ←PSW
ELEVEL ← 1
ELR1 ← PC+2
ECSR1 ←CSR
MIE ←0
PC ←TABLE[snum<<1]
```

### 説明

---

- ・*snum* で指定されたベクターテーブルの内容を PC に書き込みます。
- ・*snum* は 0-63 までの整数で、割込み先のアドレスを格納しているベクターテーブルの、ベクタ番号を指定します。
- ・割込みサイクル中に次の命令の先頭アドレスは ELR1 に退避されます。

### フラグ

---

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| - | - | - | -  | *   | -  |

MIE: 0になります。

- : 変化はありません。

### 命令フォーマット

---

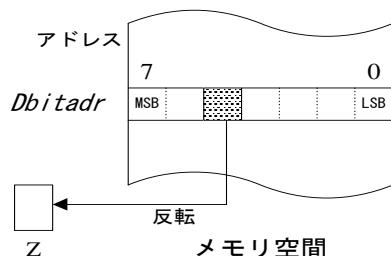
| ニーモ<br>ニック | 第1<br>オペランド   | 命令フォーマット  |   |    |             |  |  |
|------------|---------------|-----------|---|----|-------------|--|--|
|            |               | 第1<br>ワード |   |    | 第2<br>ワード   |  |  |
| SWI        | # <i>snum</i> | E         | 5 | 00 | <i>snum</i> |  |  |

## TB *Dbitadr*

テストビット命令

### 機能

$Z \leftarrow \sim [Dbitadr]$



### 説明

- ・指定したメモリの1ビットの内容をテストし、結果をZフラグに格納します。
- ・*Dbitadr*は、ビットテストするメモリのアドレスを、*Dadr16.bit*の形式で記述します。
- ・*bit*は、0-7までの整数で、リセットするビットのビット位置を記述します。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| — | * | — | —  | —   | —  |

Z : ビットの内容が0の時1になり、それ以外の時は0になります。

— : 変化はありません。

### 命令フォーマット

| ニーモニック | 第1オペランド           | DSR<br>プリフィックスコード | 命令フォーマット |   |            |             |
|--------|-------------------|-------------------|----------|---|------------|-------------|
|        |                   |                   | 第1ワード    |   | 第2ワード      |             |
| TB     | <i>Dbitadr</i>    |                   | A        | 0 | <i>bit</i> | 1           |
|        | *: <i>Dbitadr</i> | <word>            | A        | 0 | <i>bit</i> | 1           |
|        |                   |                   |          |   |            | <i>Dadr</i> |

| *                | <word> |   |                  |   |
|------------------|--------|---|------------------|---|
| <i>pseg_addr</i> | E      | 3 | <i>pseg_addr</i> |   |
| DSR              | F      | E | 9                | F |
| Rd               | 9      | 0 | <i>d</i>         | F |

## TB Rn . bit\_offset

テストビット命令

### 機能

$Z \leftarrow \sim Rn[bit\_offset]$

### 説明

- ・指定したレジスタの1ビットの内容をテストし、結果をZフラグに格納します。
- ・*bit\_offset*には、テストするビットのビット位置を記述します。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| — | * | — | —  | —   | —  |

Z : ビットの内容が0の時1になり、それ以外の時は0になります。  
— : 変化はありません。

### 命令フォーマット

| ニーモ<br>ニック | 第1<br>オペランド | 第2<br>オペランド | 命令フォーマット  |   |           |   |
|------------|-------------|-------------|-----------|---|-----------|---|
|            |             |             | 第1<br>ワード |   | 第2<br>ワード |   |
| TB         | Rn.bit      |             | A         | n | bit       | 1 |

## XOR R<sub>n</sub>, obj

排他的論理輪命令

### 機能

R<sub>n</sub> ← R<sub>n</sub> ^ obj

### 説明

- ・R<sub>n</sub> レジスタの内容と obj の内容の排他的論理和をとり、結果を R<sub>n</sub> レジスタに格納します。

### フラグ

| C | Z | S | OV | MIE | HC |
|---|---|---|----|-----|----|
| — | * | * | —  | —   | —  |

Z : 実行結果が 0 の時 1 になり、それ以外の時は 0 になります。

S : 実行結果の最上位ビットがセットされます。

— : 変化はありません。

### 命令フォーマット

| ニーモニック | 第1オペランド        | 第2オペランド        | 命令フォーマット |          |             |   |
|--------|----------------|----------------|----------|----------|-------------|---|
|        |                |                | 第1ワード    |          | 第2ワード       |   |
| XOR    | R <sub>n</sub> | R <sub>m</sub> | 8        | <i>n</i> | <i>m</i>    | 4 |
|        |                | #imm8          | 4        | <i>n</i> | <i>imm8</i> |   |

# 4付録

---

U16コアのインストラクション表を記載しています。

インストラクション表では、DSRプリフィックスコード指定は省略しています。  
各命令の詳細な記述については、第3章を参照して下さい。

## 4.1インストラクション表

### 演算命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード               |       | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|-------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC | 第1ワード               | 第2ワード |              |
| ADD    | Rn          | Rm          | *      | * | * | *  | *   | *  | 1000_nnnn_mmmm_0001 |       | 1            |
|        |             | #imm8       | *      | * | * | *  | *   | *  | 0001_nnnn_iiii_iiii |       | 1            |
| ADD    | ERn         | ERm         | *      | * | * | *  | *   | *  | 1111_nnn0_mmm0_0110 |       | 1            |
|        |             | #imm7       | *      | * | * | *  | *   | *  | 1110_nnr0_1iii_iiii |       | 1            |
| ADDC   | Rn          | Rm          | *      | * | * | *  | *   | *  | 1000_nnnn_mmmm_0110 |       | 1            |
|        |             | #imm8       | *      | * | * | *  | *   | *  | 0110_nnnn_iiii_iiii |       | 1            |
| AND    | Rn          | Rm          | *      | * |   |    |     |    | 1000_nnnn_mmmm_0010 |       | 1            |
|        |             | #imm8       | *      | * |   |    |     |    | 0010_nnnn_iiii_iiii |       | 1            |
| CMP    | Rn          | Rm          | *      | * | * | *  | *   | *  | 1000_nnnn_mmmm_0111 |       | 1            |
|        |             | #imm8       | *      | * | * | *  | *   | *  | 0111_nnnn_iiii_iiii |       | 1            |
| CMPC   | Rn          | Rm          | *      | * | * | *  | *   | *  | 1000_nnnn_mmmm_0101 |       | 1            |
|        |             | #imm8       | *      | * | * | *  | *   | *  | 0101_nnnn_iiii_iiii |       | 1            |
| MOV    | ERn         | ERm         | *      | * |   |    |     |    | 1111_nnn0_mmm0_0101 |       | 1            |
|        |             | #imm7       | *      | * |   |    |     |    | 1110_nnr0_0iii_iiii |       | 1            |
| MOV    | Rn          | Rm          | *      | * |   |    |     |    | 1000_nnnn_mmmm_0000 |       | 1            |
|        |             | #imm8       | *      | * |   |    |     |    | 0000_nnnn_iiii_iiii |       | 1            |
| OR     | Rn          | Rm          | *      | * |   |    |     |    | 1000_nnnn_mmmm_0011 |       | 1            |
|        |             | #imm8       | *      | * |   |    |     |    | 0011_nnnn_iiii_iiii |       | 1            |
| XOR    | Rn          | Rm          | *      | * |   |    |     |    | 1000_nnnn_mmmm_0100 |       | 1            |
|        |             | #imm8       | *      | * |   |    |     |    | 0100_nnnn_iiii_iiii |       | 1            |
| CMP    | ERn         | ERm         | *      | * | * | *  | *   | *  | 1111_nnn0_mmm0_0111 |       | 1            |
| SUB    | Rn          | Rm          | *      | * | * | *  | *   | *  | 1000_nnnn_mmmm_1000 |       | 1            |
| SUBC   | Rn          | Rm          | *      | * | * | *  | *   | *  | 1000_nnnn_mmmm_1001 |       | 1            |

### シフト命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード               |       | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|-------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC | 第1ワード               | 第2ワード |              |
| SLL    | Rn          | Rm          | *      |   |   |    |     |    | 1000_nnnn_mmmm_1010 |       | 1            |
|        |             | #width      | *      |   |   |    |     |    | 1001_nnnn_0www_1010 |       | 1            |
| SLLC   | Rn          | Rm          | *      |   |   |    |     |    | 1000_nnnn_mmmm_1011 |       | 1            |
|        |             | #width      | *      |   |   |    |     |    | 1001_nnnn_0www_1011 |       | 1            |
| SRA    | Rn          | Rm          | *      |   |   |    |     |    | 1000_nnnn_mmmm_1110 |       | 1            |
|        |             | #width      | *      |   |   |    |     |    | 1001_nnnn_0www_1110 |       | 1            |
| SRL    | Rn          | Rm          | *      |   |   |    |     |    | 1000_nnnn_mmmm_1100 |       | 1            |
|        |             | #width      | *      |   |   |    |     |    | 1001_nnnn_0www_1100 |       | 1            |
| SRCLC  | Rn          | Rm          | *      |   |   |    |     |    | 1000_nnnn_mmmm_1101 |       | 1            |
|        |             | #width      | *      |   |   |    |     |    | 1001_nnnn_0www_1101 |       | 1            |

## 第4章 付 錄

### ロード/ストア命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード               |                     | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|---------------------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC | 第1ワード               | 第2ワード               |              |
| L      | ERn         | [EA]        | *      | * |   |    |     |    | 1001_nnn0_0011_0010 |                     | 1            |
|        |             | [EA+]       | *      | * |   |    |     |    | 1001_nnn0_0101_0010 |                     | 1            |
|        |             | [ERm]       | *      | * |   |    |     |    | 1001_nnn0_mmn0_0010 |                     | 1            |
|        |             | Disp16[ERm] | *      | * |   |    |     |    | 1010_nnr0_mmn0_1000 | DDDD_DDDD_DDDD_DDDD | 2            |
|        |             | Dispδ[BP]   | *      | * |   |    |     |    | 1011_nnr0_00DD_DDDD |                     | 2            |
|        |             | Dispδ[FP]   | *      | * |   |    |     |    | 1011_nnr0_01DD_DDDD |                     | 2            |
|        |             | Dadr        | *      | * |   |    |     |    | 1001_nnn0_0001_0010 | DDDD_DDDD_DDDD_DDDD | 2            |
| Rn     | Rn          | [EA]        | *      | * |   |    |     |    | 1001_nnnn_0011_0000 |                     | 1            |
|        |             | [EA+]       | *      | * |   |    |     |    | 1001_nnnn_0101_0000 |                     | 1            |
|        |             | [ERm]       | *      | * |   |    |     |    | 1001_nnnn_mmn0_0000 |                     | 1            |
|        |             | Disp16[ERm] | *      | * |   |    |     |    | 1001_nnnn_mmn0_1000 | DDDD_DDDD_DDDD_DDDD | 2            |
|        |             | Dispδ[BP]   | *      | * |   |    |     |    | 1101_nnnn_00DD_DDDD |                     | 2            |
|        |             | Dispδ[FP]   | *      | * |   |    |     |    | 1101_nnnn_01DD_DDDD |                     | 2            |
|        |             | Dadr        | *      | * |   |    |     |    | 1001_nnnn_0001_0000 | DDDD_DDDD_DDDD_DDDD | 2            |
| XRn    | XRn         | [EA]        | *      | * |   |    |     |    | 1001_nr00_0011_0100 |                     | 2            |
|        |             | [EA+]       | *      | * |   |    |     |    | 1001_nr00_0101_0100 |                     | 2            |
| QRn    | QRn         | [EA]        | *      | * |   |    |     |    | 1001_r000_0011_0110 |                     | 4            |
|        |             | [EA+]       | *      | * |   |    |     |    | 1001_r000_0101_0110 |                     | 4            |

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード               |                     | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|---------------------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC | 第1ワード               | 第2ワード               |              |
| ST     | ERn         | [EA]        |        |   |   |    |     |    | 1001_nnn0_0011_0011 |                     | 1            |
|        |             | [EA+]       |        |   |   |    |     |    | 1001_nnn0_0101_0011 |                     | 1            |
|        |             | [ERm]       |        |   |   |    |     |    | 1001_nnn0_mmn0_0011 |                     | 1            |
|        |             | Disp16[ERm] |        |   |   |    |     |    | 1010_nnr0_mmn0_1001 | DDDD_DDDD_DDDD_DDDD | 2            |
|        |             | Dispδ[BP]   |        |   |   |    |     |    | 1011_nnr0_10DD_DDDD |                     | 2            |
|        |             | Dispδ[FP]   |        |   |   |    |     |    | 1011_nnr0_11DD_DDDD |                     | 2            |
|        |             | Dadr        |        |   |   |    |     |    | 1001_nnn0_0001_0011 | DDDD_DDDD_DDDD_DDDD | 2            |
| Rn     | Rn          | [EA]        |        |   |   |    |     |    | 1001_nnnn_0011_0001 |                     | 1            |
|        |             | [EA+]       |        |   |   |    |     |    | 1001_nnnn_0101_0001 |                     | 1            |
|        |             | [ERm]       |        |   |   |    |     |    | 1001_nnnn_mmn0_0001 |                     | 1            |
|        |             | Disp16[ERm] |        |   |   |    |     |    | 1001_nnnn_mmn0_1001 | DDDD_DDDD_DDDD_DDDD | 2            |
|        |             | Dispδ[BP]   |        |   |   |    |     |    | 1101_nnnn_10DD_DDDD |                     | 2            |
|        |             | Dispδ[FP]   |        |   |   |    |     |    | 1101_nnnn_11DD_DDDD |                     | 2            |
|        |             | Dadr        |        |   |   |    |     |    | 1001_nnnn_0001_0001 | DDDD_DDDD_DDDD_DDDD | 2            |
| XRn    | XRn         | [EA]        |        |   |   |    |     |    | 1001_nr00_0011_0101 |                     | 2            |
|        |             | [EA+]       |        |   |   |    |     |    | 1001_nr00_0101_0101 |                     | 2            |
| QRn    | QRn         | [EA]        |        |   |   |    |     |    | 1001_r000_0011_0111 |                     | 4            |
|        |             | [EA+]       |        |   |   |    |     |    | 1001_r000_0101_0111 |                     | 4            |

**コントロールレジスタアクセス命令**

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | C Z S OV MIE HC | フラグの変化 |                     | 命令コード | 最小実行<br>サイクル |
|--------|-------------|-------------|-----------------|--------|---------------------|-------|--------------|
|        |             |             |                 | 第1ワード  | 第2ワード               |       |              |
| ADD    | SP          | #signed8    |                 |        | 1110_0001_iii_iii   |       | 1            |
| MOV    | ECSR        | Rm          |                 |        | 1010_0000_mmmm_1111 |       | 1            |
|        | ELR         | ERm         |                 |        | 1010_mmm0_0000_1101 |       | 1            |
|        | EPSW        | Rm          |                 |        | 1010_0000_mmmm_1100 |       | 1            |
|        | ERn         | ELR         |                 |        | 1010_nnn0_0000_0101 |       | 1            |
|        |             | SP          |                 |        | 1010_nnn0_0001_1010 |       | 1            |
|        | PSW         | Rm          | * * * * *       |        | 1010_0000_mmmm_1011 |       | 1            |
|        |             | #unsigned8  | * * * * *       |        | 1110_1001_iii_iii   |       | 1            |
|        | Rn          | ECSR        |                 |        | 1010_nnnn_0000_0111 |       | 1            |
|        |             | EPSW        |                 |        | 1010_nnnn_0000_0100 |       | 1            |
|        |             | PSW         |                 |        | 1010_nnnn_0000_0011 |       | 1            |
|        | SP          | ERm         |                 |        | 1010_0001_mmm0_1010 |       | 1            |

**PUSH/POP命令**

| ニーモニック | 第1<br>オペランド   | 第2<br>オペランド | C Z S OV MIE HC | フラグの変化 |                     | 命令コード | 最小実行<br>サイクル |
|--------|---------------|-------------|-----------------|--------|---------------------|-------|--------------|
|        |               |             |                 | 第1ワード  | 第2ワード               |       |              |
| PUSH   | ERn           |             |                 |        | 1111_nnr0_0101_1110 |       | 1            |
|        | QRn           |             |                 |        | 1111_r000_0111_1110 |       | 4            |
|        | Rn            |             |                 |        | 1111_nnnn_0100_1110 |       | 1            |
|        | XRn           |             |                 |        | 1111_nr00_0110_1110 |       | 2            |
|        | register_list |             |                 |        | 1111_lepa_1100_1110 |       | 1-6          |
| POP    | ERn           |             |                 |        | 1111_nnr0_0001_1110 |       | 1            |
|        | QRn           |             |                 |        | 1111_r000_0011_1110 |       | 4            |
|        | Rn            |             |                 |        | 1111_nnnn_0000_1110 |       | 1            |
|        | XRn           |             |                 |        | 1111_nr00_0010_1110 |       | 2            |
|        | register_list |             | * * * * *       |        | 1111_lepa_1000_1110 |       | 1-9          |

## 第4章 付 錄

### コプロセッサ転送

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | C Z S OV MIE HC | フラグの変化              |       | 命令コード | 最小実行<br>サイクル |
|--------|-------------|-------------|-----------------|---------------------|-------|-------|--------------|
|        |             |             |                 | 第1ワード               | 第2ワード |       |              |
| MOV    | CRn         | Rm          |                 | 1010_nnnn_mmmm_1110 |       |       | 1            |
|        | CERn        | [EA]        |                 | 1111_nnn0_0010_1101 |       |       | 1            |
|        |             | [EA+]       |                 | 1111_nnn0_0011_1101 |       |       | 1            |
|        | CRn         | [EA]        |                 | 1111_nnnn_0000_1101 |       |       | 1            |
|        |             | [EA+]       |                 | 1111_nnnn_0001_1101 |       |       | 1            |
|        | CXRn        | [EA]        |                 | 1111_nn00_0100_1101 |       |       | 2            |
|        |             | [EA+]       |                 | 1111_nn00_0101_1101 |       |       | 2            |
|        | CQRn        | [EA]        |                 | 1111_n000_0110_1101 |       |       | 4            |
|        |             | [EA+]       |                 | 1111_n000_0111_1101 |       |       | 4            |
|        | Rn          | CRm         |                 | 1010_nnnn_mmmm_0110 |       |       | 1            |
| [EA]   | [EA]        | CERm        |                 | 1111_mmm0_1010_1101 |       |       | 1            |
|        | [EA+]       | CERm        |                 | 1111_mmm0_1011_1101 |       |       | 1            |
|        | [EA]        | CRm         |                 | 1111_mmmm_1000_1101 |       |       | 1            |
|        | [EA+]       | CRm         |                 | 1111_mmmm_1001_1101 |       |       | 1            |
|        | [EA]        | CXRm        |                 | 1111_mm00_1100_1101 |       |       | 2            |
|        | [EA+]       | CXRm        |                 | 1111_mm00_1101_1101 |       |       | 2            |
|        | [EA]        | CQRm        |                 | 1111_m000_1110_1101 |       |       | 4            |
|        | [EA+]       | CQRm        |                 | 1111_m000_1111_1101 |       |       | 4            |

### EAレジスタ転送命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | C Z S OV MIE HC | フラグの変化              |       | 命令コード               | 最小実行<br>サイクル |
|--------|-------------|-------------|-----------------|---------------------|-------|---------------------|--------------|
|        |             |             |                 | 第1ワード               | 第2ワード |                     |              |
| LEA    | [ERm]       |             |                 | 1111_0000_mmm0_1010 |       |                     | 1            |
|        | Disp16[ERm] |             |                 | 1111_0000_mmm0_1011 |       | DDDD_DDDD_DDDD_DDDD | 2            |
|        | Dadr        |             |                 | 1111_0000_0000_1100 |       | DDDD_DDDD_DDDD_DDDD | 2            |

### ALU命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | C Z S OV MIE HC | フラグの変化 |       | 命令コード               | 最小実行<br>サイクル |
|--------|-------------|-------------|-----------------|--------|-------|---------------------|--------------|
|        |             |             |                 | 第1ワード  | 第2ワード |                     |              |
| DAA    | Rn          |             | *               | *      | *     | 1000_nnnn_0001_1111 | 1            |
| DAS    | Rn          |             | *               | *      | *     | 1000_nnnn_0011_1111 | 1            |
| NEG    | Rn          |             | *               | *      | *     | 1000_nnnn_0101_1111 | 1            |

**ビットアクセス命令**

| ニーモニック | 第1オペランド              | 第2オペランド | フラグの変化 |   |   |    |     |    | 命令コード               |                     | 最小実行サイクル |
|--------|----------------------|---------|--------|---|---|----|-----|----|---------------------|---------------------|----------|
|        |                      |         | C      | Z | S | OV | MIE | HC | 第1ワード               | 第2ワード               |          |
| SB     | <i>Rn.bit_offset</i> |         | *      |   |   |    |     |    | 1010_nnnn_0bbb_0000 |                     | 1        |
|        | <i>Dbitadr</i>       |         | *      |   |   |    |     |    | 1010_0000_1bbb_0000 | DDDD_DDDD_DDDD_DDDD | 2        |
| RB     | <i>Rn.bit_offset</i> |         | *      |   |   |    |     |    | 1010_nnnn_0bbb_0010 |                     | 1        |
|        | <i>Dbitadr</i>       |         | *      |   |   |    |     |    | 1010_0000_1bbb_0010 | DDDD_DDDD_DDDD_DDDD | 2        |
| TB     | <i>Rn.bit_offset</i> |         | *      |   |   |    |     |    | 1010_nnnn_0bbb_0001 |                     | 1        |
|        | <i>Dbitadr</i>       |         | *      |   |   |    |     |    | 1010_0000_1bbb_0001 | DDDD_DDDD_DDDD_DDDD | 2        |

**PSWアクセス命令**

| ニーモニック | 第1オペランド | 第2オペランド | フラグの変化 |   |   |    |     |    | 命令コード               |       | 最小実行サイクル |
|--------|---------|---------|--------|---|---|----|-----|----|---------------------|-------|----------|
|        |         |         | C      | Z | S | OV | MIE | HC | 第1ワード               | 第2ワード |          |
| EI     |         |         |        |   | * |    |     |    | 1110_1101_0000_1000 |       | 1        |
| DI     |         |         |        |   | * |    |     |    | 1110_1011_1111_0111 |       | 3        |
| SC     |         |         | *      |   |   |    |     |    | 1110_1101_1000_0000 |       | 1        |
| RC     |         |         | *      |   |   |    |     |    | 1110_1011_0111_1111 |       | 1        |
| CPLC   |         |         | *      |   |   |    |     |    | 1111_1110_1100_1111 |       | 1        |

**条件相対分岐命令**

| ニーモニック | 第1オペランド      | 第2オペランド | フラグの変化 |   |   |    |     |    | 命令コード               |       | 最小実行サイクル |
|--------|--------------|---------|--------|---|---|----|-----|----|---------------------|-------|----------|
|        |              |         | C      | Z | S | OV | MIE | HC | 第1ワード               | 第2ワード |          |
| BGE    | <i>Raddr</i> |         |        |   |   |    |     |    | 1100_0000_rrrr_rrrr |       | 1/3      |
| BLT    |              |         |        |   |   |    |     |    | 1100_0001_rrrr_rrrr |       | 1/3      |
| BGT    |              |         |        |   |   |    |     |    | 1100_0010_rrrr_rrrr |       | 1/3      |
| BLE    |              |         |        |   |   |    |     |    | 1100_0011_rrrr_rrrr |       | 1/3      |
| BGES   |              |         |        |   |   |    |     |    | 1100_0100_rrrr_rrrr |       | 1/3      |
| BLTS   |              |         |        |   |   |    |     |    | 1100_0101_rrrr_rrrr |       | 1/3      |
| BGTS   |              |         |        |   |   |    |     |    | 1100_0110_rrrr_rrrr |       | 1/3      |
| BLES   |              |         |        |   |   |    |     |    | 1100_0111_rrrr_rrrr |       | 1/3      |
| BNE    |              |         |        |   |   |    |     |    | 1100_1000_rrrr_rrrr |       | 1/3      |
| BEQ    |              |         |        |   |   |    |     |    | 1100_1001_rrrr_rrrr |       | 1/3      |
| BNV    |              |         |        |   |   |    |     |    | 1100_1010_rrrr_rrrr |       | 1/3      |
| BOV    |              |         |        |   |   |    |     |    | 1100_1011_rrrr_rrrr |       | 1/3      |
| BPS    |              |         |        |   |   |    |     |    | 1100_1100_rrrr_rrrr |       | 1/3      |
| BNS    |              |         |        |   |   |    |     |    | 1100_1101_rrrr_rrrr |       | 1/3      |
| BAL    |              |         |        |   |   |    |     |    | 1100_1110_rrrr_rrrr |       | 3        |

**符号拡張命令**

| ニーモニック | 第1オペランド | 第2オペランド | フラグの変化 |   |   |    |     |    | 命令コード                 |       | 最小実行サイクル |
|--------|---------|---------|--------|---|---|----|-----|----|-----------------------|-------|----------|
|        |         |         | C      | Z | S | OV | MIE | HC | 第1ワード                 | 第2ワード |          |
| EXTBW  | ERn     |         | *      | * |   |    |     |    | 1000_nnnn1_nnnr0_1111 |       | 1        |

## 第4章 付 錄

### ソフトウェア割込み命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード<br>第1ワード      | 命令コード<br>第2ワード | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|----------------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC |                     |                |              |
| SWI    | #snum       |             |        |   | * |    |     |    | 1110_0101_00ii_iii  |                | 3            |
| BRK    |             |             |        |   |   |    |     |    | 1111_1111_1111_1111 |                | 7            |

### 分岐命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード<br>第1ワード      | 命令コード<br>第2ワード      | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|---------------------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC |                     |                     |              |
| B      | <i>Cadr</i> |             |        |   |   |    |     |    | 1111_gggg_0000_0000 | cccc_cccc_cccc_cccc | 2            |
|        |             | ERn         |        |   |   |    |     |    | 1111_0000_nnn0_0010 |                     | 2            |
| BL     | <i>Cadr</i> |             |        |   |   |    |     |    | 1111_gggg_0000_0001 | cccc_cccc_cccc_cccc | 2            |
|        |             | ERn         |        |   |   |    |     |    | 1111_0000_nnn0_0011 |                     | 2            |

### 乗除算命令

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード<br>第1ワード      | 命令コード<br>第2ワード | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|----------------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC |                     |                |              |
| MUL    | ERn         | Rm          |        | * |   |    |     |    | 1111_nnn0_mmmm_0100 |                | 9            |
| DIV    | ERn         | Rm          | *      | * |   |    |     |    | 1111_nnn0_mmmm_1001 |                | 17           |

### その他

| ニーモニック | 第1<br>オペランド | 第2<br>オペランド | フラグの変化 |   |   |    |     |    | 命令コード<br>第1ワード      | 命令コード<br>第2ワード | 最小実行<br>サイクル |
|--------|-------------|-------------|--------|---|---|----|-----|----|---------------------|----------------|--------------|
|        |             |             | C      | Z | S | OV | MIE | HC |                     |                |              |
| INC    | [EA]        |             |        | * | * | *  | *   |    | 1111_1110_0010_1111 |                | 2            |
| DEC    | [EA]        |             |        | * | * | *  | *   |    | 1111_1110_0011_1111 |                | 2            |
| RT     |             |             |        |   |   |    |     |    | 1111_1110_0001_1111 |                | 2            |
| RTI    |             |             | *      | * | * | *  | *   |    | 1111_1110_0000_1111 |                | 2            |
| NOP    |             |             |        |   |   |    |     |    | 1111_1110_1000_1111 |                | 1            |

## 改版履歷

---

## 改版履歴

| ドキュメント No.           | 発行日       | ページ   |       | 変更内容                                                                                                                                                                                                                                                         |
|----------------------|-----------|-------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |           | 改版前   | 改版後   |                                                                                                                                                                                                                                                              |
| FJUL-U16-100-INST-01 | 2012.5.31 | -     | -     | 正式版発行                                                                                                                                                                                                                                                        |
| FJUL-U16-100-INST-02 | 2012.7.31 | P3-11 | P3-11 | <ul style="list-style-type: none"> <li>・3.3.命令実行時間について</li> <li>(誤)条件1. RomWindows 領域のアクセスを行うと、n m (n :アクセスバイト数 m:1 バイトアクセス時のウェイト数)のウェイトサイクルが挿入されます。</li> <li>(正)条件1. RomWindows 領域のアクセスを行うと、n × m (n :アクセス回数 m:1 アクセスあたりのウェイト数)のウェイトサイクルが挿入されます。</li> </ul> |
|                      |           | P3-12 | P3-12 | <ul style="list-style-type: none"> <li>・ADD SP,#signed8 の最小実行サイクル</li> <li>(誤) 2 (正) 1</li> <li>・DIV の最小実行サイクル</li> <li>(誤) 16 (正) 17</li> </ul>                                                                                                             |
|                      |           | P3-17 | P3-17 | <ul style="list-style-type: none"> <li>・MUL 命令の最小実行サイクル</li> <li>(誤) 8 (正) 9</li> </ul>                                                                                                                                                                      |
|                      |           | P4-6  | P4-6  | <ul style="list-style-type: none"> <li>・MUL 命令の最小実行サイクル</li> <li>(誤) 8 (正) 9</li> <li>・DIV の最小実行サイクル</li> <li>(誤) 16 (正) 17</li> </ul>                                                                                                                       |