

アナログ回路でCPU を作った

2024/12/01 第4回 CPUを語る会

14:30 – 15:00

講演者: Yokogoya (X: @electrotelecast)



目次

- 自己紹介
- なぜアナログでCPUを？
- おさらい: CPUの構成要素
- CPUの構成要素をアナログにしてみる
 - 記憶装置 (メモリ)
 - 入出力装置 (I/O)
 - 計算装置 (ALU)
 - その他構成要素
- 作成したAnalog CPU
 - 仕様
 - アーキテクチャ
 - 命令一覧
 - 動作概要
- 動作デモ
- まとめ・今後の展望

自己紹介

- 名前(HN):Yokogoya (よこごや)
 - オペアンプの"4558"に由来
- 趣味:電子工作、ドライブ、ガーデニング
- 職業:会社員 (アナログ回路設計) ←CPUに関係ない
- 学生時代の専攻:物性物理 ←CPUにほぼ関係ない



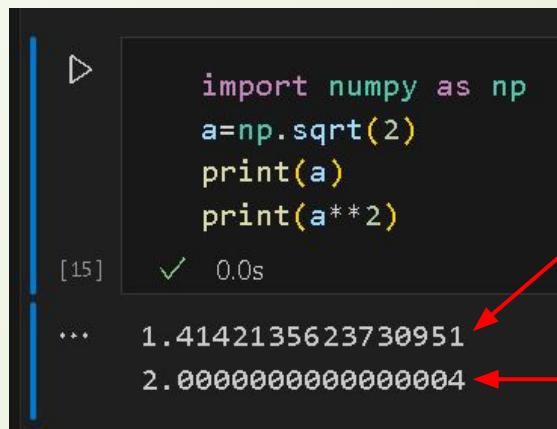
⇒ 情報工学素人でもCPUは作れる！！！

Analog CPUも難しくないのでみんな作ろう！！！

なぜアナログでCPUを？

① 無理数の取り扱い

▶ 例えば以下のPythonで $\sqrt{2}, (\sqrt{2})^2$ を計算する場合



```
import numpy as np
a=np.sqrt(2)
print(a)
print(a**2)

[15]    ✓  0.0s
...      1.4142135623730951
2.0000000000000004
```

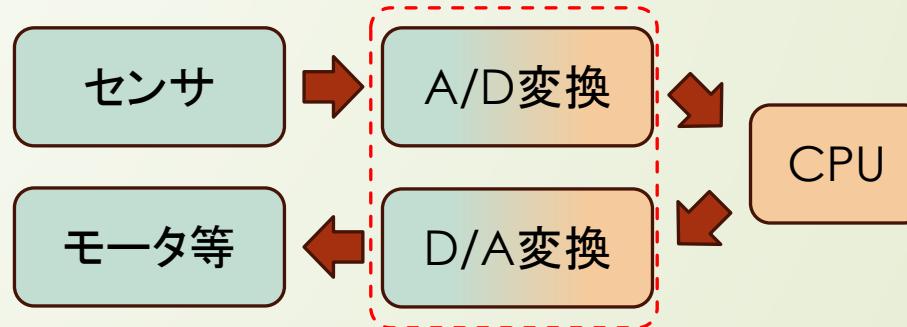
有限桁で表現
=有理数として処理

2ピッタリでは計算されない

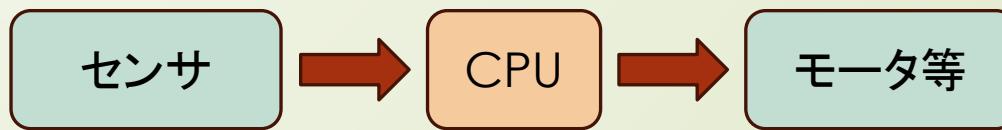
- 😢 デジタルで小数・無理数を表現するには限界がある。
- 😊 アナログでは数値は連続的に表現できるので、無理数もそのまま表現できる！！！

なぜアナログでCPUを？

2. マイコンを使用したセンシング＆自動制御システム



泣き顔の絵文字 アナログ↔デジタル変換を2回もしており回りくどい。
システムの用途・仕様によっては贅沢すぎる場合も。



笑顔の絵文字 CPUが直接アナログ信号を扱うことができるので、
アナログ↔デジタル変換が不要に！
システムのリソース最適化が可能！！！

おさらい①

そもそもCPUって何だっけ？

- 定義はいろいろあるかと思いますが…
この講演では以下のように定義します！

「外部入力もしくは記憶装置から取り込まれた値を、プログラムによって規定された順序で保持または演算し、結果を外部に出力する装置」
(ヒトマス"ソウイウコトニシテクダサイ...)

- では具体的には何でできているの？

おさらい②

CPUの構成要素

CPUは、以下の構成要素の組み合わせで実現している。

1. 記憶装置

- データを受け取り保持し、必要時に他装置にデータを渡す。

2. 入出力装置(I/O)

- 外部からデータを入力し、外部へデータを出力する。

3. 演算装置(ALU)

- 複数のデータ同士を演算し、演算結果を記憶装置に渡す。

4. 命令デコーダ・データセレクタ

- プログラムに書かれた命令を1個ずつ解読し、各装置の動作を決める

5. プログラムメモリ

- プログラムを格納する(特別な)記憶装置。

おさらい② CPUの構成要素

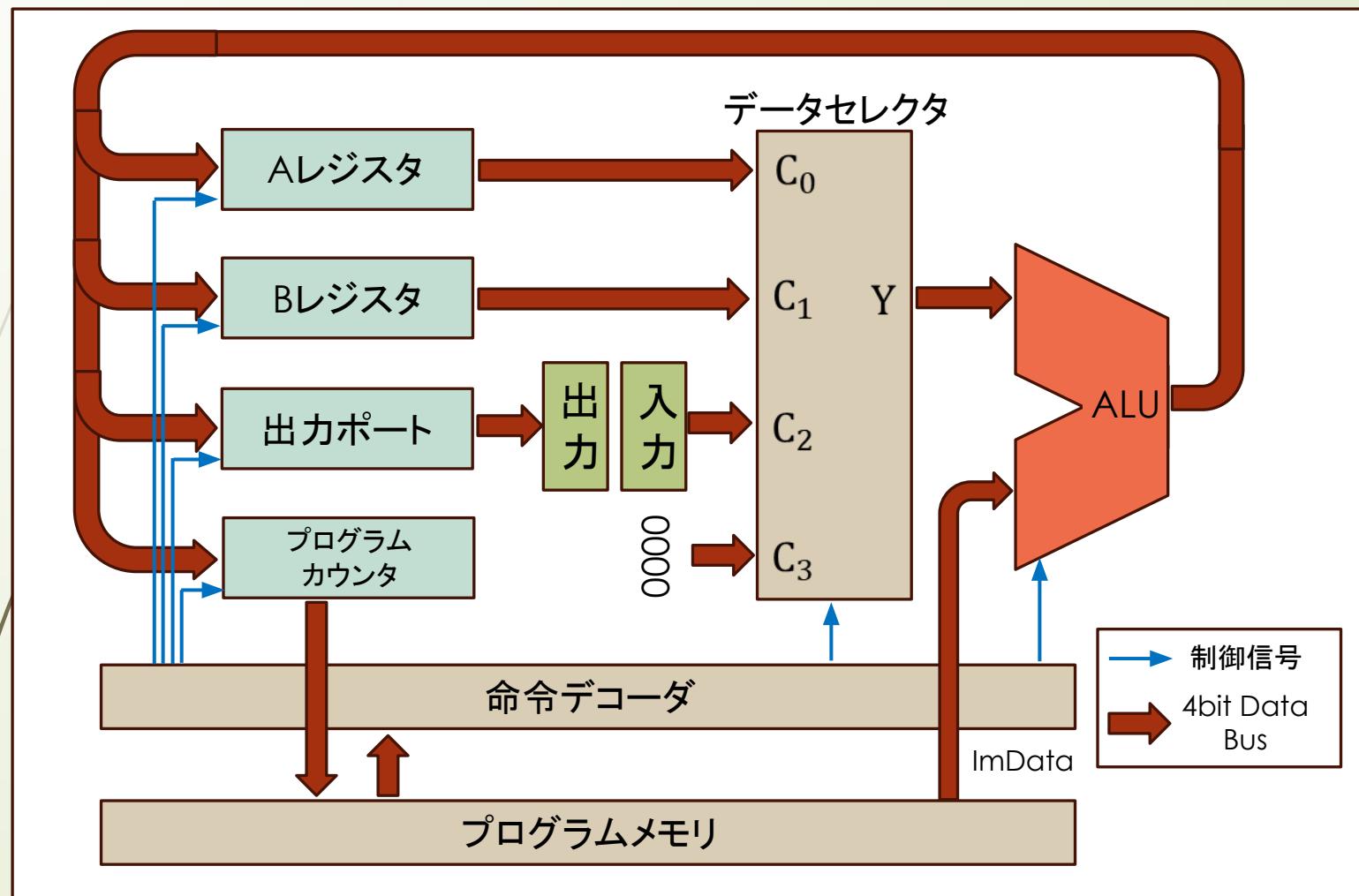


Fig: 某4bit CPUのアーキテクチャ[1]

おさらい② CPUの構成要素

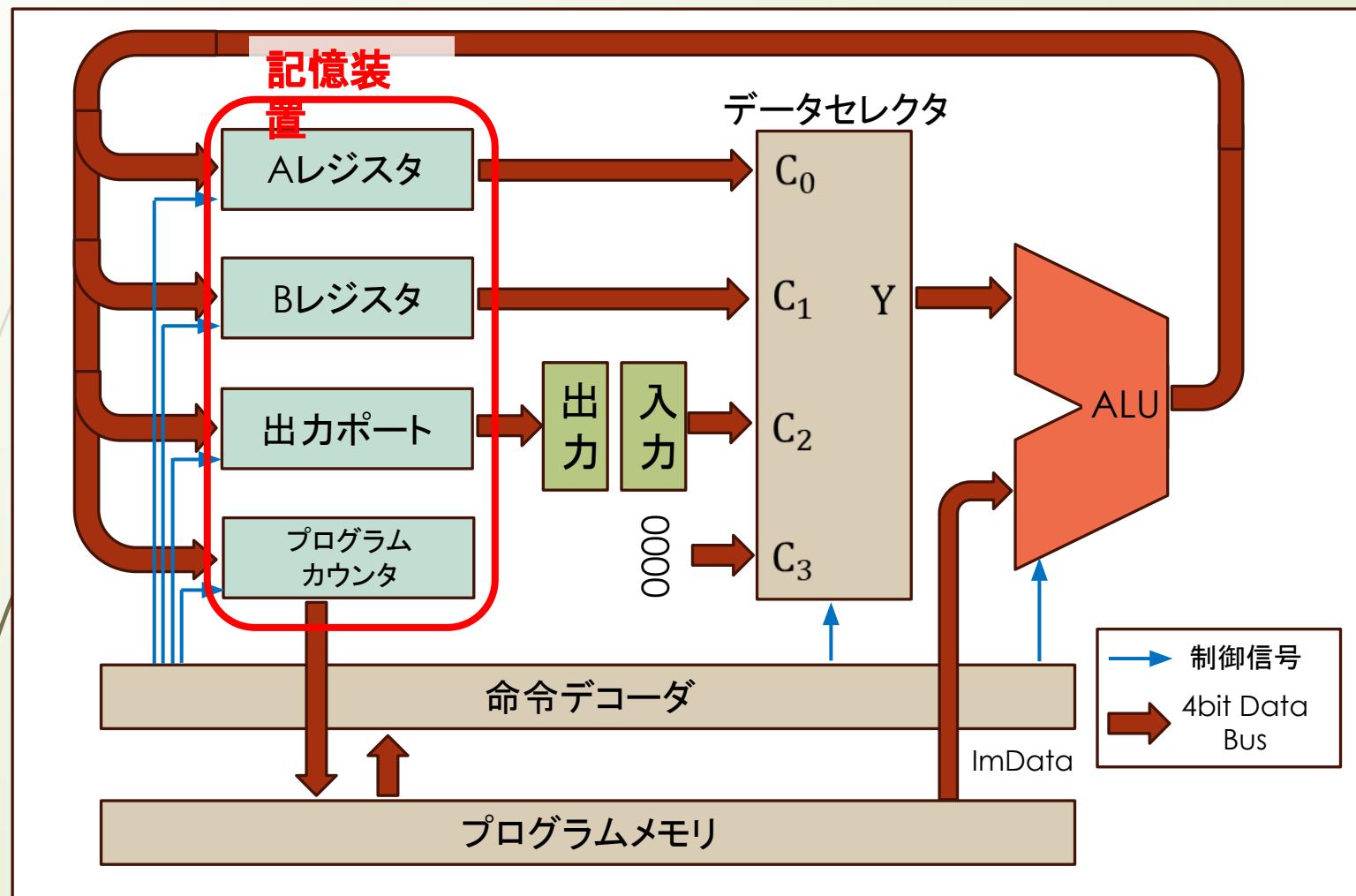


Fig: 某4bit CPUのアーキテクチャ[1]

おさらい② CPUの構成要素

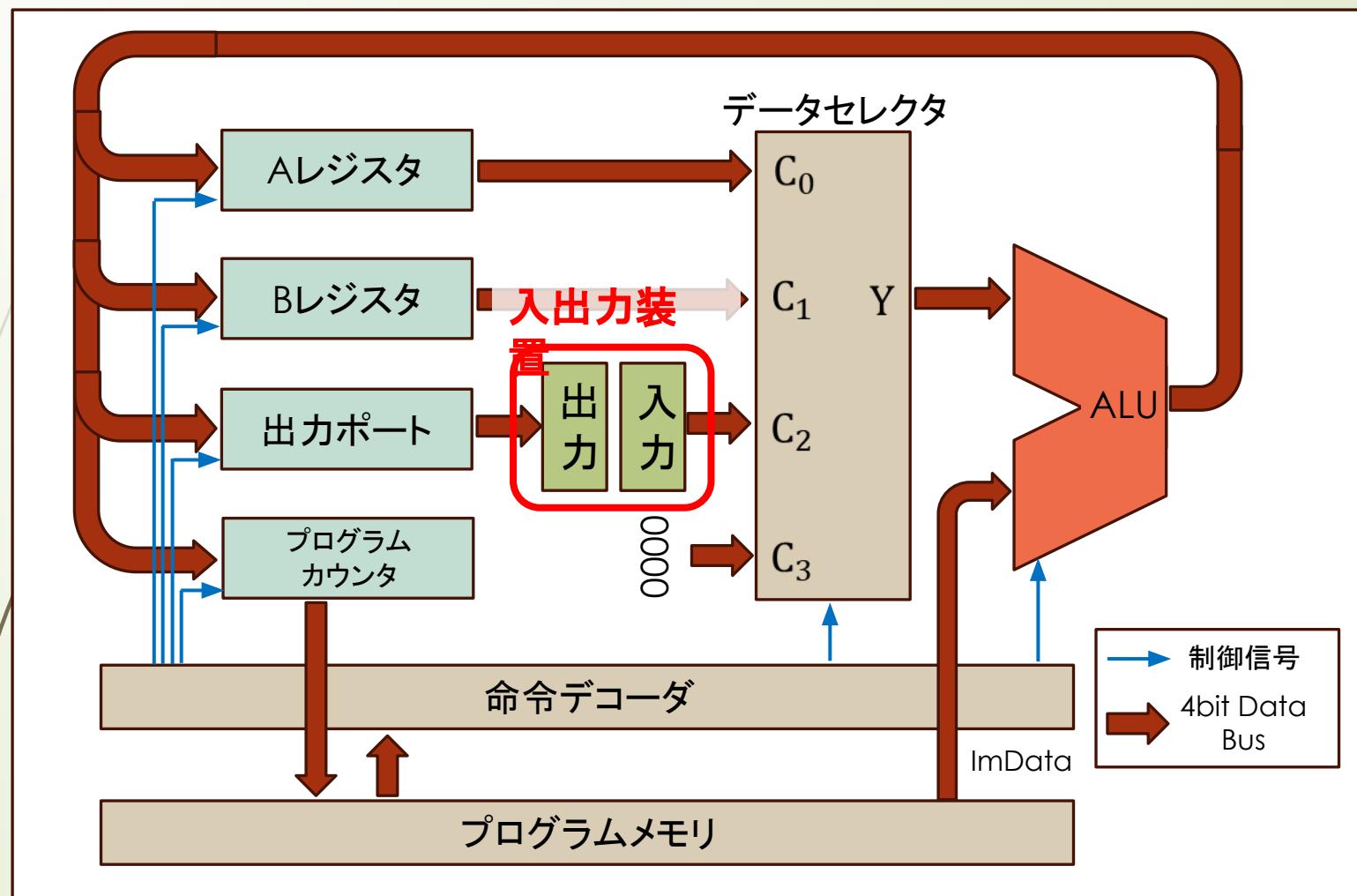


Fig: 某4bit CPUのアーキテクチャ[1]

おさらい② CPUの構成要素

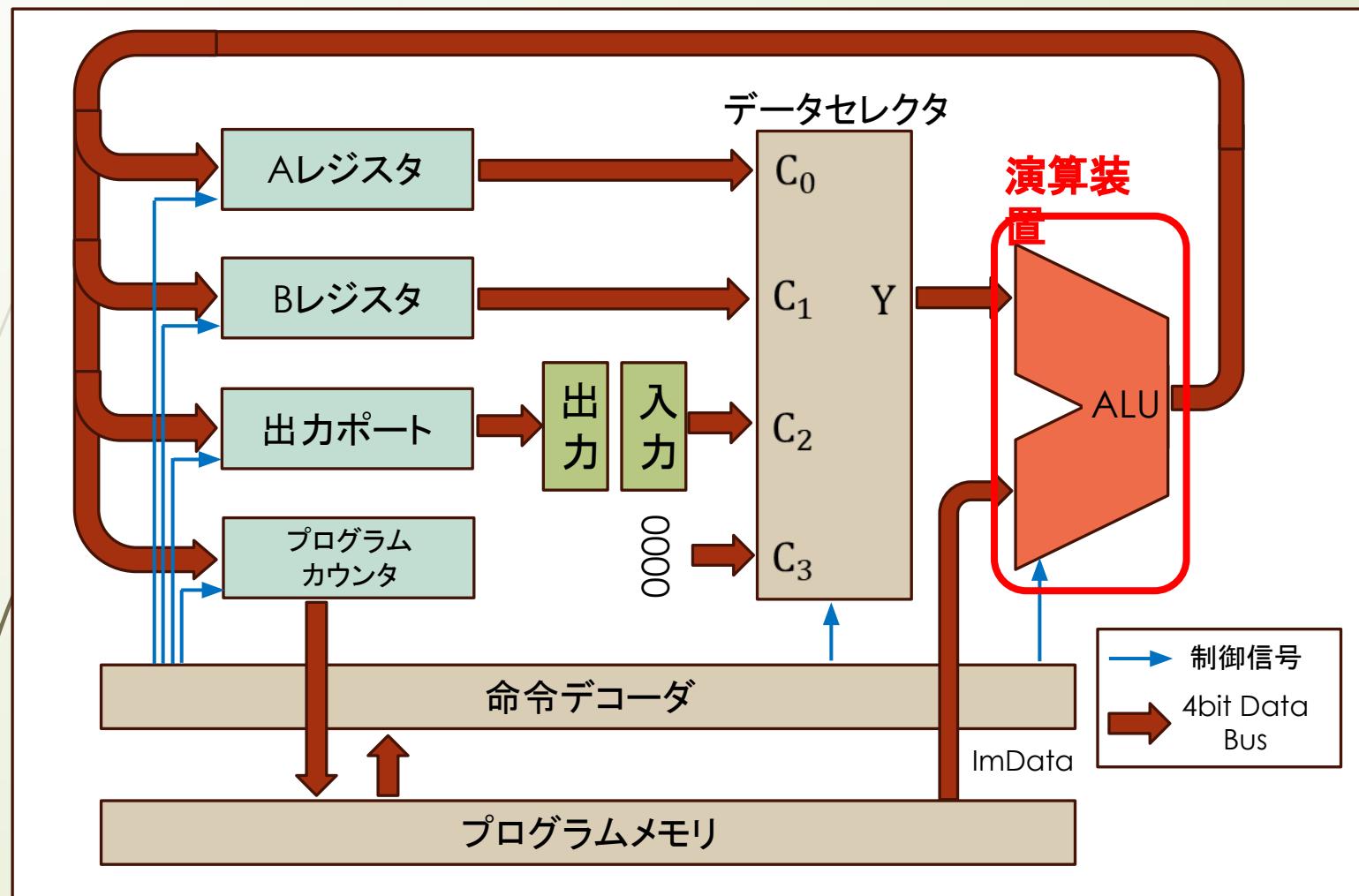


Fig: 某4bit CPUのアーキテクチャ[1]

おさらい② CPUの構成要素

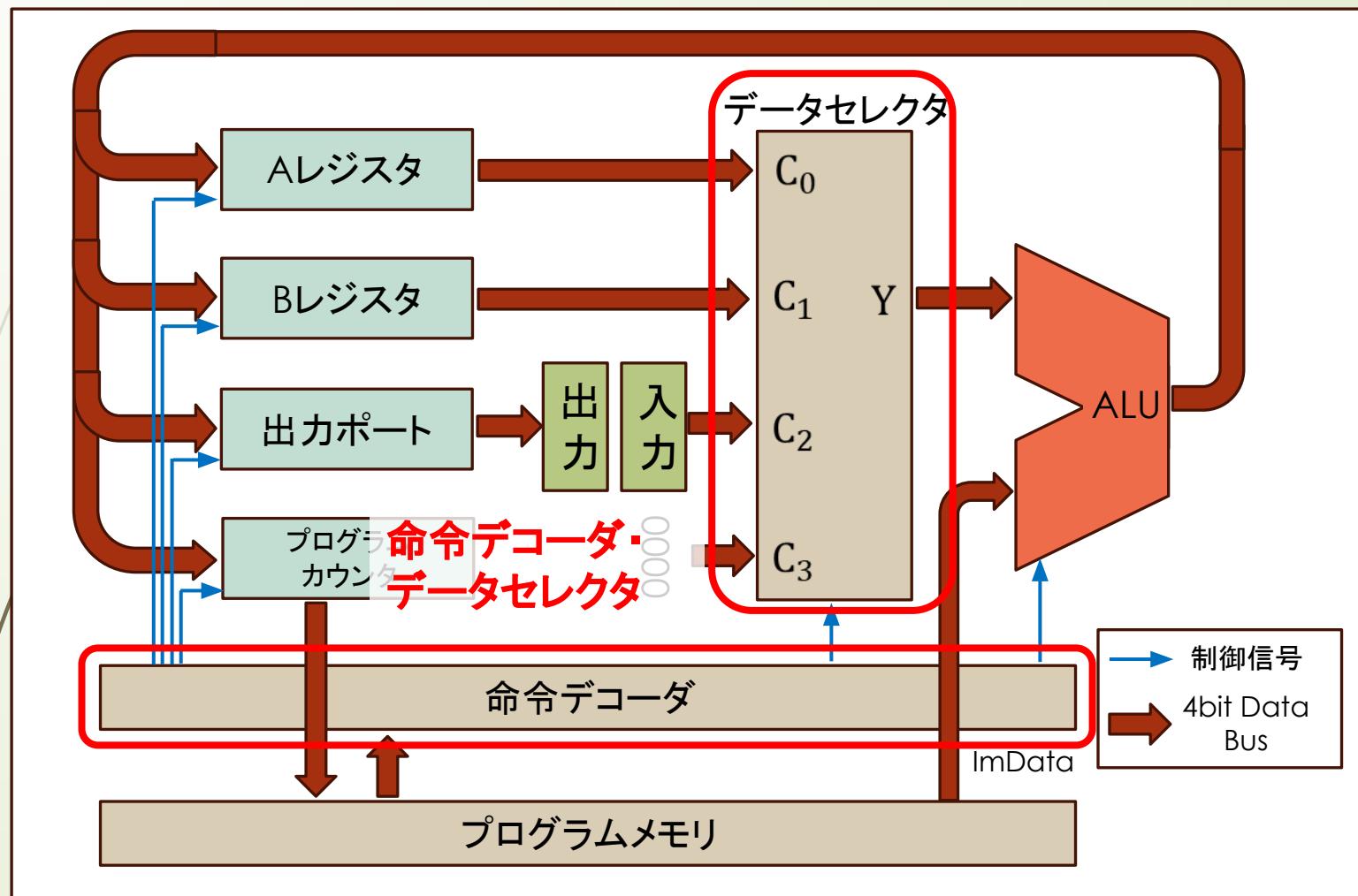


Fig: 某4bit CPUのアーキテクチャ[1]

おさらい② CPUの構成要素

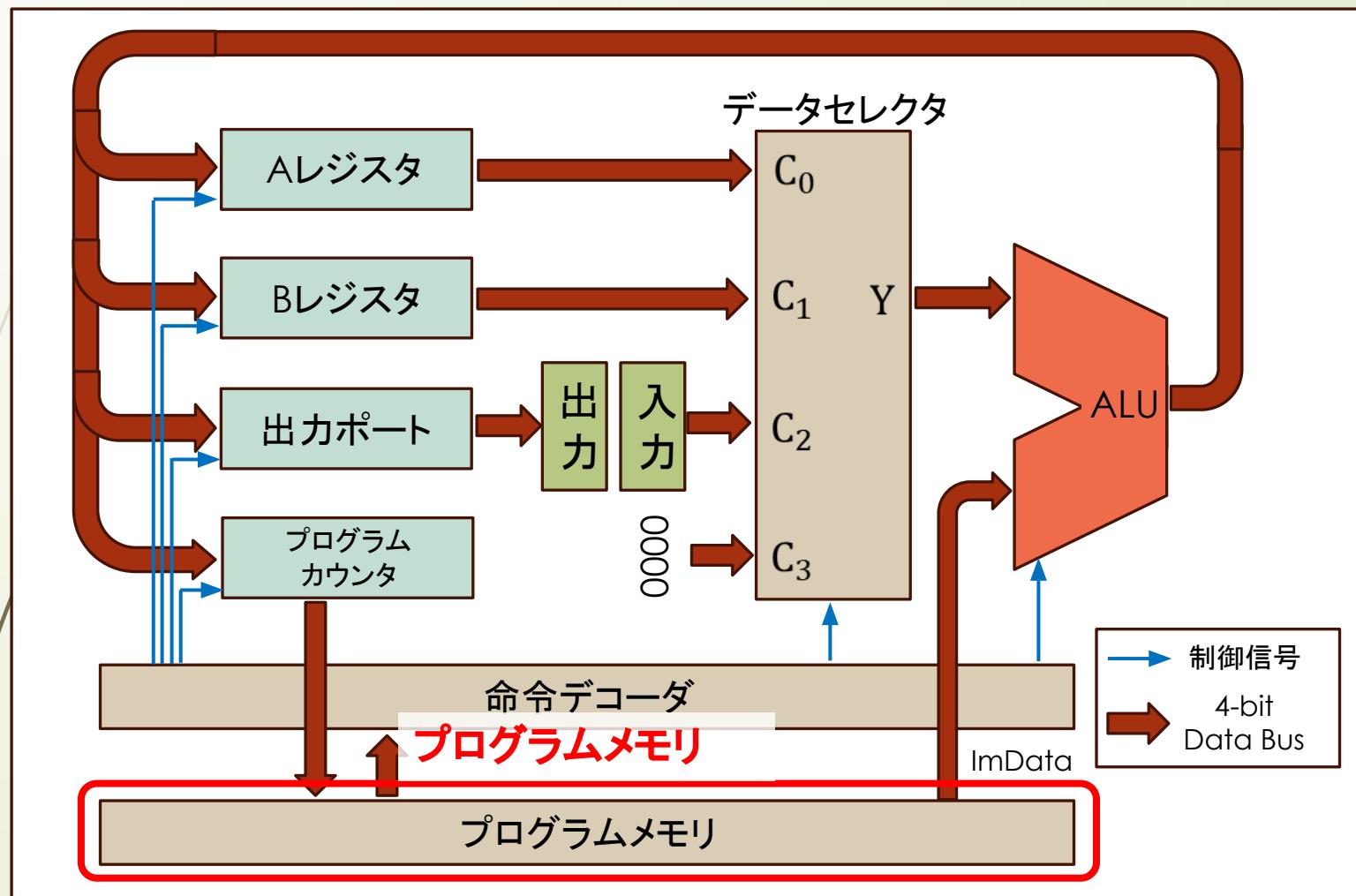
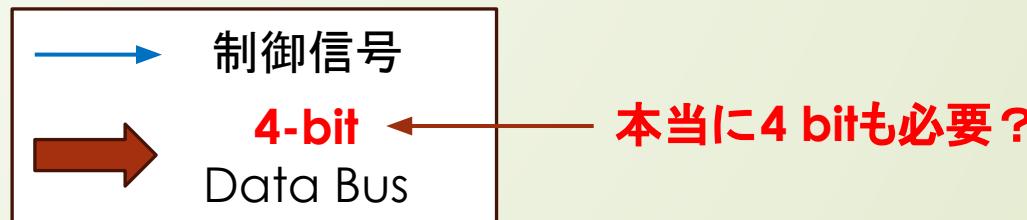


Fig: 某4bit CPUのアーキテクチャ[1]

CPUの構成要素をアナログにしてみる①(概説)

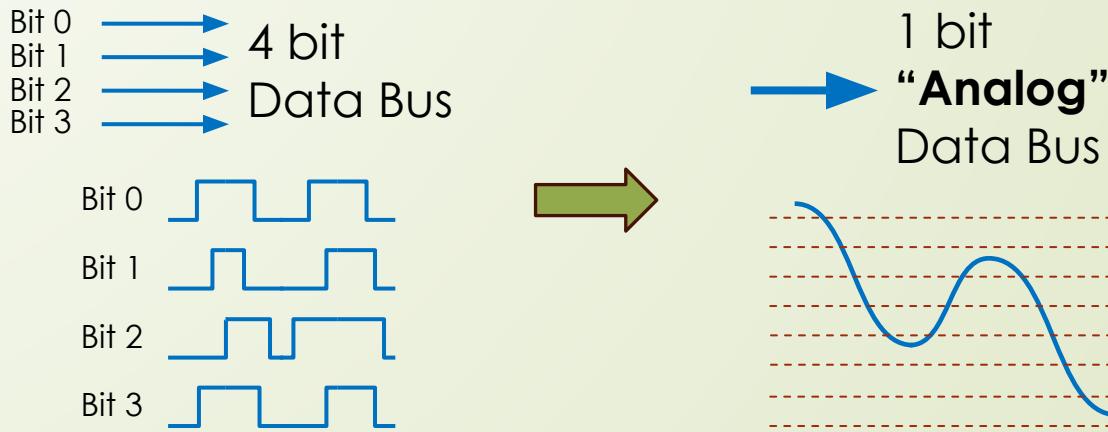
- ここから本題！！！
- 半ば当然のように受け入れられていますが…



- そもそも4 bit必要なのはなぜ?
 - 信号線の電圧の H(High)/L(Low) → 1/0 に対応させた2進法で表現しているため。
 - 例:H,L,L,H ⇒ 1001(2) ⇒ 9,
L,H,H,H ⇒ 0111(2) ⇒ 7

CPUの構成要素をアナログにしてみる①(概説)

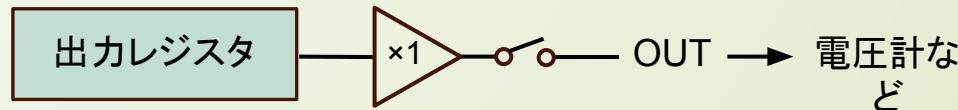
- 4 bitも必要なのは、信号電圧のH/Lのみで区別するため。
⇒ 信号電圧それ自体をそのまま情報にすれば1本でOK！！



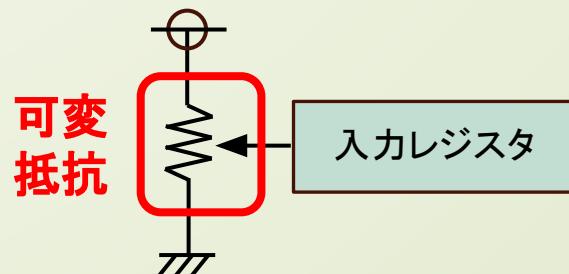
- 信号電圧を情報として扱うためには、CPUの各構成要素に工夫が必要

CPUの構成要素をアナログにしてみる③(入出力装置)

- 【復習】入出力装置(I/O)
 - 外部からデータを入力し、外部へデータを出力する。
- 必要な回路
 - 出力: 電圧をそのまま出力する回路。



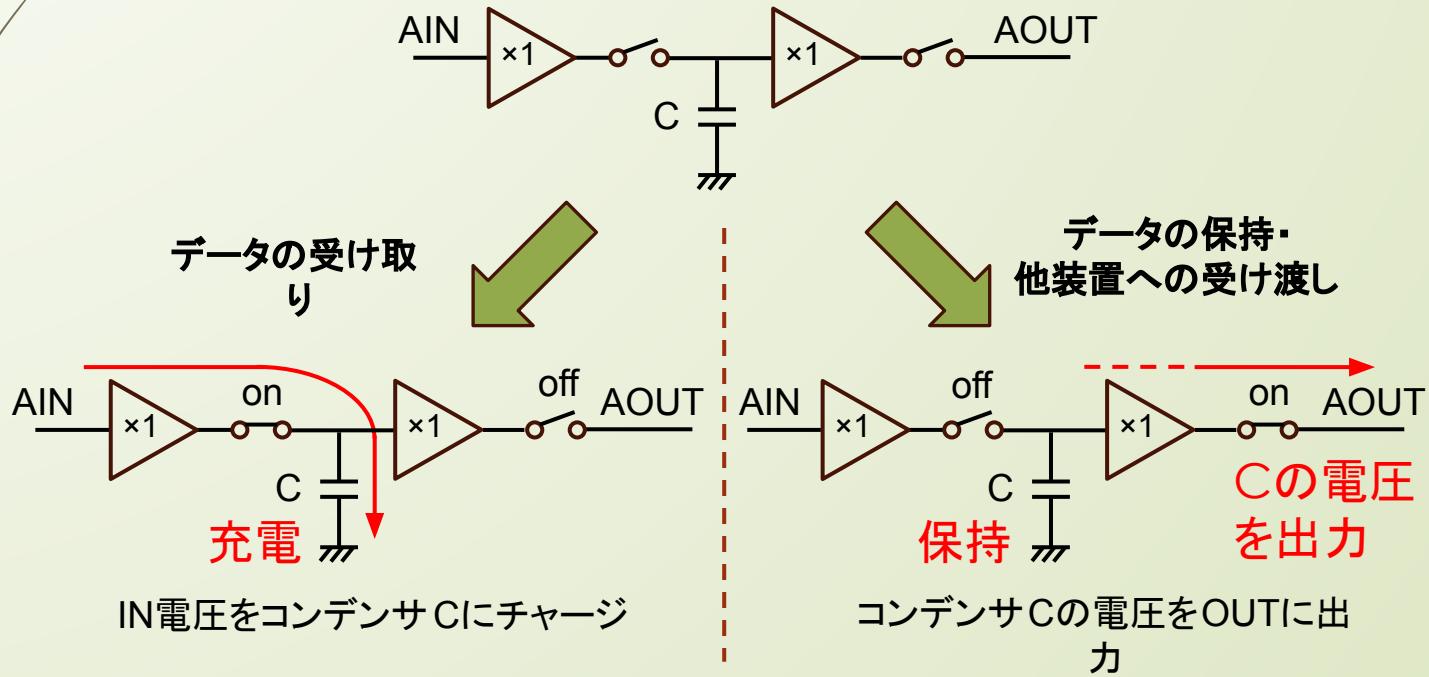
- 入力: 電圧を入力する回路。



CPUの構成要素をアナログにしてみる②(記憶装置)

□ 【復習】記憶装置

- データを受け取り保持し、必要時に他装置にデータを渡す。
- 必要な回路：サンプル & ホールド回路



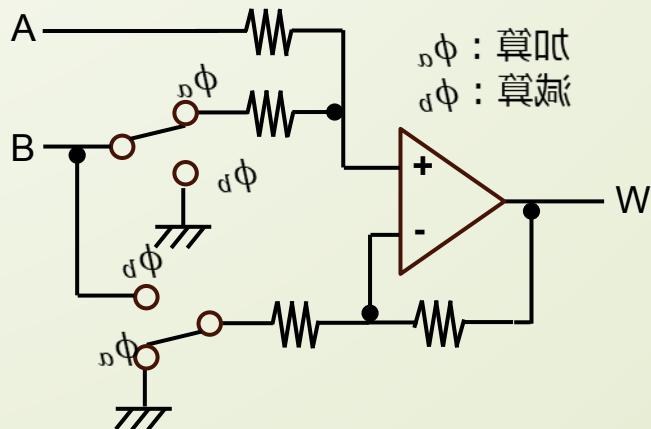
CPUの構成要素をアナログにしてみる④(演算装置)

□ 演算装置(ALU)

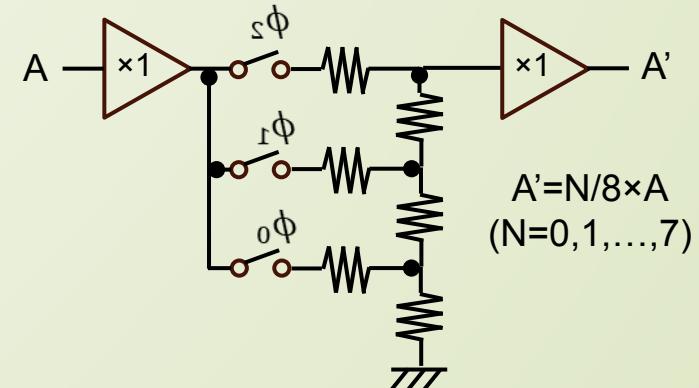
□ 複数のデータ同士を演算し、演算結果を記憶装置に渡す。

□ 必要な回路:オペアンプによる演算回路

□ 加減乗除、対数、逆対数、絶対値、微積、比較... etc.
を比較的簡易な回路で実現できる= **Analog CPUの強み！！**



加減算・比較回路



除算回路(R-2R)

Fig:演算回路の一例

CPUの構成要素をアナログにしてみる⑤(その他)

- 【復習】命令デコーダ・データセレクタ
 - プログラムに書かれた命令を1個ずつ解読し、各装置の動作を決める
- 【復習】プログラムメモリ
 - プログラムを格納する(特別な)記憶装置。
- これら要素には正確さ(≒精度)が必要。
また、そもそも計算中の計算データから動作を変更するような使い方はあまり想定されない。
よって、アナログにするメリットは薄い。

⇒ 通常のCPU同様、デジタル回路を用いて実現する。

Analog CPUの仕様

- ▶ 以上を踏まえて、以下のような仕様のCPUを作成した。
- ▶ レジスタ構成
 - ▶ アナログレジスタ
A, B, W, IN, OUT の 5個の1-bit アナログレジスタ
 - ▶ デジタルレジスタ
D, E の 2個の8-bit レジスタ。PIC16F18877でエミュレート。
- ▶ ALUの仕様
 - ▶ A, Bレジスタの電圧から演算した結果をWレジスタに格納する。
 - ▶ データ受け渡し : $W = A, W = B, A = W, B = W$
 - ▶ 計算 : $W = A + B, A - B, A + B \times \frac{N}{32}, A - B \times \frac{N}{32}, B \times \frac{N}{32}$
 - ▶ 比較 : $W = \begin{cases} \text{VDD} & (\text{if } A > B) \\ 0 \text{ V} & (\text{if } A < B) \end{cases}$

Analog CPUの仕様

- 以上を踏まえて、以下のような仕様のCPUを作成した。
- デジアナ混載ブロック
 - ADC: 1-bit アナログバスの電圧を読み取ってAD変換し、D or E レジスタに格納する。
PIC16F18877でエミュレートする。
 - DAC:D or E レジスタの値、もしくは、リテラル値をDA変換した電圧を A or B or Wレジスタに格納する。
- プログラムメモリ・命令体系
 - PIC16F18877でエミュレートする。
 - プログラムメモリの構成:ノイマン型アーキテクチャ
 - メモリの実装:RAM 2kB, EEPROM 256B

Analog CPUのアーキテクチャ

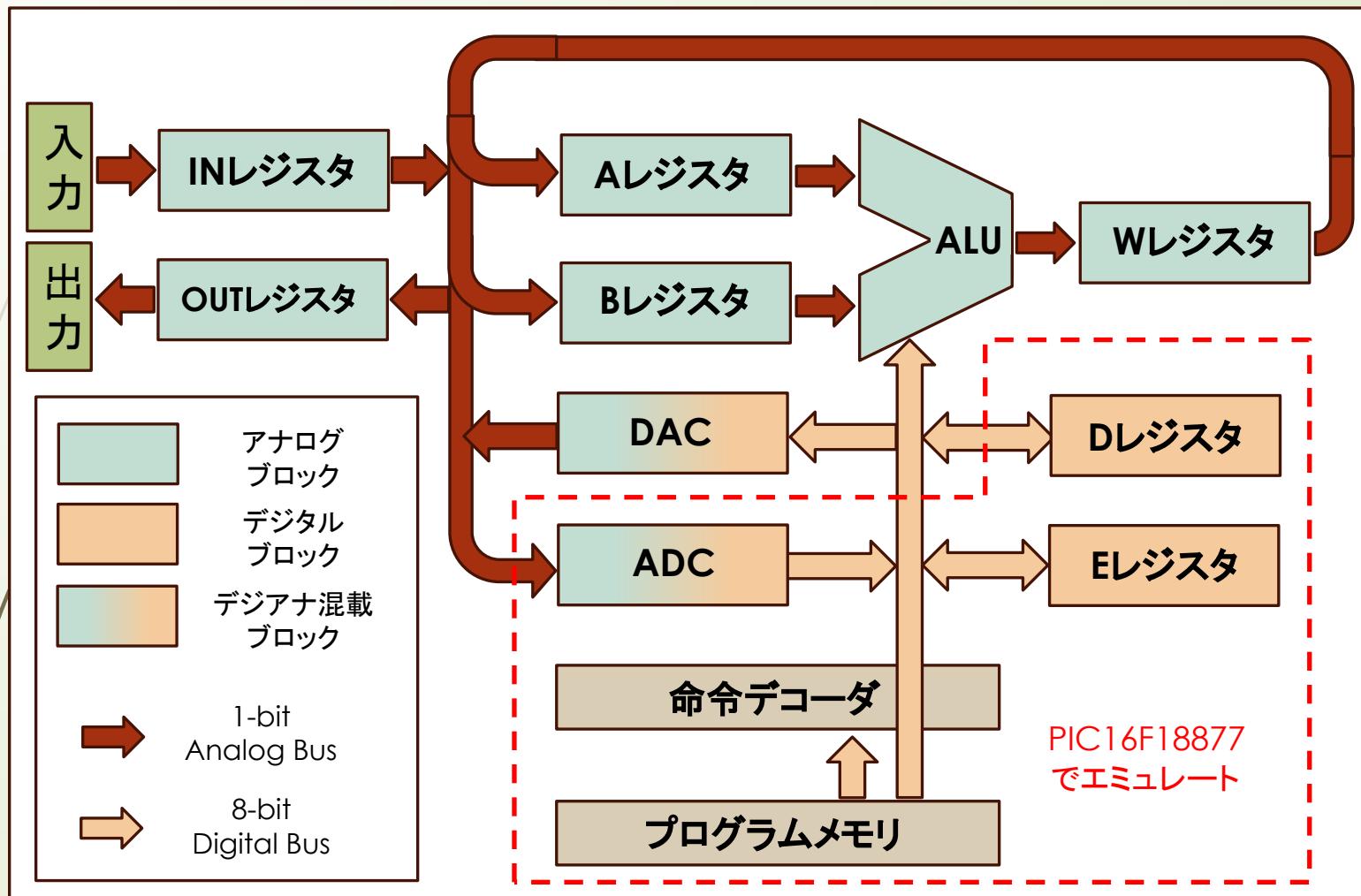
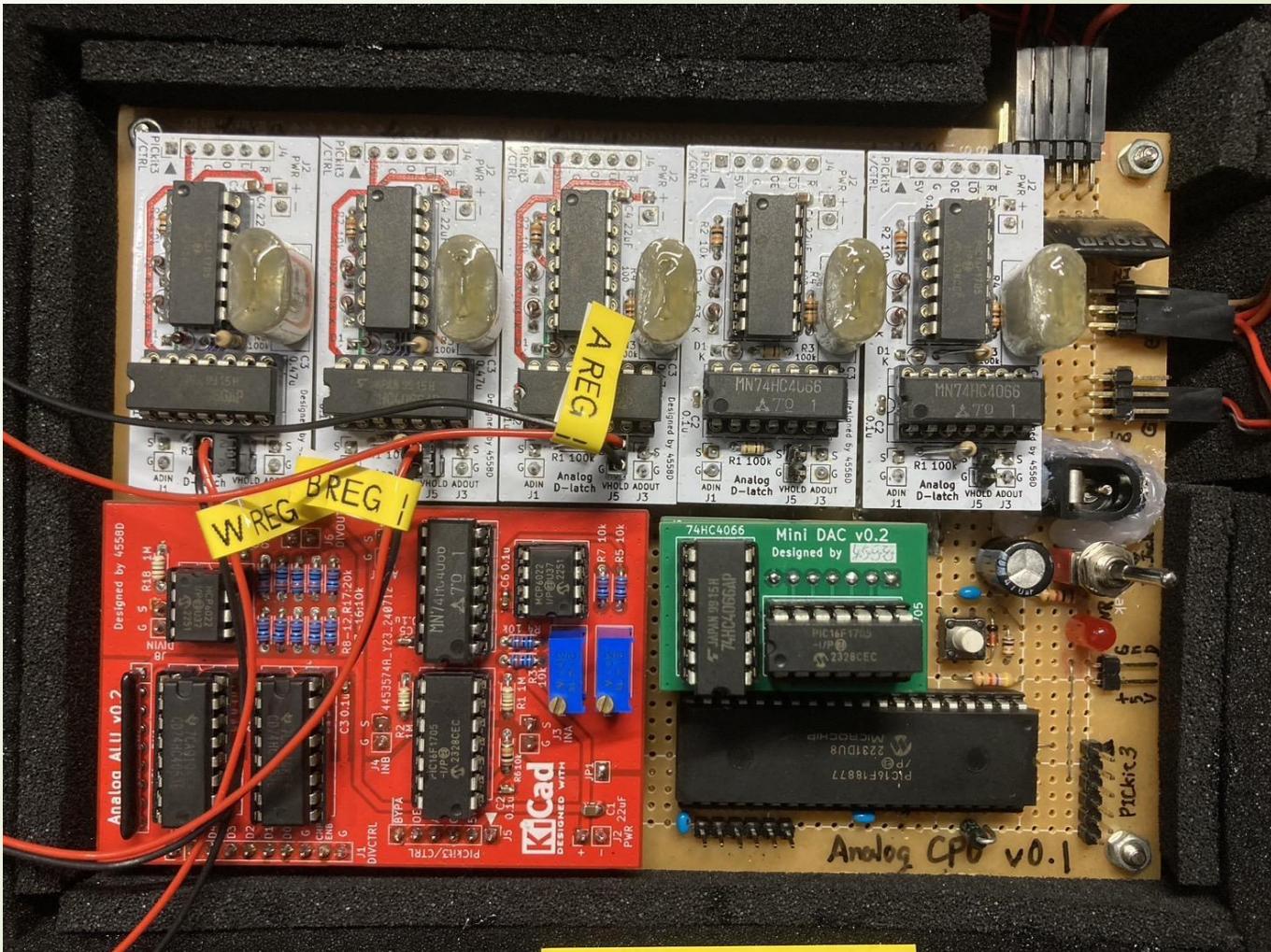


Fig: Analog CPUのアーキテクチャ

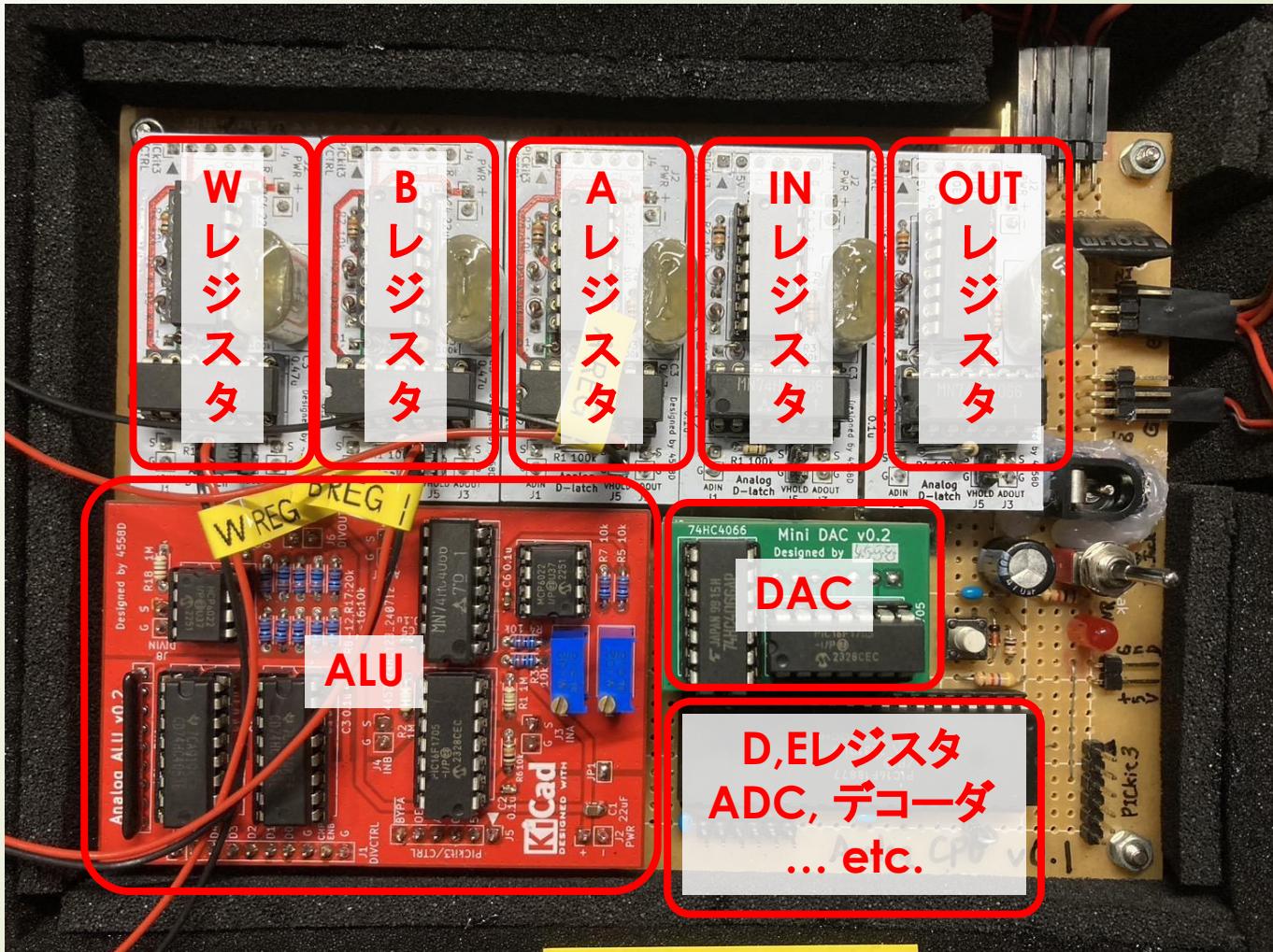
作成したAnalog CPU

□ CPU本体



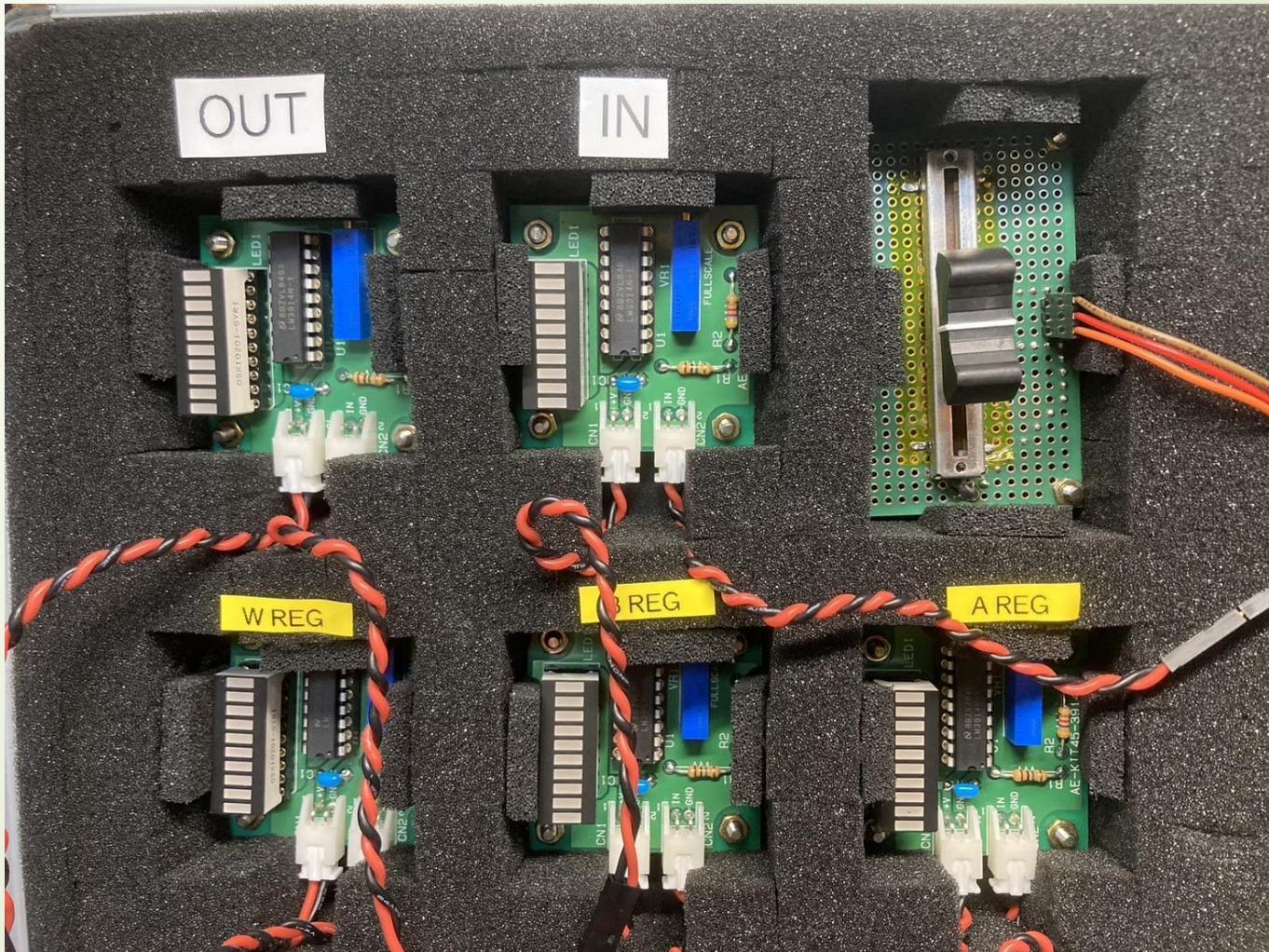
作成したAnalog CPU

□ CPU本体



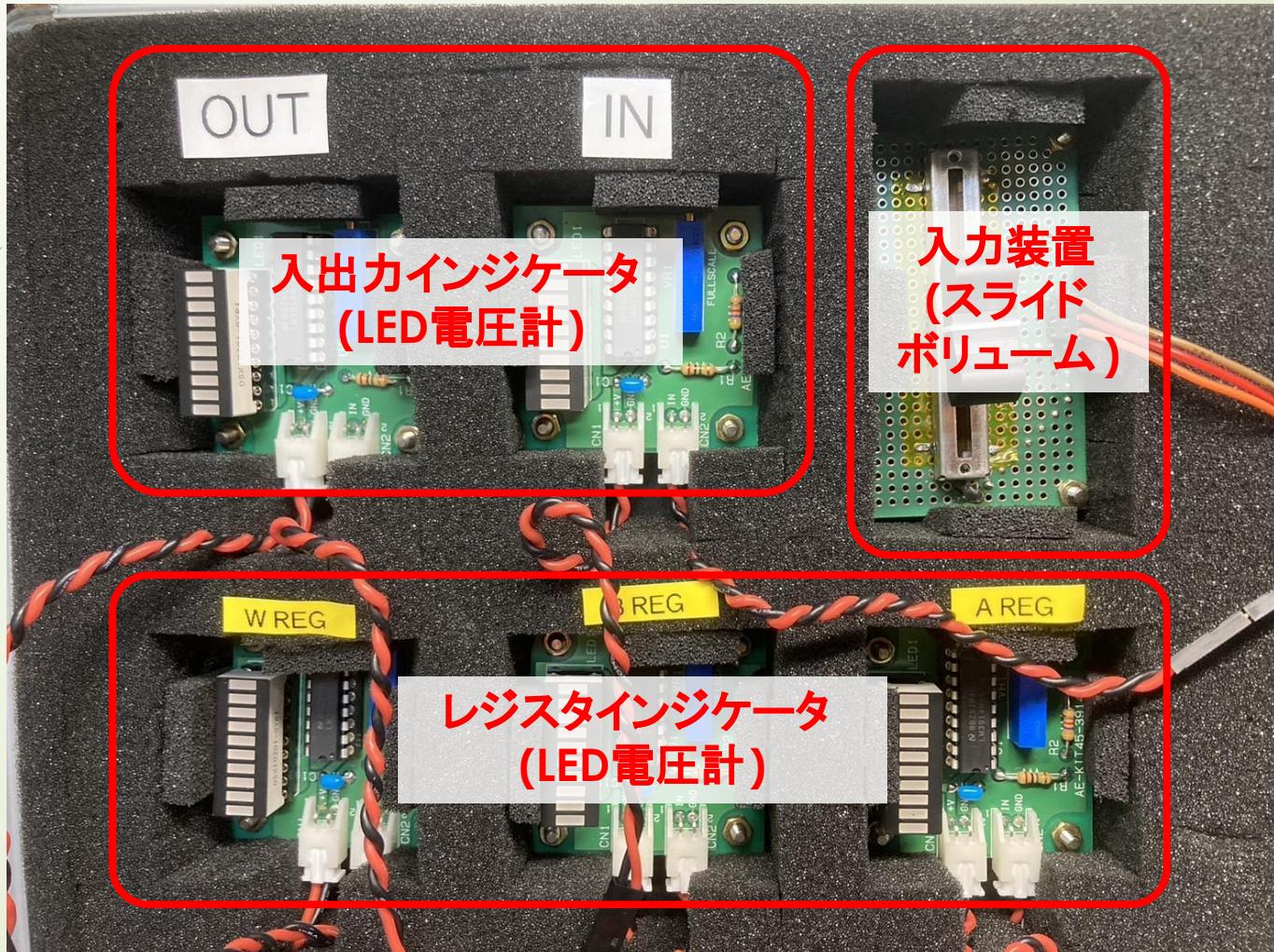
作成したAnalog CPU

□ 入出力装置・インジケータ



作成したAnalog CPU

□ 入出力装置・インジケータ



Analog CPUの命令概要

- 命令数: 78
- 命令の分類
 - CPU制御系:NOP, RESET, PAUSEなど
 - ジャンプ命令:JP, JPZ, JPNZ, JPC, JPNC
 - デジタルレジスタ操作・演算:LDD, ADDDなど
 - レジスタ-メモリ間転送:LDDRM, LDDMR
 - デジタルレジスタ入出力:IND, OUTD
 - アナログレジスタ操作・演算:LDA, ADDAなど
 - アナログレジスタ入出力:INA, OUTA
- 概念実証のため、命令数は必要最低限にした。

Analog CPUの命令一覧

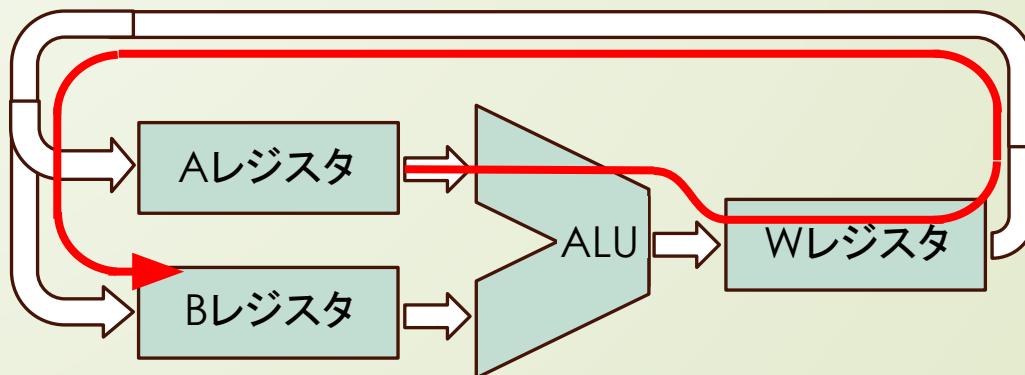
***** Mnemonic List *****						
code	length	order	operand1	operand2		
0x00	1	NOP			0x40	1
0x01	1	RESET			0x41	1
0x02	1	HALT			0x42	1
0x03	1	RESETA			0x43	1
0x04	1	PAUSE			0x44	1
0x08	2	JP	n		0x45	1
0x09	2	JPNZ	n		0x46	2
0x0a	2	JPZ	n		0x47	1
0x0b	2	JPNC	n		0x48	2
0x0c	2	JPC	n		0x49	1
0x10	1	LDD	D		0x4a	1
0x11	2	LDD	D	E	0x4b	1
0x12	1	LDD	E	D	0x4c	1
0x13	2	LDD	E	n	0x4d	1
0x20	1	ADDD	D	E	0x4e	1
0x21	2	ADDD	D	n	0x4f	1
0x22	1	ADDD	E	D	0x50	1
0x23	2	ADDD	E	n	0x51	1
0x24	1	SUBD	D	E	0x52	1
0x25	2	SUBD	D	n	0x53	2
0x26	1	SUBD	E	D	0x54	1
0x27	2	SUBD	E	n	0x55	1
0x28	1	LDDRM	D	D	0x56	1
0x29	1	LDDRM	E	E	0x57	1
0x2a	1	LDDRM	D	E	0x58	1
0x2b	1	LDDRM	E	D	0x59	1
0x2c	2	LDDRM	D	n	0x5a	1
0x2d	2	LDDRM	E	n	0x5b	1
0x2e	1	LDDMR	D	D	0x5c	1
0x2f	1	LDDMR	E	E	0x5d	1
0x30	1	LDDMR	D	E	0x5e	1
0x31	1	LDDMR	E	D	0x5f	1
0x32	2	LDDMR	n	D	0x60	1
0x33	2	LDDMR	n	E	0x61	1
0x36	1	IND	D		0x62	1
0x37	1	IND	E		0x63	1
0x38	2	OUTD	D	n	0x64	1
0x39	2	OUTD	E	n	0x65	1
					0x66	1
					0x67	1
					0x68	2
					0x69	2
					0x6a	2
					0x6b	2
					0x6c	1
					0x6d	1
					0x6e	1
					0x6f	1
					0x70	1
					0x71	1
					0x72	1
					0x73	1
					0x74	1
					0x75	1
					0x76	1
					0x77	1
					0x78	1
					0x79	1
					0x7a	1
					0x7b	1
					0x7c	1
					0x7d	1
					0x7e	1
					0x7f	1
					0x80	1
					0x81	1
					0x82	1
					0x83	1
					0x84	1
					0x85	1
					0x86	1
					0x87	1
					0x88	1
					0x89	1
					0x8a	1
					0x8b	1
					0x8c	1
					0x8d	1
					0x8e	1
					0x8f	1
					0x90	1
					0x91	1
					0x92	1
					0x93	1
					0x94	1
					0x95	1
					0x96	1
					0x97	1
					0x98	1
					0x99	1
					0x9a	1
					0x9b	1
					0x9c	1
					0x9d	1
					0x9e	1
					0x9f	1
					0x100	1
					0x101	1
					0x102	1
					0x103	1

Analog CPUの命令一覧

***** Mnemonic List *****				operand1	operand2		
code	length	order					
0x00	1	NOP					
0x01	1	RESET					
0x02	1	RESETA					
0x03	1	PAUSE					
0x04	1						
0x08	2	JP	n				
0x09	2	JPNZ	n				
0x0a	2	JPNC	n				
0x0b	2	JPC	n				
0x0c	2						
0x10	1	LDD	D	E			
0x11	2	LDD	D	n			
0x12	1	LDD	E	D			
0x13	2	LDD	E	n			
0x20	1	ADD	D	E			
0x21	2	ADD	E	n			
0x22	1	ADD	E	D			
0x23	2	ADD	E	n			
0x24	1	SUBD	D	E			
0x25	2	SUBD	D	n			
0x26	1	SUBD	E	D			
0x27	2	SUBD	E	n			
0x28	1	LDDRM	D	D			
0x29	1	LDDRM	E	E			
0x2a	1	LDDRM	D	E			
0x2b	1	LDDRM	E	D			
0x2c	2	LDDRM	D	n			
0x2d	2	LDDRM	D	n			
0x2e	1	LDDMR	D	D			
0x2f	1	LDDMR	E	E			
0x30	1	LDDMR	D	E			
0x31	1	LDDMR	E	D			
0x32	2	LDDMR	n	D			
0x33	2	LDDMR	n	F			
0x36	1	IND	D				
0x37	1	IND	E				
0x38	2	OUTD	D	n			
0x39	2	OUTD	E	n			
0x40	1	LDA	A				
0x41	1	LDA	A				
0x42	1	LDA	D				
0x43	1	LDA	B				
0x44	1	LDA	W				
0x45	1	LDA	W				
0x50	2	LDDA	A				
0x51	1	LDDA	A				
0x52	1	LDDA	A				
0x53	2	LDDA	B				
0x54	1	LDDA	B				
0x55	1	LDDA	D				
0x58	1	LDAD	D				
0x59	1	LDAD	D				
0x5a	1	LDAD	D				
0x5b	1	LDAD	E				
0x5c	1	LDAD	E				
0x5d	1	LDAD	E				
0x60	1	TNA	A				
0x61	1	TNA	B				
0x62	1	TNA	W				
0x63	1	OUTA	I				
0x64	1	ADDA					
0x65	1	SUBA					
0x67	1	CMPA					
0x68	2	ADDA				n	
0x69	2	SUBA				n	
0x6a	2	DIVA				n	
0x6b	2	ANALOG REGISTERS				n	
0x6c	1	ADDA				D	
0x6d	1	SUBA				D	
0x6e	1	DIVA				D	
0x6f	1	CMPA				D	
0x70	1	ADDA				E	
0x71	1	SUBA				E	
0x72	1	DIVA				E	
0x73	1	CMPA				E	
0x100	1	END					
0x101	1	END					
0x103	1	DB					

Analog CPUの動作

- LDA A B (Aレジスタの電圧をBレジスタにロードする)を元に動作を説明
- 下図のように、
Aレジスタ→ALU→Wレジスタ→Bレジスタ
の順に電圧を伝える。

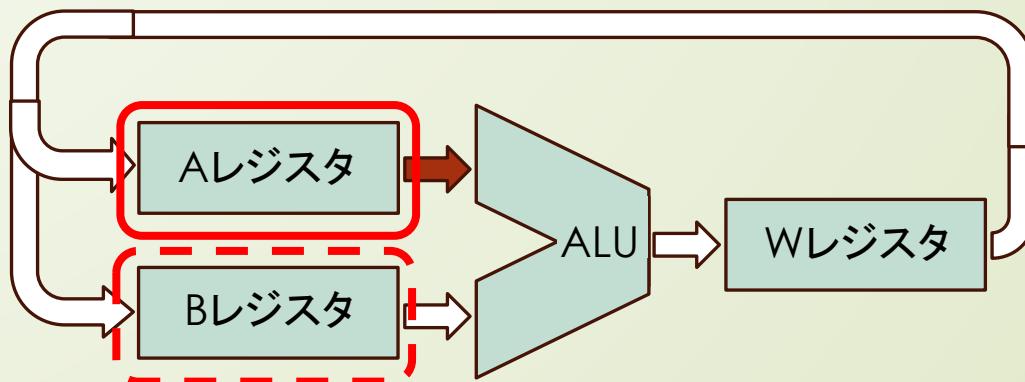


Analog CPUの動作

□ LDA A B

(Aレジスタの電圧をBレジスタにロードする)

- ① Aレジスタを出力許可、Bレジスタを出力禁止にし、
Aレジスタの電圧をALUに送る

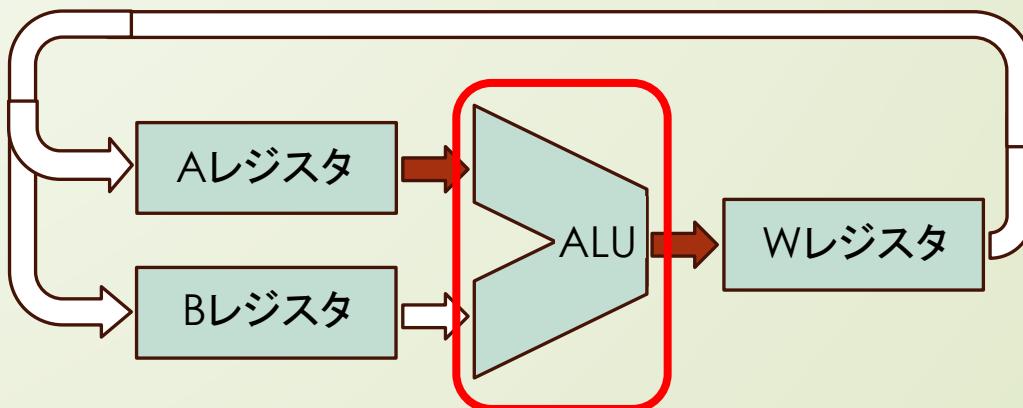


Analog CPUの動作

□ LDA A B

(Aレジスタの電圧をBレジスタにロードする)

- ② ALUを出力許可し、Wレジスタに電圧をロードする。

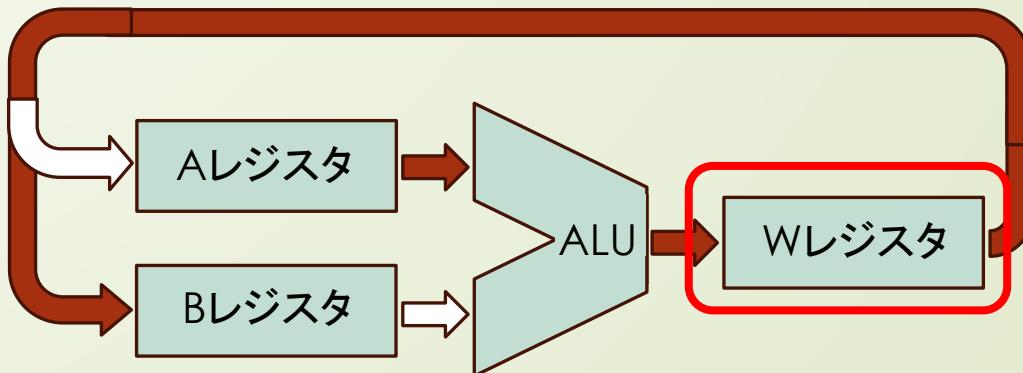


Analog CPUの動作

□ LDA A B

(Aレジスタの電圧をBレジスタにロードする)

- ③ Wレジスタを出力許可し、Bレジスタに電圧をロードする。

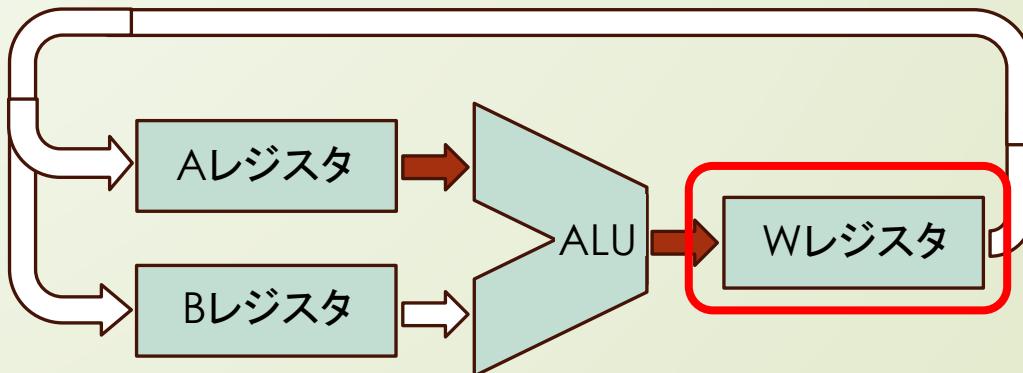


Analog CPUの動作

□ LDA A B

(Aレジスタの電圧をBレジスタにロードする)

- ④ Bレジスタのロードを止め、Wレジスタの出力を止める。

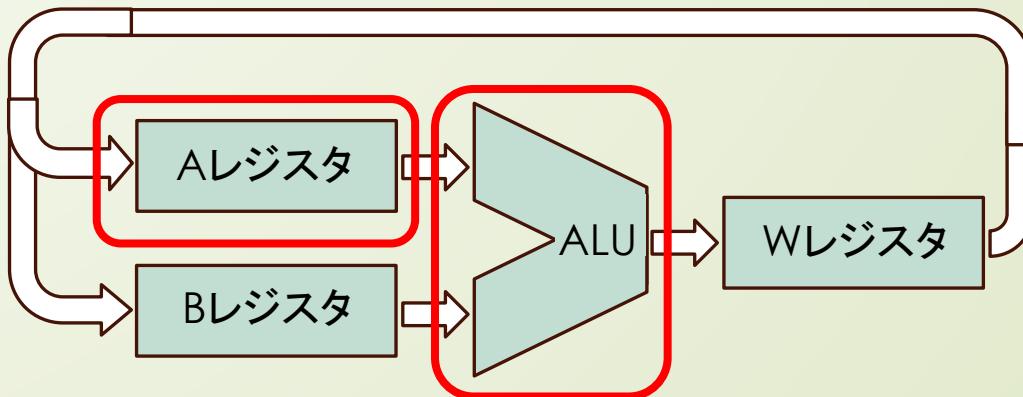


Analog CPUの動作

□ LDA A B

(Aレジスタの電圧をBレジスタにロードする)

- ⑤ ALU, Aレジスタの出力を止めてから次の命令へ。



動作デモ①

フィボナッチ数の計算

- INPUTに入力した電圧を1としたときのフィボナッチ数※に該当する電圧を、OUTPUTに出力する。

例：INPUTの電圧がLEDバー1目盛り分の場合

OUTPUT電圧 = 1, 2, 3, 5, 8... 目盛り分
の電圧を順番に出力していく。

※フィボナッチ数

フィボナッチ数列 $a_{n+2} = a_{n+1} + a_n$, $a_0 = a_1 = 1$ を
計算することで得られる数。小さい方から順に、

1, 1, 2, 3, 5, 8, 13, 21, 34, ...

動作デモ②

整数値の加減乗除

- IND DでDレジスタに格納した整数に、
IND EでEレジスタに格納した整数を加減乗除する。

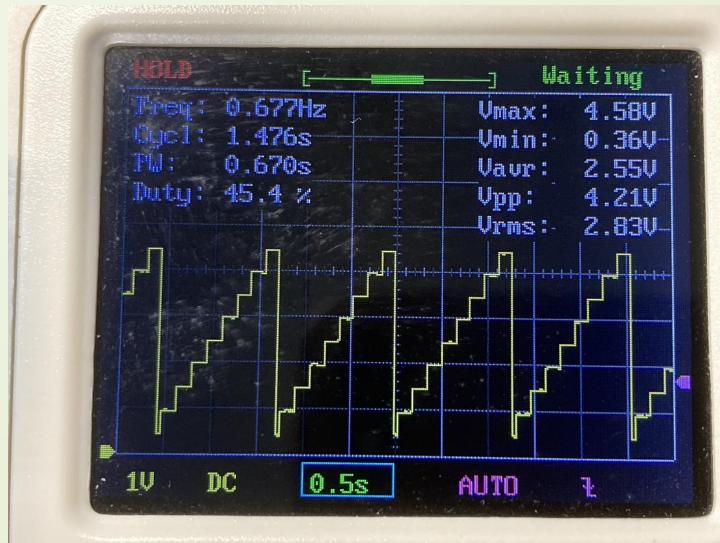
- 具体的には、以下の計算して出力する。
 - 加算: $OUTPUT = (D \times INPUT) + (E \times INPUT)$
 - 減算: $OUTPUT = (D \times INPUT) - (E \times INPUT)$
 - 乗算: $OUTPUT = (D \times INPUT) \times (E \times INPUT)$
 - 除算: $OUTPUT = (D \times INPUT) \div (E \times INPUT)$

- INPUT電圧=1としたときの、数値D,Eの計算を
アナログ電圧で計算している。

動作デモ③

LEDバーの上昇 (ノコギリ波)

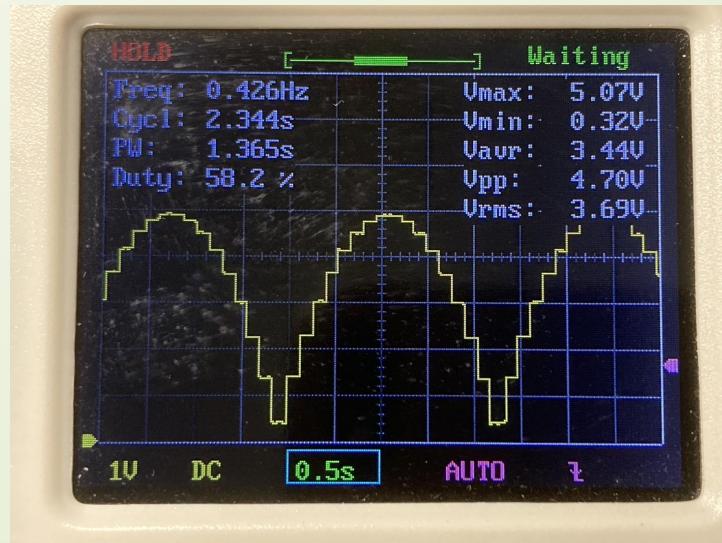
- アナログCPUならではのプログラム、その1
- 1目盛り分の電圧を繰り返し足すことで
OUTPUTの出力電圧を大きくする。
- OUTPUTの電圧 > INPUTの電圧 となったら
足し合わせるのを止め、OUTPUTの電圧をリセットする。



動作デモ④

LEDバーの上昇 (サイン波)

- アナログCPUならではのプログラム、その2
- INPUT電圧を元に、OUTPUTの電圧を
サイン波っぽく変化させる。
- $OUT = OUT \pm IN * N/32$
をNを減らしながら計算している。



まとめ

- CPUが、①記憶装置、②入出力装置(I/O)、
③演算装置(ALU)、④命令デコーダ・データセレクタ、
⑤プログラムメモリ の 5個の構成要素から成立する
ことを確認した。
- CPUの各構成要素をアナログ回路に置き換えることで、信号線
の "電圧そのもの" を情報として処理できるCPUを作成した。
- 作成したアナログCPUのアーキテクチャ、命令体系、命令処理
の動作を説明した。
- 作成したアナログCPUでいくつかのデモプログラムを実行し、ア
ナログ電圧を用いた演算ができることを確認した。

今後の展望 (やりたい改良)

- アーキテクチャの最適化
 - INレジスタを削除し、汎用アナログレジスタを増やす。
 - ALU前段にデータセレクタを増設し、計算の自由度を向上させる。
- アナログ記憶装置の改良
 - アーキテクチャを見直し、データ受け取り・データ出力の速度、および、データ保持時間を向上させる。
- ALUの改良
 - 加減算・除算の精度を向上させる。
 - 比較機能の結果出力を、H/Lのデジタル信号で出力する。専用フラグビットを増設し、プログラムの処理を簡易化する。

今後の展望 (次世代機構想)

- 正負両電圧対応: ±10Vぐらいまでは扱いたい。
- アナログレジスタ数: 8~10個以上は欲しい。
- コントロールロジックの脱マイコン化-
→ アナログ周りが解決するまではお預けかも。

- 高速化: クロック1kHzぐらいは出せるようにしたい。
- 高精度化: システムオフセットの除去、ノイズ軽減
- ワンボード化: ノイズ対策・高速化のためにも必要

- ALUの改良: 加減算、乗算、対数、逆対数、絶対値
ぐらいは最低限出来るようにしたい。できれば微積とかも。



補足資料

- デモのソースコード
- 回路図

デモ①

フィボナッチ数の計算

□ ソースコード

```
INA A          ; Input A Register  
ADDA          ; W = A+B  
OUTA W        ; Output W Register  
LDA B W       ; B = W  
PAUSE          ; Pausing until Key Pushed  
ADDA          ; W = A+B  
OUTA W        ; Output W Register  
LDA A W       ; A = W  
PAUSE          ; Pausing until Key Pushed  
JP 0x01        ; Return to 2nd line  
END
```

デモ②

整数値の加減乗除

□ ソースコード (加算)

```
RESETA

IND D      ; Input D: Target Number
INA B      ; Set Areg as Unit Voltage

ORG 0x10   ; Setting Areg Voltage to D
ADDA
LDA A W
SUBD D 001
JPNZ 0x10  ; When finished, A==D

IND E      ; Input E: Base Number
INA B      ; Set Areg as Unit Voltage

ORG 0x20   ; D=E
LDD D E

ORG 0x22   ; Calculating W<-D-E
ADDA
LDA A W
SUBD E 001
JPNZ 0x22  ; A=A-B
             ; E=E-1

LDA A W
OUTA W    ; Display Result
PAUSE
JP 0x00

END
```

デモ②

整数値の加減乗除

□ ソースコード (減算)

```
RESETA

IND D      ; Input D: Taget Number
INA B      ; Set Areg as Unit Voltage

ORG 0x10   ; Setting Areg Voltage to D
ADDA
LDA A W
SUBD D 001
JPNZ 0x10  ; When finished, A==D

IND E      ; Input E: Base Number
INA B      ; Set Areg as Unit Voltage

ORG 0x20   ; D=E
LDD D E

ORG 0x22   ; Calculating W<-D-E
SUBA
LDA A W
SUBD E 001
JPNZ 0x22  ; E=E-1

LDA A W
OUTA W    ; Display Result
PAUSE

JP 0x00

END
```

デモ②

整数値の加減乗除

□ ソースコード (乗算)

RESETA

```
IND D      ; Input D: Taget Number
INA B      ; Set Breg as Unit Voltage

ORG 0x10   ; Setting Areg Voltage to D
ADDA
LDA A W
SUBD D 001
JPNZ 0x10  ; When finished, A==D

IND E      ; Input E: Base Number
LDA B A
; A=B

ORG 0x20
SUBD E 001
JPZ 0x30   ; E=0 -> Finish Calculation
ADDA
LDA A W
JP 0x20    ; A=A

ORG 0x30
LDA A W
OUTA W    ; Calculating W<-D-E
; Display Result

PAUSE
JP 0x00

END
```

デモ②

整数値の加減乗除

□ ソースコード (除算)

```
RESETA

IND D      ; Input D: Taget Number
INA B      ; Set Breg as Unit Voltage

ORG 0x10    ; Setting Areg Voltage to D
ADDA
LDA A W    ; A=A+B
SUBD D 001
JPNZ 0x10   ; When finished, A==D

ORG 0x20
IND E      ; Input E: Base Number

ORG 0x22    ; Calculate D=D-E
SUBA
LDA A W    ; A=A-B
SUBD E 001
JPNZ 0x22   ; 

CMPA       ; A vs B
LDAD E W
ADDD E 0x7f
JPNC 0x40   ; A<B: Finish
ADDD D 001
JP 0x20

ORG 0x40      ; Calculating W<-D-E
ADDD D 001
OUTD D 010
ORG 0x48
ADDA
LDA A W      ; A=A+B
SUBD D 001
JPNZ 0x48   ; When finished, A==D
OUTA W      ; Output

PAUSE
JP 0x00

END
```

デモ③

LEDバーの上昇 (ノコギリ波)

□ ソースコード

RESETA

```
LDDA B 025      ; Set Voltage for one scale.  
ADDA             ; W=A+B  
OUTA W          ; W -> Output  
LDA A W         ; A=W  
INA B            ; B=Input (Reference)  
CMPA             ; A>B->W=VDD, A<B->W=0V  
LDAD D W         ; D <- W  
ADDD D 0x7F      ; D = D+VDD/2  
  
JPNC 0x01        ; C flag Not Set? -> JP to 0x01  
JP 0x00           ; return to RESET  
  
END
```

デモ④

LEDバーの上昇 (サイン波)

□ ソースコード

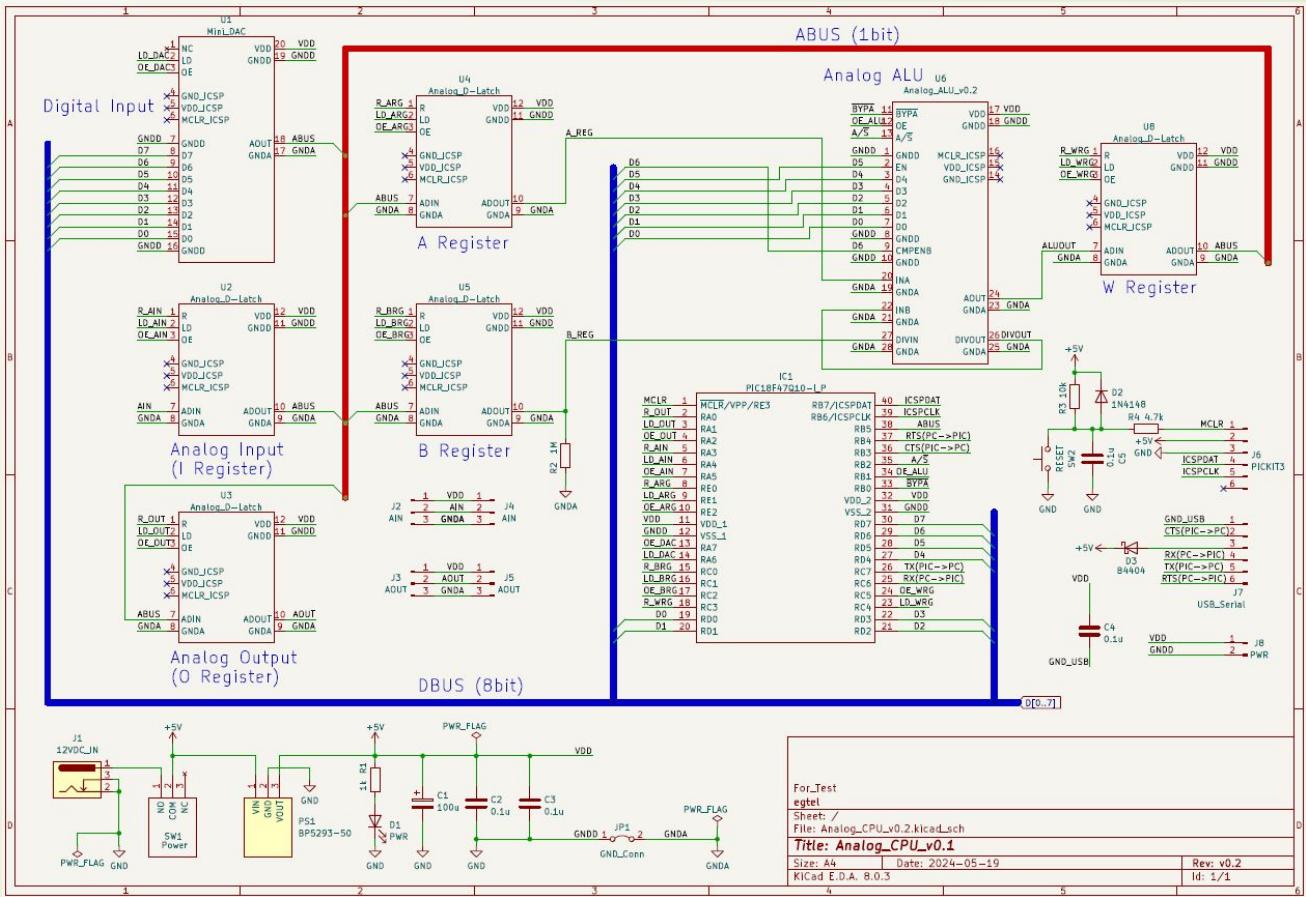
RESETA

```
ORG 0x01      ; Initialize D=16
LDD D 016
ORG 0x10      ; Address: 0x10
INA B
ADDA D        ; W=A+B*D/32
LDA A W       ; A=W
OUTA W        ; Output W value.
SUBD D 002    ; D=D-2
JPNZ 0x10     ; D>0->0x10, D=0:Next Address

ORG 0x20      ; Initialize D=0
LDD D 000
ORG 0x30      ; Address: 0x30
LDD E 018     ; E=16+2
INA B
SUBA D        ; W=A-B*D/32
LDA A W       ; A=W
OUTA W        ; Output W value
ADDD D 002    ; D=D+2
SUBD E D      ; E=E-D
JPNZ 0x30     ; E>0->0x30, E=0:Next Address

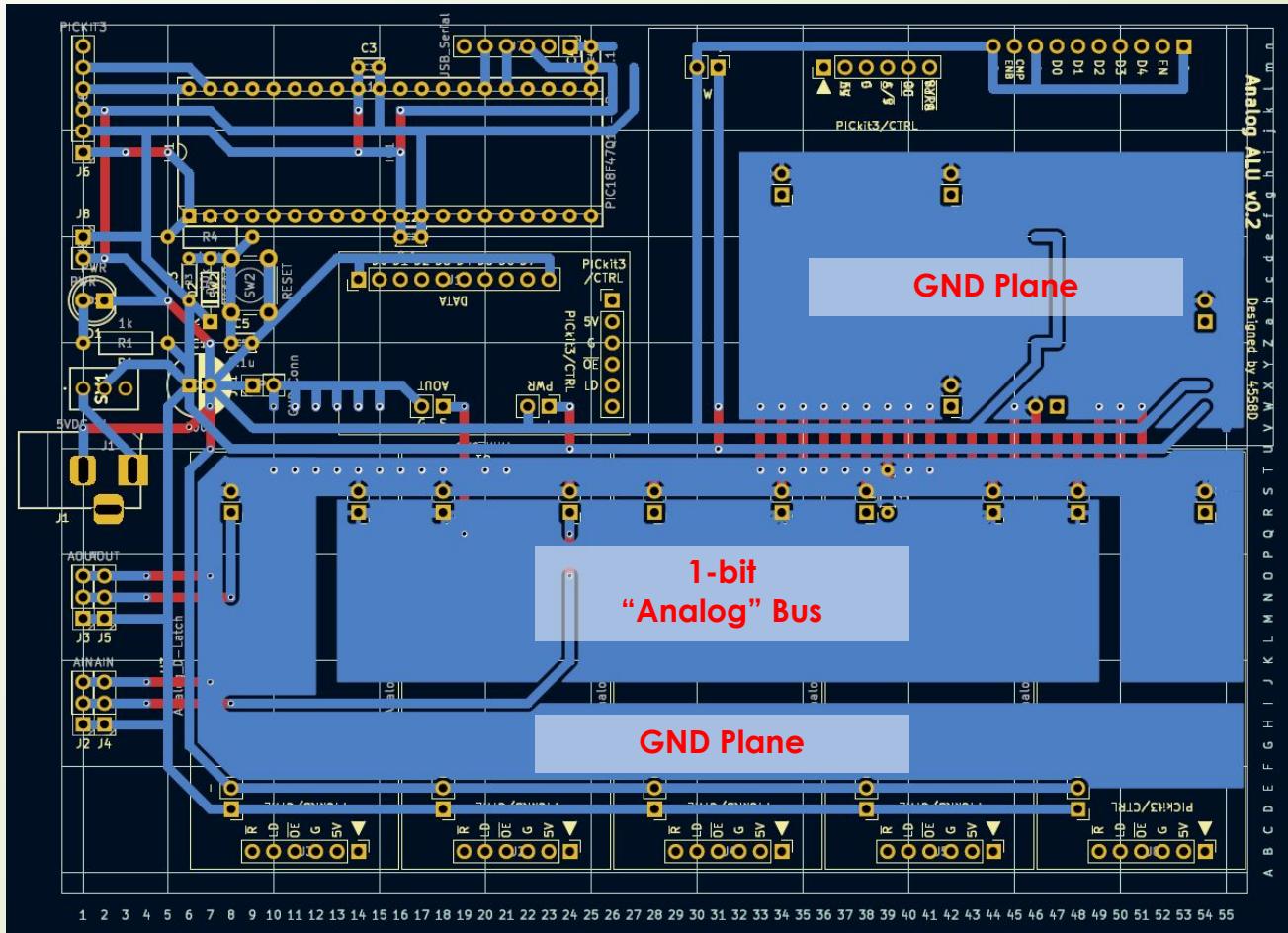
JP 0x00        ; return to Add 0x00
END
```

補足資料 CPU全体 - 回路図

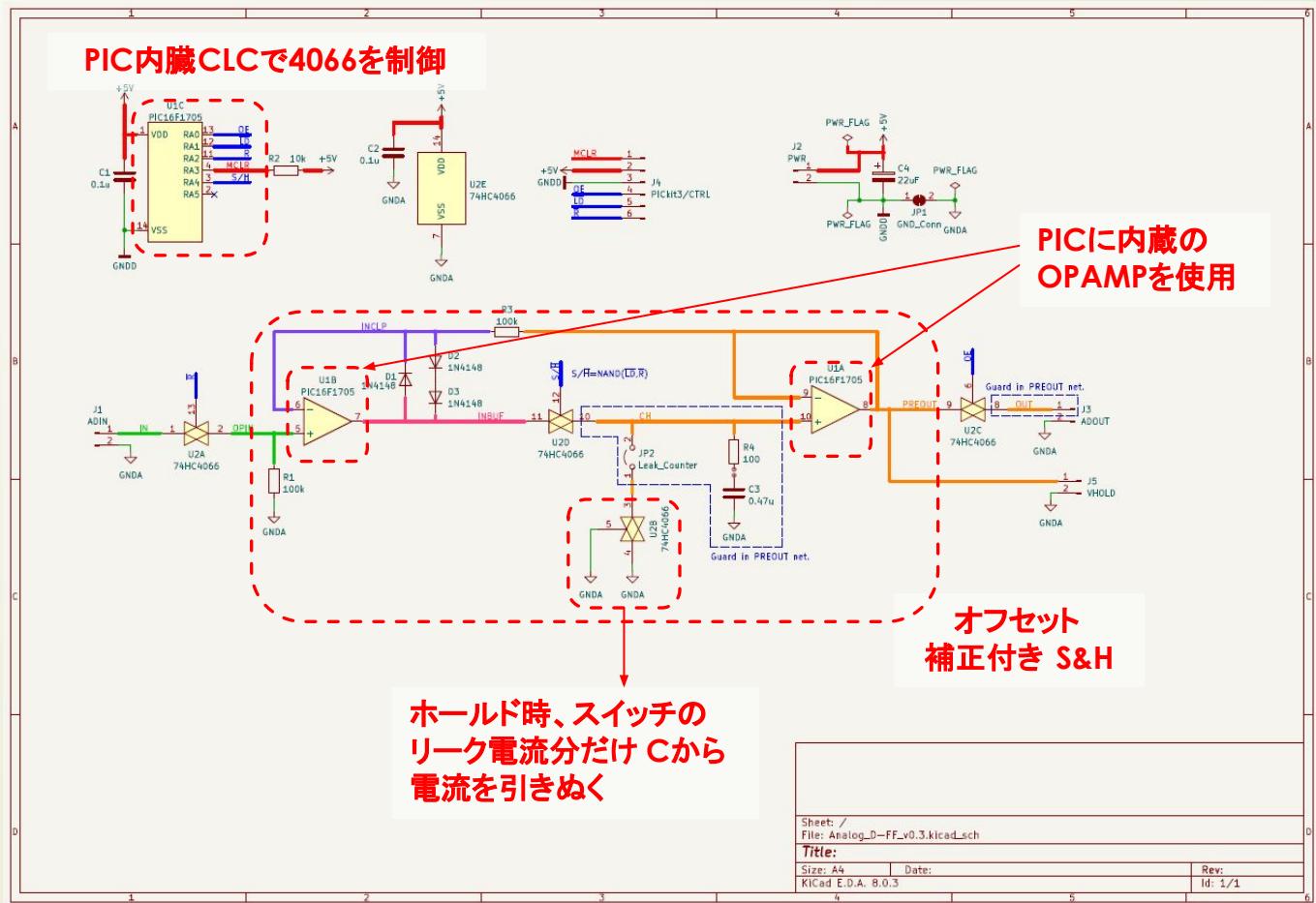


補足資料

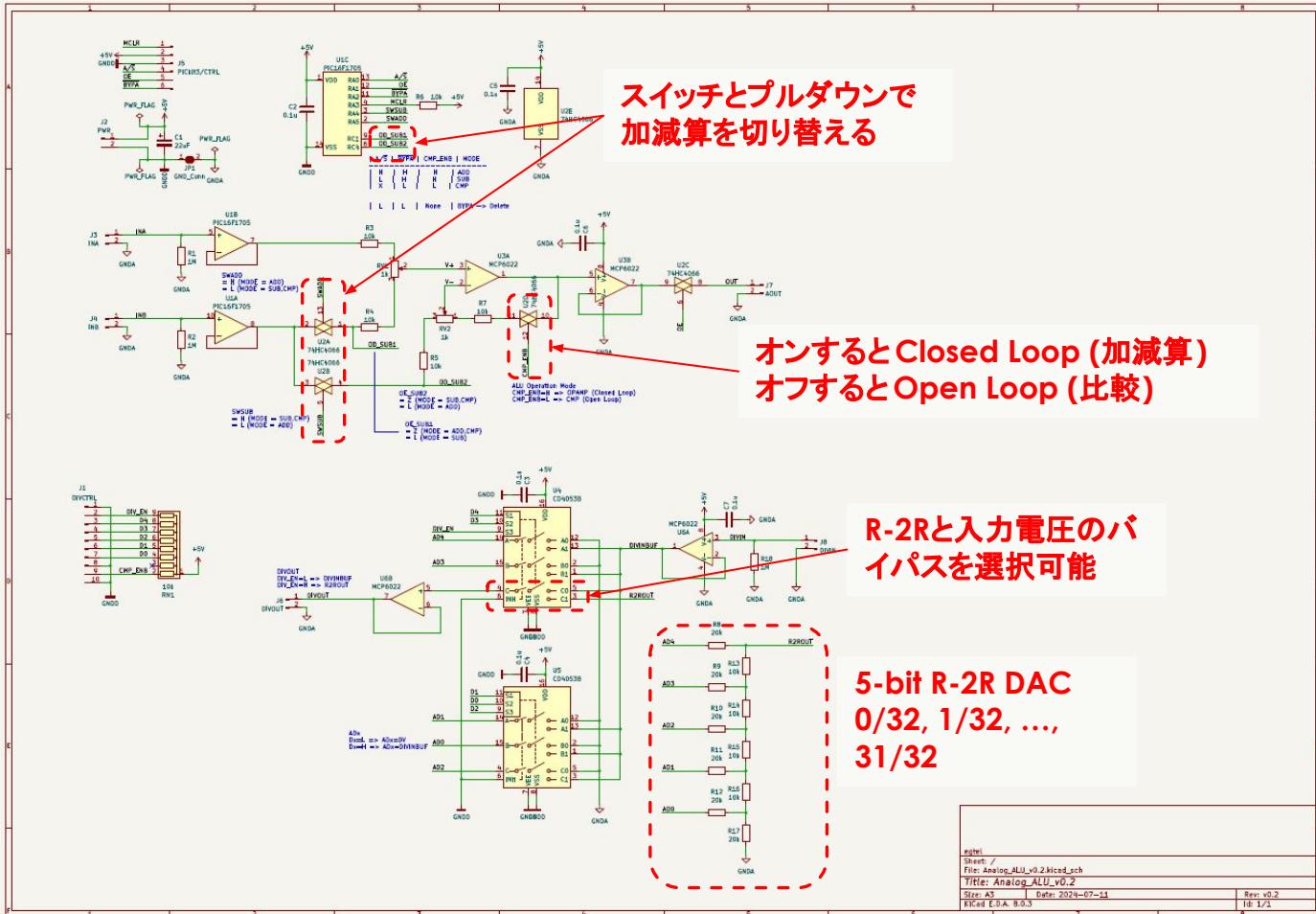
CPU全体 - PCBパターン



補足資料 アナログレジスター回路図

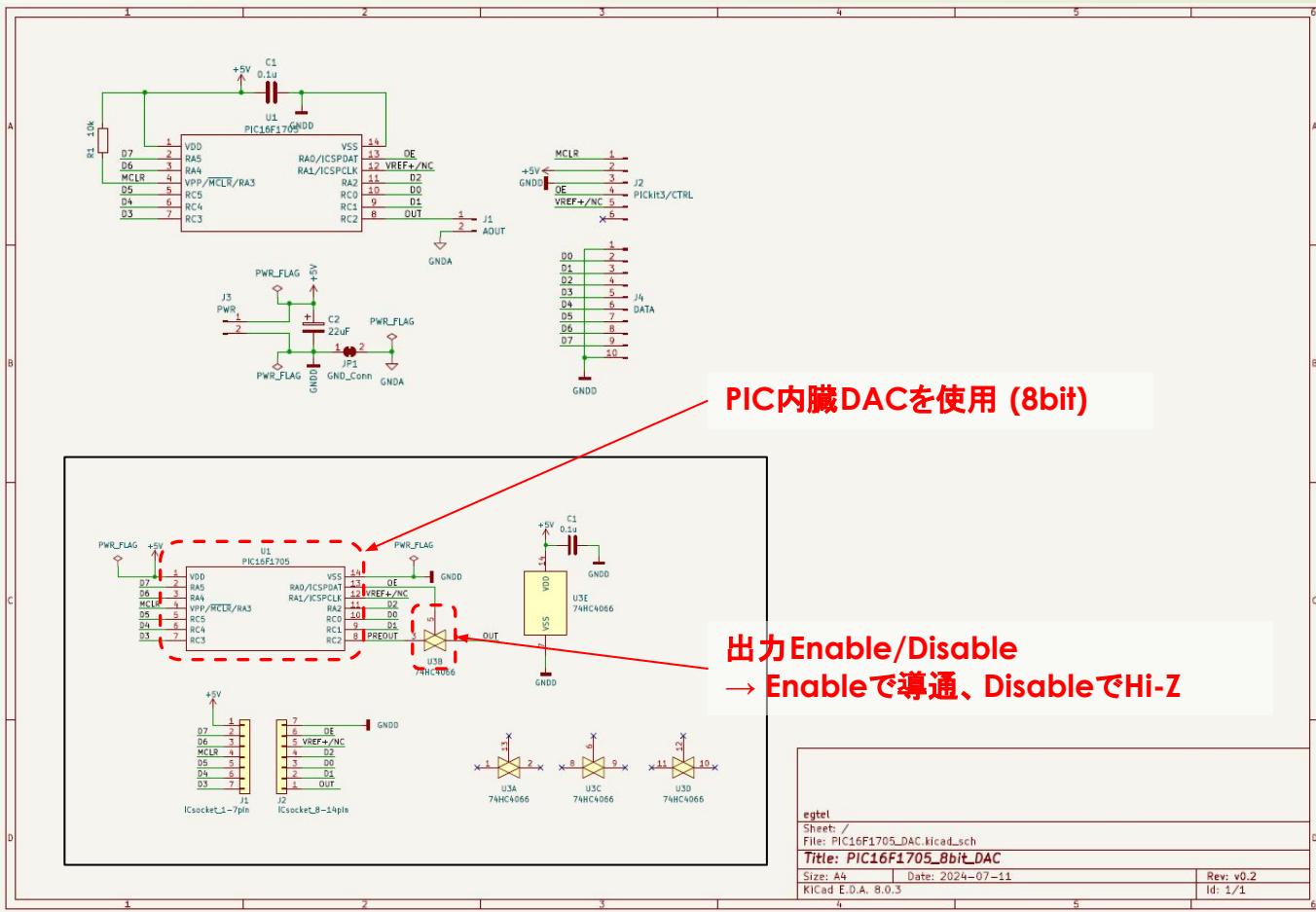


補足資料 アナログALU - 回路図



補足資料

DAC - 回路図



Analog CPUの命令一覧 (旧)

***** Mnemonic List *****			***** Mnemonic List *****						
code	length	order	operand1	operand2	code	length	order	operand1	operand2
0x00	1	NOP			0x50	2	LDDA	A	n
0x01	1	RESET			0x51	1	LDDA	A	D
0x02	1	HALT			0x52	1	LDDA	A	E
0x03	1	RESETA			0x53	2	LDDA	B	n
0x04	1	PAUSE			0x54	1	LDDA	B	D
0x08	2	JP	n		0x55	1	LDDA	B	E
0x09	2	JPNZ	n		0x58	1	LOAD	D	A
0x0a	2	JPZ	n		0x59	1	LOAD	D	B
0x0b	2	JPNC	n		0x5a	1	LOAD	D	W
0x0c	2	JPC	n		0x5b	1	LOAD	E	A
0x10	1	LDI	D	E	0x5c	1	LOAD	E	B
0x11	2	LDI	D	n	0x5d	1	LOAD	E	W
0x12	1	LDI	E	D	0x60	1	INA	A	
0x13	2	LDI	E	n	0x61	1	INA	B	
0x20	1	ADDD	D	E	0x62	1	OUTA	W	
0x21	2	ADDD	D	n	0x63	1	OUTA	I	
0x22	1	ADDD	E	D	0x64	1	ADDA		
0x23	2	ADDD	E	n	0x65	1	SUBA		
0x24	1	SUBD	D	E	0x67	1	CMPA		
0x25	2	SUBD	D	n	0x68	2	ADDA		
0x26	1	SUBD	E	D	0x69	2	SUBA		
0x27	2	SUBD	E	n	0x6a	2	DIVA		
0x36	1	IND	D		0x6b	2	CMPA		
0x37	1	IND	E		0x6c	1	ADDA		
0x38	2	OUTD	D	n	0x6d	1	SUBA		
0x39	2	OUTD	E	n	0x6e	1	DIVA		
0x40	1	LDA	A	B	0x6f	1	CMPA		
0x41	1	LDA	A	W	0x70	1	ADDA	E	
0x42	1	LDA	B	A	0x71	1	SUBA	E	
0x43	1	LDA	B	W	0x72	1	DIVA	E	
0x44	1	LDA	W	A	0x73	1	CMPA	E	
0x45	1	LDA	W	B	0x100	1	END		
					0x101	1	ORG	n	

Analog CPUの命令一覧(旧)

***** Mnemonic List *****

code length order

			operand1	operand2
0x00	1	NOP		
0x01	1	RESET		
0x02	1	CPU制御系		
0x03	1	RESETA		
0x04	1	PAUSE		
0x08	2	JP	n	
0x09	2	JPNZ	n	
0x0a	2	ジャンプ命令	n	
0x0b	2	JPNC	n	
0x0c	2	JPC	n	
0x10	1	LDD	D	E
0x11	2	LDD	D	n
0x12	1	LDD	E	D
0x13	2	LDD	E	n
0x20	1	ADDD	D	E
0x21	2	ADDD	D	n
0x22	1	ADD	D	D
0x23	2	ADD	D	n
0x24	1	SUBD	D	E
0x25	2	SUBD	D	n
0x26	1	SUBD	E	D
0x27	2	SUBD	E	n
0x36	1	IND	D	
0x37	1	IND		
0x38	2	OUTD	D	n
0x39	2	OUTD	E	n
0x40	1	LDA	A	B
0x41	1	LDA	A	W
0x42	1	ANALOGレジスタ操作命令	B	A
0x43	1	LDA	W	A
0x44	1	LDA	W	B
0x45	1	LDA	W	B

***** Mnemonic List *****

code length order

			operand1	operand2
0x50	2	LDDA	A	n
0x51	1	LDDA	A	D
0x52	1	LDDA	A	E
0x53	2	LDDA	B	n
0x54	1	LDDA	B	D
0x55	1	LDDA	B	E
0x58	1	LOAD	D	A
0x59	1	LOAD	D	B
0x5a	1	LOAD	D	W
0x5b	1	LOAD	E	A
0x5c	1	LOAD	E	B
0x5d	1	LOAD	E	W
0x60	1	TNA	A	
0x61	1	TNA		
0x62	1	OUTA	W	
0x63	1	OUTA	I	
0x64	1	ADDA		
0x65	1	SUBA		
0x67	1	CMPA		
0x68	2	ADDA		n
0x69	2	SUBA		n
0x6a	2	DIVA		n
0x6b	2	ANALOGレジスタ演算命令		
0x6c	1	ADDA	D	
0x6d	1	SUBA	D	
0x6e	1	DIVA	D	
0x6f	1	CMPA	D	
0x70	1	ADDA	E	
0x71	1	SUBA	E	
0x72	1	DIVA	E	
0x73	1	CMPA	E	
0x100	1	END		
0x101	1	ORG	n	

疑似命令