リアルタイムカーネル勉強会用資料
SH-3 プロセッサの概要

本田 晋也, 若林 隆行
豊橋技術科学大学 情報工学系
e-mail: {honda,takayuki}@ertl.ics.tut.ac.jp

1 SuperH シリーズ

SuperH（以下 SH）シリーズは日立によって開発された組込み用マイクロプロセッサである．新たな 32 ビットのプロセッサーを設計するのにあたって，コストを押さえながらも高い性能を持たせるため，過去の高機能な命令セットを持つプロセッサとは異なる新しい RISC プロセッサ[1]の考え方で設計された．RISC の考え方の中でも SH シリーズは 16 ビット固定長命令と 5 段のパイプラインを持つことで，1 命令あたりの処理時間が約 1 クロックという高いスループットを実現した．またシステムを構築する際に必要になる各種インタフェース等を内蔵しているため，コストパフォーマンスの高いシステムの構築が可能である．

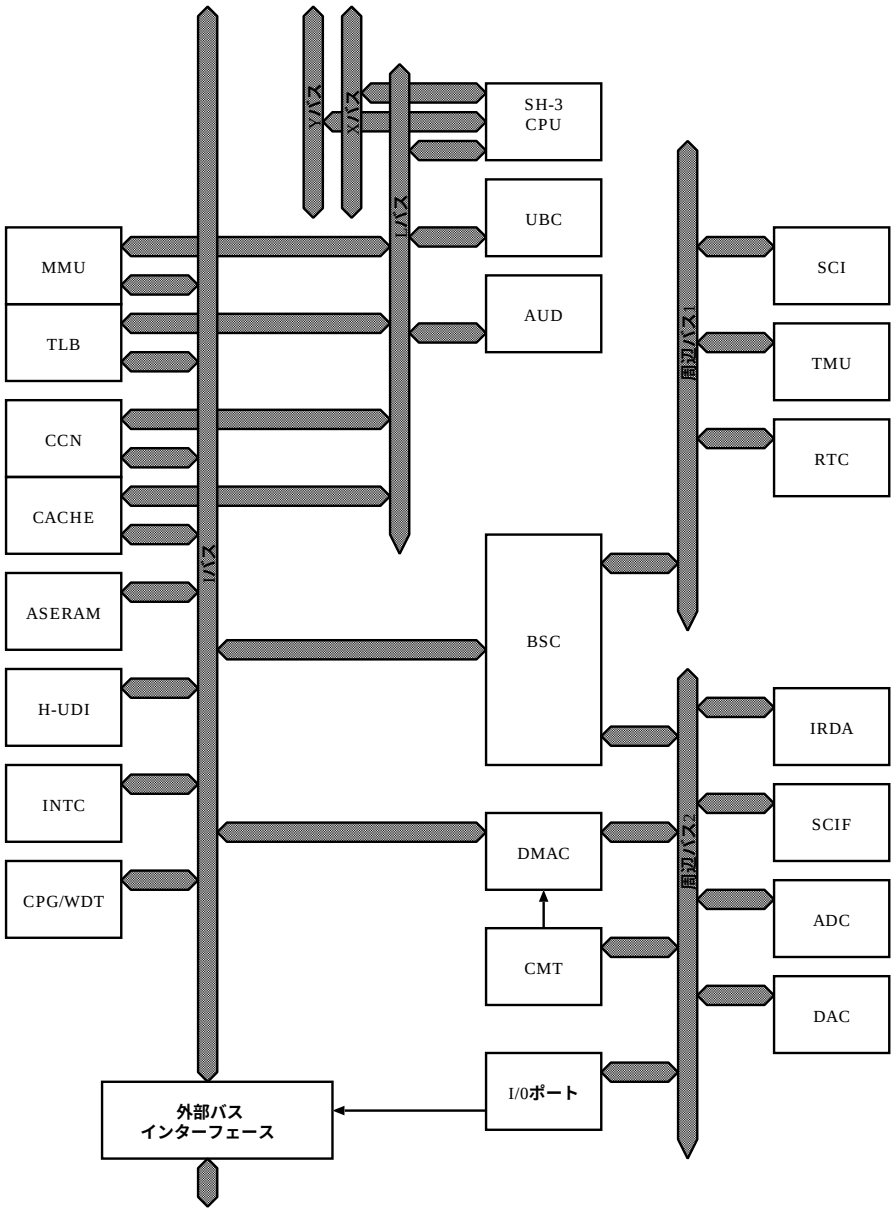
1.1 SH-3

SH-3 は，SuperH シリーズの第 3 世代目のプロセッサである．SH-1/2 と上位互換性のある命令セットを持ち，内蔵乗算機，メモリ管理機構 (Memory Management Unit, 以下 MMU)，キャッシュメモリなどを搭載している．MMU の搭載によって OS の実装が容易となり，WindowsCE ベースのターゲットや PDA 用のプロセッサとして，広く使用されている[2].

SH7709A は，SH7708 に Windows CE 向けの機能等を拡張した CPU である．具体的には，周辺機能の内蔵化，デバッグ機能 (JTAG 準拠) などがある．最大動作周波数は 133 MHz であるが，CPU の電源電圧を 1.8V とすることで，低消費電力を実現している．図1にブロック図を示す.

[1] Reduced Instruction Set Computer の略．単純な命令セットを高速で動作させることに主眼が置かれたプロセッサ．CISC (Complex Instruction Set Computer) の逆．機構が単純化しやすいため，高速化しやすいのが特徴．最近では外部 CISC/内部 RISC のプロセッサもある．

[2] SuperH シリーズで 44.8%のシェア (2000 年 トロン協会調べ)



ADC:	A/D 変換器	IRDA	シリアルコミュニケーションインターフェース (IRDA 付き)
ASERAM:	ASE メモリ	MMU:	メモリマネージメントユニット
AUD:	アドバンストユーザデバッガ	RTC:	リアルタイムクロック
BSC:	バスステートコントローラ	SCI:	シリアルコミュニケーションインターフェース (スマートカードインターフェース付き)
CACHE:	キャッシュメモリ		
CCN:	キャッシュメモリコントローラ	SCIF:	シリアルコミュニケーションインターフェース (FIFO 付き)
CMT:	コンペアマッチタイマ		
CPG/WDT:	クロック発信器/ウオッチドックタイマ	TLB:	タイマユニット
CPU:	中央演算処理装置	TMU:	タイマユニット
DAC:	D/A 変換器	UBC:	ユーザーブレークコントローラ
DMAC;	ダイレクトメモリアクセスコントローラ	H-UDI:	日立ユーザデバッグインターフェース
INTC:	割り込みコントローラ		

図 1: SH7709A のブロック図

2　SH-3（SH7709A）ハードウェア構造

2.1　メモリインターフェース機能

メモリ空間を七つのエリアに分割し，各エリアごとにメモリの種類やウェイトを制御することができる．1 つのエリアは最大 64M バイトの空間を持つ．物理空間割り付けを図2に，MMU を使用した際の論理アドレスマップを図3に示す．なお，4G バイトのメモリ空間のうち，2G バイトは特権モードでもユーザモードでもアクセスできる空間で，残りの 2G バイトは特権モードでのみアクセス可能な空間である．

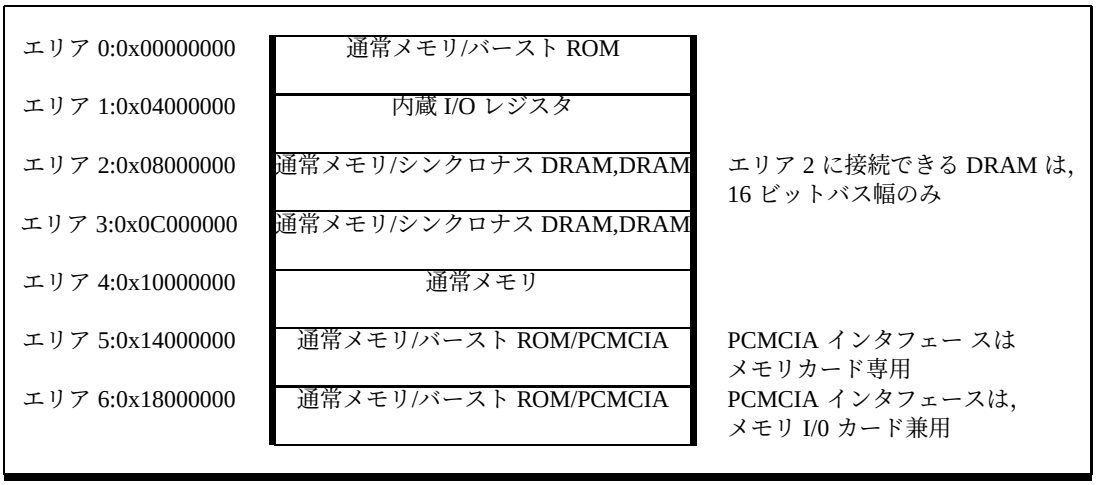


図 2: 物理空間の割り付け

図 3: 論理アドレスマップ

2.2 内部周辺機器

SH-3 SH7709A は以下の内部周辺機器をチップ内に持っており，これらのレジスタはメモリ空間にマッピングされている．マッピングを表1に示す.

表 1: 内部周辺機器のアドレスマップ

周辺機器名	アドレス	周辺機器名	アドレス	周辺機器名	アドレス
SCI	0xFFFFFE80	MMU	0xFFFFFFE0	PB	0xA4000102
SCIF	0xA4000150	CCR	0xFFFFFFEC	PC	0xA4000104
IrDA	0xA4000140	CCR2	0xA40000B0	PD	0xA4000106
TMU	0xFFFFFE90	DMAC	0xA4000060	PE	0xA4000108
TMU0	0xFFFFFE94	DMAC0	0xA4000020	PF	0xA400010A
TMU1	0xFFFFFEA0	DMAC1	0xA4000030	PG	0xA400010C
TMU2	0xFFFFFEAC	DMAC2	0xA4000040	PH	0xA400010E
RTC	0xFFFFFEC0	DMAC3	0xA4000060	PJ	0xA4000110
INTC	0xFFFFFEE0	CMT	0xA4000070	PK	0xA4000112
BSC	0xFFFFFF50	AD	0xA4000080	PL	0xA4000114
CPG	0xFFFFFF80	DA	0xA40000A0	SCP	0xA4000116
UBC	0xFFFFFF90	PA	0xA4000100	INT	0xA4000000

上記の内蔵周辺機器のレジスタは，上位 8 ビットが 0xA4 で始まる領域 (P2 領域)，または 上位 8 ビットが 0xFF で始まる領域 (制御領域 または P4 領域) に配置される．この領域には，次のような特徴がある．

- キャッシュ対象外（ノンキャッシャブル）
- MMU によるアドレス変換が行われない (物理アドレス=論理アドレス)
- ユーザモードでのアクセスが禁止される [3]

上位 8 ビットが 0xA4 で始まる領域は，上位 8 ビットが 0x04 で始まる領域 (P0 領域 エリア 1) のシャドウ[4]である．ただし，P0 領域は論理アドレス領域 かつ キャッシュ対象領域であるのに対し，P2 領域は物理アドレス領域 かつ キャッシュ対象外領域である．

以下に各内蔵機器の詳細について述べる．

- **CPG(クロック発振器)**
  CPU クロック (Iϕ)，バスクロック (CKIO)，周辺クロック (Pϕ) を生成する．発振機への入力クロックには，外部入力か水晶発振子かのいずれかが選択できる．
- **MMU(メモリ管理ユニット)**
  4G のアドレス空間をサポートする．ページ方式で，1 ページの大きさは 1k または 4K バイトを選択できる．なおリセット直後は OFF になっている．
- **CACHE(キャッシュメモリ)**
  キャッシュは 8k バイトで命令/データ混在，2k バイト×4 ウェイ・セットアソシアィブ方式[5]で，1 ラインは 16 バイトである．キャッシュがミスヒットすると，そのアドレスから 16 バイトの読み込みを行う．キャッシュへ

[3] SH-3 では 0x80000000 を超えるアドレスはユーザモードからアクセスできない

[4] SH-3 はアドレスの上位 3 ビットを出力しないため，アドレスの下位 29 ビットが同一であれば同じメモリをアクセスすることになる．このとき，アドレス空間に同一のメモリが複数配置されているように見える．この領域をシャドウと読んでいる

[5] キャッシュのエントリ位置に対し，主記憶のアドレスが一対 n 対応する (連想度が n である) 場合，n ウェイ・セットアソシアティブ方式と呼ばれる．特に一対一である場合はダイレクトマップ方式，n 対 n である場合はフルアソシアティブ方式と呼ばれる

の書き込みは，ライトスルー[6]またはコピーバック[7]の両方から選択できる．キャッシュは物理アドレス空間で動作する．アドレスの指定方法によりキャッシュ対象となる空間とキャッシュ非対象区間を区別している．また，MMU との組み合わせで，キャッシュ可能な区間であってもキャッシュしないページを設定することも可能である．キャッシュの設定はキャッシュ制御レジスタ (CCR) で行う．

- **INTC(割り込みコントローラ)**
  11 本の外部割り込み端子 (NMI,IRQ5-IRQ0,IRLA3-IRLA0) を持ち，内蔵周辺割り込みについてはモジュールごとに優先順位を設定できる．

- **UBC(ユーザーブレークコントローラ)**
  2 本のブレークチャンネルを持つ．ブレーク条件としてアドレス，データ値，アクセスタイプ，データサイズを設定可能．また，シーケンシャルブレーク機能をサポートしている．

- **BSC(バスステートコントローラ)**
  物理アドレス空間を最大 64MB の 6 つの領域 (エリア 0, エリア 2〜6) に分割し，各エリア毎にバスサイズ，ウェイトサイクル数等を設定可能．初期状態では，各エリアとも最大のウェイトが挿入されている．

- **H-UDI(日立ユーザーデバッグインタフェース)**

- **TMU(タイマユニット)**
  32 ビットのオートリロード方式の 3 チャンネルタイマでインターバルタイマとして利用することができる．1 チャンネルはインプットキャプチャの機能をもつ．カウント方式はダウンカウントである．

- **RTC(リアルタイムクロック)**
  アラーム機能を持つ時計で，CPU と別の 32.768kHz の水晶振動子で動作する．

- **SCI(シリアルコミュニケーションインターフェース)**
  調歩同期/クロック同期式のシリアルコミュニケーションインターフェースであり，スマートカードインターフェース機能を内蔵している．

- **WDT(ウオッチドッグタイマ)**
  システムのフェイルセーフ機能を実現するためのタイマで，タイマがオーバーフローするまでにカウンタをクリアしないとリセット要求を出す．リセット要求は，パワーオンとマニュアルの切り替えができるようになっている．

- **DMAC(DMA コントローラ)**
  4 チャンネルで，バスのアービトレーションにはバースト/サイクルスチール，チャンネル間の優先順位も固定/ラウンドロビンなどの選択が可能

- **ADC(A-D コンバータ)**
  10 ビットの変換制度で 8 チャンネルの入力を扱える．変換回路は逐次比較方式で 1 チャンネルあたり 134 クロックで変換が完了する．変換結果を格納するレジスタが 4 本あるので，連続で 4 チャンネルの変換が可能である．

- **DAC(ディジタル-アナログ コンバータ)**

- **SCIF(FIFO 付きシリアルコミュニケーションインターフェース)**

- **AUDSCIF(アドバンストユーザーデバッガ)**

[6]メモリ書込み時にはキャッシュもメモリも更新する方式
[7]メモリ書込み時にはキャッシュのみを更新し，エントリが解放される瞬間にメモリを更新する方式．ライトバックとも．

3 SH-3 プログラミングモデル

3.1 動作モード

CPU の動作モードには，特権モードとユーザモードの 2 種類がある[8]．動作モードは，ステータスレジスタの MD ビット (SR.MD[9]) によって変更できる．

特権モード (SR.MD=1:初期値) では，特殊命令 (LDC,STC など) が使用できることに加え，特殊レジスタ (SSR,SPC,R0_BANK1〜R7_BANK1 など)，0x8000 0000 を超える空間，内蔵周辺機器などへのアクセスが可能となる．

ユーザモード (SR.MD=0) では，特権モード時のみ利用可能なリソースの使用は禁止される．使用した場合，特権違反としてアドレスエラー例外が発生する (例外に関しては後述).

2 つのモードの切り替えは,

- **ユーザーモード → 特権モード**
  ソフトウェア例外発生命令 (TRAPA 命令) や，割込みなどの例外処理要求の発生により移行する．
- **特権モード → ユーザーモード**
  SSR.MD=0 の状態で RTE 命令を実行する．OS からユーザプログラムへ移行するような場合では，LDC 命令で直接 SR 等を操作するのでなく，復帰すべきユーザプログラムの開始番地を SPC，復帰時の SR を SSR に設定し，RTE 命令を実行することで切替を行う．

3.2 レジスタ構成

SH-3 は次に 3 種類のレジスタによって構成されている．括弧内は，特権モード時のみ使用可能なレジスタであることを意味する．また処理モード別のレジスタ構成を図4に示す.

- 汎用レジスタ : R0〜R15 (R0_BANK0〜R7_BANK0, R0_BANK1〜R7_BANK1, R8〜R15)
- 制御レジスタ : GBR, SR (SSR, SPC)
- システムレジスタ : MACH/MACL, RP, PC

3.2.1 汎用レジスタ

汎用レジスタは，R0〜R15 までの計 16 本用意されている．

R0 は汎用であるほか，インデックスレジスタとしての機能を持つ．一部の命令 (データ転送, 論理演算, ビット演算) は，GBR をベースレジスタ，R0 をインデックスレジスタとして，インデックス付 GBR 間接アドレッシング[10]が利用できる (アドレッシングモードに関しては後述)．また一部の命令 (加算, 比較, 論理演算, ビット操作) は，第 2 オペランドが R0 であるときに限り，第 1 オペランドに 8 ビット即値が利用できる．

16 本の汎用レジスタのうち R0〜R7 の 8 本は，BANK0 と BANK1 の 2 枚のレジスタバンクと成している．使用するバンクはステータスレジスタのレジスタバンクビット (SR.RB) によって切り替えることができる．R0〜R7 は，SR.RB=0 であるとき BANK0 が使用され，SR.RB=1 のときは BANK1 が使用される．ユーザモードから例外が発生した場合，自動的に SR.RB=1 となり，特権モード時は BANK1 が使用される．特権モードで実行している場合，現在使用していないバンクへのアクセスは，バンクを切り替えるか，制御レジスタ操作命令 (LDC,STC) を用いることで実現できる．ユーザモードで実行している場合，常に BANK0 のみが使用され，BANK1 へアクセスすることはできない．

[8]特権モードでは，ユーザモードでは使用が制限されている特権命令と呼ばれる特殊命令が利用でき，MMU の保護モードを独立して設定できるなどの利点がある．

[9]ステータスレジスタ (SR) の MD ビットを示すのに，以降 SR.MD と表記する

[10]@(R0,GBR) と表記．C 言語の*(GBR+R0) と等価

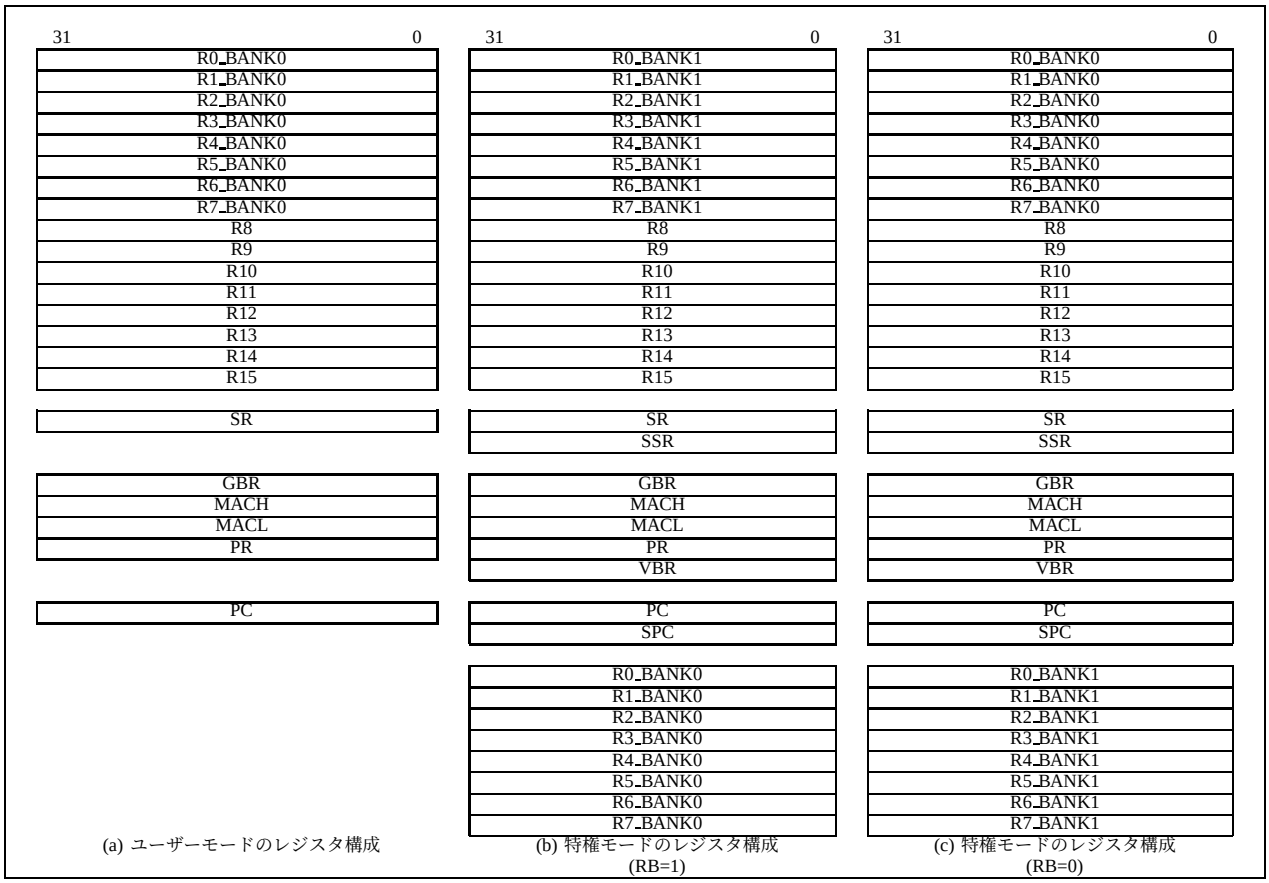


図 4: 処理モード別のレジスタ構成

汎用レジスタの一部にバンクを持つことで，例外処理の高速化が期待できる．通常，例外処理では作業領域を得るために汎用レジスタの待避/復帰処理を行うが，SH ではレジスタバンクを用いることで待避/復帰作業をハードウェアで行うため，処理にかかるオーバーヘッドが軽減される[11]．

SH ではスタックポインタは専用のレジスタとして用意していない．どの汎用レジスタをスタックポインタとして使用してもよいが，GNU C コンパイラは R15 をスタックポインタで使用する．参考資料として，GNU C Compiler(GCC) のレジスタ割当を示す．

表 2: GNU Compiler のレジスタ割当

レジスタ	機能
R0	返却値
R1 ··· R3	スクラッチレジスタ
R4 ··· R7	引数
R8 ··· R13	呼出先が待避すべき作業レジスタ
R14	フレームポインタ
R15	スタックポインタ

[11]ARM も同様の処理を行う．一方，Intel x86 や NEC V850 は一切レジスタを待避しない．中にはスタックポインタのみ切り替えるプロセッサもある

3.2.2 コントロールレジスタ

- **SR:ステータスレジスタ**

ステータスレジスタは，CPU の動作状態を示すいくつかのフラグビットによって構成される．具体的には，動作モード切り替えビット (MD)，例外処理を禁止するブロックビット (BL)，現在使用中のレジスタバンクを示す (RB)，条件付き分岐に使用するフラグ (T) と，15 レベルの割り込みマスク 4 ビット (I) と，積和演算時のオーバーフローを制御する (S)，除算時のフラグ (M,Q) などがある．

31	30	29	28	27 ——— 10	9	8	7 3	2	1	0	1	0	
0	MD	RB	BL	0 ——— 0	M	Q	I3 I2 I1 I0	0	0	S	T		SR

MD	処理モードビット 処理モードを表示 “1” のとき特権モード，“0” のときユーザーモード 例外，割り込みが発生すると “1” になる． リセットで “1” に初期化される．
RB	レジスタバンクビット R0〜R7 レジスタはバンクレジスタ 特権モードで使用する汎用レジスタ R0〜R7 を決める． “1” のとき，R0_BANK1〜R0_BANK1 と R8〜R15 が汎用レジスタになり， R0_BANK0〜R7_BANK0 は LDC/STC 命令でアクセスできる． “0” のとき，R0_BANK0〜R0_BANK0 と R8〜R15 が汎用レジスタになり， R0_BANK1〜R7_BANK1 は LDC/STC 命令でアクセスできる． 例外，割り込みが発生すると”1” になる． リセットで”1” に初期化される．
BL	ブロックビット “1” のとき，例外，割り込みの発生を抑止する． “0” のときは，例外，割り込みを受け付ける 例外，割り込みが発生すると 1 になる． リセットで”1” に初期化される
M,Q ビット	DIVOS/U,DIVI 命令で使用する．
I3〜I0 ビット	割り込みマスクビット 割り込み要求マスクレベルを表す 4 ビットデータ 割り込みが発生しても，割り込み受け付けレベルに変化しない． リセットで”1111” に初期化される
S ビット	MAC 命令で使用する．
T ビット	MOVT,CMP/COND,TAS,TST,BT,BF,SETT,CLRT, および DT 命令 で真 (1) または偽 (0) を表す． ADDV/C,SUBV/C,DIVOU/S,DIVI,NEGC,SHAR/L,SHLR/L, ROTR/L, および ROTCR/L 命令でキャリ，ボロー，オーバフローまたは アンダフローを表す
0 ビット	常に”0” が読み出される．書き込む値も常に”0” にすること

図 5: SH-3 のステータスレジスタ

- **GBR:グローバルベースレジスタ**

GBR は，GBR 間接アドレッシング時のベースアドレスを保持する．GBR 間接アドレッシングには，インデックス値として R0 または 8 ビット即値 (ゼロ拡張) を使用できる．論理演算, およびビットテスト命令では，GBR 間接アドレッシングを利用して，直接メモリ操作を行うことができる．

- **VBR:ベクタベースレジスタ**

割り込みなどの例外処理要求時に参照されるベクタアドレスのベースアドレスを格納する．

- **SPC:退避プログラムカウンタ**

例外が発生した際，復帰先の命令のアドレス (PC) を保持する

- **SSR:退避ステータスレジスタ**
  例外処理発生時，SR を待避するレジスタ.

3.2.3 システムレジスタ

- **MACH,MACL:積和上位, 下位レジスタ**
  乗算と積和の演算結果を保持するレジスタ．積和演算では MACH と MACL を合わせた 64 ビット，乗算では MACL の 32 ビットとして使用される．
- **PC:プログラムカウンタ**
  現在実行中のメモリの番地を格納する．
- **PR:プロシージャレジスタ**
  サブルーチンプロシージャに分岐する際，戻るべきアドレス (PC+2) を保持する．関数呼出などに利用する

3.3 メモリ上のデータ形式

データ長には，バイト (8 ビット)，ワード (16 ビット)，ロングワード (32 ビット) の 3 種類がある．

メモリに対してワード単位でアクセスする場合，データはワード境界 (2*n*) に配置しなければならない．同様に，ロングワード単位ではロングワード境界 (4*n*) に配置しなければならない．境界をまたぐようなアクセスを行った場合，アドレスエラー例外が発生する．

転送命令など，オペランドにディスプレースメントを指定した場合，オフセットはデータ形式のバイト数にあわせて 2 倍 または 4 倍された値となる．具体的には，ディスプレースメントが 10 であるとき，ワードアクセスでは 20 バイト先が，ロングワードアクセスでは 40 バイト先が参照される．

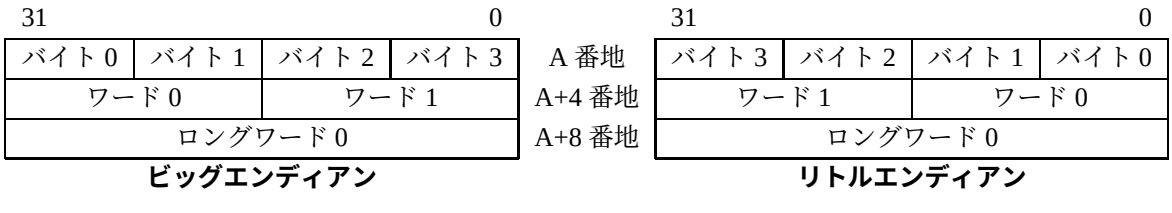
データフォーマットはビックエンディアン，リトルエンディアンのいずれかを選択可能であり，パワーオンリセット時に外部ピン (MD5 ピン) で設定する．

エンディアン

エンディアンは，CPU がデータバス幅と異なるサイズでメモリアクセスを行う際，メモリとレジスタをどのように対応付けるかを示している．多くのプロセッサはリトルエンディアンとビッグエンディアンの 2 つに分類される．

ビッグエンディアン 下位アドレスに位置するメモリの内容が，レジスタの上位に格納される

リトルエンディアン 上位アドレスに位置するメモリの内容が，レジスタの上位に格納される



4 SH-3 命令セット

SH は組み込み用途をターゲットしているため，豊富な周辺機能を内蔵している．周辺機器のレジスタはメモリ空間にマッピングされているため，メモリマップの操作以外の命令は存在しない．RISC のロード/ストアアーキテクチャの考え方では，I/O のビットを操作するのにもロード命令で読み出し，演算を行い，ストアするといった操作が必要である．組み込み用途ではこのような操作の頻度が高いため，SH では論理演算命令のみメモリを直接操作可能である．絶対アドレスでデータを参照するときには，あらかじめその値をメモリ上に配置しておき，レジスタにロードしてインデックス付き間接アドレッシングモードでオペラントにアクセスする．無条件分岐は常に遅延分岐を行うが，り，条件分岐は遅延スロットを作成するかどうかを選択できるようになっている[12]．

遅延分岐, 遅延スロット

SH-3 は 5 段のパイプラン (「命令読出し (Instruction Fetch : IF)」「命令解析 (Instruction Decode : ID)」「命令実行 (Execution : EX)」「メモリ操作 (Memory Access : MA)」「レジスタ書戻し (WriteBack : WB)」) を搭載している．そのため，ある命令を実行しているときには，すでに 2 命令先の命令を読み出している．
分岐命令のように PC が変化するような命令を発行した場合，分岐命令を実行する段階で後続する 2 命令 (IF,ID にある命令) が無効となるため，プロセッサのスループットが低下する．遅延分岐とは，プロセッサのスループットを高めるため，分岐直後の命令を無効にせず，実行を継続する．このような，直後の命令を常に実行するような分岐命令を**遅延分岐命令**とよび，条件によらず常に実行される命令が入る位置を**遅延スロット**と呼んでいる．

```
                    /* i = (i == 0) ? 1 : (i+1)*2; */
      mov.l     @R1, R0
      cmp.eq    #0, R0
      bt/s      1f
      add       #1, R0      ; この命令は分岐する前に必ず実行される
      shll1     R0
1:    mov.l     R0, @R1
```

分岐不条件成立時のパイプラインフロー

cmp.eq	IF	ID	EX	MA	WB				
bt		IF	ID	EX	MA	WB			
add			IF	ID	EX	MA	WB		
shll				IF	ID	EX	MA	WB	
mov.l					IF	ID	EX	MA	WB

分岐条件成立時のパイプラインフロー

cmp.eq	IF	ID	EX	MA	WB					
bt/s		IF	ID	EX	MA	WB				
add			IF	ID	EX	MA	WB			← 遅延スロット
shll				IF ×						
mov.l					IF	ID	EX	MA	WB	

[12]遅延スロットを作成しない場合，制御ハザードが発生し，1 ステージストールする．これは，NOP が遅延スロットに挿入されたのと等価な動作である．

4.1 命令コードおよびデータのサイズについて

- **機械語の命令コードのサイズ**
  命令コードは 16 ビット固定長であるため，メモリ上に置かれた機械語の命令コードは必ずワード境界 (偶数番地:2n 番地) に配置する必要がある．誤ってプログラムが奇数番地 (2n+1) にジャンプした場合には” 命令アドレスエラー” の例外が発生する．

- **メモリ上のデータサイズについて**
  命令セットが，メモリからロード/ストアするデータのサイズはバイト (8 ビット)，ワード (16 ビット)，ロングワード (32 ビット)，クワッドワード (64 ビット) のいずれかのサイズで行われる．

4.2 アドレッシングモード

SH がもつアドレッシングモードは,「レジスタ直接」,「レジスタ間接],「ポストインクリメントレジスタ間接」,「プリデクリメントレジスタ間接」,「ディスプレイスメント付きレジスタ間接」,「インデックス付きレジスタ間接」,「ディスプレイメント付き GBR 間接」,「インデックス付き GBR 間接」,「ディスプレイスメント付き PC 相対」,「PC 相対」,「イミディエイト」の 11 種類がある．表3にアドレッシングモードと実行アドレスを示す．表中の disp,imm は，それぞれディスプレースメントおよび即値を示し，後続する数値は設定可能なビット数である (disp:8→8 ビット長ディスプレースメント)．また u は値がゼロ拡張されることを示す．

表 3: アドレッシングモードと実行アドレス

アドレッシングモード	命令フォーマット	等価な C 表記	主な用途
レジスタ直接	Rn	Rn	
レジスタ間接	@Rn	*(Rn)	
ポストインクリメント レジスタ間接	@Rn+	*(Rn++)	スタック操作 積和演算
プリデクリメント レジスタ間接	@-Rn	*(–Rn)	スタック操作
ディスプレイスメント付き レジスタ間接	@(disp:4u,Rn)	*(Rn + (unsigned)disp)	テーブル参照
インデックス付き レジスタ間接	@(R0,Rn)	*((char *)Rn + R0)	テーブル参照
ディスプレイスメント 付き GBR 間接	@(disp:8u,GBR)	*(GBR+(unsigned)disp)	
インデックス付き GBR 間接	@(R0,GBR)	*((char *)GBR + R0)	
ディスプレイスメント 付き PC 相対	@(disp:8u,PC)	*((PC & 0xfffffffc) + (unsigned)disp)	即値読出し
PC 相対	disp:8/12 Rn Rn	PC = PC + (signed)disp * 2 PC = Rn PC += Rn	条件分岐など JMP 命令など BSR 命令など
即値	#imm:8	imm	

4.2.1 ディスプレイスメント

ディスプレイスメントは，PC 相対が符合拡張により 32 ビット符合付き 2 進数，PC 相対以外がゼロ拡張により 32 ビット値に変換された後，実行アドレスの計算に使用される．また，ディスプレイメントは，アクセスするデータサイズにしてベースアドレスから何個離れているかを示す．そのため 32 ビット化された後，アクセスするデータサイズ (1/2/4 のいずれかの値) が乗算されそれをベースアドレスに加算される．

4.2.2 即値

即値は 8 ビットであるため 32 ビットに拡張される．

論理演算命令 (TST,AND,OR,XOR) と TRAP 命令ではゼロ拡張し，その他の命令では (MOV,ADD,CMP/EQ) はイミディエイト値を符合拡張し 32 ビット値にする．

4.2.3 命令

SH-3 は 68 の命令を持っており，固定小数点転送命令，算術演算命令，論理演算命令，シフト命令，分岐命令，システム制御命令の 6 つに分類される．

- **固定小数点転送命令**
  固定小数点命令は表6に示すような命令がある．固定小数点命令では，メモリ上のバイトをやワードの値をロードする場合，メモリ上の値を 2 の補数で表された 2 進整数として扱い，符合が拡張され，汎用レジスタに格納される．

- **算術演算命令**
  表7に算術演算命令を示す．算術命令は，イミディエイトと MAC 命令を除いた，すべての命令がレジスタ-レジスタ間での演算となる．除算命令は 2 つの命令を組み合わせることで 1 つの除算を行う．まず，符合付きの場合は DIV0，符合なしの場合は DIV0U の命令を実行し，次に必要なビット数分だけ DIV1 命令を実行する．DIV1 命令は 1 ステップ除算命令である．1 ステップ除算とは，被除数を左に 1 ビットシフトし，それから除数を減算し，結果の正負によって商のビットを Q ビットに反映するという処理を実行する．乗算命令では，結果を汎用レジスタではなく MAC(結果が 32 ビットの場合は MACL, 結果が 64 ビットの場合は MACH と MACL を連結した 64 ビット) のレジスタ上に格納する．また，メモリ上のデータを直接乗算し，その結果を MAC レジスタ (MACH と MACL を連結した 64 ビット値) に加算する MAC 命令などもある．

- **論理演算命令とシフト命令**
  表8に論理演算命令，表9にシフト命令を示す．論理演算命令では，レジスタ-レジスタ間，あるいはイミディエイト値とレジスタの演算が基本だが，ディスプレイスメント付き GBR 間接では，イミディエイト値とメモリ上のバイトデータとの間で直接演算する命令も用意されている．

- **分岐命令**表10に分岐命令を示す．“BF,BT” を除く分岐命令はは RISC 特有の遅延分岐となり，分岐命令直後の命令を必ず実行する．この先に実行される分岐命令の次の 1 命令のことを，遅延スロットと呼ぶ．遅延スロットとしては以下の命令は実行できないため注意が必要である．

```
JMP     JSR     BRA     BRAF    BSR     BSRF    RTS     RTE
BT      BF      BT/S    BF/S    TRAPA
LDC     Rm, SR
LDC.L   @Rm+, SR
```

- **システム制御命令**
  表11にシステム制御命令を示す．システム制御命令は，CPU 自体をコントロールする命令なのでユーザーモードで実行不可能な命令がある．

命令は，次のようなフォーマットを成す．

```
ラベル        ニーモニック   ソース,   ディスティネーション   ; コメント
example:      mov.l          R1,       R0                    ; 転送命令 R1 → R0
example_2:    cmp.eq         #0,       R0                    ;R0 が 0 かどうかを T ビットに入れる
example_3:    shll1                    R0                    ;R0 を 2 倍する
```

4.3 他の RISC プロセッサとの相違点

SH プロセッサは，独自アーキテクチャであることや 16 ビット固定長命令を持つなどの特徴のため，コードを記述するにあたり他のプロセッサとは異なる機種依存制約を持つ．本節では前述した制約を明確にする．

- **ワード/ロングワード即値の置き方**

  SuperH プロセッサは 16 ビット固定長命令を持つという特徴のため，16/32 ビット即値を扱うには独特のテクニックを要する (8 ビットまでの即値は命令に直接記述可能).

  16/32 ビット即値の読出しを行う場合には，プログラムに 32 ビット即値を埋め込み，ディスプレースメント付き PC 相対アドレッシングを利用して読み出す．以下に 32 ビット即値読出しの具体例を示す.

```
                ⋮
                mov.l    longvalue_p, R0 (※)
                ⋮
longvalue_p:    .long    0x12345678
                ※ gas ではラベル名を入れると PC 相対に変換する
```

  ディスプレースメント付き PC 相対アドレッシングでは，ディスプレースメントは 8 ビット値をゼロ拡張したものが利用される．そのため，即値を読み出す命令と，即値を格納したアドレスが 1024 バイト以上[13]離れてはならない.

- **条件分岐命令の条件設定**

  条件分岐命令は，分岐命令を示すビット，条件のトリガを識別するために必要なビット，分岐先を示すためのビットの 3 つから構成される．16 ビット固定長命令をもつ SH プロセッサでは，分岐条件となるトリガを多くした場合，必然的に分岐先を示すビット数が減るため，分岐できる範囲が狭くなる[14].

  この問題に対処するため，SH プロセッサでは条件分岐命令は T ビットのみをトリガとするりに，8 ビット長のディスプレースメントを指定可能となっている．しかし，8 ビット長 (前後 128 命令) でも不足する場合では，無条件分岐命令を利用することで 12 ビット長のディスプレースメント，またはレジスタ相対，レジスタ絶対による分岐が可能となる.

命令	飛べる範囲
BF,BF/S,BT,BT/S	-128 命令 から 127 命令
BRA, BSR	-2048 命令 から 2047 命令
BRAF, BSRF	全空間 (相対)
JMP, JSR	全空間 (絶対)

[13] ロングワードアクセス時．ワードアクセスの場合は 512 バイト以上離れてはならない

[14] 可変長命令をもつ Intel x86 プロセッサでは，分岐命令がニーモニックレベルで 63 種類もある．また 32bit 固定長命令の ARM プロセッサでは，全ての命令に 16 個のトリガ条件を入れることができる

とても離れた場所へのトランポリンコード

```
/* if(i==0) { とても大きなコード } else { とても大きなコード } */
                cmp.eq    #0, R0
                bt        if_true
                mov.l     if_false_p, R0
                jmp       @R0
                nop       /* 遅延スロット */
if_false_p:     .long     if_false
if_true:                          条件が真の時の処理
                ⋮
if_false:                         条件が偽の時の処理
                ⋮
```

5 SH-3 の例外処理

処理状態にはリセット状態，例外処理状態，バス権解放状態，プログラム実行状態，低消費電力状態の 5 種類がある．

5.1 例外処理

SH では割込みやリセット，アドレスエラーの発生，TRAP 命令など，PC の変更を行う要求を**例外処理**と呼ぶ．

プログラム実行中に例外が発生すると，プロセッサは次の動作を行う (大まかな共通処理のみを示す．個々の詳細については後述).

1. マスク可能な例外の発生を抑止し (SR.BL=1)， 特権モードへ移行する (SR.MD=1)
2. 現在のプログラムカウンタの値を，待避プログラムカウンタ (SPC) に待避する
3. 現在のステータスレジスタの値を，待避ステータスレジスタ (SSR) に待避する
4. 汎用レジスタ R0〜R7 を BANK1 に切り替える
5. 例外処理ルーチンを実行する

   リセット A000 0000h (絶対番地)

   内部要因 VBR+100h

   TLB ミスヒット VBR+400h

   NMI, 外部要因 VBR+600h

また，例外処理中に例外復帰命令 (RTE) を実行すると，プロセッサは次の処理を行う (共通処理のみを示す).

1. 待避プログラムカウンタの値を，プログラムカウンタに格納する
2. 待避ステータスレジスタの値を，ステータスレジスタに格納する (この際，SR.BL,SR.MD が 0 になる)

SR の BL ビットが 0 のとき，例外，割り込みを受け付ける．BL ビットが 1 のときに一般例外 (ダブルフォルト) が発生した場合は CPU の内部レジスタはリセットされ，他のモジュールのレジスタは，一般例外が発生前の内容を保持したままリセットが行われる．

5.1.1 例外処理に関するレジスタ

例外処理に関するレジスタは，TRAPA 例外レジスタ (TRA), 例外イベントレジスタ (EXPEVT), 割り込みイベントレジスタ (INTEVT), 割り込みイベントレジスタ 2(INTEVT2) の 4 つである．これらのレジスタは周辺モジュールレジスタであるため P4 領域に配置され，特権モードのときのみアドレスを指定してアクセスすることができる．

- **例外事象レジスタ (EXPEVT)**
  0xFFFFFFD4 番地に配置されていて，ビット 11〜0 の例外コード 12 ビットから構成されている．EXPEVT に設置される例外コードはリセットと一般例外事象による例外コードである．例外コードは例外発生時にハードウェアにより自動的に設定される．また，ソフトウェアからも変更が可能である．

- **割り込み事象レジスタ 2(INTEVT2)**
  0x04000000 番地に配置されていて，ビット 11〜0 の例外コード 12 ビットから構成されている．INVENT2 に設定される例外コードは，割り込み要求による例外コードである．例外コードは例外発生時にハードウェアにより自動的に設定される．

- **割り込み事象レジスタ (INTEVT)**
  0xFFFFFFD8 番地に配置されていて，ビット 11〜0 の例外コード 12 ビットか，または割り込み優先順位を示すコードを格納する．割り込み発生によりどちらがセットされるかは，割り込み要因により異なる．例外コード, 割り込み優先順位は例外発生時にハードウェアにより自動的に設定される．INTEVT はソフトウェアからも変更が可能である．

- **TRAPA 例外レジスタ (TRA)**
  0xFFFFFFD0 番地に配置されていて，ビット 9〜2 の TRAPA 命令の 8 ビットイミディエイトデータ (imm) から構成されている．TRA は TRAPA 命令実行時にハードウェアにより自動的に設定される．TRA はソフトウェアからも変更可能である．

5.1.2 例外処理ベクタアドレス

リセットの場合は 0xA0000000 に分岐する．それ以外は，例外処理の発生により SH -3 は VBR+オフセットのアドレスに直接ジャンプする．例外事象ベクタを表4に示す.

ベクタは，TLB ミス例外発生時は VBR+0x400, それ以外の例外処理は VBR+0x100 へ，割り込みは VBR+0x600 となっている．このアドレスに格納するのはジャンプ先となる処理ルーチンの先頭アドレスではなく，そこから命令を実行する．そのため例外の処理はプログラムにより発生要因を一般例外なら EXPEVT から，割り込みなら INTEVT を読み込み判断する必要がある．表5に各特定例外イベントを区別するため EXPEVT レジスタのビット 11〜0，または INTEVT,INTEVT2 レジスタに書き込まれる例外コードを示す．もう 1 つの例外レジスタ，TRAPA(TRA) レジスタは無条件トラップ (TRAPA 命令) の 8 ビットイミディエイトデータを保持するために使用する．

5.2 リセットスタート

リセット時は VBR を設定できないため実行開始アドレスは 0xA0000000 に固定されている．SH-3 はアドレスの上位 3 ビットを外部に出力していないため外部バスには 0x00000000 が出力される．リセット時の SR の初期値は, MD=1(特権モード), RB=1(レジスタバンク 1),BL=1(例外処理をブロック) となっている．この状態で割り込み以外の例外処理を要求するとリセットがかかる．また，スタックポインターとして使用する R15 はハードウェアで自動的に初期化されないためプログラムによって初期化する必要がある．

表 4: 例外事象ベクタ

例外タイプ	カレント命令	例外イベント	優先順位*1	例外順位	ベクタアドレス	ベクタオフセット
リセット	中断	パワーオン	1	-	0xA0000000	-
		マニュアルリセット	1	-	0xA0000000	-
		パワーオン	1	-	0xA0000000	-
		H-UDI リセット	2	-	0xA0000000	-
一般例外イベント	中断およびリトライ	CPU アドレスエラー (命令アクセス)	2	1	-	0x100
		TLB ミス	2	2	-	0x400
		TLB 無効 (命令アクセス)	2	3	-	0x100
		TLB 保護違反 (命令アクセス)	2	4	-	0x100
		予約命令コード例外	2	5	-	0x100
		スロット不当命令例外	2	5	-	0x100
		CPU アドレスエラー (データアクセス)	2	6	-	0x100
		TLB ミス (データアクセス)	2	7	-	0x400
		TLB 無効 (データアクセス)	2	8	-	0x100
		TLB 保護違反 (データアクセス)	2	9	-	0x100
		初期ページ書き込み	2	10	-	0x100
	完了	無条件トラップ (TRAPA 命令)	2	5	-	0x100
		ユーザブレークポイントトラップ	2	*n**2	-	0x100
一般割り込み要求	完了	DMA アドレスエラー	2	12	-	0x100
		ノンマスカブル割り込み	3	-	-	0x600
		外部ハードウェア割り込み	4*3	-	-	0x600
		H-UDI 割り込み	4*3	-	-	0x600

*1 優先順位は高い方から順番に示します．1 が最高で 4 が最低です
*2 ブレークポイントトラップはユーザーが定義できます．命令実行後のブレークポイントのとき 1，命令実行後のブレークポイントのとき 11，オペラントブレークポイントのときも 11 となります．
*3 ソフトウェアで外部ハードウェア割り込みと周辺モジュール割り込みの相対的優先順位を指定してください

表 5: 例外コード

例外タイプ	例外イベント	例外コード
リセット	パワーオン	0x000
	マニュアルリセット	0x020
	H-UDI リセット	0x000
一般例外イベント	TLB ミス/TLB 無効 (ロード)	0x040
	TLB ミス/TLB 無効 (ストア)	0x060
	初期ページ書き込み	0x080
	TLB 保護違反 (ロード)	0x0A0
	TLB 保護違反 (ストア)	0x0A0
	CPU アドレスエラー (ロード)	0x0E0
	CPU アドレスエラー (ストア)	0x100
	無条件トラップ (TRAPA 命令)	0x160
	予約命令コード例外	0x180
	スロット不当命令例外	0x1A0
	ユーザーブレークポイントトラップ	0x1E0
	DMA アドレスエラー	0x5C0
一般割り込み要求	ノンマスカブル割り込み (NMI)	0x1C0
	H-UDI 割り込み	0x5E0
	外部ハードウェア割り込み IRL3-IRL0=0000	0x200
	IRL3-IRL0=0001	0x220
	IRL3-IRL0=0010	0x240
	IRL3-IRL0=0011	0x260
	IRL3-IRL0=0100	0x280
	IRL3-IRL0=0101	0x2A0
	IRL3-IRL0=0110	0x2C0
	IRL3-IRL0=0111	0x2E0
	IRL3-IRL0=1000	0x300
	IRL3-IRL0=1001	0x320
	IRL3-IRL0=1010	0x340
	IRL3-IRL0=1011	0x360
	IRL3-IRL0=1100	0x380
	IRL3-IRL0=1101	0x3A0
	IRL3-IRL0=1110	0x3C0
TMU0	TMUI0(アンダーフロー)	0x400
TMU1	TMUI1(アンダーフロー)	0x420
TMU2	TMUI2(アンダーフロー)	0x440
	TMCPI2(インプットキャプチャ)	0x460
RTC	ATI(アラーム)	0x480
	PRI(周期)	0x4A0
	CUI(桁上げ)	0x4C0
SCI	ERI(受信エラー)	0x4E0
	RXI(受信データフロー)	0x500
	TXI(受信データエンプティ)	0x520
	TEI(送信完了)	0x540
WDT	ITI(インターバルタイマ)	0x560
REF	RCMI(コンペアマッチ)	0x580
	ROVI(リフレッシュカウントオーバフロー)	0x5A0

5.3　割り込み処理

割り込みが発生するとCPUはSRのビットを確認し0なら割り込み要求を受け付け，SRの割り込みマスクビットと割り込み優先度を比較する．割り込みがマスクされていなければ割り込み処理プログラムを実行する．SRのBLビットが1のとき割り込みが発生した場合には割り込み要求は保留され0にクリアされてから受け付けられる．

5.3.1　割り込み処理の流れ

割り込み処理は次のようになる．

1. PC,SRをそれぞれSPC,SSRに退避.
2. SRのBLを1にセット，全ての例外処理をブロックする．
3. SRのMDを1にセット，特権モードへ
4. SRのRBを1にセット，レジスタバンク1を使用
5. 例外要因を識別する例外コードをINTEVT,INTEVT2のビット11〜0に書き込む.
6. VBR+0x600からの例外処理ルーチンを実行

割り込み処理ルーチンはINTEVTレジスタを見て割り込み要因を判断してそれぞれの割り込み処理プログラムにジャンプさせる必要がある．

5.3.2　多重割り込み

前述のとおり，SHプロセッサは例外処理が開始されるとSR.BLを1にするが，SR. BLが1である場合には外部要因の割込みは発生しない．そのため，多重割り込みを実現するためには，例外処理ルーチンに次のような操作を加えなければならない.

- 割込み優先レベルに設定した値を割り込みマスクビット(SR.I3〜I0)に設定
- SSP,SPCの退避
- BLビットを0クリア

A　命令セット一覧

表 6: 固定小数点転送命令

命令ニーモニック		動作	特権	T ビット
MOV	#imm, Rn	imm →符号拡張	–	–
MOV.W	@(disp,PC), Rn	(disp × 2 + PC + 4) → 符号拡張→ Rn	–	–
MOV.L	@(disp,PC), Rn	(disp × 4 + PC& 0xFFFFFFFC + 4) → Rn	–	–
MOV	Rm, Rn	Rm → Rn	–	–
MOV.B	Rm, @Rn	Rm → (Rn)	–	–
MOV.W	Rm, @Rn	Rm → (Rn)	–	–
MOV.L	Rm, @Rn	Rm → (Rn)	–	–
MOV.B	@Rm, @Rn	(Rm) →符合拡張→ Rn	–	–
MOV.W	@Rm, @Rn	(Rm) →符合拡張→ Rn	–	–
MOV.L	@Rm, @Rn	(Rm) → Rn	–	–
MOV.B	Rm, @-Rn	Rn - 1 → Rn,Rm → Rn	–	–
MOV.W	Rm, @-Rn	Rn - 2 → Rn,Rm → Rn	–	–
MOV.L	Rm, @-Rn	Rn - 4 → Rn,Rm → Rn	–	–
MOV.B	@Rm+, Rn	(Rm) →符合拡張→ Rn,Rm + 1 → Rm	–	–
MOV.W	@Rm+, Rn	(Rm) →符合拡張→ Rn,Rm + 2 → Rm	–	–
MOV.L	@Rm+, Rn	(Rm) → Rn,Rm + 4 → Rm	–	–
MOV.B	R0, @(disp,Rn)	R0 → (disp + Rn)	–	–
MOV.W	R0, @(disp,Rn)	R0 → (disp × 2 + Rn)	–	–
MOV.L	Rm, @(disp,Rn)	R0 → (disp × 4 + Rn)	–	–
MOV.B	@(disp,Rm), R0	(disp + Rm) →符号拡張→ R0	–	–
MOV.W	@(disp,Rm), R0	(disp × 2 + Rm) →符号拡張→ R0	–	–
MOV.L	@(disp,Rm), Rn	(disp × 4 + Rm) → Rn	–	–
MOV.B	Rm, @(R0,Rn)	Rm → (R0 + Rn)	–	–
MOV.W	Rm, @(R0,Rn)	Rm → (R0 + Rn)	–	–
MOV.L	Rm, @(R0,Rn)	Rm → (R0 + Rn)	–	–
MOV.B	@(R0,Rm), Rn	(R0 + Rm) →符号拡張→ Rn	–	–
MOV.W	@(R0,Rm), Rn	(R0 + Rm) →符号拡張→ Rn	–	–
MOV.L	@(R0,Rm), Rn	(R0 + Rm) → Rn	–	–
MOV.B	R0, @(disp,GBR)	R0 → (disp + GBR)	–	–
MOV.W	R0, @(disp,GBR)	R0 → (disp × 2 + GBR)	–	–
MOV.L	R0, @(disp,GBR)	R0 → (disp × 4 + GBR)	–	–
MOV.B	@(disp,GBR), R0	(disp + GBR) →符号拡張→ R0	–	–
MOV.W	@(disp,GBR), R0	(disp × 2 + GBR) →符号拡張→ R0	–	–
MOV.L	@(disp,GBR), R0	(disp × 4 + GBR) → R0	–	–
MOVA	@(disp,PC), R0	disp × 4 + PC&0xFFFFFFFC + 4 → R0	–	–
MOVT	Rn	T → Rn	–	–
SWAP.B	Rm, Rn	Rm →下位 2 バイトの上下バイト交換→ Rn	–	–
SWAP.W	Rm, Rn	Rm →上下ワード交換→ Rn	–	–
XTRCT	Rm, Rn	Rm:Rn の中央 32 ビット→ Rn	–	–

表 7: 算術演算命令

命令ニーモニック		動作	特権	T ビット
ADD	Rm, Rn	Rn + Rm → Rn	–	–
ADD	#imm, Rn	Rn + imm → Rn	–	–
ADDC	Rm, Rn	Rn + Rm + T → Rn, キャリ→ T	–	キャリ
ADDV	Rm, Rn	Rn + Rm → Rn, オーバフロー→ T	–	オーバーフロー
CMP/EQ	#imm, R0	R0 = imm のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/EQ	Rm, Rn	Rn = Rm のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/HS	Rm, Rn	無符号で Rn ≥ Rm のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/GE	Rm, Rn	有符号で Rn ≥ Rm のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/HI	Rm, Rn	無符号で Rn > Rm のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/GT	Rm, Rn	有符号で Rn > Rm のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/PZ	Rn	Rn ≥ 0 のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/PL	Rn	Rn > 0 のとき 1 → T, それ以外のとき 0 → T	–	比較結果
CMP/STR		いずれかのバイトが等しいとき 1 → T, それ以外のとき 0 → T	–	比較結果
DIV1	Rm,Rn	1 ステップ除算 (Rn ÷ Rm)	–	計算結果
DIV0S	Rm,Rn	Rn の MSB → Q,Rm の MSB → M,M^Q → T	–	計算結果
DIV0U		0 → M/Q/T	–	0
DMULS.L	Rm, Rn	符合付きで Rn × Rm → MAC, 32 × 32 → 64 ビット	–	–
DMULU.L	Rm, Rn	符合なしで Rn × Rm → MAC, 32 × 32 → 64 ビット	–	–
DT	Rn	Rn - 1 → Rn, Rn が 0 のとき 1 → T,0 以外のとき 0 → T	–	比較結果
EXTS.B	Rm, Rn	Rm をバイトから符号拡張→ Rn	–	–
EXTS.W	Rm, Rn	Rm をワードから符号拡張→ Rn	–	–
EXTU.B	Rm, Rn	Rm をバイトからゼロ拡張→ Rn	–	–
EXTU.W	Rm, Rn	Rm をワードからゼロ拡張→ Rn	–	–
MAC.L	@Rm+,@Rn+	符合付きで (Rn)×(Rm) + MAC → MAC,Rn + 4 → Rn, Rm + 4 → Rm、32 × 32 + + 64 → 64 ビット	–	–
MAC.W	@Rm+,@Rn+	符合付きで (Rn)×(Rm) + MAC → MAC,Rn + 2 → Rn, Rm + 2 → Rm、16 × 16 + 64 → 64 ビット	–	–
MUL.L	Rm,Rn	Rn × Rm → MACL 32 × 32 → 32 ビット	–	–
MULS.W	Rm, Rn	符合付きで Rn × Rm → MACL 16 × 16 → 32 ビット	–	–
MULU.W	Rm, Rn	符合なしで Rn × Rm → MACL 16 × 16 → 32 ビット	–	–
NEG	Rm, Rn	0 - Rm → Rn	–	–
NEGC	Rm, Rn	0 - Rm - T → Rn, ボロー → T	–	ボロー
SUB	Rm, Rn	Rn - Rm → Rn	–	–
SUBC	Rm, Rn	Rn - Rm - T → Rn, ボロー → T	–	ボロー
SUBV	Rm, Rn	Rn - Rm → Rn, アンダーフロー → T	–	アンダーフロー

表 8: 論理演算命令

命令ニーモニック		動作	特権	T ビット
AND	Rm, Rn	Rn&Rm → Rn	–	–
AND	#imm, R0	R0&imm → R0	–	–
AND.B	#imm, @(R0,GBR)	(R0 + GBR)&imm → (R0 + GBR)	–	–
NOT	Rm, Rn	~Rm → Rn	–	–
OR	Rm, Rn	Rn | Rm → Rn	–	–
OR	#imm, R0	R0 | imm → R0	–	–
OR.B	#imm, @(R0, GBR)	(R0 + GBR) | imm → (R0 + GBR)	–	–
TAS.B	@Rn	(Rn) が 0 のとき 1 → T, それ以外の時 0 → T 両方に対して 1 → (Rn) の MSB	–	テスト結果
TAT	Rm, Rn	Rn&Rm, 結果が 0 のとき 1 → T, それ以外の時 0 → T	–	テスト結果
TAT	#imm, R0	R0&#imm, 結果が 0 のとき 1 → T, それ以外の時 0 → T	–	テスト結果
TAT.B	#imm, @(R0, GBR)	(R0 + GBR)&#imm, 結果が 0 のとき 1 → T, それ以外の時 0 → T	–	テスト結果
XOR	Rm, Rn	Rn∧Rm → Rn	–	–
XOR	#imm, R0	R0∧imm → R0	–	–
XOR.B	#imm, @(R0, GBR)	(R0 + GBR)∧imm → (R0 + GBR)	–	–

表 9: シフト命令

命令ニーモニック		動作	特権	T ビット
ROTL	Rn	T ← Rn ← MSB	–	MSB
ROTR	Rn	LSB → Rn → T	–	LSB
ROTCL	Rn	T ← Rn ← T	–	MSB
ROTCR	Rn	T → Rn → T	–	LSB
SHAD	Rm,Rn	Rm ≥ 0 のとき Rn ≪ Rm → Rn Rm < 0 のとき Rn ≫ Rm → (MSB → Rn)	– –	– –
SHAL	Rn	T ← Rn ← 0	–	MSB
SHAR	Rn	MSB → Rn → T	–	LSB
SHLD	Rm,Rn	Rm *geq* 0 のとき Rn ≪ Rm → Rn Rm ¡ 0 のとき Rn ≫ Rm → (0 → Rn)	– –	– –
SHLL	Rn	T ← Rn ← 0	–	MSB
SHLR	Rn	0 → Rn → T	–	LSB
SHLL2	Rn	Rn ≪ 2 → Rn	–	–
SHLR2	Rn	Rn ≫ 2 → Rn	–	–
SHLL8	Rn	Rn ≪ 8 → Rn	–	–
SHLR8	Rn	Rn ≫ 8 → Rn	–	–
SHLL16	Rn	Rn ≪ 16 → Rn	–	–
SHLR16	Rn	Rn ≫ 16 → Rn	–	–

表 10: 分岐制御命令

命令ニーモニック		動作	特権	T ビット
BF	label	T=0 のとき disp × + PC + 4 → PC,	–	–
		T=1 のとき nop	–	–
BF/S	label	遅延分岐、T=0 のとき disp × + PC + 4 → PC,	–	–
		T=1 のとき nop	–	–
BT	label	T=1 のとき disp × + PC + 4 → PC,	–	–
		T=0 のとき nop	–	–
BT/S	label	遅延分岐、T=1 のとき disp × + PC + 4 → PC,	–	–
		T=0 のとき nop	–	–
BRA	label	遅延分岐、disp × 2 + PC + 4 → PC	–	–
BRAF	Rn	遅延分岐、Rn + PC + 4 → PC	–	–
BSR	label	遅延分岐、PC + 4 → PR, disp × 2 + PC + 4 → PC	–	–
BSRF	Rn	遅延分岐、PC + 4 → PR, Rn + PC + 4 → PC	–	–
JMP	@Rn	遅延分岐、Rn → PC	–	–
JSR	@Rn	遅延分岐、PC + 4 → PR, Rn → PC	–	–
RTS		遅延分岐、PR → PC	–	–

表 11: システム制御命令

命令ニーモニック		動作	特権	T ビット
CLRMAC		0 → MACH,MACL	–	–
CLRS		0 → S	–	–
CLRT		0 → T	–	0
LDC	Rm, SR	Rm → SR	特権	LSB
LDC	Rm, GBR	Rm → GBR	–	–
LDC	Rm, VBR	Rm → VBR	特権	–
LDC	Rm, SRR	Rm → SSR	特権	–
LDC	Rm, SPC	Rm → SPC	特権	–
LDC	Rm, Rn_BANK	Rm → Rn_BANK(n=0〜7)	特権	–
LDC.L	@Rm+, SR	(Rm) → SR, Rm + 4 → Rm	特権	–
LDC.L	@Rm+, GBR	(Rm) → GBR, Rm + 4 → Rm	–	–
LDC.L	@Rm+, VBR	(Rm) → VBR, Rm + 4 → Rm	特権	–
LDC.L	@Rm+, SRR	(Rm) → SSR, Rm + 4 → Rm	特権	–
LDC.L	@Rm+, SPC	(Rm) → SPC, Rm + 4 → Rm	特権	–
LDC.L	Rm, Rn_BANK	(Rm) → Rn_BANK(n=0〜7),Rm + 4 → Rm	特権	–
LDS	Rm, MACH	Rm → MACH	–	–
LDS	Rm, MACL	Rm → MACL	–	–
LDS	Rm, PR	Rm → PR	–	–
LDS.L	@Rm+, MACH	(Rm) → MACH, Rm + 4 → Rm	–	–
LDS.L	@Rm+, MACL	(Rm) → MACL, Rm + 4 → Rm	–	–
LDS.L	@Rm+, PR	(Rm) → PR, Rm + 4 → Rm	–	–
LDTLB		PTEH/PTEL → TLB	特権	–
PREF	@Rn	(Rn) → オペラントキャッシュ	–	–
RTE		遅延分岐,SSR/SPC → SR/PC	特権	–
SETS		1 → S	–	–
SETT		1 → T	–	1
SLEEP		スリープもしくはスタンバイ	特権	–
STC	SR, Rn	SR → Rn	特権	–
STC	GBR, Rn	GBR → Rn	–	–
STC	VBR, Rn	VBR → Rn	特権	–
STC	SSR, Rn	SSR → Rn	特権	–
STC	SPC, Rn	SPC → Rn	特権	–
STC	Rm_BANK, Rn	Rm_BANK → Rn(m=0〜7)	特権	–
STC.L	SR, @-Rn	Rn - 4 → Rn,SR → (Rn)	特権	–
STC.L	GBR, @-Rn	Rn - 4 → Rn,GBR → (Rn)	–	–
STC.L	VBR, @-Rn	Rn - 4 → Rn,VBR → (Rn)	特権	–
STC.L	SSR, @-Rn	Rn - 4 → Rn,SSR → (Rn)	特権	–
STC.L	SPC, @-Rn	Rn - 4 → Rn,SPC → (Rn)	特権	–
STC.L	Rm_BANK, @-Rn	Rn - 4 → Rn,Rm_BANK → (Rn)(m=0〜7)	特権	–
STS	MACH, Rn	MACH → Rn	–	–
STS	MACL, Rn	MACL → Rn	–	–
STS	PR, Rn	PR → Rn	–	–
STS.L	MACH, @-Rn	Rn - 4 → Rn MACH → (Rn)	–	–
STS.L	MACL, @-Rn	Rn - 4 → Rn MACL → (Rn)	–	–
STS.L	PR, @-Rn	Rn - 4 → Rn PR → (Rn)	–	–
TRAPA	#imm	PC → SPC, SR → SSR #imm → TRA,0x160 → EXPEVT VBR + 0x100 → PC 1 → SR,MD/BL/RB	–	–