

ユーザースマニュアル (Group 18)

1029-28-9483 勝田 峻太郎

1029-28-1547 住江 祐哉

2018 年 5 月 25 日

目 次

概要	2
性能と特徴	2
命令セットアーキテクチャ	2
命令形式	2
命令の動作略記法	2
算術・論理演算命令	3
条件フラグ	3
シフト演算	3
ロード・ストア命令	3
分岐命令	4
即値命令	4
その他の命令	4
IN 命令	5
構造と動作	5

概要

性能と特徴

- ほげほげ (mHz) の 5 段マルチサイクルプロセッサである.
- ハーバード・アーキテクチャ
- 16bit 命令セット

命令セットアーキテクチャ

命令形式

命令形式は, 以下の 2 つからなる.

Listing 1 R 形式

```
#-----|-----|-----|-----|---#  
| op1 | rs | rd | op3 | d |  
#15---|13--|10--|7----|3--#
```

Listing 2 I 形式

```
#-----|-----|-----|-----|---#  
| op1 | ra | rb |    d    |  
#15---|13--|10--|7-----#
```

命令の動作略記法

以下の図において, 命令の動作を略記する際に以下の略記法を用いる.

Table 1: 略記法一覧

略記法	意味
+	加算
-	減算
&&	bitwise and
	bitwise or
^	bitwise xor
signext()	値を 16bit へ拡張
=	右辺の値を左辺に代入する
*[<n>]	メインメモリの<n>番地に格納されているデータを示す.
r[<n>]	<n>番レジスタに格納されている値を示す.

算術・論理演算命令

算術演算命令には, 以下の R 形式の命令セットが用意されている.

Table 2: 算術・論理演算命令

ニモニック	タイプ	op1	rs	rd	op3	d	操作
ADD	算術演算	11	input	input	0000	input	$r[rd] = r[rs] + r[rd]$
SUB	算術演算	11	input	input	0001	input	$r[rd] = r[rs] - r[rd]$
AND	論理演算	11	input	input	0010	input	$r[rd] = r[rs] \&\& r[rd]$
OR	論理演算	11	input	input	0011	input	$r[rd] = r[rs] r[rd]$
XOR	論理演算	11	input	input	0100	input	$r[rd] = r[rs] \wedge r[rd]$
CMP	比較演算	11	input	input	0101	input	$r[rd] - r[rs]$
MOV	移動演算	11	input	input	0110	input	$r[rd] = r[rs]$
SLL	シフト演算	11	input	input	1000	input	$r[rd] = \text{shift_left_logical}(r[rd], d)$
SLR	シフト演算	11	input	input	1001	input	$r[rd] = \text{shift_left_rotate}(r[rd], d)$
SRL	シフト演算	11	input	input	1010	input	$r[rd] = \text{shift_right_logical}(r[rd], d)$
SRA	シフト演算	11	input	input	1011	input	$r[rd] = \text{shift_right_arithmetic}(r[rd], d)$

条件フラグ

各演算において, プロセッサ内のレジスタに, それぞれ 4 つのフラグ S, V, Z, C を設定する. これらは, 条件分岐時に用いられる.

S 演算結果が負の場合 1, そうでなければ 0 を設定する.

V 演算結果が符号付き 16 ビットで表せる範囲を超えた場合は 1, そうでなければ 0 を設定する.

Z 演算結果がゼロならば 1, そうでなければ 0 を設定する.

C 桁上げがあれば 1, そうでなければ 0 を設定する.

シフト演算

SLL(左論理シフト) 左シフト後, 空いた部分の 0 を入れる.

SLR(左循環シフト) 左シフトによって空いた部分に, シフトアウトされたビット列を入れる.

SRL(右論理シフト) 右シフト後, 空いた部分の 0 を入れる.

SRA(右算術シフト) 右シフト後, 空いた部分に符号ビットの値を入れる.

ロード・ストア命令

ロード・ストア命令には, 以下の I 形式の命令セットが用意されている.

Table 3: ロード・ストア命令

ニモニック	タイプ	op1	rs	rd	d	操作
LD	ロード命令	00	input	input	input	$r[ra] = *r[rb] + \text{sign_ext}(d)$

ニモニック	タイプ	op1	rs	rd	d	操作
ST	ストア命令	01	input	input	input	$*[r[rb] + \text{sign_ext}(d)] = r[ra]$

分岐命令

分岐命令には, 以下の I 形式の命令セットが用意されている.

Table 4: 分岐命令

ニモニック	タイプ	op1	rs	rd	d	操作
B	無条件分岐	10	100	input	input	$PC = PC + 1 + \text{sign_ext}(d)$
BE	条件分岐	10	111	000	input	if (Z) $PC = PC + 1 + \text{sign_ext}(d)$
BLT	条件分岐	10	111	001	input	if ($S \wedge V$) $PC = PC + 1 + \text{sign_ext}(d)$
BLE	条件分岐	10	111	010	input	if ($Z \vee (S \wedge V)$) $PC = PC + 1 + \text{sign_ext}(d)$
BNE	条件分岐	10	111	011	input	if (!Z) $PC = PC + 1 + \text{sign_ext}(d)$

即値命令

即値命令には, 以下の I 形式の命令セットが用意されている.

Table 5: 即値命令

ニモニック	タイプ	op1	rs	rd	d	操作
LI	即値ロード	10	000	input	input	$r[rb] = \text{sign_ext}(d)$
ADDI	拡張命令	10	001	input	input	$r[rd] = r[rd] + \text{sign_ext}(d)$
SUBI	拡張命令	10	010	input	input	$r[rd] = r[rd] - \text{sign_ext}(d)$
CMPI	拡張命令	10	011	input	input	$r[rd] = r[rd] - \text{sign_ext}(d)$

3つの演算拡張命令は, tbl. 2 に示されているものと同様, 4つの条件コードをレジスタに保存する.

その他の命令

その他の命令には, 以下の R 形式の命令セットが用意されている.

Table 6: その他命令

ニモニック	タイプ	op1	rs	rd	d	op3	操作
IN	入出力命令	11	input	input	1100	input	$r[rd] = \text{input}$
OUT	入出力命令	11	input	input	1101	input	$\text{output} = r[rs]$
HLT	入出力命令	11	input	input	1111	input	HALT

IN 命令

IN 命令は, 基板上の

構造と動作