第１回　本日の予定と宿題

本資料の２ページ目からを使用して TeC を解説する次のビデオを視聴し TeC に関する理解を深める．

1. TeC の設計思想，データパスと制御手順を理解する．（ビデオ）　13:40
   約３０分のビデオが３本 YouTube に置いてある．
   ① 設計思想と基本構造
   https://youtu.be/47zMYxJmblw
   ② TeC6 の命令と基本動作
   https://youtu.be/FK6xdkRf00E
   ③ TeC6 の制御手順
   https://youtu.be/thUJyMmIJ9M

2. データパスの制御手順を自分で組立てられるようになる．　15:30
   （理解度テストの問題を解きながら理解を深める．）

3. 時間が余るようなら下の「宿題」をやる．

終了　16:50

宿題：

1. 次週から２週間でデータパスの設計を行う．
   各自，データパス図を描くためのアプリケーションを準備しておくこと．
   使い慣れたお絵かきソフト，または，draw.io のデスクトップ版がお勧め

2. TeC の復習
   TeC6 の教科書を Teams に置いてある．

情報電子工学総合実験の目標

1. 現代の技術を用いて TeC を再設計し実装して動作させる。
    - 構造が分かりやすい(美しい)
    - 高速に(少ないクロック数で) 動作する
2. TeC を再設計する中で、順序立てた設計の重要性を認識する。

従来の TeC の設計目標

1990 年頃に安価に利用できた技術・素子を用いて、IE 実験室の設備で可能な範囲で、教育用の綺麗な命令体系を持った独自 CPU を搭載したコンソール付きのコンピュータを製作し、安定して動作させる。

制約

(1) 入手可能な部品の品種は限られ、部品の仕様は変更できない。
例：ALU は 74381 を使用するしかない。レジスタファイルは 74???を…

(2) IE 実験室で製作可能なプリント基板で実装する必要がある。
最大サイズ A4、配線はピン間 1 本、部品数、配線数はとにかく少なくする。

(3) 詳細な解析はできない。
配線の遅延などは未知のままになる。安全サイドの設計をするしかない。

設計方針

少ない部品をなるべく使い回すことで機能を実現する。例えば、一つの ALU でデータ計算、アドレス計算(インデクスモード)、レジスタ類のインクリメンドデクリメント、マルチプレクサの役割を果たす。

少ない部品を使い回すために複雑な制御手順を実行できる制御部が必要になる。そこで、制御部はマイクロプログラム方式を採用する。

総合実験で作る TeC の設計目標

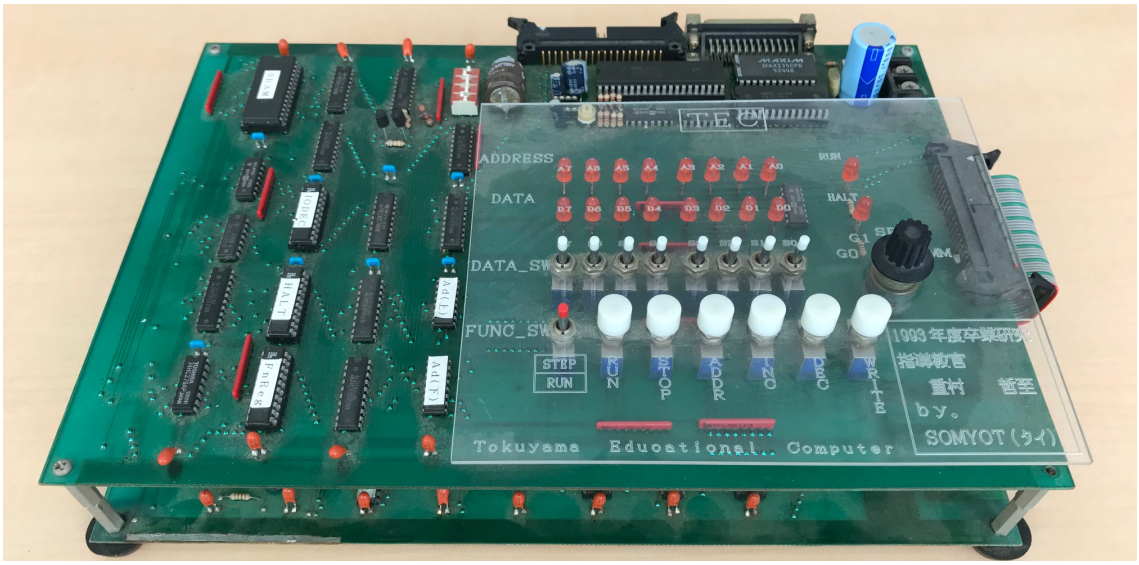
現在では、FPGA を用いることで上記の制約は全て解決できる。現代の FPGA を用いて、美しく高速な TeC CPU を再設計する。

(1) FPGA を用いれば、ALU、レジスタファイル等の仕様を自由に決定可能

(2) FPGA には十分な素子と十分な配線を集積可能

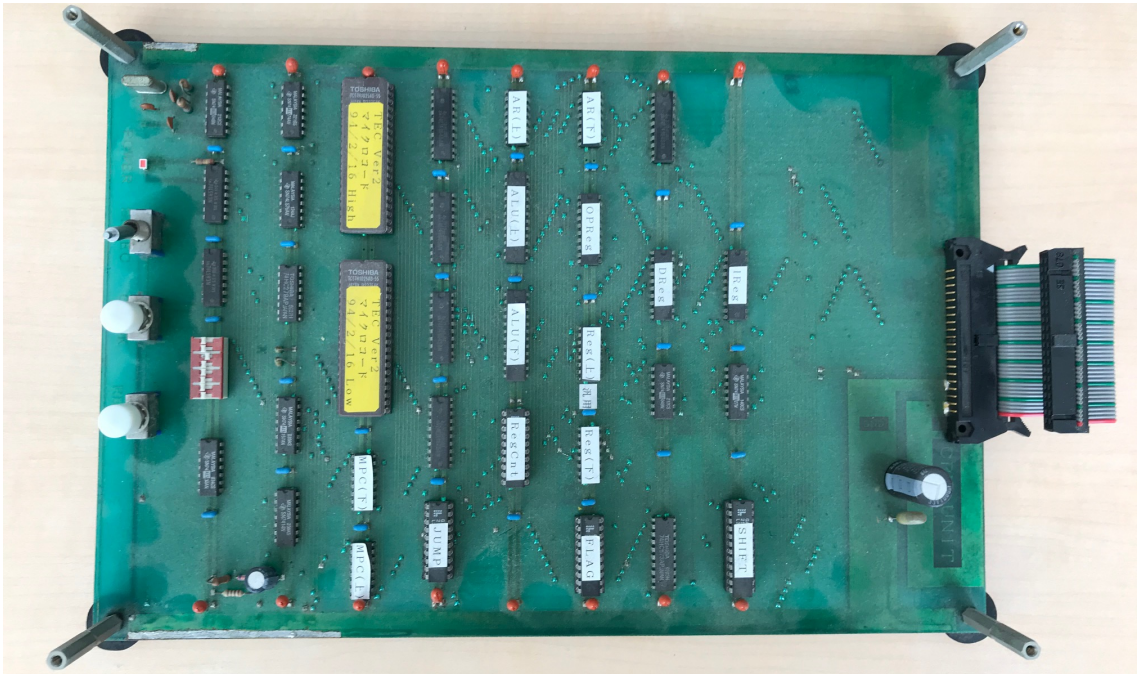
(3) FPGA ではクロック専用の配線によりクロックスキューを無視できる。
FPGA の開発システムはセットアップタイム、ホールドタイムが不足する箇所を自動的に見つけてくれる。

初代 TeC



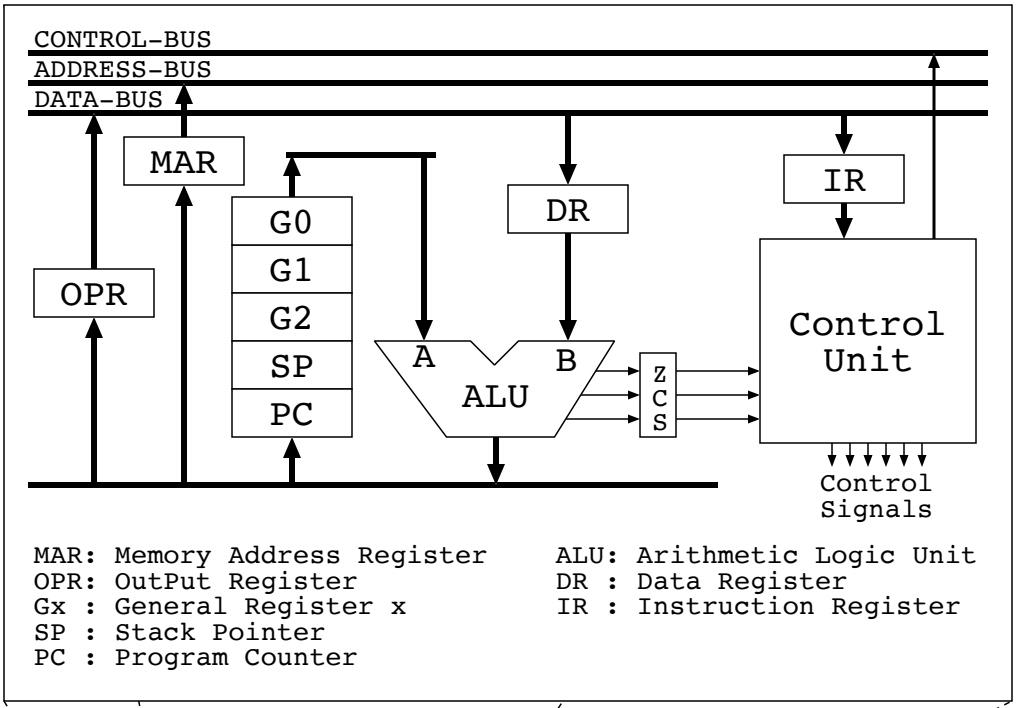
1. TeC6 と基本機能はほぼ同じ
2. 基板は A4×２枚
3. 上段の基板がメモリ（S-RAM），I/O（8251，8255），コンソールパネル

初代 TeC の CPU



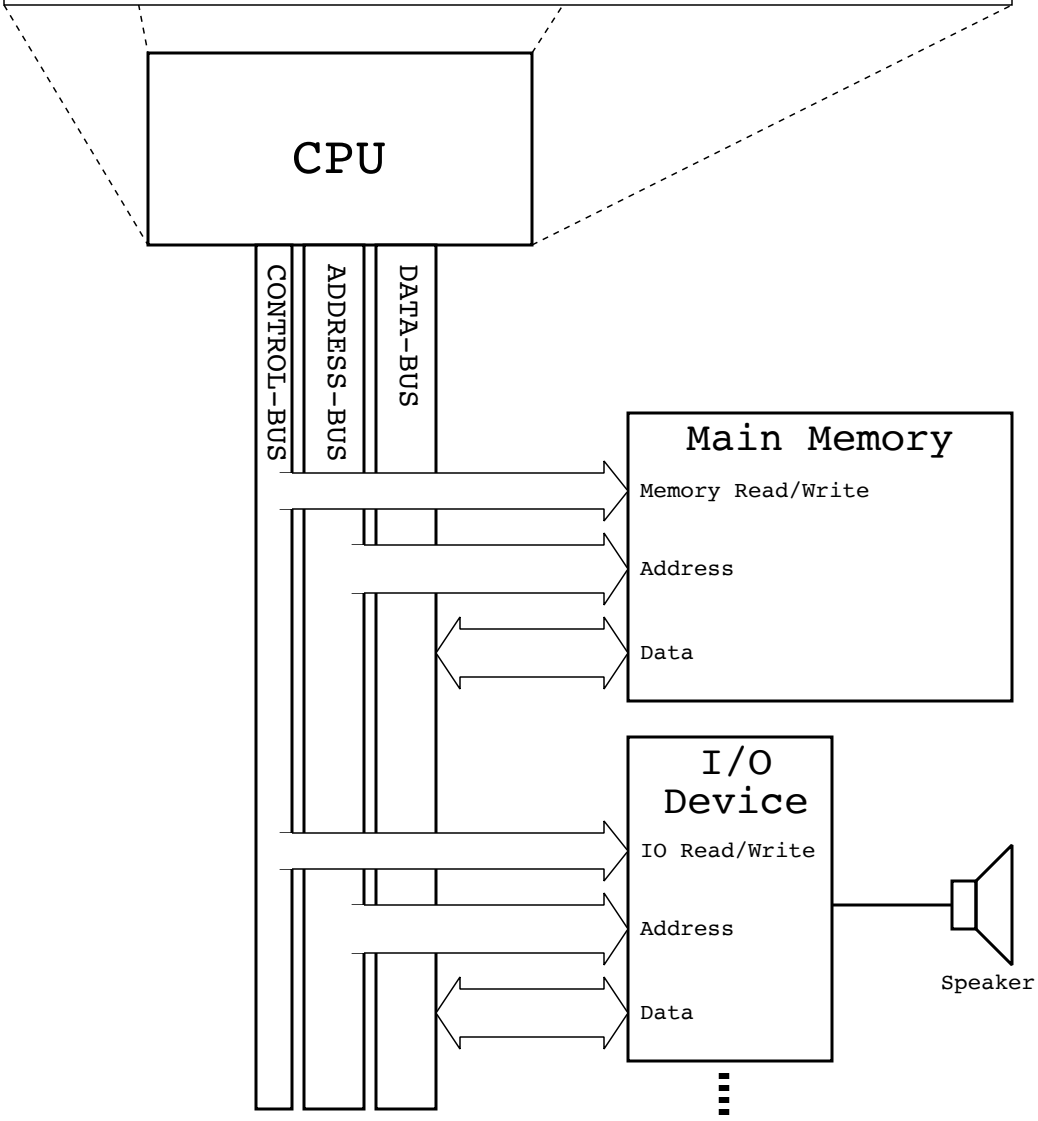
1. 汎用部品（TTL，ROM）と GAL で構成
2. 実際に製作するために，部品点数と配線を少なくする必要があった.
3. マイクロプログラムで複雑な制御をすることで解決した.

TeC CPUの内部構造



ALUの機能

A		
B		
A	+	B
A	-	B
A	+	1
A	-	1
A	and	B
A	or	B
A	xor	B
...		



TEC6命令表

Ver. 2.3 (2006.3.25)

※ステート数：イミディエイト／ダイレクト／インデクスド
(ジャンプ命令では、条件不成立／ダイレクト／インデクスド)

ニーモニック	命令	第１バイト OP	第１バイト GRXR	第２バイト	フラグ変化	ステート数※	動作
NO	No Opration	0000	00 00	————	×	3	何もしない
LD	Load	0001	GR XR	aaaa aaaa	×	5/7/7	GR <- [EA]
ST	Store	0010	GR XR	aaaa aaaa	×	-/7/7	[EA] <- GR
ADD	Add	0011	GR XR	aaaa aaaa	○	5/7/7	GR <- GR + [EA]
SUB	Subtract	0100	GR XR	aaaa aaaa	○	5/7/7	GR <- GR - [EA]
CMP	Compare	0101	GR XR	aaaa aaaa	○	5/7/7	GR - [EA]
AND	Logical And	0110	GR XR	aaaa aaaa	○	5/7/7	GR <- GR & [EA]
OR	Logical Or	0111	GR XR	aaaa aaaa	○	5/7/7	GR <- GR | [EA]
XOR	Logical Xor	1000	GR XR	aaaa aaaa	○	5/7/7	GR <- GR ^ [EA]
SHLA	Shift Left Arithmetic	1001	GR 00	————	○	4	GR <- GR << 1
SHLL	Shift Left Logical	1001	GR 01	————	○	4	GR <- GR << 1
SHRA	Shift Right Arithmetic	1001	GR 10	————	○	4	GR <- GR >> 1
SHRL	Shift Right Logical	1001	GR 11	————	○	4	GR <- GR >>> 1
JMP	Jump	1010	00 XR	aaaa aaaa	×	-/5/6	PC <- EA
JZ	Jump on Zero	1010	01 XR	aaaa aaaa	×	4/5/6	if Zero PC <- EA
JC	Jump on Carry	1010	10 XR	aaaa aaaa	×	4/5/6	if Carry PC <- EA
JM	Jump on Minus	1010	11 XR	aaaa aaaa	×	4/5/6	if Sign PC <- EA
CALL	Call subroutine	1011	11 XR	aaaa aaaa	×	-/6/7	[--SP]<-PC, PC<-EA
~~IN~~	~~Input~~	~~1100~~	~~GR 00~~	~~0000 pppp~~	×	8	~~GR <- IO[P]~~
~~OUT~~	~~Output~~	~~1100~~	~~GR 11~~	~~0000 pppp~~	×	8	~~IO[P] <- GR~~
PUSH	Push Register	1101	GR 00	————	×	6	[--SP] <- GR
~~PUSHF~~	~~Push Flag~~	~~1101~~	~~11 01~~		×	6	~~[--SP] <- FLAG~~
POP	Pop Register	1101	GR 10	————	×	6	GR <- [SP++]
~~POPF~~	~~Pop Flag~~	~~1101~~	~~11 11~~		○	6	~~FLAG <- [SP++]~~
~~EI~~	~~Enable Interrupt~~	~~1110~~	~~00 00~~		×	4	~~割り込み許可~~
~~DI~~	~~Disable Interrupt~~	~~1110~~	~~00 11~~		×	4	~~割り込み禁止~~
RET	Return from subroutine	1110	11 00	————	×	6	PC <- [SP++]
~~RETI~~	~~Return from Interrupt~~	~~1110~~	~~11 11~~		×	6	~~PC <- [SP++], EI~~
HALT	Halt	1111	11 11	————	×	4	停止

GR	意味
00	G0
01	G1
10	G2
11	SP

XR	意味
00	ダイレクトモード
01	G1インデクスドモード
10	G2インデクスドモード
11	イミディエイトモード

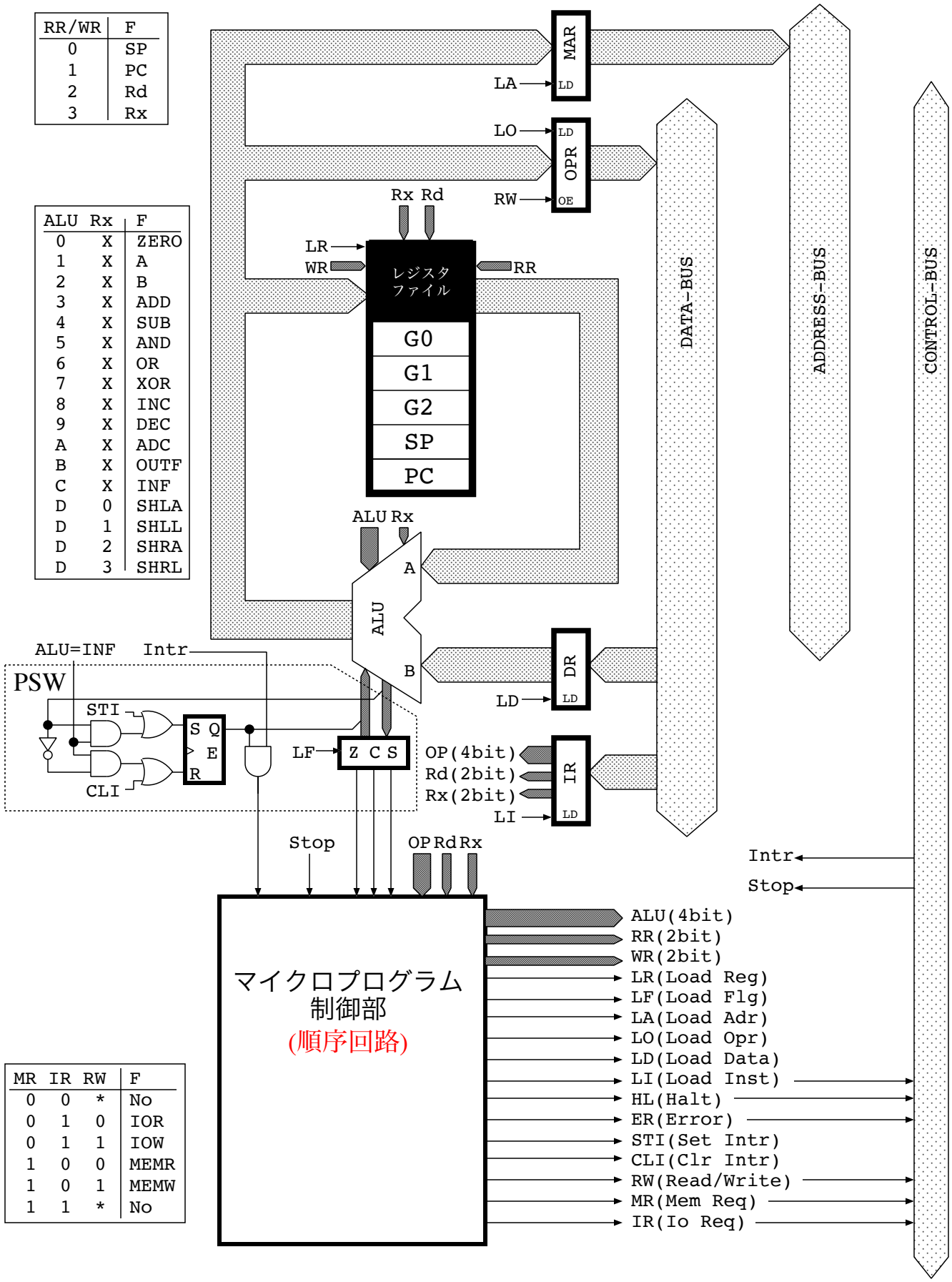
メモリマップ

Addr	内容
00 | ~~DB~~ FF	RAM
~~DC~~	~~INT0 割り込みベクタ~~
~~DD~~	~~INT1 割り込みベクタ~~
~~DE~~	~~INT2 割り込みベクタ~~
~~DF~~	~~INT3 割り込みベクタ~~
~~E0~~ | ~~FF~~	~~ROM(IPL)~~

INT1 : SIO 受信割り込み
INT2 : SIO 送信割り込み

I/Oマップ

Addr	Read/Write
0	Data-Sw/b0:Beep
1	Data-Sw/b0:Speaker
2	SIO-Data/SIO-Data
3	b7:Tx Ready ,b7:Tx STI b6:Rx Ready/b6:Rx STI
4	空き/空き
...	....
F	空き/空き



TeC6 CPUのブロック図

TeC の CPU が命令を実行する様子

1. TeC が実行するプログラムの例

```
番地   機械語   ラベル   命令     オペランド
00     14               LD       G1,03H
01     03
02     FF               HALT
03     AB               DC       0ABH
```

2. 上のプログラムがメモリに格納されている状態での CPU の動作

① PC を００H にする（ユーザの操作）
② RUN ボタンを押す（ユーザの操作）
③ 実行が開始されると CPU はクロック毎に動作する

マイクロ操作		制御信号								
ステート	操作	ALU	RR	WR	LR	LA	LI	LD	MR	HL
1 共通	PC→MAR, Stop チェック	1	1			1				
2 共通	メモリ→IR、PC++	8	1	1	1		1		1	
3 共通	命令デコード、PC→MAR	1	1			1				
4LD	メモリ→DR、PC++	8	1	1	1			1	1	
5LD	DR→MAR	2				1				
6LD	メモリ→DR							1	1	
7LD	DR→G1、１共通に戻る	2		2	1					
１共通	PC→MAR, Stop チェック	1	1			1				
2 共通	メモリ→IR、PC++	8	1	1	1		1		1	
3 共通	命令デコード、PC→MAR	1	1			1				
4HALT	Halt 出力、１共通に戻る									1

制御信号で空白は「０」の意味