ユーザーズマニュアル (Group 18)

1029-28-9483 勝田 峻太朗　　1029-28-1547 住江 祐哉

2018 年 6 月 13 日

目 次

概要

性能と特徴

- 111(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
- 16(bit/語) × 4096(語) のメインメモリ
- 16(bit/語) × 4096(語) の命令メモリ

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] ^ 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] = shift_left_logical(r[rd],d)`
SLR	シフト演算	11	input	input	1001	input	`r[rd] = shift_left_rotate(r[rd], d)`
SRL	シフト演算	11	input	input	1010	input	`r[rd] = shift_right_logical(r[rd], d)`
SRA	シフト演算	11	input	input	1011	input	`r[rd] = 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 桁上げげあれば 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] + 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] + 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 + sign_ext(d)`
BE	条件分岐	10	111	000	input	`if (Z) PC = PC + 1 + sign_ext(d)`
BLT	条件分岐	10	111	001	input	`if (S ^ V) PC = PC + 1 + sign_ext(d)`
BLE	条件分岐	10	111	010	input	`if (Z || (S ^ V)) PC = PC + 1 + sign_ext(d)`
BNE	条件分岐	10	111	011	input	`if (!Z) PC = PC + 1 + 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] = sign_ext(d)`
ADDI	拡張命令	10	001	input	input	`r[rd] = r[rd] + sign_ext(d)`
SUBI	拡張命令	10	010	input	input	`r[rd] = r[rd] + sign_ext(d)`
CMPI	拡張命令	10	011	input	input	`r[rd] - 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] = input
OUT	入出力命令	11	input	input	1101	input	output = r[rs]
HLT	入出力命令	11	input	input	1111	input	HALT

IN 命令

IN 命令は, 基板上のディップスイッチで, ディップスイッチによって表された 16 ビットの値が,rd で指定されたレジスタに代入される.

OUT 命令

OUT 命令では, 指定されたレジスタに格納されている値を,MU500-7SEG ボード上の,7SEG-LED 部,A0 から A3 までに,16 進数で表示する.

HLT 命令

HLT 命令では, 各モジュールに出力される generated clock を停止することで, プロセッサの処理を停止する.

構造

このプロセッサはステージ 4 つ (p1~p4) とコントローラに分かれている. それらの機能を以下で説明する.

IF:命令フェッチ (p1.v)

プログラムカウンタ, プログラムの命令が書かれた大容量メモリを含むモジュールである.
マルチプレクサ, 加算器も含む.
このモジュールで 16bit 固定長命令が書かれたプログラムを命令メモリから読み出し, このプロセスで行う命令を決定する.

ID&WB:命令デコードとレジスタフェッチ/レジスタ書き込み (p2.v)

読み出したり, 書き込んだりするためのデータが保存されたレジスタ, 符号拡張器が含まれているモジュールである.
このモジュールではレジスタからの読み出し・書き込みと, 命令に書かれている定数のビット変換 (拡張) を行う.
制御もここで命令を受け取って行う.
ただし, 命令の解釈/レジスタからの読み出し/レジスタへの書き込みのタイミングが異なり, それは後述する.

EX:命令実行とアドレス生成 (p3.v)

ここで演算命令の演算を行う.
ビットシフタ, 加算器,ALU, マルチプレクサが含まれる.
演算結果により,4 種類のフラグ (v,z,c,s) を決定し, 条件分岐命令で使用する.
また, 出力命令による 7SEG-LED への出力の制御もこのステージで行われている.

MEM:メモリ・アクセス (p4.v)

データメモリとそれに読み込む/書き込む機能を持つ.

コントローラー (Controller.v)

全体に適切なクロックを送り, マルチサイクル実行を実現するモジュール. プロセッサ全体の実行, 停止, リセットもこのモジュールで扱う. コントローラーは自身に入力されるクロックの半分の周波数 (2 倍の周期) のクロックをジェネレーティッドクロックとして, 各モジュールに順番にクロックを出力する.

動作

動作モジュール自体は 4 つに分かれているが,5 サイクルである.

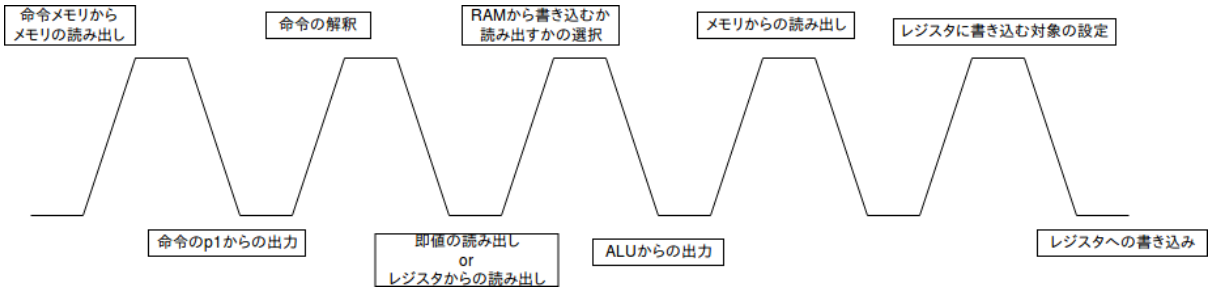

Figure 1: クロック画像

1 サイクル目:予め命令が格納されている RAM(命令メモリ) から命令の読み出しを行う.
2 サイクル目:命令セット・アーキテクチャに従って、命令をそれぞれの形式ごとに解釈する. また、命令から即値の読み出し、レジスタから値の読み出しもこのサイクルで行う.
3 サイクル目:主に ALU での計算である. 演算命令が入力された時、このサイクルで演算が行われる. 分岐の決定もこのサイクルである. また、ここで、主記憶に対する動作入力も決定される.
4 サイクル目:主記憶に対して、読み出し、書き込みを行う.
5 サイクル目:レジスタに値を書き込む動作をここで行う. その際、どの値を書き込むかも設定する.