

目次

| | | |
|--------------|------------------|-----------|
| 第 1 章 | CPU の概要 | 2 |
| 1.1 | 基板 | 2 |
| 1.2 | CPU | 3 |
| 第 2 章 | 製作 | 5 |
| 2.1 | 部品表 | 5 |
| 2.2 | 部品の実装 | 5 |
| 第 3 章 | 動作確認と使用 | 8 |
| 3.1 | ROM | 8 |
| 3.2 | アセンブラー | 10 |
| 第 4 章 | 最後に | 12 |

第1章

CPU の概要

CPU の基板をご購入いただきありがとうございます。と説明書っぽいことを言ってみる。まあ書くこともないし、紙が無駄なのでこれぐらいにする。

1.1 基板

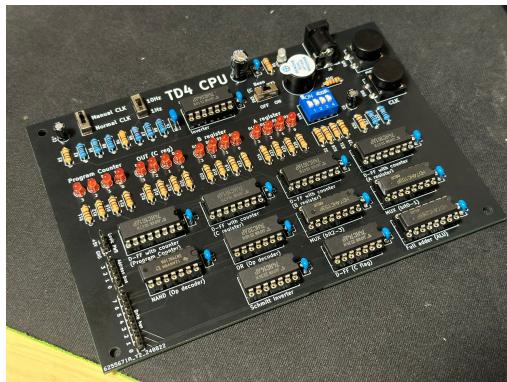


図1.1 完成した基板

このCPUは、書籍「CPUの創りかた^{*1}」で製作するTD4 CPUの回路を、筆者が基板に起こしたものになる。

書籍ではユニバーサル基板上に配線していたが、そんなことしたくなかったので基板を作ることにした。基板設計など一切したことなかったので、専

^{*1} <https://book.mynavi.jp/ec/products/detail?id=22065>
(ISBN4-8399-0986-5)

門家が見たら怒る点もあると思うが、気にしないでほしい。部品を実装すれば動くと思うが、書籍も参照しながら作ることをおすすめする。

基板の設計は KiCad 8.0 を利用した。無料の CAD だが非常に使いやすかった。また、基板は JLCPCB に発注した。

1.2 CPU

CPU の概要を以下に示す。

- 命令データ長 8bit
- アドレスバス長 4bit
- 2 つの 4bit レジスタ (A,B)
- 4bit 入出力 (IN, OUT)
- 4bit プログラムカウンタ
- ALU は全加算器のみ

命令フォーマットは、図1.2のように、上位 4bit が命令コード、下位 4bit がイミディエイトデータの計 8bit 構成となっている。

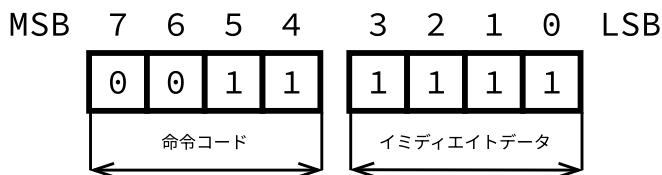


図1.2 命令フォーマット

命令コードの一覧と意味、ニーモニックを表1.1に示す。

| ニーモニック | 命令コード | 意味 |
|-------------|-----------|----------------------------|
| MOV A, [Im] | 0011 [Im] | A←Im |
| MOV B, [Im] | 0111 [Im] | B←Im |
| MOV A, B | 0001 0000 | A←B |
| MOV B, A | 0100 0000 | B←A |
| ADD A, [Im] | 0000 [Im] | A←A + Im |
| ADD B, [Im] | 0101 [Im] | B←B + Im |
| IN A | 0010 0000 | A←IN |
| IN B | 0110 0000 | B←IN |
| OUT [Im] | 1011 [Im] | OUT←Im |
| OUT B | 1001 [Im] | OUT←Im |
| JMP [Im] | 1111 [Im] | PC←Im |
| JNC [Im] | 1110 [Im] | if (C is not 1) then PC←Im |

表1.1 命令コードの一覧

第 2 章

製作

CPU の概要や回路を説明するのは書籍に任せるとして、この基板の製作について述べる。

2.1 部品表

基板以外に別途、別紙 1 に示した部品が必要となる。部品は必要個数のみ表記しているので、予備も含めて買うのをお勧めする。

部品については秋月電子通商^{*1}と桜木総業株式会社^{*2}ですべて購入できる。秋月電子通商で買える部品については秋葉原店の店頭ですべて購入できた(2024 年 8 月現在)。74HC14 だけバックヤードにあるためレジで店員に言わなければならないので注意。

秋葉原には秋月以外にもマルツや千石通商など、電子部品の店が多くある。この機会に秋葉原に出向いて色々なお店を見ながら選ぶと楽しいと思う。

2.2 部品の実装

はんだごて片手にひたすらはんだ付けする。表面実装部品は無いのではんだ付けの難易度は低いと思う。ただ量が多い。

暇だったのでアニメを見ながらはんだ付けをしていた。

この CPU の製作ではんだ付けを初めて行うという人は、はんだ付けの動画などを見て学ぶことをお勧めする。くれぐれも火傷には気を付けてほしい。

^{*1} <https://akizukidenshi.com/catalog/default.aspx>

^{*2} <https://www.kashinoki.shop/>

2.2.1 注意点

部品実装の注意点としては、電解コンデンサ (MLCC と書いていないものの) は極性^{*3}がある。

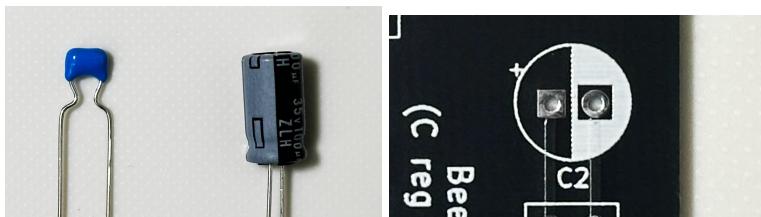


図2.1 積層セラミックコンデンサ(左)と電解コンデンサ(右)

図2.2 基板上の電解コンデンサの表記

電解コンデンサを見ると半分が白くなっていて、でかでかと- (マイナス) が書かれていると思う。これと基板上の白い部分の向きを合わせれば問題ない。また、積層セラミックコンデンサについては極性はない。



図2.3 基板上のLEDの表記

電源やレジスタの値の表示用にLED(発光ダイオード)を利用している。これも極性があるので注意してほしい。部品を刺す穴の周りの金属が四角いほう(図2.3では左側)がマイナス(LEDの足が短い方)になる。最悪こちらは逆につないでも光らないだけで回路に影響は無いはず。

^{*3} 簡単に言えば部品のプラスマイナスの向きのこと。逆につなぐと爆発などの危険がある

ICは印刷されている切り欠きとICの切り欠きを合わせて差し込むようにする。逆だと正しく動かなかったり、ICを壊す可能性もある。というか絶対に壊す。ICソケットを使用する場合はICソケットにも切り欠きがあるので合わせるのが良い。

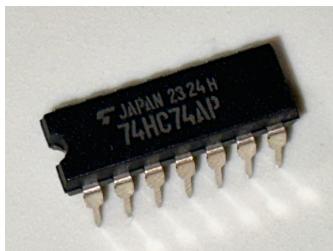


図2.4 ICの外観

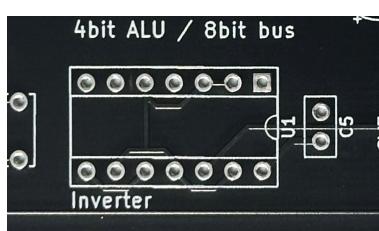


図2.5 基板上のICの印刷

第3章

動作確認と使用

はんだ付けが終わって完成しても TD4 CPU には ROM がないため、正しく動かない。そのためにまずは ROM を用意する必要がある。

3.1 ROM

書籍では ROM は 16 個 8PDIP スイッチを並べて製作していた。この方法はビジュアルとしては良いが結構部品代がかかるので断念した。アドレスバスとデータバスを引き出しているので、ここに適当な ROM を接続すれば良いが、筆者は Arduino で ROM を作ることにした。

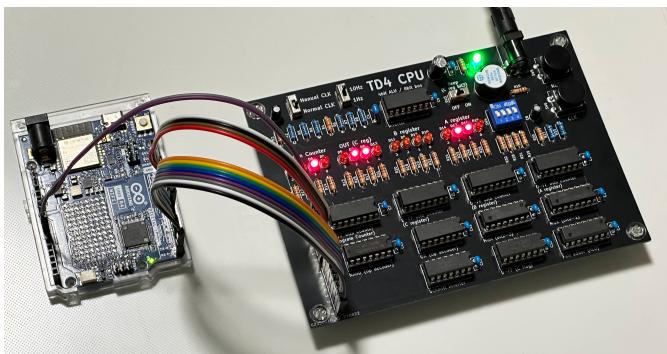


図3.1 Arduino で作った ROM で動く TD4 CPU

プログラムは以下の通り。

```
1 // rom data define
2 uint8_t rom[] = {0xb3, 0xb6, 0xbc, 0xb8, 0xb8, 0xbc, 0xb6, 0xb3, 0xb1, 0
3   xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
4
5 void writeDataBus(uint8_t data)
6 {
7   digitalWrite(D0, data & 0b00000001);
8   digitalWrite(D1, data & 0b00000010);
9   digitalWrite(D2, data & 0b00000100);
10  digitalWrite(D3, data & 0b00001000);
11  digitalWrite(D4, data & 0b00010000);
12  digitalWrite(D5, data & 0b00100000);
13  digitalWrite(D6, data & 0b01000000);
14  digitalWrite(D7, data & 0b10000000);
15 }
16
17 void setup()
18 {
19   // pin mode define
20   pinMode(D8, INPUT);
21   pinMode(D9, INPUT);
22   pinMode(D10, INPUT);
23   pinMode(D11, INPUT);
24
25   pinMode(D0, OUTPUT);
26   pinMode(D1, OUTPUT);
27   pinMode(D2, OUTPUT);
28   pinMode(D3, OUTPUT);
29   pinMode(D4, OUTPUT);
30   pinMode(D5, OUTPUT);
31   pinMode(D6, OUTPUT);
32   pinMode(D7, OUTPUT);
33
34   writeDataBus(0x00);
35 }
36
37 void loop()
38 {
39   uint8_t address = (bool)digitalRead(D8) | ((bool)digitalRead(D9) << 1)
40     | ((bool)digitalRead(D10) << 2) | ((bool)digitalRead(D11) << 3);
41   writeDataBus(rom[address]);
42 }
```

Arduino UNO R4 WiFi で動作確認しているが、他でも動くと思う。あとは Arduino 側の D0-D7 ピンをデータバスの 0-7 に、D8-D11 をアドレスバスの 0-3 にそれぞれ接続し、+5V と Vin を接続し、GND 同士を接続してどちらかに電源を入れればプログラムが実行される。

また PWR ピンがあるがこれは入出力どちらでも使えるので、Arduino 側から供給しても^{*1}、CPU 側から供給しても良い。

3.2 アセンブラー

書籍ではアセンブル^{*2}はハンドアセンブルのみが書かれていたが、めんどくさかったのでアセンブラーを作った。Linux 上で動かすことを想定しているので、Windows を使っている場合 WSL 上で行うことをお勧めする。

ビルドには `gcc`, `make`, `git`, `flex`, `bison` が必要なのでインストールする。Ubuntu を利用している場合は以下の通りになる。

```
1 $ sudo apt update
2 $ sudo apt upgrade
3 $ sudo apt install gcc make git flex bison libbison-dev
```

ビルドは以下のコマンドで行う。

```
1 $ cd ~
2 $ git clone https://github.com/nikachu2012/td4asm
3 $ cd td4asm
4 $ make all
```

`test.asm` というファイルを `td4asm` と同じ場所に作り、中身を以下のようにする。

ソースコード 3.1 LED ちかちかさせるプログラム `test.asm`

```
1 main: out 0b0011
2         out 0b0110
3         out 0b1100
4         out 0b1000
5         out 0b1000
6         out 0b1100
7         out 0b0110
8         out 0b0011
9         out 0b0001
10        jmp main
```

以下のコマンドを実行すると `test.txt` が作られる。

^{*1} Arduino 側から供給する場合、電流制限に注意

^{*2} 人間に読みやすいように書かれたニーモニックを機械語に変換する処理

```
1 $ ./td4asm -i test.asm -o test.txt -c
```

test.txt は以下のようになる。

```
1 {0xb3, 0xb6, 0xbc, 0xb8, 0xb8, 0xbc, 0xb6, 0xb3, 0xb1, 0xf0, 0x00, 0x00, 0
   x00, 0x00, 0x00}
```

このように、C 言語の配列形式で機械語が output される。

第4章

最後に

この本の PDF ファイル、回路図などは以下の URL に用意した。

<https://github.com/nikachu2012/td4cpu>

また、すべての質問に答えられるかわからないが、この CPU の基板やアセンブラーについてわからない点などがあれば、奥付に記載のメールアドレスまで連絡していただければできるだけ質問に答える。

4bit CPU 製作基板 説明書

2024 年 11 月 1 日 初版第 1 刷発行 (沼津高専 第 59 回高専祭)

著者 @nikachu2012 (荻野 壮)

発行者 @nikachu2012 (荻野 壮)
work@nikachu.net

発行所 沼津高専 情報工学同好会 (非公認)

印刷者 自分

Printed in Japan(多分)