NT東京
2025

DDD　E-4ブース

Z80 Study Group
from Odawara

目次

1.DDD
2.今回の展示内容
　1.8080ALUを動かしてみた
3.Smart knob

DDDとは

Deadline-Driven Developers

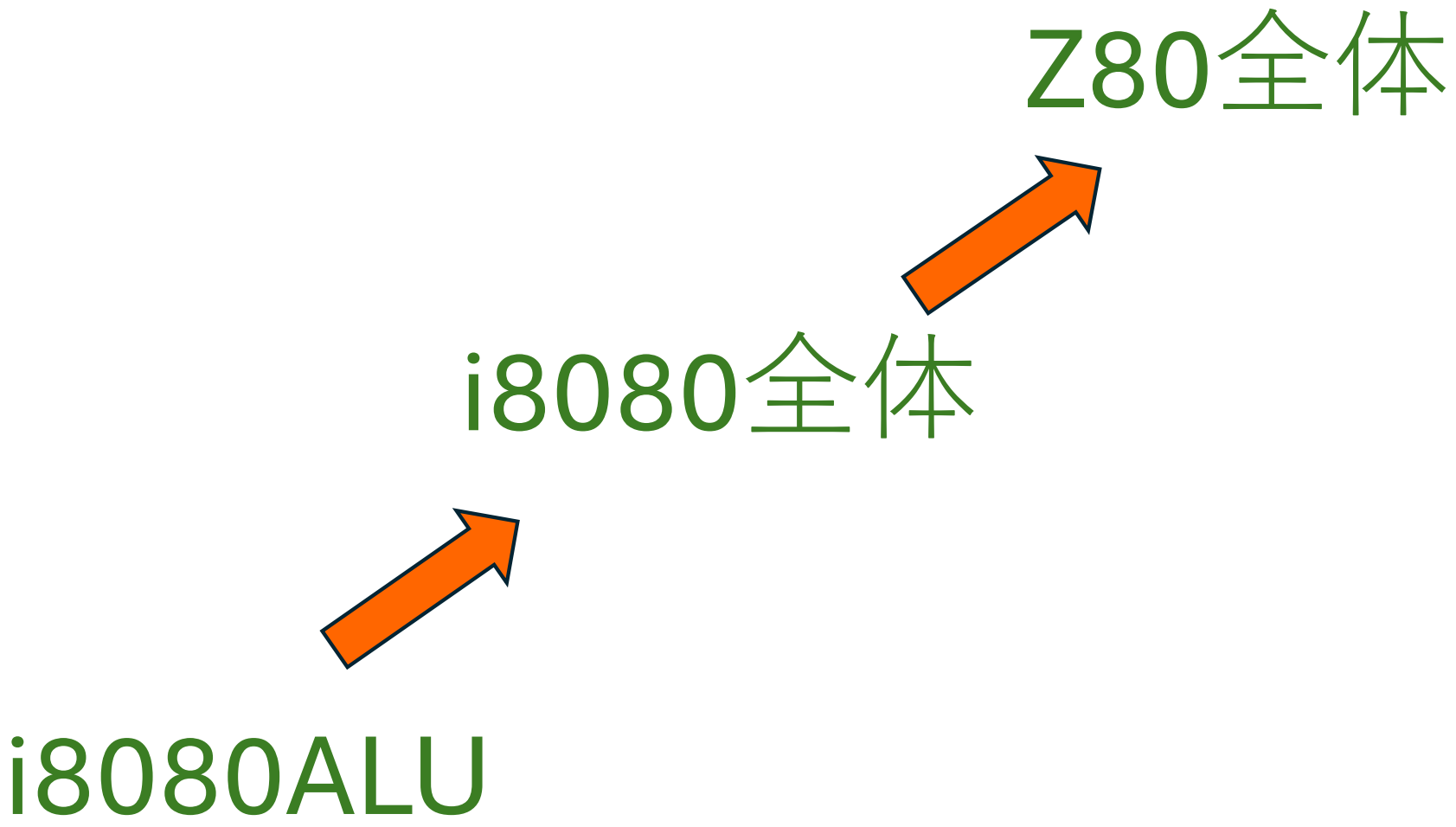
https://dddev.group/

今回の展示内容
CPUの動作を可視化してみた

- CPUの内部を理解したい
- そのためには８bitが最適
  - 8bitマシンが普及したから
  - Z80が特に普及したから
    - 16bit機はそれ程
    - それ以上は複雑

- 最終的にはZ80を動かしたい
- レジスターや内部バスの状態を見える化したい

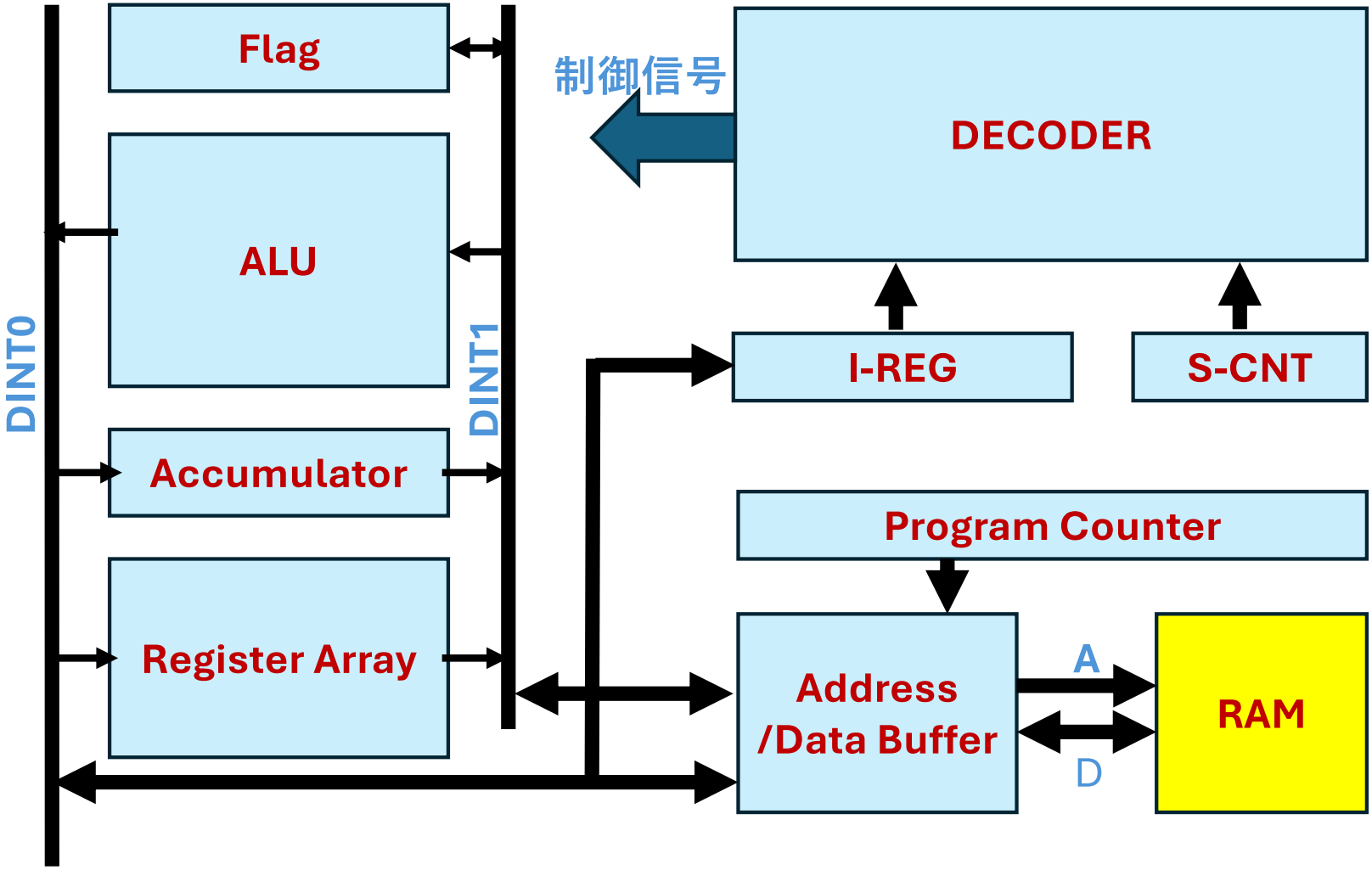
i8080 ALUのロードマップ



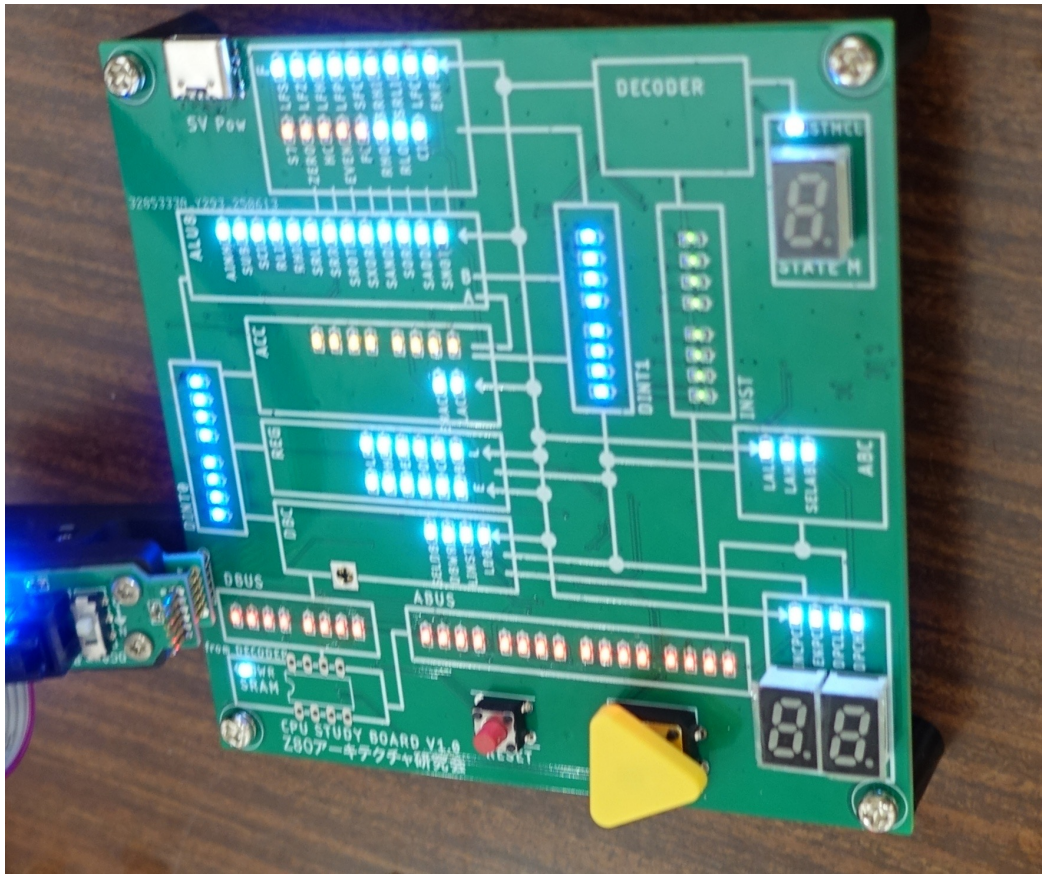
山岡の履歴

- 1975 3月の大学の卒業論文として「i8080を使ったフーリエ変換機の製作」でPCを開発
  - 原始アセンブラーからFFTまで開発
- 1977 ヤマハ入社 半導体に所属
  - FM音源の最初の開発チームに所属
  - 楽器用音源等各種開発
  - 各種CPUアーキテクチャーの調査
  - MSX2用のグラフィック・プロセッサーV9938開発
  - セガ社ゲーム用LSIの開発
    - １チップ・メガドライブの開発他

I8080 ALU Block Diagram

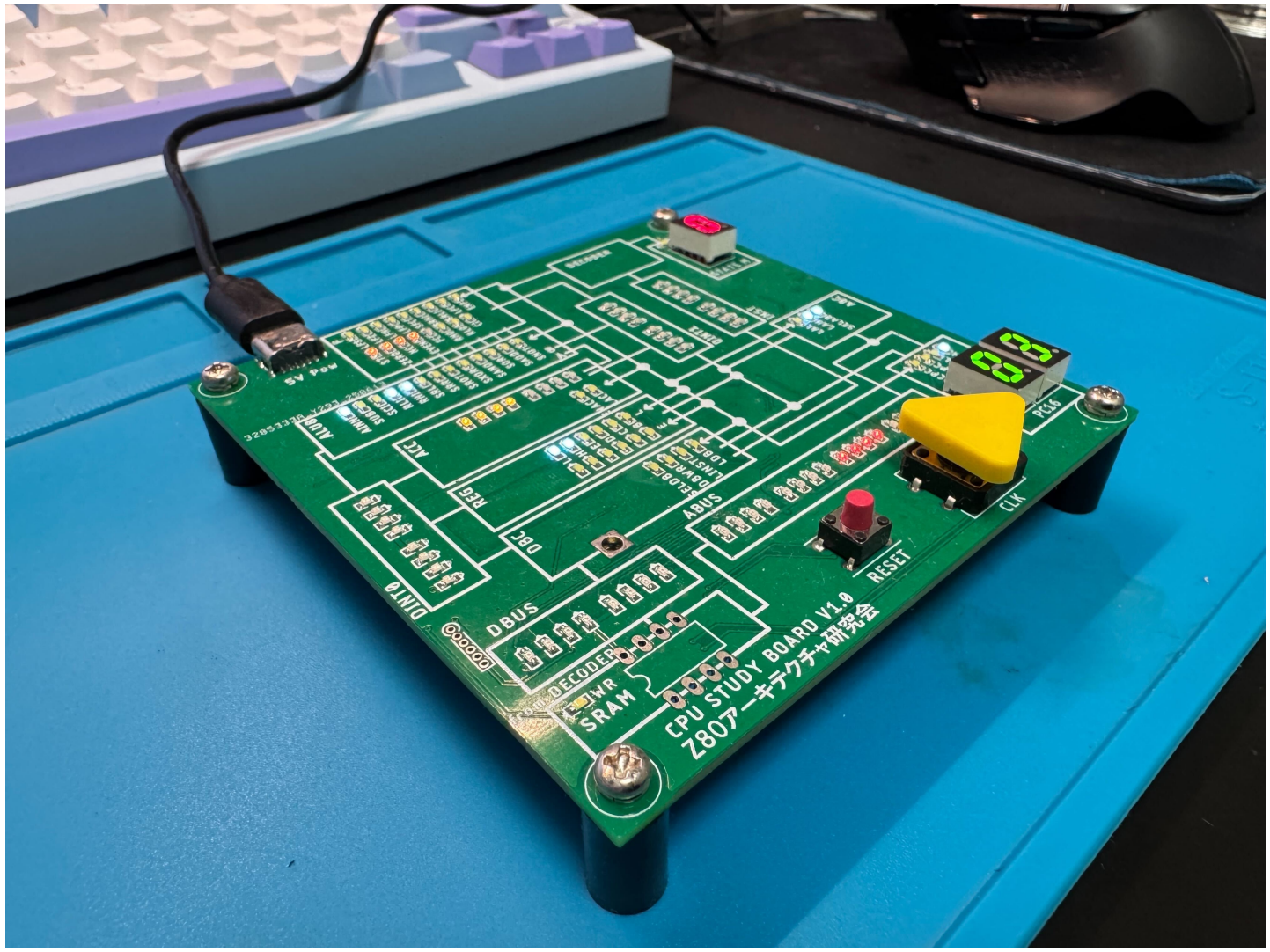


i8080ALU基板

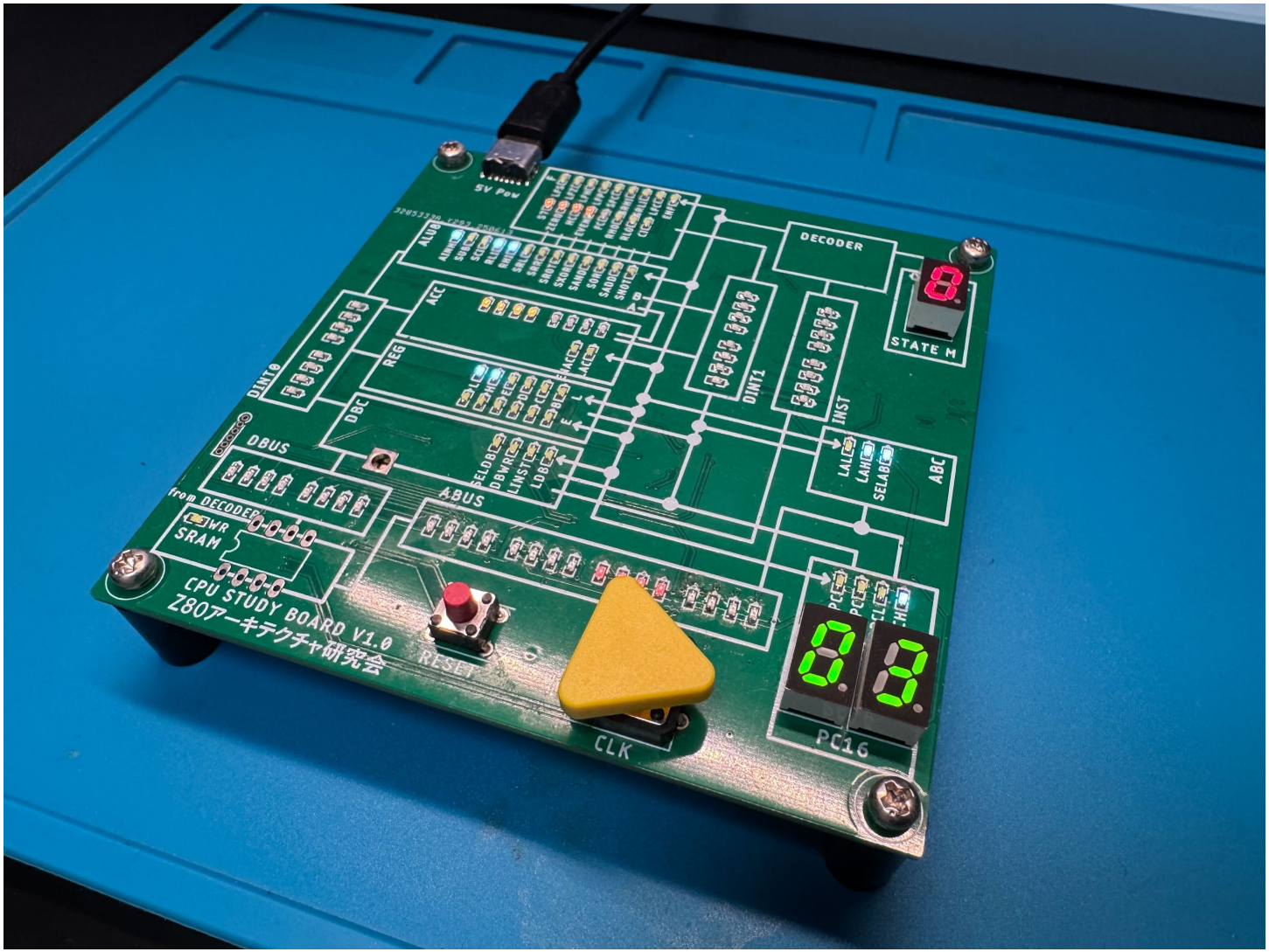


10cm X 10cm

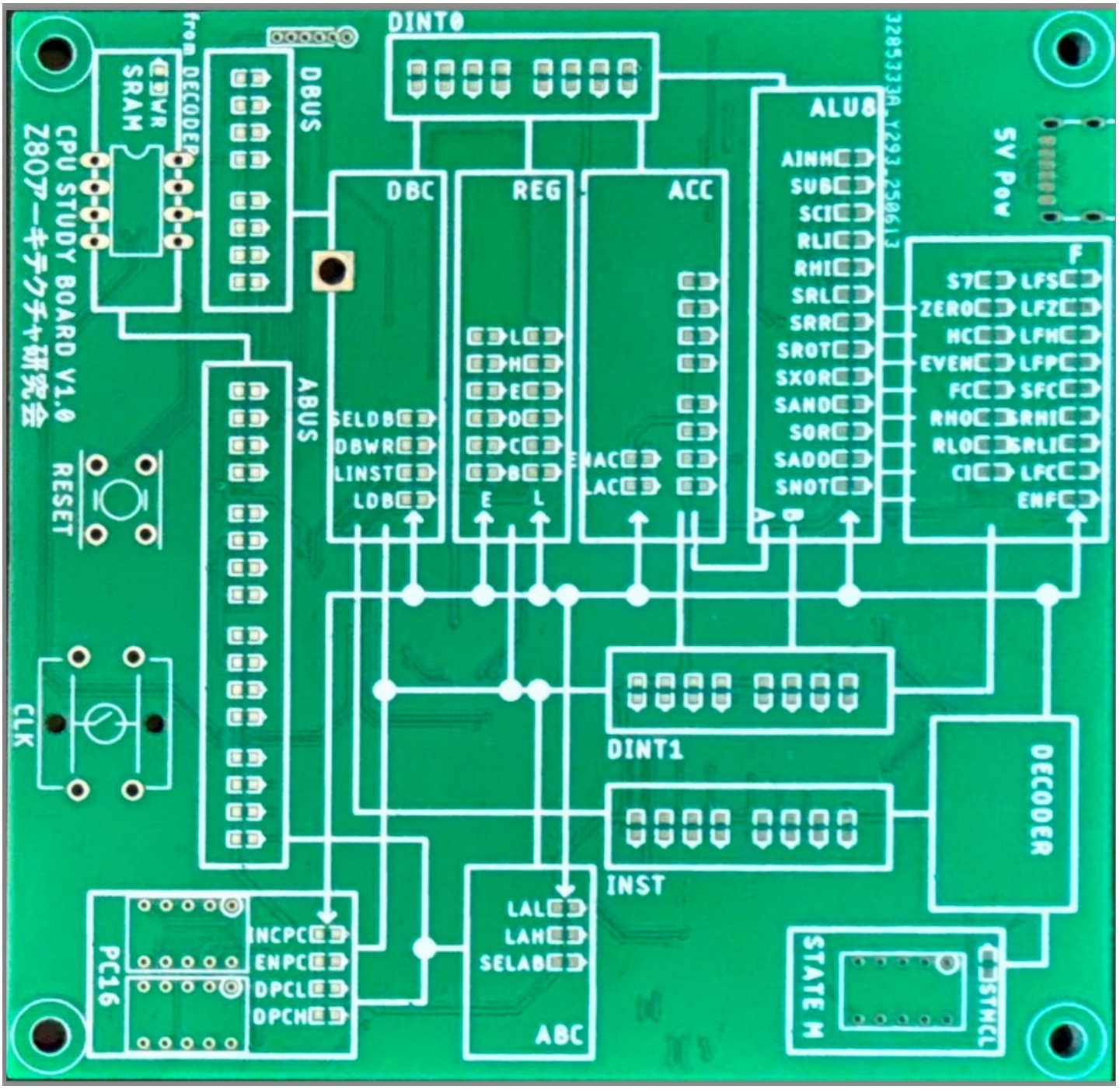
i8080ALU基板


5V Pow
ADDR
ACC
REG
DBC
ABUS
DBUS
DECODER
WR
SRAM
DIN10
RESET
CLK
CPU STUDY BOARD V1.0
Z80アーキテクチャ研究会

i8080ALU基板


DECODER
STATE M
DINT0
DINT1
INST
ACC
REG
DBC
ABC
DBUS
ABUS
SRAM
CPU STUDY BOARD V1.0
Z80アーキテクチャ研究会
RESET
CLK
PC16

i8080ALU基板（生基板）



i8080ALUで使える命令語

命令語	code	バイト数	機能	フラグ
ADD B	80	1	A=A+r	CHZSP
ADI	C6	2	A=A+B2	CHZSP
ADC B	88	1	A=A+r+C	CHZSP
ACI	CE	2	A=A+B2+C	CHZSP
SUB B	90	1	A=A-r	CHZSP
SUI	D6	2	A=A-B2	CHZSP
SBB B	98	1	A=A-r-C	CHZSP
SBI	DE	2	A=A-B2-C	CHZSP
ANA B	A0	1	A=A and r	ZSP
ANI	E6	2	A=A and B2	ZSP
ORA B	B0	1	A=A or r	ZSP
ORI	F6	2	A=A or B2	ZSP
XRA B	A8	1	A=A xor r	ZSP
XRI	EE	2	A=A xor B2	ZSP

命令語	code	バイト数	機能	フラグ
NOP	00	1		
HLT	76	1		
RLC	07	1	Lシフト	C
RAL	17	1	Lシフト w/C	C
RRC	0F	1	Rシフト	C
RAR	1F	1	Rシフト w/C	C
STA	32	3		
LDA	3A	3		
MOV B,A	47	1	B=A	

制御信号の説明

番号	制御信号	DESTINATION	信号名説明	内容
0	LFS	F	Load Flag S(sign)	Sign Flag に入力する
1	LFZ	F	Load Flag Z(sero)	Zero Flag に入力する
2	LFH	F	Load Flag H(Half carry)	Half Carry Flagに入力する
3	LFP	F	Load Flag P(parity)	Parity Flagに入力する
4	LFC	F	Load Flag C(Carry)	Carry Flag に入力する
5	ENF	F	Enable Flags to BUS	Flag全てをBUSに出力
6	SFC	F	Select "fc" for Carry	Carry入力として"fc"を選択する
7	SRHI	F	Select "RHO" for Carry	Carry入力として"RHO"を選択する
8	SRLI	F	Select "RLO" for Carry	Carry入力として"RLO"を選択する
9	SNOT	ALU8	Select NOT 出力	Accの出力の反転を選択しBUSに出力
10	SADD	ALU8	Select ADD 出力	ADD出力を選択しBUSに出力
11	SOR	ALU8	Select OR 出力	OR出力を選択しBUSに出力
12	SAND	ALU8	Select AND 出力	AND出力を選択しBUSに出力
13	SXOR	ALU8	Select XOR 出力	Exclusive OR出力を選択しBUSに出力
14	SROT	ALU8	Select ROT 出力	Rotation出力を選択しBUSに出力
15	SRR	ALU8	Rotate Right	右Rotateを選択
16	SRL	ALU8	Rotate Left	左Rotateを選択
17	WC	ALU8	With Carry flag	RotateにCarry Flagを含める、WCが"0"の時は含めない
18	SCI	ALU8	Select Carry In	ALUの__Carry InとしてCIを選択
19	SUB	ALU8	Subtract	引き算　Adderの一方の入力を反転しCIを通して"1"を足す
20	AINH	ALU8	Acc Inhibit	ALUへのAccからの入力を強制的に"0"とする
21	LAC	ACC	Load BUS入力	DINT0 BUSの値をAccに入力する
22	ENAC	ACC	Enable Acc出力	Accの内容をBUS（DO)に出力する
23	LRGB	REGSTER ARRAY	Load to Register B	BUS信号をRegister Bに入力する
24	LRGC	REGSTER ARRAY	Load to Register C	BUS信号をRegister Cに入力する
25	LRGD	REGSTER ARRAY	Load to Register D	BUS信号をRegister Dに入力する
26	LRGE	REGSTER ARRAY	Load to Register E	BUS信号をRegister Eに入力する
27	LRGH	REGSTER ARRAY	Load to Register H	BUS信号をRegister Hに入力する
28	LRGL	REGSTER ARRAY	Load to Register L	BUS信号をRegister Lに入力する
29	ERGB	REGSTER ARRAY	Enable Register B	Register B の内容をBUSに出力する
30	ERGC	REGSTER ARRAY	Enable Register C	Register C の内容をBUSに出力する
31	ERGD	REGSTER ARRAY	Enable Register D	Register D の内容をBUSに出力する
32	ERGE	REGSTER ARRAY	Enable Register E	Register E の内容をBUSに出力する
33	ERGH	REGSTER ARRAY	Enable Register H	Register H の内容をBUSに出力する
34	ERGL	REGSTER ARRAY	Enable Register L	Register L の内容をBUSに出力する
35	LINST	DBC	Load Instruction	外部データBUSの内容（Instruction)をInstruction Reg.に取込む
36	DBWR	DBC	Write Data	外部データBUSに書き込みを行う
37	SELDB	DBC	Select Data BUS for BUS1	外部データBUSの内容を内部BUS1(DINT1)に取込む
38	ENDB	DBC	Select Data BUS for BUS0	外部データBUSの内容を内部BUS0(DINT0)に取込む
39	RDMN	SRAM	/(Read Memory)	SRAMデータの読み出しの反転（書き込み信号）
40	DPCH	PC16	Load Data BUS to PC high	内部データBUSの内容をPCの上位バイトに取込む
41	DPCL	PC16	Load Data BUS to PC low	内部データBUSの内容をPCの下位バイトに取込む
42	ENPC	PC16	Enable PC counter	Program Counterの内容をAddress BUSに出力
43	INCPC	PC16	Increment PC	Program Counterの内容をIncrement(Halt命令の時は除く)
44	SELAB	ABC	Select Address Bus	Address BUSにAddress Register の内容を出力
45	LAH	ABC	Load BUS data to Addr reg H	内部BUS(DINT1)の内容をアドレスレジスターの上位バイトに入力
46	LAL	ABC	Load BUS data to Addr reg L	内部BUS(DINT1)の内容をアドレスレジスターの下位バイトに入力
47	STMCL	STATE MACHINE	Clear State Macine	ステートマシンの内容をリセットする

プログラム例

動作の流れ

1. (B) ← “AA” //B=AA
2. (A)← “55” //A=55
3. (A) ← (A) + (B) //A=FF
4. (A) ← (A) +”01” //A=00, Carry=1
5. (A) ← (A) or (B) //A=AA
6. Rotate Left (A) with Carry //A=55
7. (A) ← (A) ex-or “FF” //A=AA
8. (A) ← (A) and “3F” //A=2A
9. (M 1000番地) ← (A)
10. (A) ← “00” //A=00
11. (A) ← (M 1000番地) //A=2A

機械語プログラム

```
ANI “00”
ORI “AA”
MOV B,A
ANI “00”
ORI “55”
ADD B
ADI “01”
ORA B
RAL
XRI “FF”
ANI “3F”
STA 1000
ANI “00”
LDA 1000
```

SRAM 内容

Addr	Data	
00	E6	ANI
01	00	
02	F6	ORI
03	AA	
04	47	MOV
05	E6	ANI
06	00	
07	F6	ORI
08	55	
09	80	ADD
0A	C6	ADI
0B	01	
0C	B0	ORA
0D	17	RAL
0E	EE	XRI
0F	FF	
10	E6	ANI
11	3F	
12	32	STA
13	00	
14	10	
15	E6	ANI
16	00	
17	3A	LDA
18	00	
19	10	

プログラム例

動作の流れ

1. (B) ← “AA” //B=AA
2. (A)← “55” //A=55
3. (A) ← (A) + (B) //A=FF
4. (A) ← (A) +”01” //A=00, Carry=1
5. (A) ← (A) or (B) //A=AA
6. Rotate Left (A) with Carry //A=55
7. (A) ← (A) ex-or “FF” //A=AA
8. (A) ← (A) and “3F” //A=2A
9. (M 1000番地) ← (A)
10. (A) ← “00” //A=00
11. (A) ← (M 1000番地) //A=2A

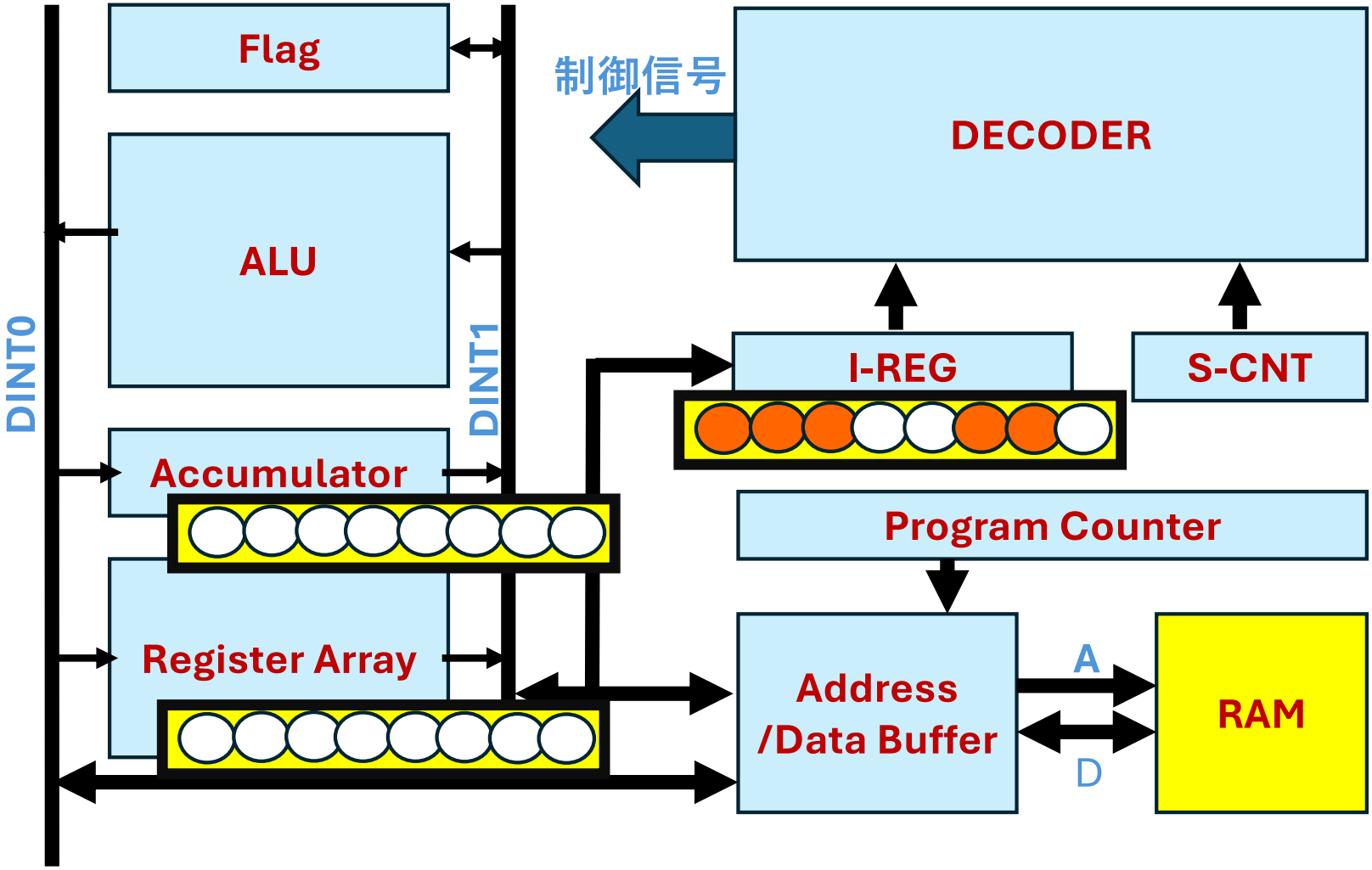
機械語プログラム

```
ANI “00”
ORI “AA”
MOV B,A
ANI “00”
ORI “55”
ADD B
ADI “01”
ORA B
RAL
XRI “FF”
ANI “3F”
STA 1000
ANI “00”
LDA 1000
```

SRAM 内容

Addr	Data	
00	E6	ANI
01	00	
02	F6	ORI
03	AA	
04	47	MOV
05	E6	ANI
06	00	
07	F6	ORI
08	55	
09	80	ADD
0A	C6	ADI
0B	01	
0C	B0	ORA
0D	17	RAL
0E	EE	XRI
0F	FF	
10	E6	ANI
11	3F	
12	32	STA
13	00	
14	10	
15	E6	ANI
16	00	
17	3A	LDA
18	00	
19	10	

Block Diagram



プログラム例

動作の流れ

1. (B) ← “AA” //B=AA
2. (A)← “55” //A=55
3. (A) ← (A) + (B) //A=FF
4. (A) ← (A) +”01” //A=00, Carry=1
5. (A) ← (A) or (B) //A=AA
6. Rotate Left (A) with Carry //A=55
7. (A) ← (A) ex-or “FF” //A=AA
8. (A) ← (A) and “3F” //A=2A
9. (M 1000番地) ← (A)
10. (A) ← “00” //A=00
11. (A) ← (M 1000番地) //A=2A

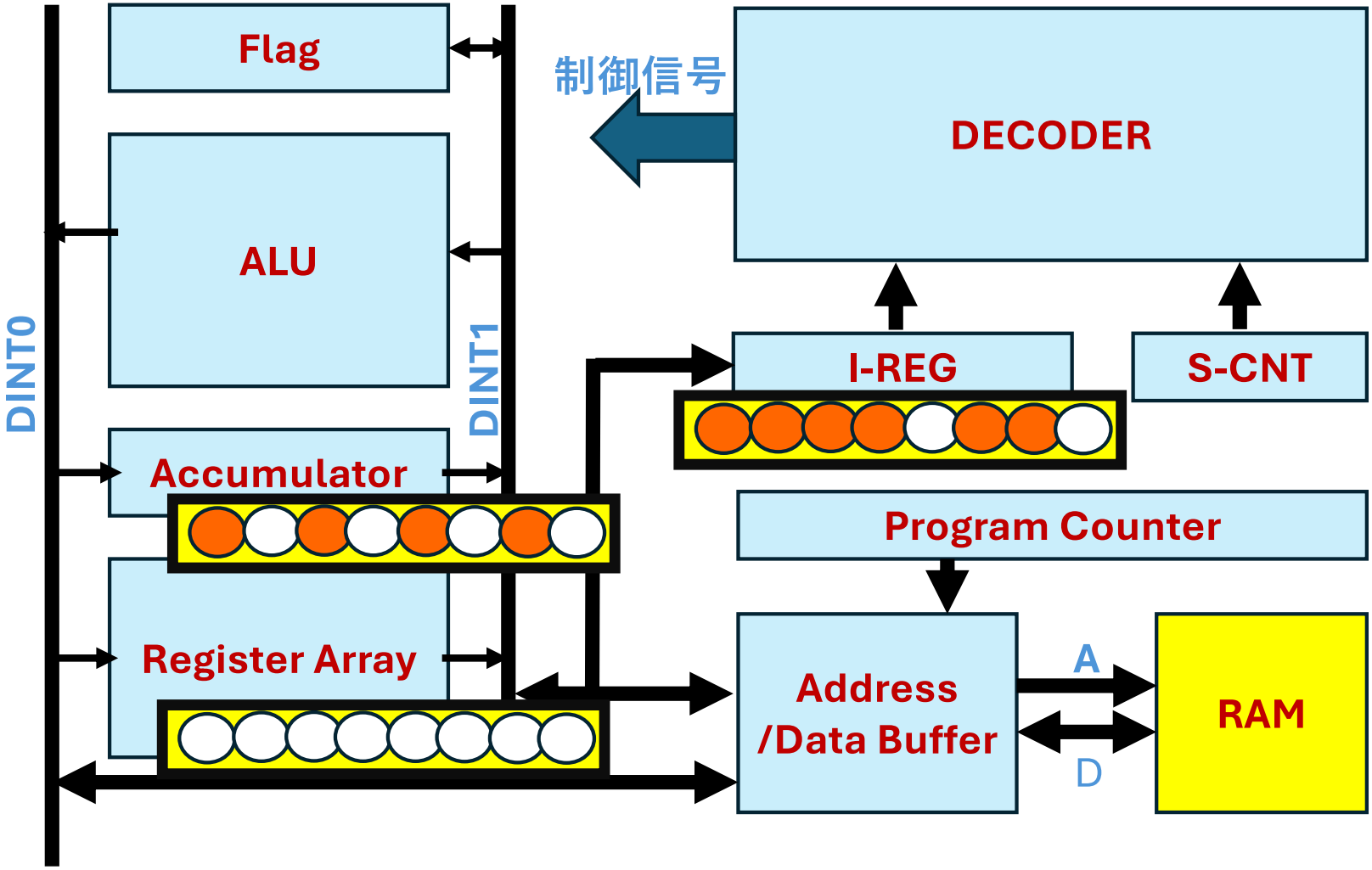
機械語プログラム

```
ANI “00”
ORI “AA”
MOV B,A
ANI “00”
ORI “55”
ADD B
ADI “01”
ORA B
RAL
XRI “FF”
ANI “3F”
STA 1000
ANI “00”
LDA 1000
```

SRAM 内容

Addr	Data	
00	E6	ANI
01	00	
02	F6	ORI
03	AA	
04	47	MOV
05	E6	ANI
06	00	
07	F6	ORI
08	55	
09	80	ADD
0A	C6	ADI
0B	01	
0C	B0	ORA
0D	17	RAL
0E	EE	XRI
0F	FF	
10	E6	ANI
11	3F	
12	32	STA
13	00	
14	10	
15	E6	ANI
16	00	
17	3A	LDA
18	00	
19	10	

Block Diagram



プログラム例

動作の流れ

1. (B) ← “AA” //B=AA
2. (A)← “55” //A=55
3. (A) ← (A) + (B) //A=FF
4. (A) ← (A) +”01” //A=00, Carry=1
5. (A) ← (A) or (B) //A=AA
6. Rotate Left (A) with Carry //A=55
7. (A) ← (A) ex-or “FF” //A=AA
8. (A) ← (A) and “3F” //A=2A
9. (M 1000番地) ← (A)
10. (A) ← “00” //A=00
11. (A) ← (M 1000番地) //A=2A

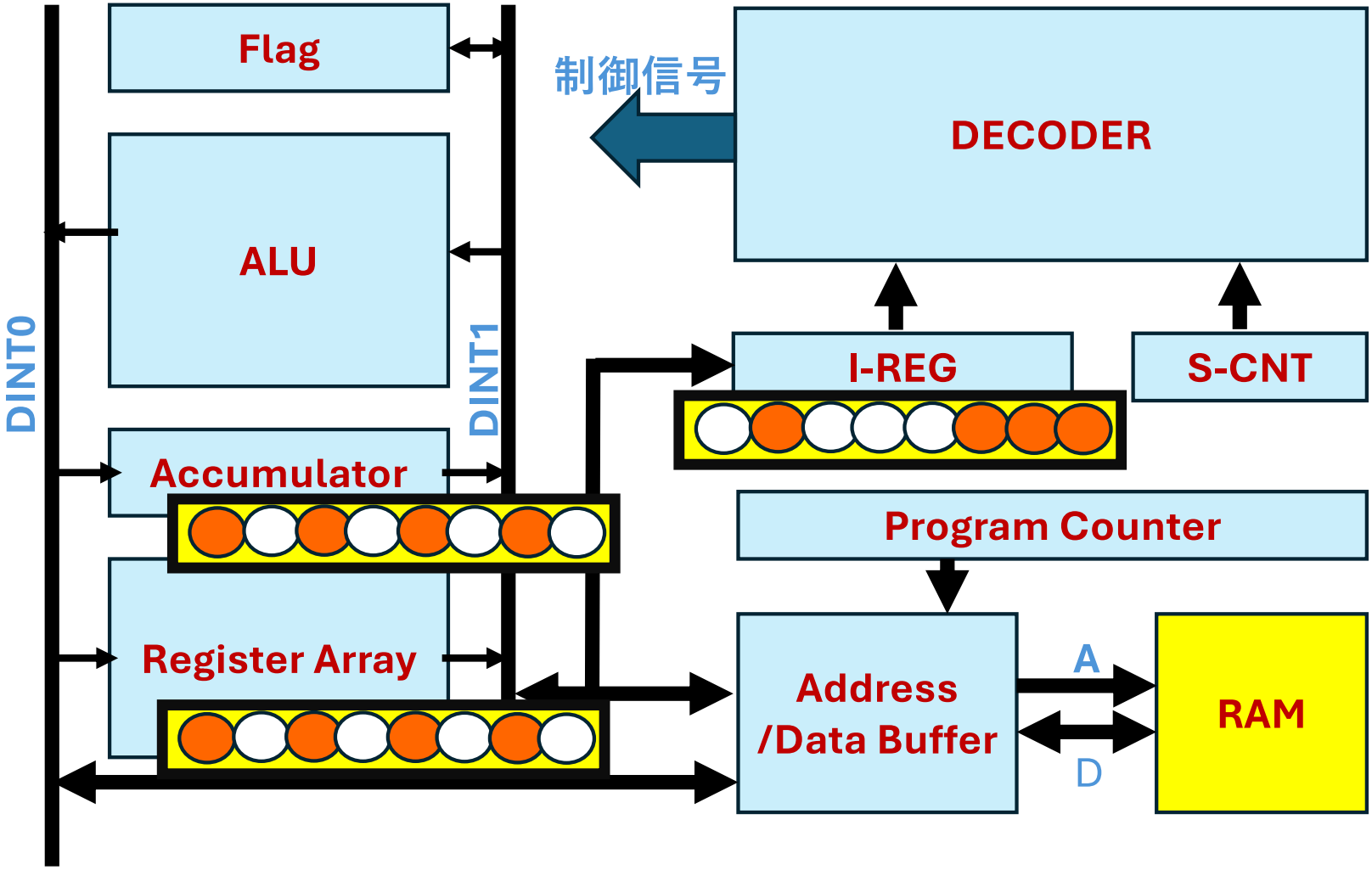
機械語プログラム

```
ANI “00”
ORI “AA”
MOV B,A
ANI “00”
ORI “55”
ADD B
ADI “01”
ORA B
RAL
XRI “FF”
ANI “3F”
STA 1000
ANI “00”
LDA 1000
```

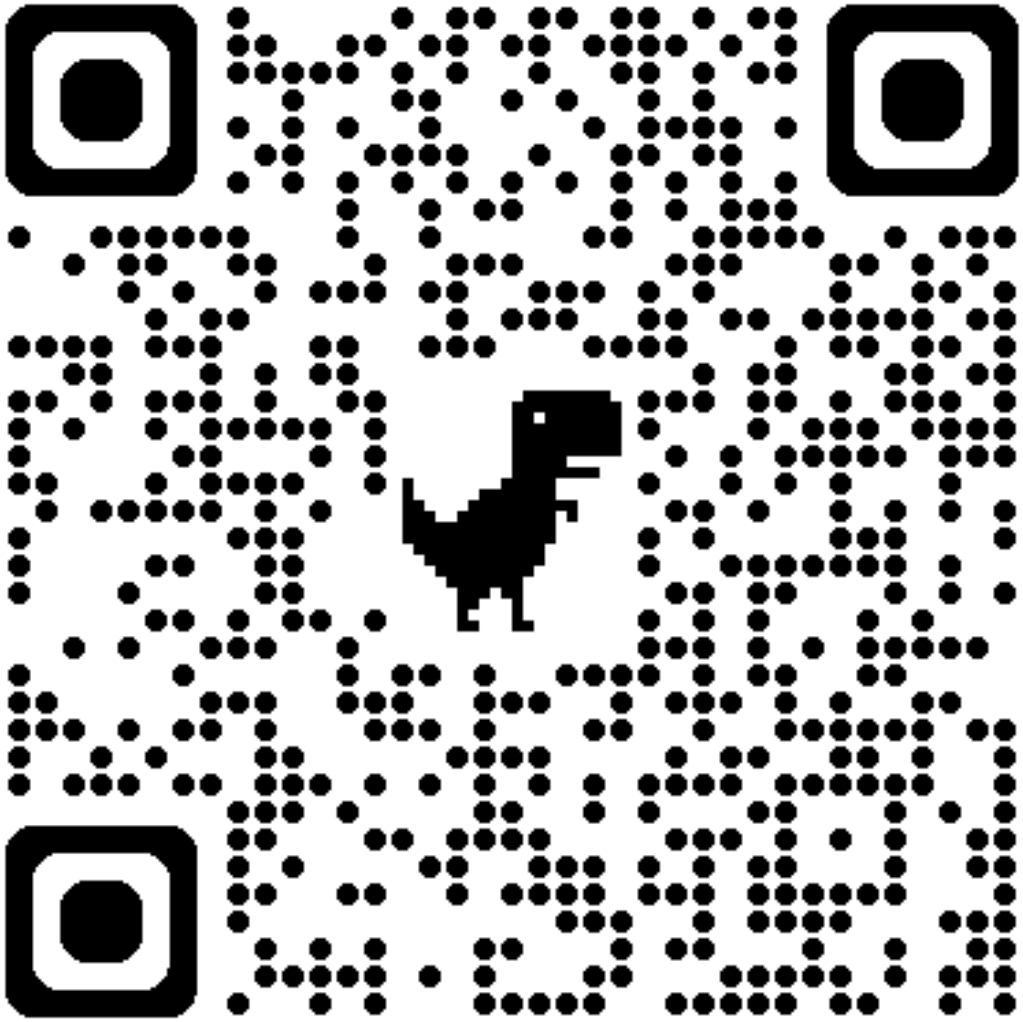
SRAM 内容

Addr	Data	
00	E6	ANI
01	00	
02	F6	ORI
03	AA	
04	47	MOV
05	E6	ANI
06	00	
07	F6	ORI
08	55	
09	80	ADD
0A	C6	ADI
0B	01	
0C	B0	ORA
0D	17	RAL
0E	EE	XRI
0F	FF	
10	E6	ANI
11	3F	
12	32	STA
13	00	
14	10	
15	E6	ANI
16	00	
17	3A	LDA
18	00	
19	10	

Block Diagram



有難うございました



E-4 ブース

←DDDのホームページ

https://dddev.group/

Block Diagram