第 1 章

自作 CPU

電気電子工学科 4 回生

本田卓

はじめに

一昨年の 3 月から今年の 3 月ごろにかけて自作の 8 bit CPU を TTL のみで製作しました。ですから今回はその CPU の基本仕様、製作中のエピソードなどをまとめました。かなりマニアックなものですが興味のある方は是非お読みください。

1.1 CPU 解説

1.1.1 基本仕様

- bit 数：8 bit CPU
- レジスタ数：1 個
- 動作周波数：12 MHz
- RISC 命令長：16 bit
- 1 命令 16 クロック
- 命令数：18 個

1.1.2　基本仕様の解説

まずこの CPU の基本仕様について解説します。普通 CPU の仕様で気になるのはビット数でしょう。最近のパソコンなら 32 bit か 64 bit のものが使われています。CPU のビット数の定義は詳しく書きませんが、簡単に言えば「その CPU が一度（1 命令）に処理できるデータサイズのこと」です。この CPU では 1 命令で 8 bit データを処理できるので 8 bit CPU となります。

次にレジスタですが、これは CPU 内にある小さなメモリです。CPU の主な処理は「2 進数データの演算」と「メインメモリへのデータの格納」です。二つ目のメインメモリへ格納ですが、メインメモリというのは、私たちは「メモリ」とよんでいます、小さければ 4 GB くらいで大きければ 16 GB くらいのあれです。このメインメモリですが、CPU 本体とは離れた場所にあります。ですから CPU とメインメモリが通信するのに時間がかかります。そこで、CPU とメインメモリの間にもう一つ CPU から高速にアクセスすることができるメモリを用意します。それがレジスタです（レジスタは CPU の内部に含まれます)。CPU はレジスタに値を格納して、その格納された値をメインメモリへ格納する、という手順を踏みます。普通の CPU ならレジスタは複数個備えているのですが、この CPU は回路を単純にするためレジスタを 1 個しか備えていません。

動作周波数はクロックを発生させる IC を使っているのでこの IC の上限である 12 MHz が上限となります。

次に命令についてです。CPU の命令は大きく分けて RISC と CISC に分けられます。簡単に説明すると RISC が全命令の命令長が同じ、CISC が命令によって命令長が違うというものです。この CPU ではすべての命令が 16 bit なので RISC ということになります。次に 1 命令にかかるクロック数ですが、これはいろいろな資料を参考に考えた結果、16 クロックで 1 命令実行されるような回路になったというだけで、特に深い意味はありません。頑張って設計すればもっと少ないクロック数で 1 命令を実行できるようになります。簡単で作りやすかったのでこうなりました。そして最後に命令数ですが、これがこの CPU の一番の特徴を表しているといえます。

本当に特徴を表しているのはその命令の中身なのですが、とにかく、命令の内容で CPU になにができるかが決まります。詳しくは後で説明しますが、この CPU の命令の特徴を一言でいうと「必要最低限」です。いろいろな命令を実行させる場合はもちろん回路が複雑になります。それが一番嫌だったので、これがあれば最低限のことはできるという程度しか命令を用意していません。次の章で命令について詳しく説明します。

1.1.3 命令セット

この CPU が実行できる全命令とその説明をします。まずは説明に使われる用語の説明を行います。

- A：レジスタのこと。複数あればレジスタ A, レジスタ B, ... となります。今回は A レジスタしかないので A しか出てきません。
- 機械語：2 進数 (0,1) で表現された実際に CPU が読み取る命令のこと。
- アセンブラ：機械語の 1 命令をわかりやすいように 2 進数から文字に変換したもの。
- IM(Immediate data)：機械語命令の後半部分の値のこと（前半部分はオペコード)。
- プログラムカウンタ (PC)：メインメモリの中で実際に実行される命令が格納されているアドレス（番地）を示す値。通常 1 命令実行されるたびインクリメントされる。

次に命令の例を表 1.1 に、全命令とその説明をまとめた命令表を表 1.2 に示します。

表 1.1 命令の例

	オペコード	IM
アセンブリ	LDIM	3
機械語	0101 0000	0000 0011
意味	A レジスタに 3 を格納する	

表 1.2　命令表

命令 アセンブラ表記	機械語 (16 進数)	説明	式
LDIM	50	IM データを A レジスタにロードする	A = IM
ADDIM	91	A レジスタの値に IM の値を足したものを A レジスタに格納する	A = A + IM
SUBIM	62	A レジスタの値に IM の値を引いたものを A レジスタに格納する	A = A − IM
ANDIM	D3	A レジスタの値と IM の値の AND をとったものを A レジスタに格納する	A = A AND IM
NOTIM	3	A レジスタの値と IM の値の NOT をとったものを A レジスタに格納する	A = A AND IM
ORIM	74	A レジスタの値と IM の値の OR をとったものを A レジスタに格納する	A = A AND IM
LDMEM	55	A レジスタにメインメモリの IM 番地の値を格納する	A = [IM]
ADDMEM	96	A レジスタの値にメインメモリの IM 番地の値を足したものを A レジスタに格納する	A = A + [IM]
SUBMEM	67	A レジスタの値にメインメモリの IM 番地の値を引いたものを A レジスタに格納する	A = A − [IM]
ANDMEM	D8	A レジスタの値とメインメモリの IM 番地の値の AND をとったものを A レジスタに格納する	A = A AND [IM]
NOTMEM	8	A レジスタの値とメインメモリの IM 番地の値の NOT をとったものを A レジスタに格納する	A = A NOT [IM]
ORMEM	79	A レジスタの値とメインメモリの IM 番地の値の OR をとったものを A レジスタに格納する	A = A OR [IM]
JMP	5A	プログラムカウンタの値を IM に変更する	PC = IM
JNC	5B	キャリーフラグが 1 の場合プログラムカウンタを IM にする	PC = IM (CF = 1)
JNZ	5C	ゼロフラグが 1 の場合プログラムカウンタを IM にする	PC = IM (ZF = 1)
STR	D	A レジスタの値をメインメモリの IM 番地に格納する	[IM] = A
IN	5E	IN ポートの値を A レジスタに格納する	A = IN
OUT	F	A レジスタの値を OUT ポートに出力する	OUT = A

1.1.4 サンプルコード

3×5 の計算

```
1      LDIM 0
2      STR 0
3      STR 1
4  bbb:LDIM 5
5      SUBMEM 1
6      JNZ aaa
7      LDMEM 0
8      ADDIM 3
9      STR 0
10     LDMEM 1
11     ADDIM 1
12     STR 1
13     JMP bbb
14 aaa:LDMEM 0
15     OUT
16 ccc:JMP ccc
```

■解説 最初の LDIM 0, STR 0, STR 1 でメインメモリの 0 番地と 1 番地に 0 を格納します。次に LDIM 5, SUBMEM 1 で 5 からメインメモリの 1 番地の値を引きます。はじめは 0 が格納されているので答えは 5 が返ってきます。ここで答えが 0 だった場合次の JNZ aaa に引っかかり aaa アドレス（プログラム中の aaa: LDMEM 0）にジャンプしますが、今回は 5 なのでそのまま次の命令に行きます。LDMEM 0, ADDIM 3, STR 0 でメインメモリの 0 番地の値を A レジスタに持ってきて、それに 3 を足してまた 0 番地に戻します。次の LDMEM 1, ADDIM 1, STR 1 で同様に 1 番地の値を持ってきて 1 足してまた 1 番地に戻します。ここで JMP bbb により bbb の地点に戻るのでこれまでの命令が繰り返されるのですが、1 番地の値が 1 足されているので最初の LDIM 5, SUBMEM 1, JNZ aaa のとろこが $5-1=4$ となります。これでも 0 ではないので JNZ で aaa にはジャンプしないのですが、これをあと 4 回繰り返すと $5-5$ となり JNZ aaa が実行されて aaa 地点の実行へ移ります。そこからは LDMEM 0 で 0 番地の値を呼び出して OUT で OUT ポートに出力（A レジスタの値を人が見えるように LED で表示）します。JMP ccc はその場無限ループなのでこれ以上もう CPU は何もしません。

1 から 10 まで足し算

```
1      LDIM 0
2      STR 0
3      LDIM 1
4      STR 1
5      LDIM 10
6      STR 2
7  bbb:LDMEM 0
8      ADDMEM 1
9      STR 0
10     LDMEM 1
11     ADDIM 1
12     STR 1
13     LDMEM 2
14     SUBIM 1
15     JNZ aaa
16     STR 2
17     JMP bbb
18 aaa:LDMEM 0
19     OUT
20 ccc:JMP ccc
```

■解説　前の 3×5 のプログラムで詳しく説明したので今回は少しざっくりと説明します。最初の LDIM 0, STR 0,LDIM 1 STR 1, LDIM 10, STR 2 でメインメモリの 0 番地に 0、1 番地に 1、2 番地に 10 を格納します。つぎに LDMEM 0, ADDMEM 1, STR 0 でメインメモリ 0 番地の値と 1 番地の値を足します。次の LDMEM 1, ADDIM 1, STR 1 で 1 番地の値を 1 増やします。これで次は 0 番地の値に 2 が足されます。最後に LDMEM 3, SUBIM 1, STR 3 で 3 番地の値から 1 引きます。3 番地にはもともと 10 が入っており、引き算の答えが 0 になったとき（10 ループしたとき）JNZ ccc でループを抜けて 0 番地の値を OUT で出力します。

1.2 製作小話

1.2.1 製作に至った経緯

もともとサークル活動として電子工作、マイコンを使ったプログラミングをしていました。そんな中で「マイコンの中身ってどうなっているんだろう」という疑問を持っていましたが、同時に「ものすごい複雑な構造をしていて、簡単にわかるものではない」とも思っていました。ですが、ある日偶然「論理回路だけで電卓を作る」というサイトを見つけました。そこにはとても分かりやすく論理回路で計算機を作る方法が解説されており、論理回路の基礎しか知らない僕でも簡単に理解することができました。これをきっかけに「もしかしたら CPU の構造も論理回路だけで理解できるかもしれない」と思いました。そこから名著「CPU の創り方」を読んだり、自分で勉強したりしてある程度 CPU を理解することができました。ですが、この時はまだ CPU 自作をしたいとは思っていても、できるとは思えず行動はしていませんでした。そんなとき、詳しくは説明できませんが、とんでもないミラクルが起き、回路のスペシャリストの方と知り合うことができ、僕が CPU を自作したいと話すと、「ぜひ一緒に作りましょう」という流れになり、その結果 CPU を作ることになりました。そこから一年かかりましたが無事 CPU を完成させることができました。かなり運がよかったと思います。

1.2.2 はんだ挫折

今完成している CPU は外注基板でできていますが最初はユニバーサル基板でジャンパー配線の手はんだで作ろうとしていました。書き込み回路だけは手はんだで作りましたが（僕ではなくはんだ付けが得意な友達がやってくれました）あまりの面倒くささに手はんだは断念して外注基板に変更することになりました。下図が実際にはんだ付けされた基板です。

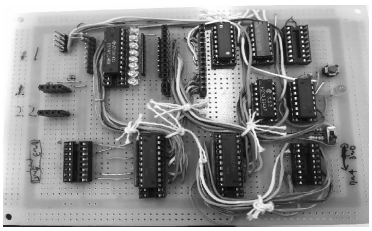

図 1.1 ユニバーサル基盤にはんだ付けした回路

1.2.3　お世話になった資料

この CPU を作るまでにいろいろな資料にお世話になりました。その中でも特に役に立った資料（本、サイトなど）を紹介します。

- 電卓製作サイト
  `http://www7b.biglobe.ne.jp/~yizawa/logic2/chap1/index.html`
  先ほども書いたように僕が CPU を作ることになったきっかけです。普通「CPU は論理回路だけでできている」と言われても想像がつきません。ですが、このサイトの丁寧な解説で自分が思っているほど CPU の内部回路は難しくないのかもしれないと思えるようになりました。
- CPU の創り方
  有名なやつです。TTL を 10 個だけで CPU を自作するという本です。これが CPU なのかというほど簡単な CPU を作りますが、基本構造は変わらないのでこれを理解できれば CPU の基本は理解できます。いきなり最新の複雑な CPU を理解できるわけがないので、最初はこの本のような最小構造の CPU を勉強するのが一番いいと思います（趣味で勉強するなら）。

図 1.2　CPU の創り方

- がたろうさんのサイト
  `http://diode.matrix.jp/`
  自作 CPU を作るおじいさんのサイトです。この人が作った CPU の回路を参考に今回の CPU を製作しました。CPU だけでなく、独自のコンパイラ、OS などほとんどのものを自作しています。この人のまねをしたといっても間違いではないです。

1.2.4 こんなことしないでシミュレーションソフトを使いましょう

最後にこれを見て少しでも CPU に興味を持たれた方がいるなら一つアドバイスですが、はんだ付けして実物が欲しい！などと思わなければ、論理回路シミュレータという素晴らしいものがあるのでそちらを使いましょう。フリーソフトでだれでも簡単に使えます。僕が使っているのは Logisim というソフトです。作った回路が実際に CPU として動作するか確認として使っていましたが、とても便利でした。正直途中から「シミュレータで動いたんだからもうはんだ付けしなくてもいいか」とも思っていました。変更点があれば一瞬で変更できるし、コピペできるし、ほかの人が作った部品流用できるし、完璧です。ネットにほかの人が logisim で作った CPU が上がっているのでそれを参考にすることもできます。「CPU の創り方」の CPU なんてすぐ作れます。まあ、実物は思い出になりますし、いいところもあります。(でももう作りません)

参考文献

[1] 渡波 郁, CPU の創り方, 毎日コミュニケーションズ.