
AZ Processor
Data Sheet
Ver. 1.01

平成 24 年 10 月 22 日

目次

第 1 章	履歴	3
第 2 章	CPU	5
2.1	命令一覧	5
2.2	例外一覧	6
2.3	CPU 制御レジスタ	6
2.3.1	アドレスマップ	7
2.3.2	ビットマップ	7
第 3 章	Bus	11
3.1	仕様	11
3.2	バスマスタのマップ	11
3.3	バススレーブのアドレスマップ	11
第 4 章	メモリ	13
4.1	Boot ROM (Read Only Memory)	13
4.1.1	概要	13
4.1.2	アドレスマップ	13
4.2	スクラッチパッドメモリ	13
4.2.1	概要	13
4.2.2	アドレスマップ	13
第 5 章	Timer	15
5.1	概要	15
5.2	制御レジスタ	15
5.2.1	アドレスマップ	15
5.2.2	ビットマップ	16
第 6 章	UART	17
6.1	概要	17
6.2	制御レジスタ	17
6.2.1	アドレスマップ	17
6.2.2	ビットマップ	18
第 7 章	General Purpose I/O	19
7.1	概要	19
7.2	制御レジスタ	19
7.2.1	アドレスマップ	19
7.2.2	ビットマップ	20

付録 A Instruction Set Architecture 詳細	21
A.1 命令の形式	21
A.2 命令一覧	22
A.3 命令セットの詳細	24
A.3.1 論理演算命令	24
A.3.2 算術演算命令	25
A.3.3 シフト命令	27
A.3.4 分岐命令	29
A.3.5 メモリアクセス命令	30
A.3.6 システムコール命令	31
A.3.7 特権命令	32

1

履歴

Version: 1.00 **Date:** 2010/10/20

- 新規作成

Version: 1.01 **Date:** 2010/10/22

- 履歴を追加
- 例外ベクタに関する記述を修正
- ビットマップの誤りを修正
- アセンブリ言語による制御レジスタの記述方法を変更

2

CPU

2.1 命令一覧

表 2.1: 命令一覧

種類	命令	意味
論理演算 命令	ANDR	AND Register
	ANDI	AND Immediate
	ORR	OR Register
	ORI	OR Immediate
	XORR	XOR Register
	XORI	XOR Immediate
算術演算 命令	ADDSR	Add Signed Register
	ADDSI	Add Signed Immediate
	ADDUR	Add Unsigned Register
	ADDUI	Add Unsigned Immediate
	SUBSR	Subtract Signed Register
	SUBUR	Subtract Unsigned Register
シフト 命令	SHRLR	SHift Right Logical Register
	SHRLI	SHift Right Logical Immediate
	SHLLR	SHift Left Logical Register
	SHLLI	SHift Left Logical Immediate

表 2.2: 命令一覧(つづき)

種類	命令	意味
分岐命令	BE	Branch Equal
	BNE	Branch Not Equal
	BSGT	Branch Signed Grater Than
	BUGT	Branch Unsigned Grater Than
	JMP	JuMP
	CALL	CALL
メモリ命令	LDW	LoaD Word
	STW	STore Word
特殊命令	TRAP	Trap
特権命令	RDCR	ReaD Control Register
	WRCR	WRite Control Register
	EXRT	EXception ReTurn

2.2 例外一覧

表 2.3: 例外一覧

例外	例外コード
例外なし	0x0
外部割り込み	0x1
未定義命令	0x2
算術オーバフロー	0x3
アドレスミスアライン	0x4
トラップ	0x5
特権違反	0x6

CPU は例外が発生すると Cause レジスタに例外コードを格納し、例外ベクタにジャンプする。例外ベクタのアドレスは Exception Vector レジスタ(制御レジスタ 0x04)で決定する。

2.3 CPU 制御レジスタ

CPU の制御レジスタの読み出しは RDCR (ReaD Control Register) 命令、CPU の制御レジスタへの書き込みは WRCR (WRite Control Register) 命令を用いて行う。

2.3.1 アドレスマップ

表 2.4: 制御レジスタのアドレスマップ

名称	アドレス	アクセス	詳細
Status	0x00	R/W	ステータス
Previous Status	0x01	R/W	例外直前のステータス
Program Counter	0x02	R	プログラムカウンタ
Exception Program Counter	0x03	R/W	例外プログラムカウンタ
Exception Vector	0x04	R/W	例外ベクタ
Exception Code	0x05	R/W	例外コード
Interrupt Mask	0x06	R/W	割り込みマスク
Interrupt Request	0x07	R	割り込みリクエスト
ROM Size Information	0x1d	R	ROM サイズ情報
SPM Size Information	0x1e	R	SPM サイズ情報
CPU Information	0x1f	R	CPU 情報

2.3.2 ビットマップ

Status : ステータス

Access : Read/Write

Register Address: 0x00

31	<i>Reserved</i>		2	1	0
			IE	EM	
1	Interrupt Enable (IE)	Default: 0x0			
外部割り込み有効ビット．本ビットがセットされている場合外部割り込みが有効になる．					
0	Execution Mode (EM)	Default: 0x0			
現在実行モード．					
設定値	意味				
0	Kernel Mode				
1	User Mode				

第2章 CPU

Previous Status : 例外直前のステータス

Access : Read/Write

Register Address: 0x01

31			2	1	0
Reserved				IE	EM
1	Interrupt Enable (IE)	Default: 0x0			
例外が発生した場合直前の Interrupt Enable が格納される .					
0	Execution Mode (EM)	Default: 0x0			
例外が発生した場合直前の実行モードが格納される .					

Program Counter : プログラムカウンタ

Access : Read Only

Register Address: 0x02

31			2	1	0
PC				00	
31:2	Program Counter (PC)	Default: 0x0			
現在実行している命令のアドレスが格納される .					

Exception Program Counter : 例外プログラムカウンタ

Access : Read/Write

Register Address: 0x03

31			2	1	0
EPC				00	
31:2	Exception Program Counter (EPC)	Default: 0x0			
例外が発生した命令のアドレスが格納される .					

Exception Vector : 例外ベクタ

Access : Read/Write

Register Address: 0x04

31			2	1	0
Exception Vector (EXP_VEC)				00	
31:2	Exception Vector (EXP_VEC)	Default: 0x0			
例外ハンドラのアドレスを格納する . 例外が起きた際このアドレスに制御が移る .					

Exception Code : 例外コード

Access : Read/Write

Register Address: 0x05

31	<i>Reserved</i>	4 3 2 0	D CODE
----	-----------------	---------	--------

3	Delay Slot Flag (D)	Default: 0x0
---	---------------------	---------------------

遅延スロットで例外が起きた場合このビットがセットされる。

2:0	Exception Code (CODE)	Default: 0x0
-----	-----------------------	---------------------

例外が起きた際例外コードが格納される。

Interrupt Mask : 割り込みマスク

Access : Read/Write

Register Address: 0x06

31	<i>Reserved</i>	8 7	INT_MASK	0
----	-----------------	-----	----------	---

7:0	Interrupt Mask (INT_MASK)	Default: 0xFF
-----	---------------------------	----------------------

割り込みマスク。本ビットがセットされた場合該当する IRQ はマスクされる。MASK の n ビット目が IRQ の n 番に対応する。

Interrupt Request : 割り込みリクエスト

Access : Read Only

Register Address: 0x07

31	<i>Reserved</i>	8 7	IRQ	0
----	-----------------	-----	-----	---

7:0	Interrupt ReQuest(IRQ)	Default: 0x00
-----	------------------------	----------------------

外部からの割り込み要求信号。IRQ の n ビット目が IRQ の n 番に対応する。

ROM Size : ROM のサイズ

Access : Read Only

Register Address: 0x1d

31	ROM Size	0
----	----------	---

31:0	ROM Size	Default: 合成時に決定
------	----------	------------------------

実装されている ROM のサイズをバイト単位で表す。

Scratch Pad Memory Size : スクラッチパッドメモリのサイズ

Access : Read Only

Register Address: 0x1e

31

0

Scratch Pad Memory Size

31:0	Scratch Pad Memory Size	Default: 合成時に決定
------	-------------------------	-----------------

実装されているスクラッチパッドメモリのサイズをバイト単位で表す。

Release Information : リリース情報

Access : Read Only

Register Address: 0x1f

31

24 23

16 15

8 7

0

Year	Month	Version	Revision
------	-------	---------	----------

31:24	Year	Default: -
-------	------	------------

リリースされた年を表す。このレジスタ値に 1970 を足したものが西暦となる。

23:16	Month	Default: -
-------	-------	------------

リリースされた月を表す。

15:8	Version	Default: -
------	---------	------------

リリースされたバージョンを表す。

7:0	Revision	Default: -
-----	----------	------------

リリースされたリビジョンを表す。

3

Bus

3.1 仕様

表 3.1: バスの仕様

バス幅	32bit
バス形態	共有バス (マルチプレックス)
転送方式	クロック同期式
アービトレーション方式	ラウンドロビン
マスターのチャンネル数	4ch (内 2ch は CPU が使用)
スレーブのチャンネル数	8ch

3.2 バスマスタのマップ

バスマスタは 4 チャンネル接続可能である。以下にバスマスタのマップを示す。

表 3.2: バスマスタのマップ

チャンネル	割り当て
CH 00	CPU 0 (命令フェッチ)
CH 01	CPU 0 (データアクセス)
CH 02	<i>Reserved</i>
CH 03	<i>Reserved</i>

3.3 バススレーブのアドレスマップ

バススレーブは 8 チャンネル接続可能である以下にバススレーブのアドレスマップを示す。

表 3.3: バススレーブのアドレスマップ

チャンネル	アドレス	割り当て
CH 00	0x0000_0000 ~ 0x1fff_fff	ROM
CH 01	0x2000_0000 ~ 0x3fff_fff	スクラッチパッドメモリ(注1)
CH 02	0x4000_0000 ~ 0x5fff_fff	タイマ
CH 03	0x6000_0000 ~ 0x7fff_fff	UART
CH 04	0x8000_0000 ~ 0x9fff_fff	General Purpose I/O
CH 05	0xa000_0000 ~ 0xbfff_fff	<i>Reserved</i>
CH 06	0xc000_0000 ~ 0xdfff_fff	<i>Reserved</i>
CH 07	0xe000_0000 ~ 0xffff_fff	<i>Reserved</i>

(注1) スクラッチパッドメモリは CPU 内部に実装されているため CPU 以外のバスマスタからのアクセスは不可能。

4

メモリ

4.1 Boot ROM (Read Only Memory)

4.1.1 概要

Boot 用 ROM .

4.1.2 アドレスマップ

ベースアドレス : 0x0000_0000
サイズ : 4KByte

4.2 スクラッチパッド メモリ

4.2.1 概要

CPU が高速にアクセス可能なメモリ . CPU は専用バスを介して 1 クロックでアクセス可能 . スクラッチパッドメモリは 0x2000_0000 番地にアドレスマップされている . CPU 以外から 0x2000_0000 番地へアクセスした際の動作は保証しない .

4.2.2 アドレスマップ

ベースアドレス : 0x2000_0000
サイズ : 32KByte

5

Timer

5.1 概要

タイマは一定時間毎に割り込みを発生させるユニットである。タイマは CPU と同じクロックで動作する。

5.2 制御レジスタ

5.2.1 アドレスマップ

ベースアドレス : 0x4000_0000

表 5.1: 制御レジスタのアドレスマップ

名称	オフセット	アクセス	詳細
Control	0x0	R/W	コントロール
Interrupt	0x4	R/W	割り込み
Expiration	0x8	R/W	満了値
Counter	0xC	R/W	カウンタ

5.2.2 ビットマップ

Control : コントロール

Access : Read / Write

Offset: 0x0

31	2 1 0
<i>Reserved</i>	
	P S

1	Periodic (P)	Default: 0x0
この bit が 0 の場合タイマはワンショットタイマとして動作し、この bit が 1 の場合タイマはピリオディックタイマとして動作する。		
0	Start (S)	Default: 0x0
この bit に 1 がセットされた場合タイマが動作する。		

Interrupt : 割り込み

Access : Read / Write

Offset: 0x4

31	1 0
<i>Reserved</i>	
	I

0	Interrupt (I)	Default: 0x0
タイマが満了すると自動的にセットされる。この bit が 1 の場合割り込みが発生する。		

Expiration : 満了値

Access : Read / Write

Offset: 0x8

31	0
EXPR	

31:0	Expiration (EXPR)	Default: 0x0
カウンタがこの値に達した場合に割り込みを発生させる。		

Counter : カウンタ

Access : Read / Write

Offset: 0xC

31	0
COUNTER	

31:0	Counter (COUNTER)	Default: 0x0
カウンタが満了値に達した場合に割り込みを発生させる。		

6

UART

6.1 概要

Universal Asynchronous Receiver Transmitter (UART) は非同期シリアル通信を行う。

表 6.1: UART の仕様

ボーレート	:	合成時に設定 (Default:38400)
フロー制御	:	なし
パリティビット	:	なし
ストップビット	:	1 bit

6.2 制御レジスタ

6.2.1 アドレスマップ

ベースアドレス : 0x6000_0000

表 6.2: 制御レジスタのアドレスマップ

名称	オフセット	アクセス	詳細
Status	0x0	R/W	ステータス
Data	0x4	R/W	データ

6.2.2 ビットマップ

Status : ステータス

Access : Read / Write

Offset: 0x0

31	<i>Reserved</i>	2	1	0
			TxI	RxI

1	Transmit Interrupt (TxI)	Default: 0x0
---	--------------------------	---------------------

データの送信が完了した場合にセットされる。この bit が 1 の場合送信割り込みが発生する。

0	Receive Interrupt (RxI)	Default: 0x0
---	-------------------------	---------------------

データを受信した場合にセットされる。この bit が 1 の場合受信割り込みが発生する。

Data : データ

Access : Read / Write

Offset: 0x4

31	<i>Reserved</i>	8	7	0
			Data	

31:0	Data (DATA)	Default: 0x0
------	-------------	---------------------

このレジスタに書き込んだ場合データを送信する。このレジスタを読み出した場合受信データを読み出す。

7

General Purpose I/O

7.1 概要

General Purpose I/O (GPIO) はビット単位でのデジタル入出力を行う。GPIO には入力ポート、出力ポート、入出力 (双方向) ポートの 3 種類のポートがある。それぞれのポート数は合成時に決定する。デフォルトでは以下のようになっている。

表 7.1: デフォルトのポート数

ポートの種類	AZPR EvBoard
入力ポート	: 4ch (プッシュSW)
出力ポート	: 18ch (7 セグ LED , LED)
入出力ポート	: 16ch (V-Port)

制御レジスタの各ビットが GPIO の各チャンネルに対応している。

7.2 制御レジスタ

7.2.1 アドレスマップ

ベースアドレス : 0x8000_0000

表 7.2: 制御レジスタのアドレスマップ

名称	オフセット	アクセス	詳細
Input Port	0x0	R/W	入力ポート
Output Port	0x4	R/W	出力ポート
Inout Port	0x8	R/W	入出力ポート
Inout Direction	0xc	R/W	入出力方向

7.2.2 ビットマップ

Input Port : 入力ポート

Access : Read / Write Offset: 0x0

31 0

IN_DATA

31:0	Input Port Data (IN_DATA)	Default: 0x00000000
このアドレスを読み出すと入力ポートの値が読み出せる。		

Output Port : 出力ポート

Access : Read / Write Offset: 0x4

31 0

OUT_DATA

31:0	Output Port Data (OUT_DATA)	Default: 0x00000000
このアドレスに書き込むと出力ポートに値を出力する。このアドレスを読み出した場合は現在出力している値が読み出せる。		

Inout Port : 入出力ポート

Access : Read / Write Offset: 0x8

31 0

INOUT_DATA

31:0	Inout Port Data (INOUT_DATA)	Default: 0x00000000
入出力方向が出力の場合、このアドレスに書き込むと入出力ポートに値を出力する。入出力方向が出力の場合にこのアドレスを読み出した場合は現在出力している値が読み出せる。入出力方向が入力の場合、このアドレスを読み出すと入力ポートの値が読み出せる。		

Inout Direction : 入出力方向

Access : Read / Write Offset: 0xc

31 0

INOUT_DIR

31:0	Inout Direction (INOUT_DIR)	Default: 0x00000000
入出力ポートの方向を設定する。この bit が 0 の場合対応する bit は入力になり、1 の場合は出力になる。		

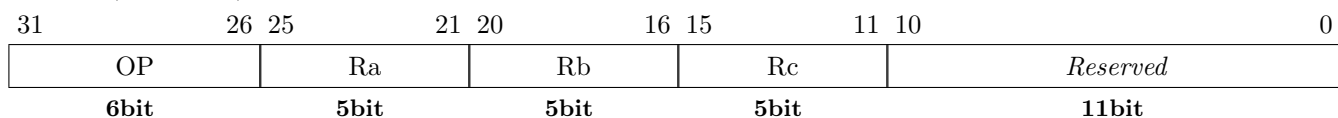
A

Instruction Set Architecture 詳細

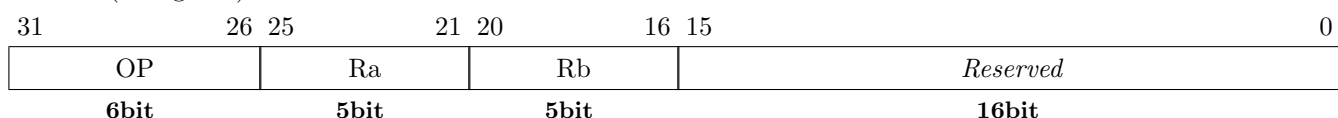
i

A.1 命令の形式

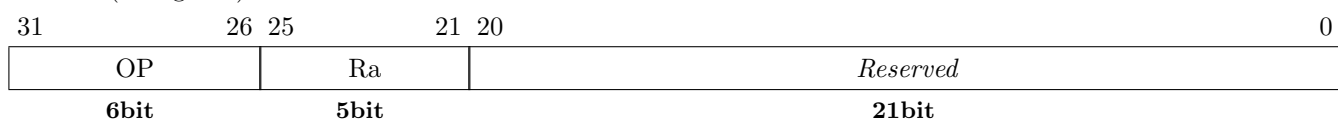
- R3(3 Register) 形式 : 3つのレジスタをオペランドに指定する形式



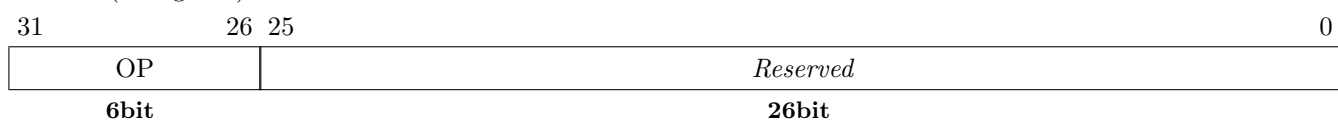
- R2(2 Register) 形式 : 2つのレジスタをオペランドに指定する形式



- R1(1 Register) 形式 : 1つのレジスタをオペランドに指定する形式



- R0(0 Register) 形式 : オペランドの無い形式



- R2I(2 Register & Immediate) 形式 : 2つのレジスタと 16bit の即値をオペランドに指定する形式

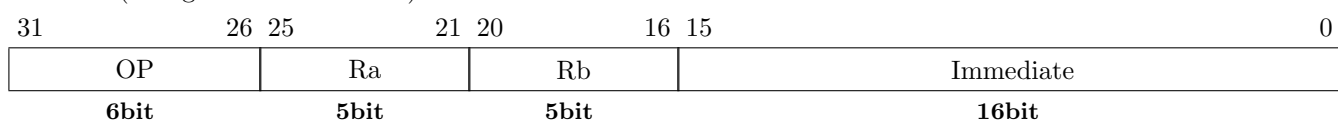


表 A.1: 命令のビットマップ

名前	位置	ビット幅	意味
OP (Opcode)	31:26	6bit	オペレーションコード
Ra (Register A)	25:21	5bit	レジスタ A のアドレス
Rb (Register B)	20:16	5bit	レジスタ B のアドレス
Rc (Register C)	15:11	5bit	レジスタ C のアドレス
Immediate	15:0	16bit	即値

A.2 命令一覧

表 A.2: 命令一覧

種類	命令	OP	形式	意味
論理演算 命令	ANDR	000000 (0x00)	R3	AND Register
	ANDI	000001 (0x01)	R2I	AND Immediate
	ORR	000010 (0x02)	R3	OR Register
	ORI	000011 (0x03)	R2I	OR Immediate
	XORR	000100 (0x04)	R3	XOR Register
	XORI	000101 (0x05)	R2I	XOR Immediate
算術演算 命令	ADDSR	000110 (0x06)	R3	Add Signed Register
	ADDSI	000111 (0x07)	R2I	Add Signed Immediate
	ADDUR	001000 (0x08)	R3	Add Unsigned Register
	ADDUI	001001 (0x09)	R2I	Add Unsigned Immediate
	SUBSR	001010 (0x0A)	R3	Subtract Signed Register
	SUBUR	001011 (0x0B)	R3	Subtract Unsigned Register

表 A.3: 命令一覧 (つづき)

種類	命令	OP	形式	意味
シフト命令	SHRLR	001100 (0x0C)	R3	SHift Right Logical Register
	SHRLI	001101 (0x0D)	R2I	SHift Right Logical Immediate
	SHLLR	001110 (0x0E)	R3	SHift Left Logical Register
	SHLLI	001111 (0x0F)	R2I	SHift Left Logical Immediate
分岐命令	BE	010000 (0x10)	R2I	Branch Equal
	BNE	010001 (0x11)	R2I	Branch Not Equal
	BSGT	010010 (0x12)	R2I	Branch Signed Grater Than
	BUGT	010011 (0x13)	R2I	Branch Unsigned Grater Than
	JMP	010100 (0x14)	R1	JuMP
	CALL	010101 (0x15)	R1	CALL
メモリ命令	LDW	010110 (0x16)	R2I	LoaD Word
	STW	010111 (0x17)	R2I	STore Word
システムコール命令	TRAP	011000 (0x18)	R0	Trap
特権命令	RDCR	011001 (0x19)	R2	ReaD Control Register
	WRCR	011010 (0x1a)	R2	WRite Control Register
	EXRT	011011 (0x1b)	R0	EXception ReTurn

A.3 命令セットの詳細

A.3.1 論理演算命令

ANDR AND Register : レジスタ同士の論理積 (R3 形式)

31	26 25	21 20	16 15	11 10	0
000000 (0x00)	Ra	Rb	Rc	000_0000_0000	
ANDR	Register A	Register B	Register C	<i>Reserved</i>	

書式 : `ANDR Ra, Rb, Rc`

動作 : `GPR[Rc] = GPR[Ra] AND GPR[Rb];`

例外 : No Exception

説明 : 汎用レジスタ Ra と Rb の論理積を Rc に格納する。

『 ANDR r0, r0, r0 (0x0000_0000) 』を NOP 命令として代用可能である。

用例 : `ANDR r1, r2, r3 // GPR[3] = GPR[1] AND GPR[2]`

ANDI AND Immediate : レジスタと定数の論理積 (R2I 形式)

31	26 25	21 20	16 15	0
000001 (0x01)	Ra	Rb	Immediate	
ANDI	Register A	Register B		

書式 : `ANDI Ra, Rb, Immediate`

動作 : `GPR[Rb] = GPR[Ra] AND ZeroExtention(Immediate);`

例外 : No Exception

説明 : 汎用レジスタ Ra とゼロ拡張した Immediate の論理積を Rb に格納する。

用例 : `ANDI r1, r2, 0xFFFF // GPR[2] = GPR[1] AND 0x0000FFFF`

ORR OR Register : レジスタ同士の論理和 (R3 形式)

31	26 25	21 20	16 15	11 10	0
000010 (0x02)	Ra	Rb	Rc	000_0000_0000	
ORR	Register A	Register B	Register C	<i>Reserved</i>	

書式 : `ORR Ra, Rb, Rc`

動作 : `GPR[Rc] = GPR[Ra] OR GPR[Rb];`

例外 : No Exception

説明 : 汎用レジスタ Ra と Rb の論理和を Rc に格納する。

用例 : `ORR r1, r2, r3 // GPR[3] = GPR[1] OR GPR[2]`

ORI OR Immediate : レジスタと定数の論理和 (R2I 形式)

31	26 25	21 20	16 15	0
000011 (0x03)	Ra	Rb	Immediate	
ORI	Register A	Register B		

書式 : $\text{ORI } Ra, Rb, Immediate$
 動作 : $\text{GPR}[Rb] = \text{GPR}[Ra] \text{ OR } ZeroExtention(Immediate)$;
 例外 : No Exception
 説明 : 汎用レジスタ Ra とゼロ拡張した Immediate の論理和を Rb に格納する。
 用例 : $\text{ORI } r1, r2, 0xFFFF // \text{GPR}[2] = \text{GPR}[1] \text{ OR } 0x0000FFFF$

XORR XOR Register : レジスタ同士の排他的論理和 (R3 形式)

31	26 25	21 20	16 15	11 10	0
000100 (0x04)	Ra	Rb	Rc	000_0000_0000	
XORR	Register A	Register B	Register C	<i>Reserved</i>	

書式 : $\text{XORR } Ra, Rb, Rc$
 動作 : $\text{GPR}[Rc] = \text{GPR}[Ra] \text{ XOR } \text{GPR}[Rb]$;
 例外 : No Exception
 説明 : 汎用レジスタ Ra と Rb の排他的論理和を Rc に格納する。
 用例 : $\text{XORR } r1, r2, r3 // \text{GPR}[3] = \text{GPR}[1] \text{ XOR } \text{GPR}[2]$

XORI XOR Immediate : レジスタと定数の排他的論理和 (R2I 形式)

31	26 25	21 20	16 15	0
000101 (0x05)	Ra	Rb	Immediate	
XORI	Register A	Register B		

書式 : $\text{XORI } Ra, Rb, Immediate$
 動作 : $\text{GPR}[Rb] = \text{GPR}[Ra] \text{ XOR } ZeroExtention(Immediate)$;
 例外 : No Exception
 説明 : 汎用レジスタ Ra とゼロ拡張した Immediate の排他的論理和を Rc に格納する。
 用例 : $\text{XORI } r1, r2, 0xFFFF // \text{GPR}[2] = \text{GPR}[1] \text{ XOR } 0x0000FFFF$

A.3.2 算術演算命令

ADDSR Add Signed Register : レジスタ同士の符号あり加算 (R3 形式)

31	26 25	21 20	16 15	11 10	0
000110 (0x06)	Ra	Rb	Rc	000_0000_0000	
ADDSR	Register A	Register B	Register C	<i>Reserved</i>	

書式 : `ADDSR Ra, Rb, Rc`
 動作 : `SUM = GPR[Ra] + GPR[Rb];`
 if (((GPR[Ra] >= 0) AND (GPR[Rb] >= 0) AND (SUM < 0)) OR
 ((GPR[Ra] < 0) AND (GPR[Rb] < 0) AND (SUM >= 0)))
 Exception(ArithmeticOverflow) ;
 else
 GPR[Rc] = SUM;
 例外 : 算術オーバーフロー
 説明 : 汎用レジスタ Ra と Rb を足した値を Rc に格納する。
 演算結果がオーバーフローした場合算術オーバーフロー例外を発生させる。
 用例 : `ADDSR r1, r2, r3 // GPR[3] = GPR[1] + GPR[2]`

ADDSI Add Signed Immediate : レジスタと定数の符号あり加算 (R2I 形式)

31	26 25	21 20	16 15	0
000111 (0x07)	Ra	Rb	Immediate	
ADDSI	Register A	Register B		

書式 : `ADDSI Ra, Rb, Immediate`
 動作 : `SUM = GPR[Ra] + SignExtention(Immediate) ;`
 if (((GPR[Ra] >= 0) AND (Immediate >= 0) AND (SUM < 0)) OR
 ((GPR[Ra] < 0) AND (Immediate < 0) AND (SUM >= 0)))
 Exception(ArithmeticOverflow) ;
 else
 GPR[Rb] = SUM;
 例外 : 算術オーバーフロー
 説明 : 汎用レジスタ Ra と符号拡張した Immediate を足した値を Rb に格納する。
 演算結果がオーバーフローした場合算術オーバーフロー例外を発生させる。
 用例 : `ADDSI r1, r2, 1 // GPR[2] = GPR[1] + 1`
 `ADDSI r1, r2, -1 // GPR[2] = GPR[1] - 1`

ADDUR Add Unsigned Register : レジスタ同士の符号なし加算 (R3 形式)

31	26 25	21 20	16 15	11 10	0
001000 (0x08)	Ra	Rb	Rc	000_0000_0000	
ADDUR	Register A	Register B	Register C	<i>Reserved</i>	

書式 : `ADDUR Ra, Rb, Rc`
 動作 : `GPR[Rc] = GPR[Ra] + GPR[Rb];`
 例外 : No Exception
 説明 : 汎用レジスタ Ra と Rb を足した値を Rc に格納する。
 演算結果がオーバーフローした場合も例外は発生しない。
 用例 : `ADDUR r1, r2, r3 // GPR[3] = GPR[1] + GPR[2]`

ADDUI Add Unsigned Immediate : レジスタと定数の符号なし加算 (R2I 形式)

31	26 25	21 20	16 15	0
001001 (0x09)	Ra	Rb	Immediate	
ADDUI	Register A	Register B		

書式 : ADDUI *Ra, Rb, Immediate*

動作 : $GPR[Rb] = GPR[Ra] + \text{SignExtention}(Immediate)$

例外 : No Exception

説明 : 汎用レジスタ Ra と符号拡張した Immediate を足した値を Rb に格納する。
演算結果がオーバーフローした場合も例外は発生しない。

用例 : ADDUI r1, r2, 1 // $GPR[2] = GPR[1] + 1$
ADDUI r1, r2, -1 // $GPR[2] = GPR[1] - 1$

SUBSR Subtract Signed Register : レジスタ同士の符号あり減算 (R3 形式)

31	26 25	21 20	16 15	11 10	0
001010 (0x0A)	Ra	Rb	Rc	000_0000_0000	
SUBSR	Register A	Register B	Register C	Reserved	

書式 : SUBSR *Ra, Rb, Rc*

動作 : $DIFF = GPR[Ra] - GPR[Rb];$
if ((($GPR[Ra] \geq 0$) AND ($GPR[Rb] < 0$) AND ($DIFF < 0$)) OR
($GPR[Ra] < 0$) AND ($GPR[Rb] \geq 0$) AND ($DIFF \geq 0$)))
Exception(ArithmeticOverflow);

else

$GPR[Rc] = DIFF;$

例外 : 算術オーバーフロー

説明 : 汎用レジスタ Ra から Rb を引いた値を Rc に格納する。
演算結果がオーバーフローした場合算術オーバーフロー例外を発生させる。

用例 : SUBSR r1, r2, r3 // $GPR[3] = GPR[1] - GPR[2]$

SUBUR Subtract Unsigned Register : レジスタ同士の符号なし減算 (R3 形式)

31	26 25	21 20	16 15	11 10	0
001011 (0x0B)	Ra	Rb	Rc	000_0000_0000	
SUBUR	Register A	Register B	Register C	Reserved	

書式 : SUBUR *Ra, Rb, Rc*

動作 : $GPR[Rc] = GPR[Ra] - GPR[Rb];$

例外 : No Exception

説明 : 汎用レジスタ Ra から Rb を引いた値を Rc に格納する。
演算結果がオーバーフローした場合も例外は発生しない。

用例 : SUBUR r1, r2, r3 // $GPR[3] = GPR[1] - GPR[2]$

A.3.3 シフト命令

SHRLR SHift Right Logical Register : レジスタ同士の論理右シフト (R3 形式)

31	26 25	21 20	16 15	11 10	0
001100 (0x0C)	Ra	Rb	Rc	<i>000_0000_0000</i>	
SHRLR	Register A	Register B	Register C	<i>Reserved</i>	

書式 : SHRLR *Ra, Rb, Rc*

動作 : $GPR[Rc] = GPR[Ra] \gg GPR[Rb]_{4\sim 0}$;
Shift In Value : 0

例外 : No Exception

説明 : 汎用レジスタ Ra を Rb の下位 5bit の値だけ右シフトし、Rc に格納する。

用例 : SHRLR r1, r2, r3 // $GPR[3] = GPR[1] \gg GPR[2]_{4\sim 0}$

SHRLI SHift Right Logical Immediate : レジスタと定数の論理右シフト (R2I 形式)

31	26 25	21 20	16 15	0
001101 (0x0D)	Ra	Rb	Immediate	
SHRLI	Register A	Register B		

書式 : SHRLI *Ra, Rb, Immediate*

動作 : $GPR[Rb] = GPR[Ra] \gg Immediate_{4\sim 0}$;
Shift In Value : 0

例外 : No Exception

説明 : 汎用レジスタ Ra を Immediate の下位 5bit の値だけ右シフトし、Rb に格納する。

用例 : SHRLI r1, r2, 0xF // $GPR[2] = GPR[1] \gg 0xF$

SHLLR SHift Left Logical Register : レジスタ同士の論理左シフト (R3 形式)

31	26 25	21 20	16 15	11 10	0
001110 (0x0E)	Ra	Rb	Rc	<i>000_0000_0000</i>	
SHLLR	Register A	Register B	Register C	<i>Reserved</i>	

書式 : SHLLR *Ra, Rb, Rc*

動作 : $GPR[Rc] = GPR[Ra] \ll GPR[Rb]_{4\sim 0}$;
Shift In Value : 0

例外 : No Exception

説明 : 汎用レジスタ Ra を Rb の下位 5bit の値だけ左シフトし、Rc に格納する。

用例 : SHLLR r1, r2, r3 // $GPR[3] = GPR[1] \ll GPR[2]_{4\sim 0}$

SHLLI SHift Left Logical Immediate : レジスタと定数の論理左シフト (R2I 形式)

31	26 25	21 20	16 15	0
001111 (0x0F)	Ra	Rb	Immediate	
SHLLI	Register A	Register B		

書式 : SHLLI $Ra, Rb, Immediate$
 動作 : $GPR[Rb] = GPR[Ra] \ll Immediate_{4\sim0}$;
 Shift In Value : 0
 例外 : No Exception
 説明 : 汎用レジスタ Ra を $Immediate$ の下位 5bit の値だけ左シフトし、 Rb に格納する。
 用例 : SHLLI $r1, r2, 0xF$ // $GPR[2] = GPR[1] \ll 0xF$

A.3.4 分岐命令

BE Branch Equal : レジスタ同士の符号あり比較 (==) による条件分岐 (R2I 形式)

31	26 25	21 20	16 15	0
010000 (0x10)	Ra	Rb	Immediate	
BE	Register A	Register B		

書式 : BE $Ra, Rb, Immediate$
 動作 : if ($GPR[Ra] == GPR[Rb]$)
 GOTO ($NextPC + SignExtention(Immediate)$) ;
 例外 : No Exception
 説明 : 汎用レジスタ Ra と Rb を比較し、 $Ra == Rb$ の場合分岐する。
 用例 : BE $r1, r2, LABEL$ // if ($GPR[1] == GPR[2]$) GOTO LABEL

BNE Branch Not Equal : レジスタ同士の符号あり比較 (!=) による条件分岐 (R2I 形式)

31	26 25	21 20	16 15	0
010001 (0x11)	Ra	Rb	Immediate	
BNE	Register A	Register B		

書式 : BNE $Ra, Rb, Immediate$
 動作 : if ($GPR[Ra] != GPR[Rb]$)
 GOTO ($NextPC + SignExtention(Immediate)$) ;
 例外 : No Exception
 説明 : 汎用レジスタ Ra と Rb を比較し、 $Ra != Rb$ の場合分岐する。
 用例 : BNE $r1, r2, LABEL$ // if ($GPR[1] != GPR[2]$) GOTO LABEL

BSGT Branch Signed Grater Than : レジスタ同士の符号あり比較 (<) による条件分岐 (R2I 形式)

31	26 25	21 20	16 15	0
010010 (0x12)	Ra	Rb	Immediate	
BSGT	Register A	Register B		

書式 : BSGT $Ra, Rb, Immediate$
 動作 : if ($GPR[Ra] < GPR[Rb]$)
 GOTO ($NextPC + SignExtention(Immediate)$) ;
 例外 : No Exception
 説明 : 汎用レジスタ Ra と Rb を符号ありで比較し、 $Ra < Rb$ の場合分岐する。
 用例 : BSGT $r1, r2, LABEL$ // if ($GPR[1] < GPR[2]$) GOTO LABEL

BUGT Branch Unsigned Grater Than : レジスタ同士の符号なし比較 (<) による条件分岐 (R2I 形式)

31	26 25	21 20	16 15	0
010011 (0x13)	Ra	Rb	Immediate	
BUGT	Register A	Register B		

書式 : BUGT Ra, Rb, Immediate

動作 : if (GPR[Ra] < GPR[Rb])

GOTO (NextPC + *SignExtention*(Immediate));

例外 : No Exception

説明 : 汎用レジスタ Ra と Rb を符号なしで比較し、Ra < Rb の場合分岐する。

用例 : BRU.GT r1, r2, LABEL // if (GPR[1] < GPR[2]) GOTO LABEL

JMP JuMP: レジスタ指定アドレスへの絶対分岐 (R1 形式)

31	26 25	21 20	0
010100 (0x14)	Ra	0_0000_0000_0000_0000	
JMP	Register A	<i>Reserved</i>	

書式 : JMP Ra

動作 : GOTO GPR[Ra]_{31~2} ;

例外 : No Exception

説明 : 汎用レジスタ Ra で指定したアドレスへ絶対分岐する。

用例 : JMP r1 // GOTO GPR[1]

CALL CALL: レジスタ指定アドレスへのサブルーチンコール (R1 形式)

31	26 25	21 20	0
010101 (0x15)	Ra	0_0000_0000_0000_0000	
CALL	Register A	<i>Reserved</i>	

書式 : CALL Ra

動作 : GOTO GPR[Ra]_{31~2} ;

GPR[31] = NextPC;

例外 : No Exception

説明 : 汎用レジスタ Ra で指定したアドレスへ絶対分岐する。

呼び出し元の次のアドレスを汎用レジスタ 31 番に格納する。

用例 : CALL r1 // GOTO GPR[1], GPR[31] = NextPC

A.3.5 メモリアクセス命令

LDW LoaD Word : ワード読み出し (R2I 形式)

31	26 25	21 20	16 15	0
010110 (0x16)	Ra	Rb	Immediate	
LDW	Register A	Register B		

書式 : LDW $Ra, Rb, Immediate$
 動作 : $ADDR = GPR[Ra] + SignExtention(Immediate)$
 if ($ADDR_{1\sim0} == 00$)
 $GPR[Rb] = MEMORY[ADDR_{31\sim2}]$
 else
 $Exception(AddressMissAlign)$;
 例外 : アドレスミスアライン
 説明 : 汎用レジスタ Ra と符号拡張した $Immediate$ を足したアドレスから 1 語 (4 バイト) 読み出し、その値を汎用レジスタ Rb に格納する。
 語 (4 バイト) 境界を跨ぐアドレスへのアクセスは、アドレスミスアライン例外を発生させる。
 用例 : LDW $r1, r2, 0x4$ // $GPR[2] = MEMORY[GPR[1] + 0x4]$

STW STore Word : ワード書き込み (R2I 形式)

31	26 25	21 20	16 15	0
010111 (0x17)	Ra	Rb	Immediate	
STW	Register A	Register B		

書式 : STB $Ra, Rb, Immediate$
 動作 : $ADDR = GPR[Ra] + SignExtention(Immediate)$
 if ($ADDR_{1\sim0} == 00$)
 $MEMORY[ADDR_{31\sim2}] = GPR[Rb]$
 else
 $Exception(AddressMissAlign)$;
 例外 : アドレスミスアライン
 説明 : 汎用レジスタ Ra と符号拡張した $Immediate$ を足したアドレスへ、汎用レジスタ Rb の値を 1 語 (4 バイト) 書き込む。
 語 (4 バイト) 境界を跨ぐアドレスへのアクセスは、アドレスミスアライン例外を発生させる。
 用例 : STB $r1, r2, 0x4$ // $MEMORY[GPR[1] + 0x4] = GPR[2]$

A.3.6 システムコール命令

TRAP Trap : トラップ (RR 形式)

31	26 25	0
011000 (0x18)	<i>00_0000_0000_0000_0000_0000_0000</i>	
TRAP	<i>Reserved</i>	

書式 : TRAP
 動作 : $Exception(Trap)$
 例外 : トラップ
 説明 : トラップ例外を発生させる。
 用例 : TRAP // Invoke Trap Exception

A.3.7 特権命令

RDCR ReaD Control Register : 制御レジスタの読み出し (R2 形式)

31	26 25	21 20	16 15	0
011001 (0x19)	Ra	Rb	0000_0000_0000_0000	
RDCR	Register A	Register B	<i>Reserved</i>	

書式 : RDCR Ra, Rb

動作 : if (ExeMode != KernelMode)
 Exception(PrivilegeViolation);
 else

GPR[Rb] = CTRL[Ra];

例外 : 特権違反

説明 : 制御レジスタ Ra の値を読み出し、汎用レジスタ Rb に格納する。
 カーネルモード以外でこの命令を実行した場合特権違反例外が発生する。

用例 : RDCR c1, r2 // GPR[2] = CTRL[1]

WRCR WRite Control Register : 制御レジスタへの書き込み (R2 形式)

31	26 25	21 20	16 15	0
011010 (0x1a)	Ra	Rb	0000_0000_0000_0000	
WRCR	Register A	Register B	<i>Reserved</i>	

書式 : WRCR Ra, Rb

動作 : if (ExeMode != KernelMode)
 Exception(PrivilegeViolation);
 else

CTRL[Rb] = GPR[Ra];

例外 : 特権違反

説明 : 汎用レジスタ Ra の値を、制御レジスタ Rb に書き込む。
 カーネルモード以外でこの命令を実行した場合特権違反例外が発生する。

用例 : WRCR r1, c2 // CTRL[2] = GPR[1]

EXRT EXception ReTurn : 例外からの復帰 (R0 形式)

31	26 25	0
011011 (0x1b)	00_0000_0000_0000_0000_0000_0000	
EXRT	<i>Reserved</i>	

書式 : EXRT

動作 :

```
if (ExeMode != KernelMode) {
    Exception(PrivilegeViolation);
} else {
    ExeMode = PreviousExeMode;
    GOTO ExceptionProgramCounter;
}
```

例外 : 特権違反

説明 : 実行モードを例外発生前のモードに戻し、例外が発生したアドレスに復帰する。
カーネルモード以外でこの命令を実行した場合特権違反例外が発生する。

用例 : EXRT // Exception Return (GOTO EPC)