3章 インストラクション

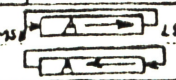
1chip型のROM内蔵のものは187命令で、EVACHIP及びROM外付型は191命令有ります。

全て1ワード(16ビット)命令で 命令実行サイクルはテーブル参照命令は2マシンサイクルですが、それ以外は全て 1マシンサイクルであり 非常に高速にリアルタイムに素片合成方式の音声合成ができる命令体系になっています。

3.1 インストラクションでの記号

A	アキュムレータ
STK	スタックレジスタ(12ビットまは16ビット)
PC	プログラムカウンタ(12ビットまは16ビット)
H	データポインタ(6ビット)
(H)	データポインタH指定のメモリ内容(8ビット)
Rr	イミディエイトデータr指定のレジスタ (8ビット)
PRr.	イミディエイトデータr指定のペアレジスタ(16ビット)
SP	スタックポインタ(3ビット)
(SP)	スタックポインタ指定のメモリー内容(16ビット)
X	Xレジスタ(被乗数レジスタ, 7ビット)
Y	Yレジスタ(乗数レジスタ, 5ビット)
N	Nレジスタ(分周比レジスタ, 8ビット)
MD1	モードレジスタ上位3ビット (MD5~7)
MD0	モードレジスタ下位7ビット (MD0~4,8,9)
MD	モードレジスタ (MD0~7 8ビット)
RG	擬似ランダムジェネレータ(7ビット)
DA	DAコンバータラッチ(符号+8ビット)
PA	PAポート(I/Oポート)ラッチ
PB	PBポート(I/Oポート)
TS	トーンサイン
NS	ノイズサイン
SS	サウンド出力サイン
FL0	フラグ0
n	イミディエイトデータ(3~8ビット, 12ビット)
r	レジスタ指定のイミディエイトデータ(5ビット)
A'	アキュムレータ退避レジスタ
(Rr)	ペアレジスタPRrで指定されたプログラムメモリ(ROM)の内容
+	加算
−	減算
∧	論理積
∨	論理和
∀	排他的論理和
Ca	8ビット目キャリー有
Bo	8ビット目ボロー有
Z	演算結果=0
$\overline{Ca}$	8ビット目キャリー無
$\overline{Bo}$	8ビット目ボロー無
$\overline{Z}$	演算結果≠0

C_5	5ビット目キャリー出力
C_6	6ビット目キャリー出力
C_5INH	5ビット目から6ビット目へのキャリーを禁止
C_5 or C_6 INH	5ビット目から6ビット目へのキャリーか6ビット目から7ビット目へのキャリーを禁止

命令群	ニーモニック	命令コード $D_{15}D_{14}D_{13}D_{12}D_{11}D_{10}D_9D_8D_7D_6D_5D_4D_3D_2D_1D_0$	HEXA				オペレーション	スキップ条件	マシンサイクル	コメント
データ転送命令（イミディエイト）	MVI Rr, n	0 1 0 ← r → ← n →	4+r2	r1	n2	n1	Rr ← n		1	Rrはr指定のレジスタ
	MVI (H), n	0 0 1 1 0 0 1 0 ← n →	3	2	n2	n1	(H) ← n			H指定のデータメモリー
	MVI A, n	0 0 1 1 0 1 0 0 ← n →	3	4	n2	n1	A ← n			アキュームレータ
	MVI H, n	0 0 1 1 1 0 0 0 0 0 ← n →	3	8	n2	n1	H ← n			データポインタ6ビット
	MVI MD1, n	0 0 1 1 0 0 0 1 ← n → 0 0 0 0 0	3	1	2n	0	MD1 ← n			MD1=MD5~7
	MVI MD0, n	0 0 1 0 0 0 0 1 0 ← n →	2	1	n2	n1	MD0 ← n			MD0=MD0~4 8,9
データ転送命令	MOV N, A	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1	0	2	0	1	N ← A			
	MOV X, A	0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0	0	2	0	8	X ← A			TSには転送しない
	MOV (H), A	0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1	1	6	0	1	(H) ← A			
	MOV A, (H)	0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1	1	4	0	5	A ← (H)			
	XCHG (H), A	0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1	1	6	0	5	(H) ↔ A			
	MOV Rr, A	0 0 0 1 0 0 1 ← r → 0 0 0 1	1	2+r2	r1	1	Rr ← A			
	MOV A, Rr	0 0 0 1 0 0 0 ← r → 0 1 0 1	1	r2	r1	5	A ← Rr			
	XCHG Rr, A	0 0 0 1 0 0 1 ← r → 0 1 0 1	1	2+r2	r1	5	Rr ↔ A			
	MOV Rr, H	0 0 0 1 0 0 1 ← r → 0 0 1 0	1	2+r2	r1	2	Rr ← H			
	MOV H, Rr	0 0 0 1 0 0 0 ← r → 1 0 1 0	1	r2	r1	A	H ← Rr			
	XCHG Rr, H	0 0 0 1 0 0 1 ← r → 1 0 1 0	1	2+r2	r1	A	Rr ↔ H			
	MOV Y, Rr	0 0 0 1 0 0 0 ← r → 0 0 0 0	1	r2	r1	0	Y ← Rr			Yは5ビット
	MOV X, RG	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	0	0	0	8	X ← RG			RGはポリノミナルカウンター
回転命令	RAR	0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0	0	4	0	4	A ← RRT			
	RAL	0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0	0	4	0	8	A ← LRT			
入出力命令	IN PA	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1	0	4	0	1	A ← PA			} 入力 PAポート PBポート
	IN PB	0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0	0	4	0	2	A ← PB			
	OUT PA	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	0	0	0	2	PA ← A			} 出力 PAポート PBポート
	OUT PB	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0	0	0	4	PB ← A			
	OUT DA	0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0	0	5	0	2	DA ← A			D/Aコンバータに出力
乗算命令	MUL1	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0	0	5	0	4	MULT1 (A←RS(A) if Y0=0; A←A if Y0=1) Y→RS			
	MUL2	0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0	0	5	0	C	MULT2 (A←RS(A) if Y0=0; A←RS(A+X) if Y0=1) Y→RS			
混合命令	MIX Rr	0 0 0 1 0 1 0 ← r → 1 0 0 1	1	4+r2	r1	9	A←Rr±A, SS←TS∀Bo		↓	Rr<A∀同符号—ADD 〃 異符号—SUB

命令群	ニーモニック	命令コード $D_{15}D_{14}D_{13}D_{12}D_{11}D_{10}D_9D_8D_7D_6D_5D_4D_3D_2D_1D_0$	HEXA				オペレーション	スキップ条件	マシンサイクル	コメント
テーブル参照命令	TBLO A,(Rr)	0 0 0 1 1 0 0 ← r → 0 0 0 1	1	8+r_2	r_1	1	A←(PRr)$_0$ (PC15←0, PC0~14←RS(PRr), A←Π0~7 (ODF=0), A←Π8~15 (ODF=1))		2	(PRr)は16ビット出力 (PRr)がEVEN→ODF=0 ODD→ODF=1
	TBLO X,(Rr)	0 0 0 1 1 0 0 ← r → 0 0 1 0	1	8+r_2	r_1	2	X←(PRr)$_0$ (PC15←0, PC0~14←RS(PRr), X←Π0~6, TS←Π7)		2	Xは7ビット TSはトーン符号
	TBLO Y,(Rr)	0 0 0 1 1 0 0 ← r → 0 1 0 0	1	8+r_2	r_1	4	Y←(PRr)$_0$ (PC15←0, PC0~8←RS(PRr), Y←Π0~4, Π8~12)		2	
	TBLI A,(Rr)	0 0 0 1 1 0 1 ← r → 0 0 0 1	1	A+r_2	r_1	1	A←(PRr)$_1$ (PC15←1, PC0~14←RS(PRr), A←Π0~7 (ODF=0), A←Π8~15 (ODF=1))		2	(PRr)は16ビット出力 (PRr)がEVEN→ODF=0 ODD→ODF=1
	TBLI X,(Rr)	0 0 0 1 1 0 1 ← r → 0 0 1 0	1	A+r_2	r_1	2	X←(PRr)$_1$ (PC15←1, PC0~14←RS(PRr), X←Π0~6, TS←Π15)		2	
	TBLI Y,(Rr)	0 0 0 1 1 0 1 ← r → 0 1 0 0	1	A+r_2	r_1	4	Y←(PRr)$_1$ (PC15←1, PC0~12←RS(PRr), Y←Π0~4 (ODF=0), Y←Π8~12 (ODF=1))		2	
ジャンプ命令	JMP n	0 1 1 0 ← n →	6	n_3	n_2	n_1	JP n (PC0~11←n)		1	PC12~15変らず
		0 0 1 0 0 0 0 0 ← n → 0 0 0 0	2	0	n	0	JPP n (PC12~15←n)			ページジャンプ PC0~7変らず
	JMPA	0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1	0	5	0	1	JPA (PC0~7←A)			PC8~15変らず
	JMPFZ n	0 0 1 0 1 0 0 0 ← n →	2	8	n_2	n_1	FLOJR (ifFLO=0, PC0~7←n; ifFLO=1, PC0~7←PC+1)			PC8~15変らず FLQリセット
コール命令	CALL n	0 1 1 1 ← n →	7	n_3	n_2	n_1	JSn (PC0~11←n, PC12~15←0; (ST)←PC+1, SP←SP+1)			必らず0ページへコール。 PC+1は16ビット
	CALLO (Rr)	0 0 0 1 1 0 0 ← r → 1 0 0 0	1	8+r_2	r_1	8	PC←(PRr)$_0$ (PC15←0, PC0~14←RS(PRr), (SP)←PC+1, SP←SP+1)		2	テーブル参照コール命令。 PC+1は16ビット
	CALLI (Rr)	0 0 0 1 1 0 1 ← r → 1 0 0 0	1	A+r_2	r_1	8	PC←(PRr)$_1$ (PC15←1, PC0~14←RS(PRr), (SP)←PC+1, SP←SP+1)		2	テーブル参照コール命令 PC+1は16ビット
リターン命令	RET	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	0	8	0	0	RET (SP←SP-1, PC←(SP))		1	コール命令のリターン
	RETS	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1	0	8	0	1	RET+1 (SP←SP-1, PC←(SP), NEXT ADR SKIP)	○無条件		
	RETI	0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1	0	9	0	F	RETI (SP←SP-1, PC←SP, INTR, A←A', SKIP←SKIP')			割込みのリターン
その他の命令	STF	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1	0	0	0	5	FLQS			フラグをセット
	OFF	0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0	0	6	0	2	PWOFF			パワーOFF命令
	NOP	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0	0	0	0	NOP			

命令群	ニーモニック	命令コード D_{15} D_{14} D_{13} D_{12} D_{11} D_{10} D_9 D_8	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	HEXA				オペレーション	スキップ条件	マシンサイクル	コメント
演算命令（アキュムレータ）イミーディエト・データ算術論理	ADI A,n	1 0 0 0 0 0 0 0	← n →	8	0	n_2	n_1	A←A+n		1	
	ANDI A,n	0 0 0 0 1 0		8	2			A←A∧n			
	SBI A,n	0 0 0 1 0 0		8	4			A←A−n			
	ORI A,n	0 0 0 1 1 0		8	6			A←A∨n			
	ADIS A,n	0 1 0 0 0 0		8	8			A←A+n	Ca		
	ANDIS A,n	0 1 0 0 1 0		8	A			A←A∧n	Z		
	SBIS A,n	0 1 0 1 0 0		8	C			A←A−n	Bo		
	XORI A,n	0 1 0 1 1 0		8	E			A←A∀n			
	TADINC A,n	1 0 0 0 0 0		9	0			A+n	$\overline{Ca}$		
	TANDINZ A,n	1 0 0 1 0 0		9	2			A∧n	$\overline{Z}$		
	TSBINC A,n	1 0 1 0 0 0		9	4			A−n	$\overline{Bo}$		
	TSBINZ A,n	1 0 1 1 0 0		9	6			A=n	≠		
	TADIC A,n	1 1 0 0 0 0		9	8			A+n	Ca		
	TANDIZ A,n	1 1 0 1 0 0		9	A			A∧n	Z		
	TSBIC A,n	1 1 1 0 0 0		9	C			A−n	Bo		
	TSBIZ A,n	1 1 1 1 1 0		9	E			A=n	=		
演算命令（モードレジスタ）イミーディエト・データ算術論理	ADI MDI,n	1 0 0 0 0 0 0 1	← n_1 → 0 0 0 0 0 0	8	1	2n	0	MDI←MDI+n			MDIはMD5~7だけである。 MD_5 = EXT INT E MD_6 = OUT MD_7 = IF ~~n = n1×20H~~
	ANDI MDI,n	0 0 0 1 1		8	3			MDI←MDI∧n			
	SBI MDI,n	0 0 1 0 1		8	5			MDI←MDI−n			
	ORI MDI,n	0 0 1 1 1		8	7			MDI←MDI∨n			
	ADIS MDI,n	0 1 0 0 1		8	9			MDI←MDI+n	Ca		
	ANDIS MDI,n	0 1 0 1 1		8	B			MDI←MDI∧n	Z		
	SBIS MDI,n	0 1 1 0 1		8	D			MDI←MDI−n	Bo		
	XORI MDI,n	0 1 1 1 1		8	F			MDI←MDI∀n			
	TADINC MD,n	1 0 0 0 1	← n →	9	1	n_2	n_1	MD+n	$\overline{Ca}$		MDはMD0~7でNSF1, NSF2は除く, MD_0 = 64/32 MD_1 = TONE INT E MD_2 = NS INT E MD_3 = NSS MD_4 = TIME INT E
	TANDINZ MD,n	1 0 0 1 1		9	3			MD∧n	$\overline{Z}$		
	TSBINC MD,n	1 0 1 0 1		9	5			MD−n	$\overline{Bo}$		
	TSBINZ MD,n	1 0 1 1 1		9	7			MD=n	≠		
	TADIC MD,n	1 1 0 0 1		9	9			MD+n	Ca		
	TANDIZ MD,n	1 1 0 1 1		9	B			MD∧n	Z		
	TSBIC MD,n	1 1 1 0 1		9	D			MD−n	Bo		
	TSBIZ MD,n	1 1 1 1 1		9	F			MD=n	=		

命令群	ニーモニック	命令コード $D_{15}D_{14}D_{13}D_{12}D_{11}D_{10}D_9D_8D_7D_6D_5D_4D_3D_2D_1D_0$	HEXA		オペレーション	スキップ条件	マシンサイクル	コメント
演算命令(データメモリ)イミーディエト・データ算術論理	ADI (H),n	1 0 1 0 0 0 0 0 n	A	0	(H)←(H)+n		1	
	ANDI (H),n	0 0 1 n	A	2	(H)←(H)∧n			
	SBI (H),n	0 1 0 n	A	4	(H)←(H)−n			
	ORI (H),n	0 1 1 n	A	6	(H)←(H)∨n			
	ADIS (H),n	1 0 0 n	A	8	(H)←(H)+n	Ca		
	ANDIS (H),n	1 0 1 n	A	A	(H)←(H)∧n	Z		
	SBIS (H),n	1 1 0 n	A	C	(H)←(H)−n	Bo		
	XORI (H),n	1 1 1 n	A	E	(H)←(H)∀n			
	TADINC (H),n	1 0 0 0 n	B	0	(H)+n	Ca		
	TANDINZ (H),n	0 0 1 n	B	2	(H)∧n	Z		
	TSBINC (H),n	0 1 0 n	B	4	(H)−n	Bo		
	TSBINZ (H),n	0 1 1 n	B	6	(H)=n	キ		
	TADIC (H),n	1 0 0 n	B	8	(H)+n	Ca		
	TANDIZ (H),n	1 0 1 n	B	A	(H)∧n	Z		
	TSBIC (H),n	1 1 0 n	B	C	(H)−n	Bo		
	TSBIZ (H),n	1 1 1 n	B	E	(H)=n	=		
演算命令(データポインタ)イミーディエト・データ算術論理	ADI H,n	1 0 1 0 0 0 0 1 0 0 ←n→	A	1	H←H+n			nは6ビット
	ANDI H,n	0 0 1 n	A	3	H←H∧n			
	SBI H,n	0 1 0 n	A	5	H←H−n			
	ORI H,n	0 1 1 n	A	7	H←H∨n			
	ADIS H,n	1 0 0 1 1 $C+n_2$	A	9	H←H+n	Ca		
	ANDIS H,n	1 0 1 0 0 n_2	A	B	H←H∧n	Z		
	SBIS H,n	1 1 0	A	D	H←H−n	Bo		
	XORI H,n	1 1 1	A	F	H←H∀n			
	TADINC H,n	1 0 0 0 1 1 $C+n_2$	B	1	H+n	Ca		
	TANDINZ H,n	0 0 1 0 0 n_2	B	3	H∧n	Z		
	TSBINC H,n	0 1 0	B	5	H−n	Bo		
	TSBINZ H,n	0 1 1	B	7	H=n	キ		
	TADIC H,n	1 0 0 1 1 $C+n_2$	B	9	H+n	Ca		
	TANDIZ H,n	1 0 1 0 0 n_2	B	B	H∧n	Z		
	TSBIC H,n	1 1 0	B	D	H−n	Bo		
	TSBIZ H,n	1 1 1	B	F	H=n	=		

命令群	ニーモニック	命令コード $D_{15}D_{14}D_{13}D_{12}D_{11}D_{10}D_9D_8D_7D_6D_5D_4D_3D_2D_1D_0$	HEXA	r_2	r_1	n	オペレーション	スキップ条件	マシンサイクル	コメント
演算命令（レジスタ）イミーディエト・データ算術論理	ADI Rr,n	1 1 1 0 0 0 0 ← r → ← n →	E				Rr←Rr+n		1	n は4ビット
	ADIS Rr,n	0 0 1	E	2+			Rr←Rr+n	Ca		
	SBI Rr,n	0 1 0	E	4+			Rr←Rr−n			
	SBIS Rr,n	0 1 1	E	6+			Rr←Rr−n	$\overline{Bo}$		
	TADINC Rr,n	1 0 0	E	8+			Rr+n	$\overline{Ca}$		
	TADIC Rr,n	1 0 1	E	A+			Rr+n	Ca		
	TSBINC Rr,n	1 1 0	E	C+			Rr−n	$\overline{Bo}$		
	TSBIC Rr,n	1 1 1	E	E+			Rr−n	Bo		
	ADI5 Rr,n	1 1 1 1 0 0 0	F				Rr←Rr+n (C_5 INH)			
	ADIMS Rr,n	0 0 1	F	2+			Rr←Rr+n (C_5 or C_6 INH) (if MD 6分32=0…C_5, 6分32=1…C_6)	C_5 or C_6 (if 6分32=0 / if 6分32=1)		MD 6分32 = MD_0
	TADI5 Rr,n	1 0 0	F	8+			Rr+n	C_5		
命令モニター	MON	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1	0	1	0	1	MONI←SP [ML7←TM INTF; ML6←TM INTF; ML5←NS INTF; ML4←EXT INTF; ML3←SKIP F; ML2−0←SP2−0]			EVACHIP モニター用命令.

命令群	ニーモニック	命令コード $D_{15}D_{14}D_{13}D_{12}D_{11}D_{10}D_9D_8D_7D_6D_5D_4D_3D_2D_1D_0$	HEXA				オペレーション	スキップ条件	マシンサイクル	コメント
算術論理演算命令（アキュムレータ・レジスタ間）	AD A, Rr	1 1 0 0 0 0 0 r r r r 0 0 0 0	C	r_2	r_1	0	A ← A + Rr		1	
	AND A, Rr	0 0 1	C	2+		0	A ← A ∧ Rr			
	SB A, Rr	0 1 0	C	4+		0	A ← A − Rr			
	OR A, Rr	0 1 1	C	6+		0	A ← A ∨ Rr			
	ADS A, Rr	1 0 0	C	8+		0	A ← A + Rr	Ca		
	ANDS A, Rr	1 0 1	C	A+		0	A ← A ∧ Rr	Z		
	SBS A, Rr	1 1 0	C	C+		0	A ← A − Rr	Bo		
	XOR A, Rr	1 1 1	C	E+		0	A ← A ∀ Rr			
	TADNC A, Rr	1 0 0 0	D			0	A + Rr	$\overline{Ca}$		
	TANDNZ A, Rr	0 0 1	D	2+		0	A ∧ Rr	$\overline{Z}$		
	TSBNC A, Rr	0 1 0	D	4+		0	A − Rr	$\overline{Bo}$		
	TSBNZ A, Rr	0 1 1	D	6+		0	A = Rr	≠		
	TADC A, Rr	1 0 0	D	8+		0	A + Rr	Ca		
	TANDZ A, Rr	1 0 1	D	A+		0	A ∧ Rr	Z		
	TSBC A, Rr	1 1 0	D	C+		0	A − Rr	Bo		
	TSBZ A, Rr	1 1 1	D	E+		0	A = Rr	=		
	AD Rr, A	1 1 0 0 0 0 0 r r r r 1 0 0 0	C			8	Rr ← Rr + A			
	AND Rr, A	0 0 1	C	2+		8	Rr ← Rr ∧ A			
	SB Rr, A	0 1 0	C	4+		8	Rr ← Rr − A			
	OR Rr, A	0 1 1	C	6+		8	Rr ← Rr ∨ A			
	ADS Rr, A	1 0 0	C	8+		8	Rr ← Rr + A	Ca		
	ANDS Rr, A	1 0 1	C	A+		8	Rr ← Rr ∧ A	Z		
	SBS Rr, A	1 1 0	C	C+		8	Rr ← Rr − A	Bo		
	XOR Rr, A	1 1 1	C	E+		8	Rr ← Rr ∀ A			
	TADNC Rr, A	1 0 0 0	D			8	Rr + A	$\overline{Ca}$		
	TANDNZ Rr, A	0 0 1	D	2+		8	Rr ∧ A	$\overline{Z}$		
	TSBNC Rr, A	0 1 0	D	4+		8	Rr − A	$\overline{Bo}$		
	TSBNZ Rr, A	0 1 1	D	6+		8	Rr = A	≠		
	TADC Rr, A	1 0 0	D	8+		8	Rr + A	Ca		
	TANDZ Rr, A	1 0 1	D	A+		8	Rr ∧ A	Z		
	TSBC Rr, A	1 1 0	D	C+		8	Rr − A	Bo		
	TSBZ Rr, A	1 1 1	D	E+		8	Rr = A	=		

命令群	ニーモニック	命令コード D_{15} D_{14} D_{13} D_{12} D_{11} D_{10} D_9 D_8 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	HEXA				オペレーション	スキップ条件	マシンサイクル	コメント
算術論理演算命令（アキュムレータ・データメモリ間）	AD A,(H)	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1	C	0	0	1	A ← A +(H)		1	
	AND A,(H)	0 0 1	C	2	0	1	A ← A ∧(H)			
	SB A,(H)	0 1 0	C	4	0	1	A ← A −(H)			
	OR A,(H)	0 1 1	C	6	0	1	A ← A ∨(H)			
	ADS A,(H)	1 0 0	C	8	0	1	A ← A +(H)	Ca		
	ANDS A,(H)	1 0 1	C	A	0	1	A ← A ∧(H)	Z		
	SBS A,(H)	1 1 0	C	C	0	1	A ← A −(H)	Bo		
	XOR A,(H)	1 1 1	C	E	0	1	A ← A ∀(H)			
	TADNC A,(H)	1 0 0 0	D	0	0	1	A +(H)	$\overline{Ca}$		
	TANDNZ A,(H)	0 0 1	D	2	0	1	A ∧(H)	$\overline{Z}$		
	TSBNC A,(H)	0 1 0	D	4	0	1	A −(H)	$\overline{Bo}$		
	TSBNZ A,(H)	0 1 1	D	6	0	1	A =(H)	≠		
	TADC A,(H)	1 0 0	D	8	0	1	A +(H)	Ca		
	TANDZ A,(H)	1 0 1	D	A	0	1	A ∧(H)	Z		
	TSBC A,(H)	1 1 0	D	C	0	1	A −(H)	Bo		
	TSBZ A,(H)	1 1 1	D	E	0	1	A =(H)	=		
	AD (H),A	1 1 0 0 0 0 0 1 0 0 1	C	0	0	9	(H) ← (H) + A			
	AND (H),A	0 0 1	C	2	0	9	(H) ← (H) ∧ A			
	SB (H),A	0 1 0	C	4	0	9	(H) ← (H) − A			
	OR (H),A	0 1 1	C	6	0	9	(H) ← (H) ∨ A			
	ADS (H),A	1 0 0	C	8	0	9	(H) ← (H) + A	Ca		
	ANDS (H),A	1 0 1	C	A	0	9	(H) ← (H) ∧ A	Z		
	SBS (H),A	1 1 0	C	C	0	9	(H) ← (H) − A	Bo		
	XOR (H),A	1 1 1	C	E	0	9	(H) ← (H) ∀ A			
	TADNC (H),A	1 0 0 0	D	0	0	9	(H) + A	$\overline{Ca}$		
	TANDNZ (H),A	0 0 1	D	2	0	9	(H) ∧ A	$\overline{Z}$		
	TSBNC (H),A	0 1 0	D	4	0	9	(H) − A	$\overline{Bo}$		
	TSBNZ (H),A	0 1 1	D	6	0	9	(H) = A	≠		
	TADC (H),A	1 0 0	D	8	0	9	(H) + A	Ca		
	TANDZ (H),A	1 0 1	D	A	0	9	(H) ∧ A	Z		
	TSBC (H),A	1 1 0	D	C	0	9	(H) − A	Bo		
	TSBZ (H),A	1 1 1	D	E	0	9	(H) = A	=		

3.6. 乗算命令

MUL1　　(Multiply Accumulator Contents by Y Register Contents)

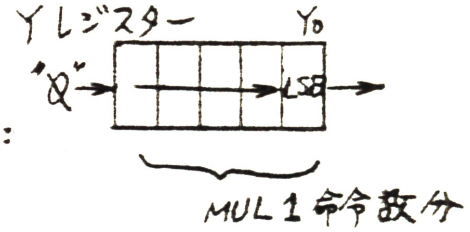
if $Y_0=0$, $A \leftarrow RS(A)$ and $Y \leftarrow RS(Y)$

if $Y_0=1$, $A \leftarrow A$ and $Y \leftarrow RS(Y)$

もしYレジスタのLSB Y_0が"0"なら，アキュムレータを1ビットライトシフトし，Yレジスタも1ビットライトシフトします．
もしYレジスタのLSB Y_0が"1"なら，アキュムレータはそのままで，Yレジスタは1ビットライトシフトします。
ライトシフトされたアキュムレータ及びYレジスタのMSBには"0"が入り，そのLSBは切り捨てられます．
次のように命令を実行すると，アキュムレータの値は およそ

```
MOV  Y, Rr
MUL  1
MUL  1
MUL  1
MUL  1
MUL  1
```

MUL1命令数: 最大5個



(A)× 1/2(Yレジスタの下位からMUL1命令数分に含まれる0の個数) となります．(ライトシフトされて切り捨てられた"1"が誤差になって現れます.)

MUL2　　(Multiply X Register Contents by Y Register Contents)

if $Y_0=0$, $A \leftarrow RS(A)$ and $Y \leftarrow RS(Y)$

if $Y_0=1$, $A \leftarrow RS(A+X)$ and $Y \leftarrow RS(Y)$

もしYレジスタのLSB Y_0が"0"ならば，アキュムレータの内容を1ビットライトシフトし，Yレジスタも1ビットライトシフトします。
もし，Yレジスタの LSB Y_0が"1"ならば アキュムレータの内容と Xレジスタの内容を加算して その結果を1ビットライトシフトして アキュムレータにストアします。Yレジスタも1ビットライトシフトします。
ライトシフトされたアキュムレータ及びYレジスタのMSBには"0"が入り，そのLSBは切り捨てられます．

次のようにMUL2命令を5個使用してXとYを乗算させると、

```
MVI   A, 00
TBLQ  X, R0
TBLQ  Y, R2
MUL2  ┐
MUL2  │
MUL2  │ MUL2 命令数：最大 5個
MUL2  │
MUL2  ┘
```

(X)×(Y/32) がアキュムレータに演算結果としてストアされます。アキュムレータのMSBには"0"がストアされます。

例えば X=7FH (1111111)
　　　Y=17H (　10111)　とすると

```
                 1 1 1 1 1 1 1      (X)
Y0          ×        1 0 1 1 1      (Y)
            ---------------------
1             0 0 1 1 1 1 1 1 1     ---第1回目のMUL2結果
1         +     1 1 1 1 1 1 1
          ---------------------
            0 1 0 1 1 1 1 1 0       ---第2回目のMUL2結果
1       +     1 1 1 1 1 1 1
        ---------------------
          0 1 1 0 1 1 1 1 0         ---第3回目のMUL2結果
0
      ---------------------
      0 0 1 1 0 1 1 1 1             ---第4回目のMUL2結果 (Y0=0なのでAは1ビットライトシフトされる)
1   +     1 1 1 1 1 1 1
    ---------------------
    0 1 0 1 1 0 1 1 0               ---第5回目のMUL2結果
```

〰のデータは1ビットシフトされて切り捨てられる。

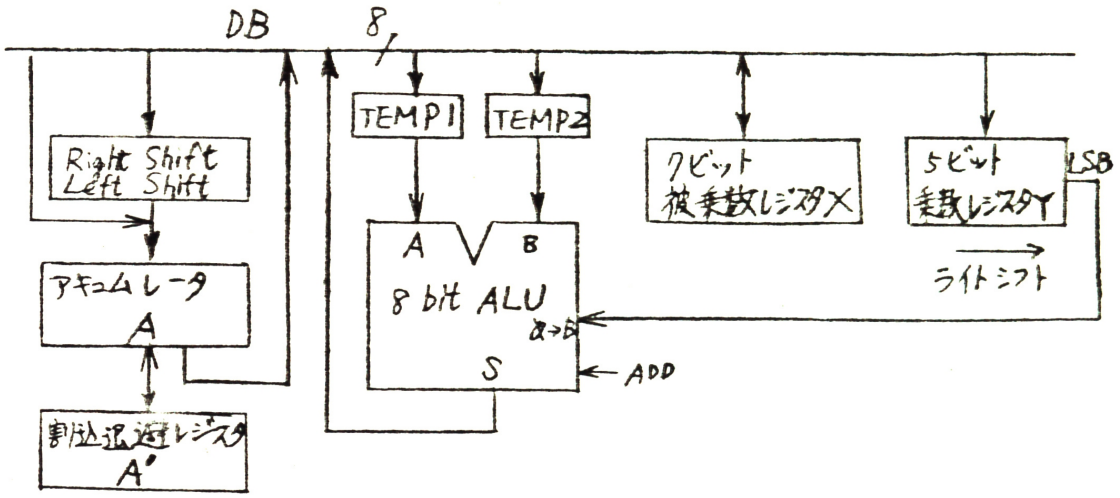
A = 5BH (01011011) になります。

X=7FH, Y=1FHの場合は A=XY=7BHになり乗算結果の最大値です。

通常Xレジスタにはトーン波形データ(素片データ)の絶対値, Yレジスタにはエンベロープデータをストアさせて MUL2命令をYレジスタのビット数だけ実行すれば トーン波形データがエンベロープデータでAM変調された結果を得る事ができます。

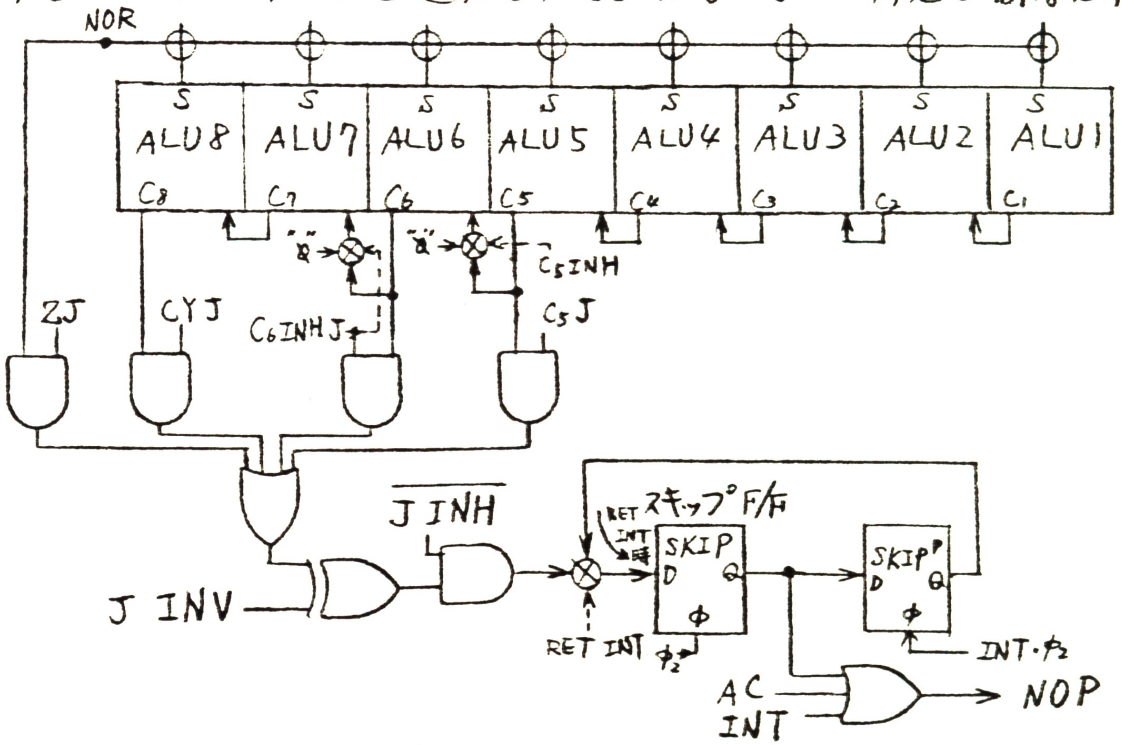
また擬似ノイズデータに対してもエンベロープデータと乗算する事によりAM変調された結果を得る事ができます。

○乗算用加算 ---- 乗数レジスタYのLSBが"0"ならアキュムレータを1ビットライトシフトして アキュムレータにストアし，乗数レジスタYも1ビットライトシフトする。
乗数レジスタYのLSBが"1"なら アキュムレータと被乗数レジスタXを加算して1ビットライトしてアキュムレータにストアし，乗数レジスタYも1ビットライトシフトする。
これをYレジスタのビット数だけ行なえば 乗算が完了したことになります。つまり最大5マシンサイクルで乗算を終了し 加算のものに比べて非常に高速です



○32進・64進モード切替えによる32/64進数加算とキャリー判定

モードレジスタに64/32進切替え用フリップフロップがあり，特殊命令により，そのフリップフロップによって6ビット目のキャリー出力C_6を上位へ上げるのを禁止するか，5ビット目のキャリー出力C_5を上位へ上げるのを禁止するかを選択し，C_5又はC_6の判定も同時に行ないます。



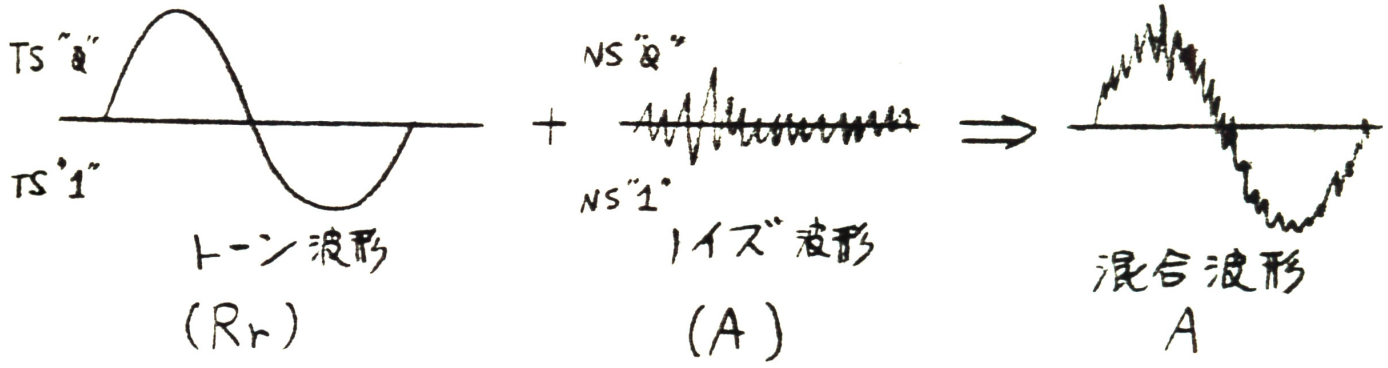
従って 9倍の速さで演算を実行できROM使用ワード数も 1/15 になります

3.7 混合命令

MIX Rr (Mix Tone Data with Noise Data)

A ← Rr ± A, SS ← TS ∀ (Bo or Ca) if TS = NS ADD
if TS ≠ NS SUB
(r = 0 ~ 1FH)

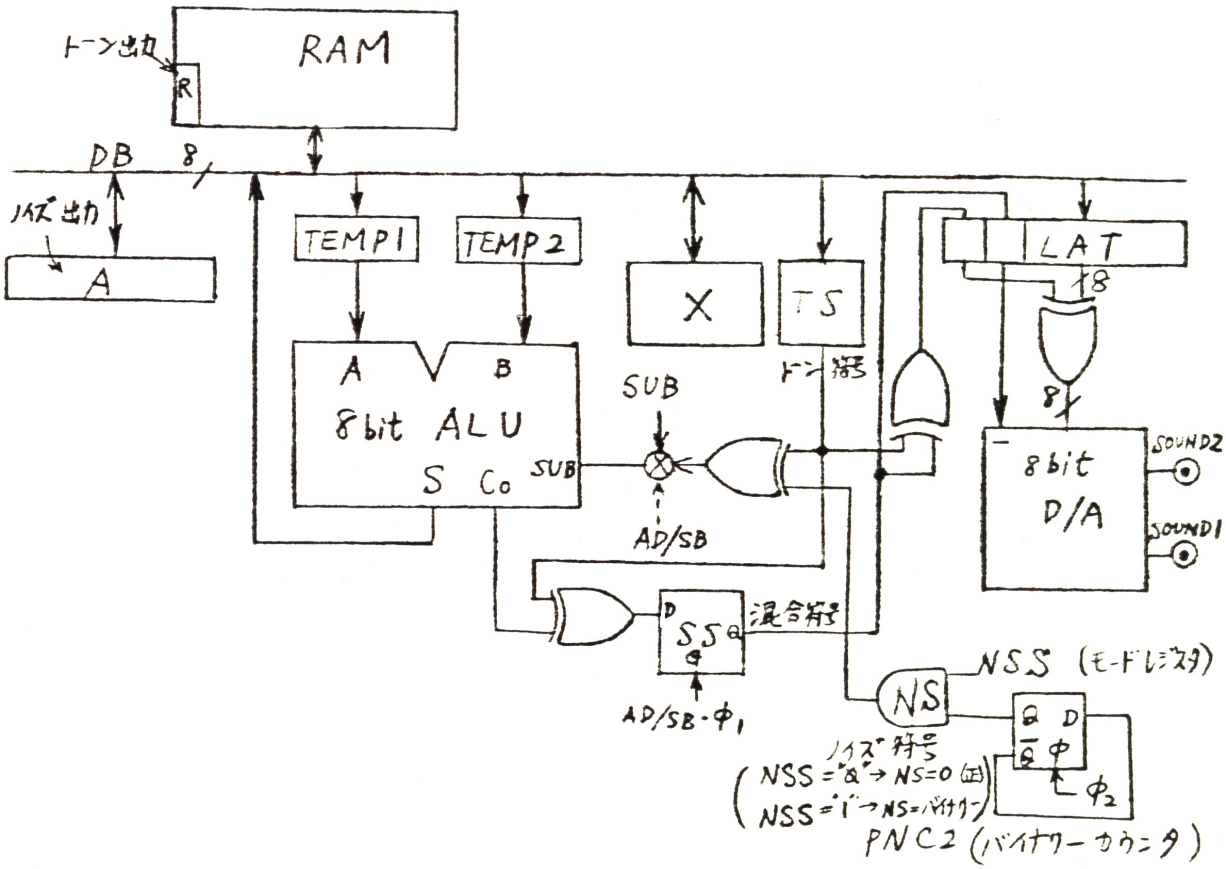
もし、トーン符号TSとノイズ符号NSが一致していれば イミーディエト・データr(0~1FH)で指定されたレジスタRrとアキュムレータを加算して アキュムレータにストアされ、トーン符号TSはサウンド符号SSにロードされます。
もし、トーン符号TSとノイズ符号NSが不一致であれば、レジスタRrからアキュムレータを減算し、その結果をアキュムレータにストアし、もし レジスタRrのデータがアキュムレータの値より小さければ ALUよりボローBoが発生し、トーン符号TSを反転してサウンド符号SSにロードし、もしレジスタRrのデータがアキュムレータのデータ以上の場合は、ボローBoが発生しないので、トーン符号TSがそのままサウンド符号SSにロードされます。レジスタRrにはトーン出力データ、アキュムレータにはノイズ出力データをあらかじめストアしておいて下さい。また加算した結果 キャリーがでないようなデータ構成(例えば Rr, A共にMSBが"0"の状態 つまり7ビットデータ)にて行なって下さい。もし 8ビット構成の場合はキャリーが出て SSにはTSの反転データが入ります。但し、rが0の時はその限りでない。
本命令のみがサウンド符号SSを決定する命令です。
本命令実行後にOUT DA実行を実行する場合は OUT DAで説明した動作になります。



トーン波形 (Rr) + ノイズ波形 (A) ⇒ 混合波形 A

○トーン・ノイズ混ぜ合わせ用加減算（混合命令）

トーン出力データとノイズ出力データを混合する場合に使用します。ノイズ出力データがアキュムレータにストアされており，トーン出力がワーキングレジスタ（RAMのR0～R1F）にストアされている場合にトーン±ノイズの演算を実行することができますのでトーン出力データとノイズ出力データとの混ぜ合わせをすることができます。



乗算の結果得られたトーン出力の7ビットがアキュムレータAにストアされており，その符号がトーン符号レジスタTSにストアされている。また乗算されたノイズ出力の7ビットはRAMのレジスタにストアされており，ノイズ符号はNSSによって決定され モードレジスタの1ビットのNSSが"0"であればノイズ符号は0で（正）になり，NSSが"1"であればノイズ符号は1マシンサイクル毎に0（正）→1（負）→0（正）→1（負）とバイナリーに変化します。

A←Rr±Aという混合命令が出ると，AD/SBが"1"になりTSとNSのEXOR出力がALUのSUB入力に入り，同符号であればEXOR出力が"0"となり加算，異符号ならばEXOR出力が"1"となり減算を実行します。そして加算の場合は符号が変化しませんのでサウンド符号レジスタSSにはトーン符号レジスタTSが入ります。Co出力はトーン（7ビット）＋ノイズ（7ビット）なので"0"になります。そしてトーン＋ノイズ出力がアキュムレータにストアされます。

また減算の場合にはトーン－ノイズとなりトーンがノイズより絶対値が小さければCoよりボローがでますのでTSが反転されてSSに入り符号が反転され アキュムレータには 補数がストアされます。また減算でトーンがノイズより絶対値が大きければCoよりボローは出ず TSがそのままSSに入り符号は反転されずアキュレータにも

トーンーノイズの減算結果がストアされます。

その後でOUT DAという命令を実行すると，アキュムレータからDAコンバータのラッチへデータが転送され，サウンドサイン（混合符号）SSよりD/Aコンバータの符号ラッチに転送され，TS出力とSS出力のEXOR出力が"1"であればALUのCoよりボローが出たということでアキュムレータが補数になっていることがわかりアキュムレータからD/Aコンバータへラッチされた後でEXORゲートにより"1"の補数"をとることによりレベルを正常にするようになっている。正確にD/Aコンバータに送るには"2の補数"をとらなければならないが8ビットD/Aコンバータの精度からみて1の補数でも全く問題がない。（1ビット加算うだけである。）
この一連の混合演算命令を従来のソフトのみで実行しようとすると10倍以上のマシンサイクル数が必要でリアルタイムに音声出力をたすのは困難であり，この混合命令は非常に高速でわずか1マシンサイクルで演算を処理してしまいます。また実行のためのROM使用ワード数も1/15位になります。

○キャリー　ボロー，演算結果が0であったり，その反対であったりすると，命令によりSKIPフリップフロップに"1"をセットし，次の命令をNOPにしてSKIPします。

○演算はデータメモリ（RAM）及びアキュムレータ（A）をソース及びディストネーション及びその逆の関係にして演算できますのでアキュムレータをソースにするだけのシステムに比較して，直接メモリーに対し演算ができますのでソフトウェアの軽減と，高速演算が可能となっています。

2.11.2　テンポラリー・ラッチ（TEMP1，TEMP2）

○TEMP1，TEMP2はそれぞれALUのA入力及びB入力に入力されるテンポラリーラッチで8ビット構成になっています。

○データバスのデータをALUに入力するラッチとして使用されます。

3.8 テーブル参照命令

○テーブル参照命令は割り込みプログラム中か，割り込みが絶対入らないプログラム中で実行するようにして下さい。

TBLA A,(Rr)　(Move Table Data to Accumulator)

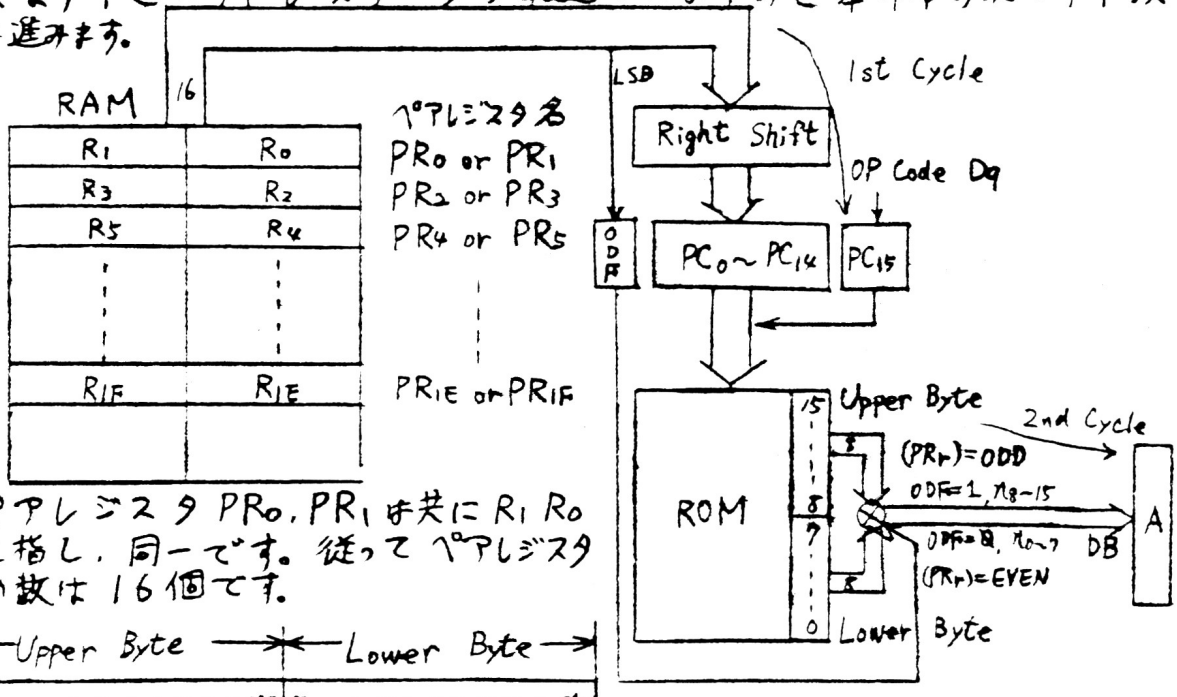
A ← (PRr)$_0$　(r = 0 ~ 1FH)

STK ← PC+1, PC$_{15}$ ← 0, PC$_{0\sim14}$ ← RS(PRr)　（1マシンサイクル）

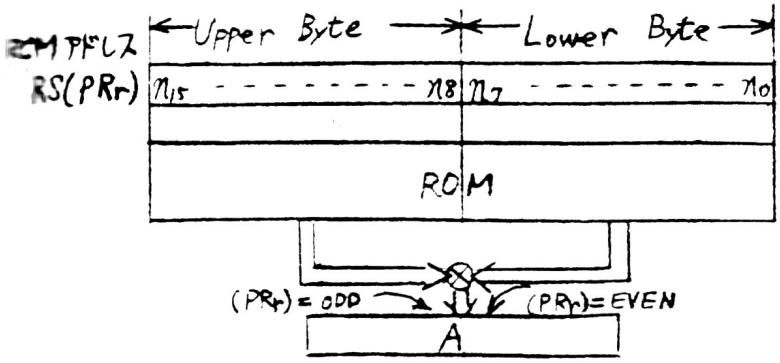
A ← $n_{0\sim7}$, if(PRr) = EVEN
A ← $n_{8\sim15}$ if(PRr) = ODD　PC ← STK　（2マシンサイクル）

データメモリ中にあるr(0~1FH)指定のペアレジスタPRrにストアされているアドレスデータで0000~7FFFH番地のプログラムデータメモリ(ROM)を指定し，その8ビット単位のテーブルデータをアキュムレータに転送するテーブル参照命令です。1マシンサイクル目においてPCはインクリメントされてSTKにストアされ，イミーディエト・データr(0~1FH)で指定されたペアレジスタPRrの16ビットのデータを1ビットライトシフトしてPC$_{0\sim14}$に転送しPC$_{15}$には0がストアされます。PRrの内容のLSBは奇数フリップフロップODFにストアされます。

2マシンサイクル目において，奇数フリップフロップODFが"0"つまり(PRr)が偶数ならテーブルデータ16ビットの下位8ビット$n_{0\sim7}$がアキュムレータに転送されます。奇数フリップフロップODFが"1"つまり(PRr)が奇数ならテーブルデータ16ビットの上位8ビット$n_{8\sim15}$がアキュムレータに転送されます。そしてSTKよりPCにアドレスデータが転送されますので本命令の次のアドレスに進みます。



(注) ペアレジスタPR0, PR1は共にR1 R0を指し，同一です。従ってペアレジスタの数は16個です。



TBL& X, (Rr)　(Move Table Data to X Register and TS Flip Flop)

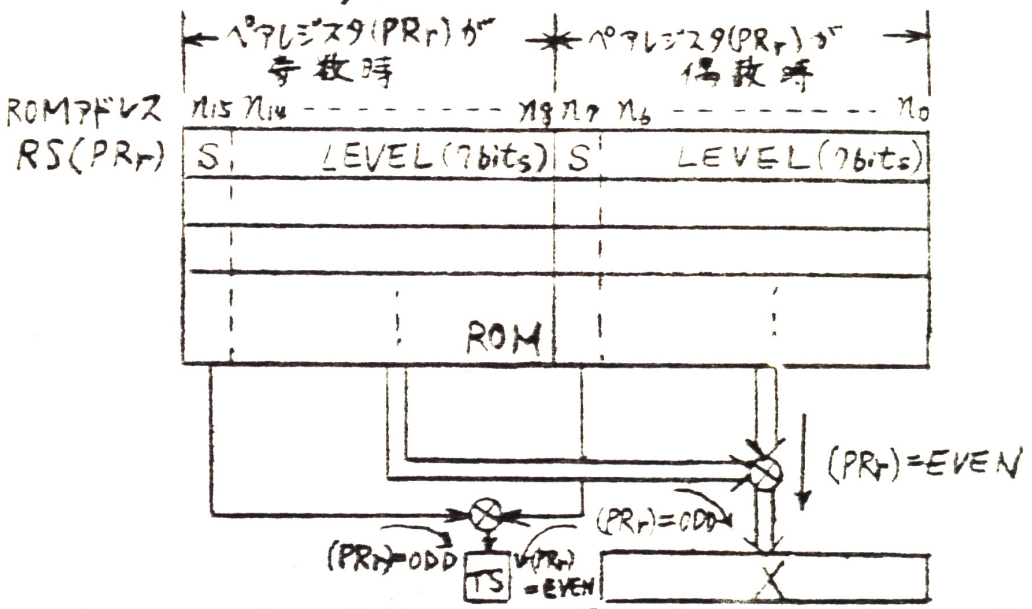
X ← (PRr)$_0$

(r = &~1FH)

STK ← PC+1, PC$_{15}$ ← &
PC$_{0\sim14}$ ← RS(PRr)

X ← n$_{0\sim6}$, TS ← n$_7$, if (PRr) = EVEN, PC ← STK
X ← n$_{8\sim14}$, TS ← n$_{15}$, if (PRr) = ODD

r(&~1FH)指定のペアレジスタPRrにストアされているアドレスデータにより指定されたテーブルデータをXレジスタおよびトーン符号フリップフロップTSに転送するテーブル参照命令で、&&&&~7FFF番地にある8ビット単位のトーン波形(素片)データの符号をTSに、その絶対値7ビットをXレジスタに転送します。TBL& A,(PRr)命令と同様な動作によりテーブルデータ転送先がアキュムレータの代りに



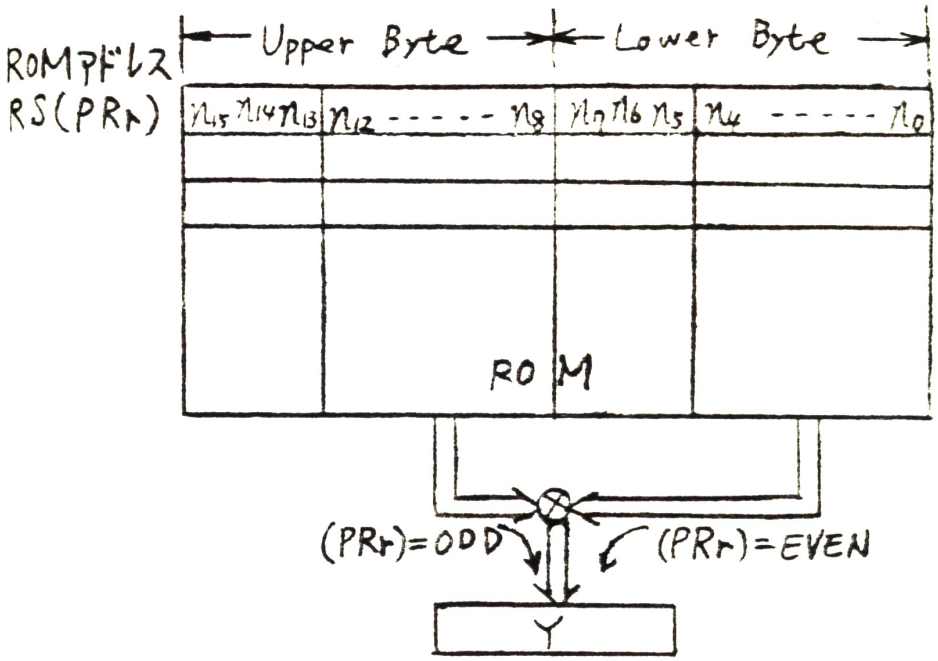
Xレジスタと TS フリップフロップとなり、8ビットデータのうち下位7ビットがXレジスタに、MSBがTSフリップフロップに転送されます。トーン波形データ(素片データ)をテーブル参照する場合にこの命令を使用します。

TBL& Y, (Rr)　(Move Table Data to Y Register)

Y ← (PRr)$_0$
(r = &~1FH)

PC$_{15}$ ← &
PC$_{0\sim14}$ ← RS(PRr)
Y ← n$_{0\sim4}$　if (PRr) = EVEN
Y ← n$_{8\sim12}$　if (PRr) = ODD

r(&~1FH)指定のペアレジスタPRrにストアされているアドレスデータにより指定されるテーブルデータをYレジスタに転送するテーブル参照命令で、&&&&~7FFF番地にある8ビット単位のデータのうち下位5ビットのデータをYレジスタに転送します。TBL& A,(PRr)命令と同様な動作によりテーブルデータ転送先がアキュムレータの代りに5ビットのYレジスタとなります。エンベロープデータの絶対値をテーブル参照する場合にこの命令を使用します。



TBL1 A,(Rr) (Move Table Data to Accumulator)

A ← (PRr)1 (r=0~1FH)

STK←PC+1, PC15 ← 1, PC0~14 ← RS(PRr)

A ← n0~7, if (PRr)=EVEN ; PC←STK

A ← n8~15, if (PRr)=ODD ; PC←STK

r(0~1FH)指定のペアレジスタPRrにストアされているアドレスデータで指定されるテーブルデータをアキュムレータに転送するテーブル参照命令で、800H~FFFFH番地にある8ビット単位のテーブルデータをアキュムレータに転送します。

TBL0 A,(Rr)命令と異なる点は800H~FFFFH番地のテーブルデータを参照する点だけです。1chip型ではPC12~PC15はありませんのでTBL0 A,(Rr)命令と同一の動作をします。

TBL1 X,(Rr) (Move Table Data to X Register)

X ← (PRr)1 (r=0~1FH)

STK←PC+1, PC15 ← 1, PC0~14 ← RS(PRr)

X←n0~n6, TS←n7, if (PRr)=EVEN, PC←STK

X←n8~n12, TS←n15, if (PRr)=ODD, PC←STK

r(0~1FH)指定のペアレジスタPRrにストアされているアドレスデータで指定されるテーブルデータをXレジスタ及びトーン符号フリップフロップTSに転送するテーブル参照命令で800H~FFFFH番地にあるテーブルデータを転送します。

TBL0 X,(Rr)命令と異なる点は800~FFFFH番地のデータを参照する点だけです。1chip型ではPC12~PC15はありませんのでTBL0 X,(Rr)命令と同一の動作をします。

TBL1 Y, (Rr)　(Move Table Data to Y Register)

Y ← (PRr)　STK ← PC+1, PC15 ← 1, PC0~14 ← RS(PRr)

(r = 0~1FH)　Y ← n0~4, if (PRr) = EVEN

Y ← n8~12, if (PRr) = ODD

r(0~1FH)指定のページアドレスレジスタ PRrにストアされているアドレスデータで指定されるテーブルデータをYレジスタに転送するテーブル参照命令で，8000H~FFFFH番地にある8ビット単位のデータのうち下位5ビットのデータをYレジスタに転送します。
TBL0 Y, (PRr) 命令と異なる点は 8000H~FFFFH番地のデータを参照する点だけです。1Chip型では PC12~PC15はありませんので TBL0 Y, (Rr) 命令と同一な動作をします。

3.9 ジャンプ命令

JMP n　(Direct Jump Within 4K Words Block)

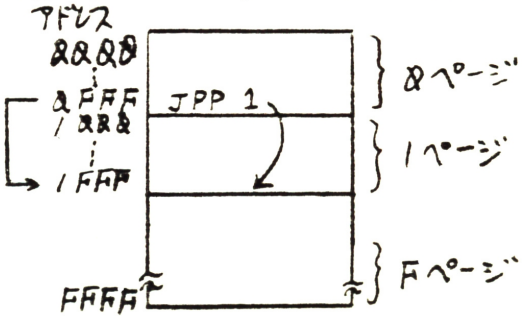
PC0~11 ← n　(n = 000~FFFH)

プログラムカウンタの下位12ビットに直接指定アドレスn(000~FFFH)をロードし，4K Words (1ページ)内をダイレクトジャンプします。PC12~PC15は変わりません。

ページジャンプ

JPP n　(PC12~15 ← n)　(n = 0~F)

ページジャンプは プログラムカウンタ上位4ビットのPC12~PC15だけにページアドレスデータ n (0~F)をロードして，4K Words ブロック外，つまり他のページにジャンプします。アセンブラにおいてはページジャンプ命令はありませんが，マニュアル操作にてアセンブラを使用しない場合は本命令を使用できます。ROMの64K Words全領域へのページ変更だけを行ないます。PC0~11は変わりませんのでページだけが変わることになります。PC0~11はインクリメントされません。

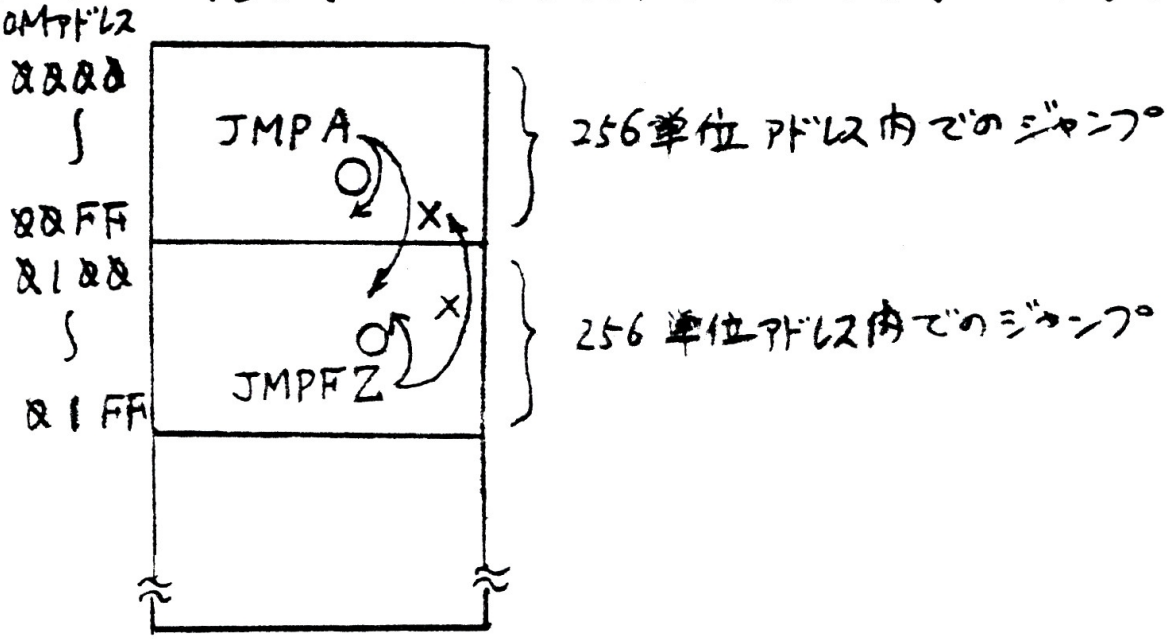


例えば 0FFFH番地で JPP 1 命令を実行すると 1FFFH番地にページジャンプします。

JMPA　　(Indirect Jump by Accumulator Contents)

PC0~7 ← A

アキュムレータの内容をプログラムカウンタ PC0~7 にロードしますので，アキュムレータの内容が指定する番地にジャンプします。
プログラムカウンタ PC8~15 は変化しません。
256単位毎のアドレス内でのジャンプになります。
JMPA 命令を実行する下位8ビットのアドレスとアキュムレータの内容が同一の場合はプログラムカウンタは止まったままになります。



JMPFZ n　(Jump if FL0 is Zero and Reset FL0)

if FL0 = 0 , PC0~7 ← n , FL0 ← 0　(n=0~FFH)
if FL0 = 1 , PC ← PC+1 ,

もし，フラグフリップフロップ FL0 が0であればプログラムカウンタの下位 PC0~7 に直接指定アドレスデータ（0~FFH）をロードし，256単位アドレス内でジャンプします。上位の PC8~15 はそのままです。
もし，フラグフリップフロップ FL0 が1であればプログラムカウンタをインクリメントします。
フラグフリップフロップをリセットします。

3.10 コール命令

CALL n　　(Call Subroutine)

(SP) ← PC+1.
SP ← SP+1.
$PC_{0\sim11}$ ← n,　$PC_{12\sim15}$ ← α　　(n = 000～FFFH)

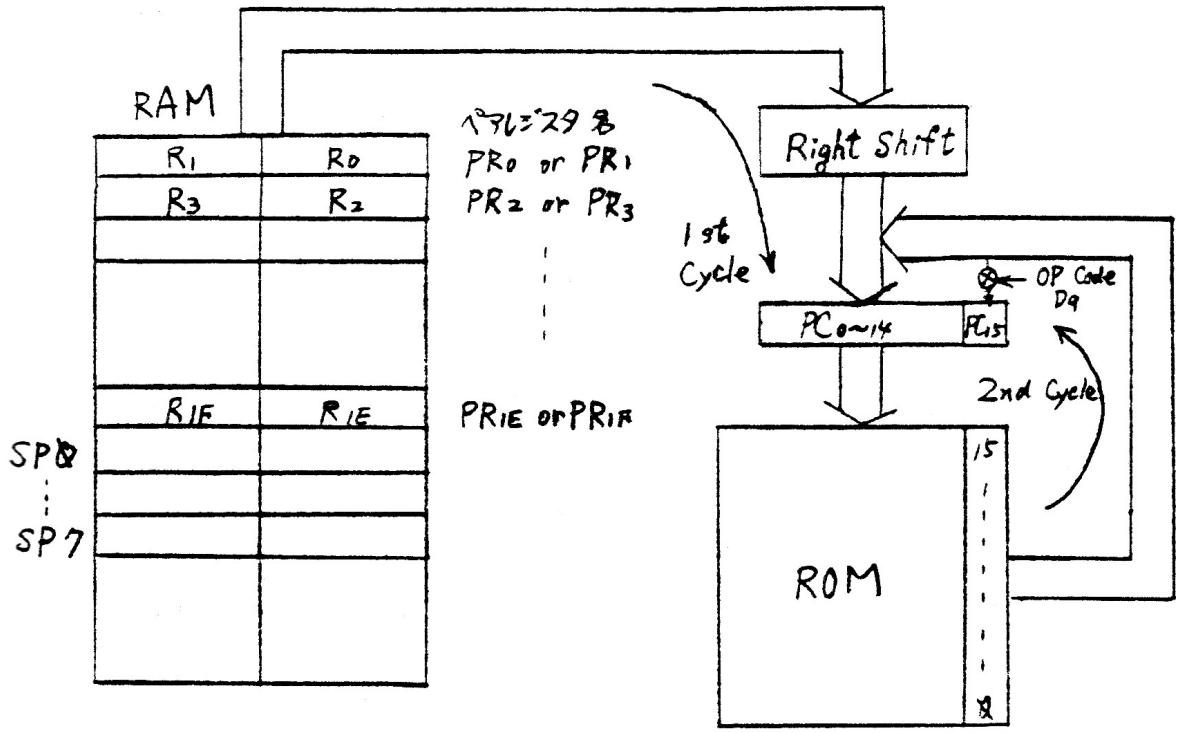
プログラムカウンタの内容をインクリメントしたデータをスタックポインタSPで指定したデータメモリ中のスタックに退避させます。スタックポインタをインクリメントします。
プログラムカウンタ $PC_{0\sim11}$ にn(000～FFFH)をロードし，$PC_{12\sim15}$ にαをロードしますので αページ内にあるサブルーチンをコールします。
従って CALL n 命令を使用してコールされるサブルーチンは必らず αページ(4k Words)内，つまり α000～αFFFH番地に置いて下さい。
1chip 型のものは αページ以内ですので，その全アドレスに対してサブルーチンコールができます。

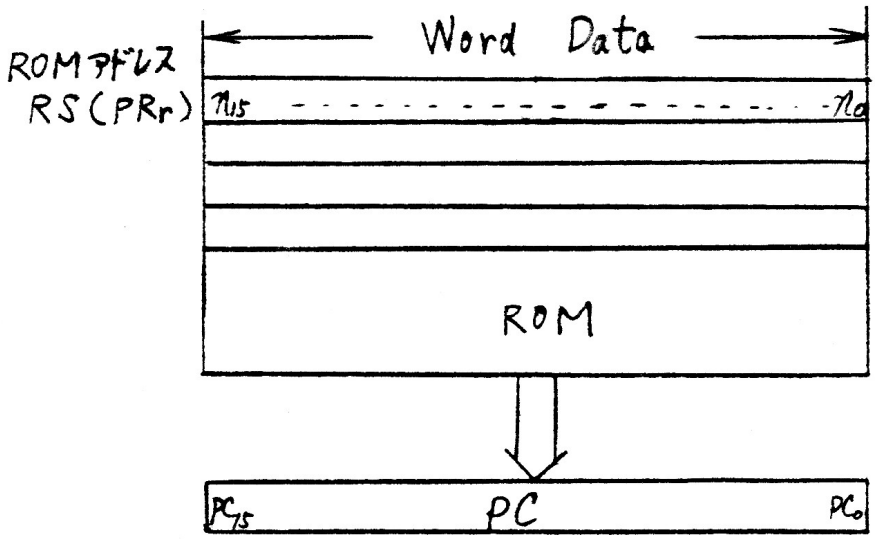
CALLα (Rr)　(Indirect Call Subroutine by Table Data)

(r=0～1FH)
1st マシンサイクル: STK ← PC+1, PC_{15} ← α ; $PC_{0\sim14}$ ← RS(PRr)
2nd マシンサイクル: (SP) ← STK ; SP ← SP+1 ; PC ← (PC)

r(0～1FH)指定のペアレジスタ PRrにストアされているアドレスデータで，0000～7FFFH番地のプログラムデータメモリ(ROM)を指定し，その16ビット単位のテーブルデータを先頭番地とするサブルーチンをコールします。
1マシンサイクル目にプログラムカウンタの内容をインクリメントしてSTKレジスタにストアし プログラムカウンタ $PC_{0\sim14}$ にr(0～1FH)指定のペアレジスタの内容を1ビットライトシフトしてロードし，PC_{15} にαをロードします。
2マシンサイクル目にSTKレジスタの内容 つまり復帰アドレスをスタックポインタSPで指定されたデータメモリ中のスタックに退避させます。そしてスタックポインタSPをインクリメントします。1マシンサイクル目でロードされたプログラムカウンタでアドレス指定されたプログラム・データメモリ(ROM)の内容つまりテーブルデータ(PC)(0000～FFFFH)をプログラムカウンタPCにロードして，そのテーブルデータを先頭番地とするサブルーチンにジャンプします。
リターン命令のRET命令又はRETS命令によりコールされたサブルーチンから復帰して CALLα (Rr) 命令の次のアドレスに進みます。

(注) 本命令は割り込みプログラム中か，割り込みが絶対に入ってこないプログラム中で実行するようにして下さい。





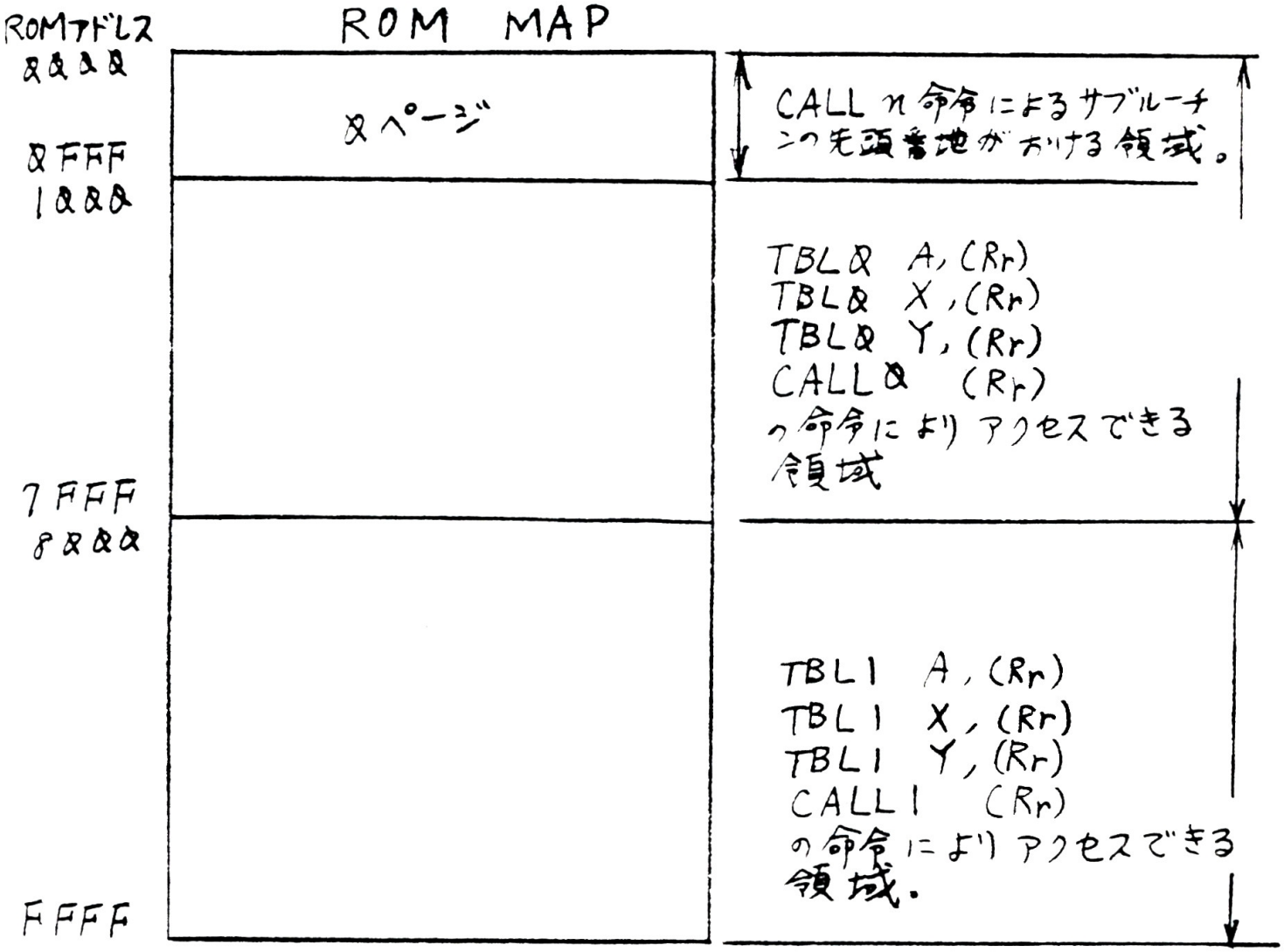
CALL1 (Rr) (Indirect Call Subroutine by Table Data)

(r=0~1FH)

STK ← PC + 1, PC15 ← 1
PC0~14 ← RS(PRr)
(SP) ← STK
SP ← SP + 1
PC ← (PC)

r(0~1FH)指定のペアレジスタ PRrにストアされているアドレスデータで 8000~FFFF番地のプログラム・データメモリ(ROM)を指定し、その16ビット単位のテーブルデータを先頭番地とするサブルーチンをコールします。
CALL 0 (Rr) 命令と異なる点は 8000~FFFF番地のROMのテーブルデータを参照してサブルーチンをコールする点だけです。

(注) CALL 1 (Rr) 命令も CALL 0 (Rr) 命令と同様に割り込みプログラム中か，絶対に割り込みが入ってこないプログラム中で実行して下さい。



3.12 その他の命令

STF　　(Set Flag α)

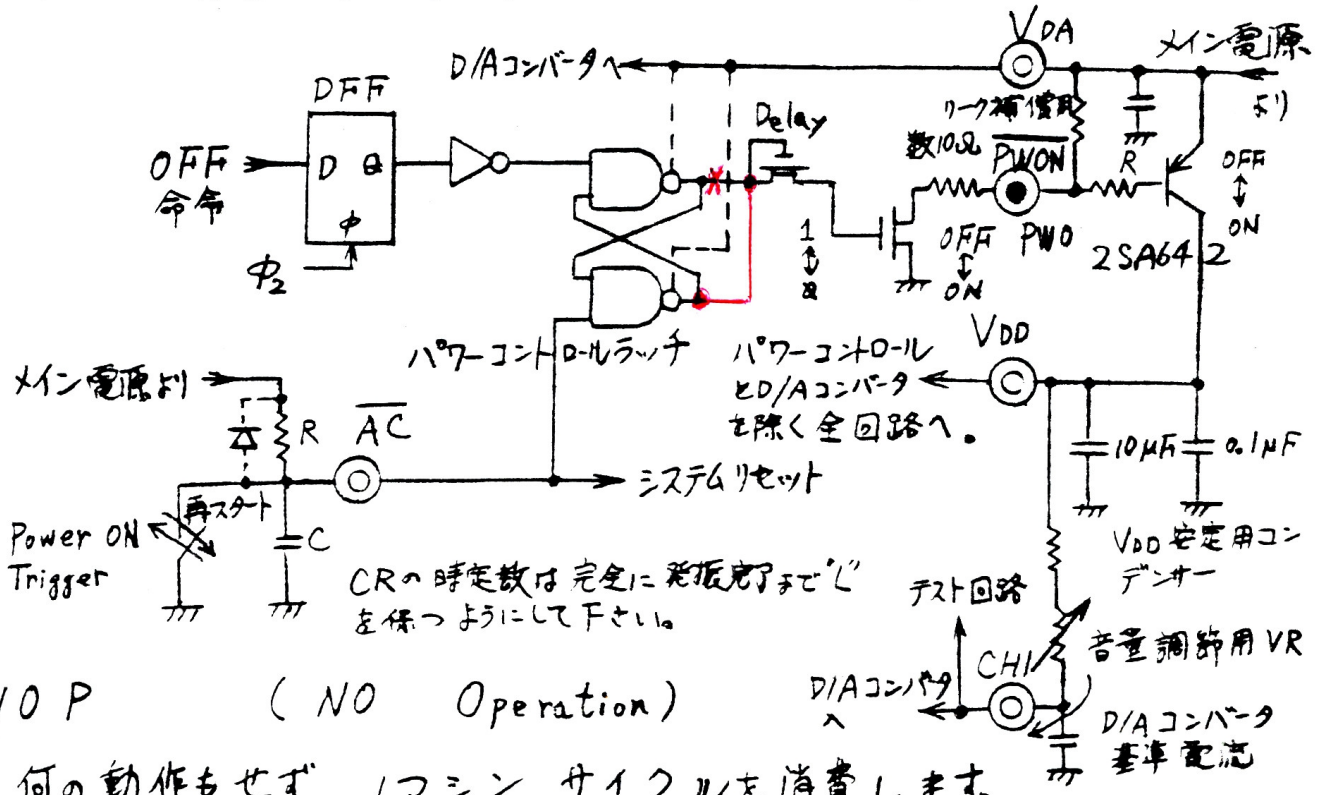
FLα ← 1

フラグαのフリップフロップFLαをセットします。
FLαのジャッジとリセットはJMPF2命令にて行ないます。

OFF　　(Power OFF)

$\overline{PWON}$ ← 1

オープンドレイン構造の $\overline{PWON}$ 端子をオフにして外付のPNPトランジスタをOFFに，電源端子の V_{DD} 端子に電流を供給させないように働らき V_{DD} 電源が切れてPower OFFモードになります。
$\overline{PWON}$ 端子がオフになるには本命令実行後のマシンサイクルからとなります。
Power ONモードにするには $\overline{AC}$ 端子を"L"にすると常に V_{DA} 電源が供給されているパワーコントロールラッチが反転し，$\overline{PWON}$ がオンになり，外付のPNPトランジスタがONし，V_{DD} 電源に電流を供給し，発振回路が発振し，プログラムカウタがαなり，システムがリセットされます。完全に発振した後に，$\overline{AC}$ 端子を"H"にするか，CRの積分回路にてHになるとα番地よりプログラムがスタートします。



NOP　　(NO Operation)

何の動作もせず 1マシン サイクルを消費します。

3.13 算術論理演算命令

算術論理演算命令には主に次のようなものがあります

演算の種類			スキップ条件	ニーモニック イミーディエート・データとの演算	ニーモニック レジスタ間演算
演算	加減算	キャリー判定無し2進加算		ADI	AD
		キャリー判定有り2進加算	Ca	ADIS	ADS
		ボロー判定無し2進減算		SBI	SB
		ボロー判定有り2進減算	Bo	SBIS	SBS
	論理積	ビットテスト無し論理積		ANDI	AND
		ビットテスト有り論理積	Z	ANDIS	ANDS
	論理和			ORI	OR
	排他的論理和			EORI	EOR
テスト	キャリーテスト (加算)		Ca	TADIC	TADC
	ボローテスト (減算)		Bo	TSBIC	TSBC
	ゼロテスト (論理積)		Z	TANDIZ	TANDZ
	イコールテスト (減算)		=	TSBIZ	TSBZ
	ノットキャリーテスト(加算)		$\overline{Ca}$	TADINC	TADNC
	ノットボローテスト (減算)		$\overline{Bo}$	TSBINC	TSBNC
	ノットゼロテスト(論理積)		Z	TANDINZ	TANDNZ
	ノットイコールテスト(減算)		≠	TSBINZ	TSBNZ

3.13-1表

演算 --- Distnation ← Distnation * Source (No Skip / Next Skip or Not)

テスト --- Distnation * Source (Next Skip or Not)

上の表の16種類があり、それぞれ Distnation, Source は次の組合せです。

	Distnation (1st Operand)	Source (2nd Operand)	実行できる算術論理演算命令
イミーディエート・データnとの演算	アキュムレータ A	n (00~FF)	3.13-1表の全命令
	データメモリ (H)	n (00~FF)	〃
	データポインタ H	n (00~3F)	〃
	モードレジスタ MDI	n (0~7)	演算のみ (ADI~EORI)
	〃 MD	n (00~FF)	テストのみ (TADIC~TSBINZ)
	レジスタ Rr	n (0~F)	加算(ADI,ADIS),減算(SBI,SBIS) テスト(TADIC,TSBIC,TADINC, TSBINC) 特殊加算命令 (ADI5,ADIMS) 特殊テスト命令 (TADI5) イミーディエート・データnは4ビットですがそれを下位にして,上位4ビットを0にしたnHをソースデータとして8ビット演算を行なう。
レジスタ間の演算	アキュムレータ A	レジスタ Rr	3.13-1表の全命令
	レジスタ Rr	アキュムレータ A	〃
	アキュムレータ A	データメモリ (H)	〃
	データメモリ (H)	アキュムレータ A	〃

3.13.2 イミーディエト・データ 特殊算術演算命令

ADI5 Rr, n (ADD Immediate Data to Lower 5 bits of Register)

$Rr_{0\sim4} \leftarrow Rr_{0\sim4} + n$
$Rr_{5\sim7} \leftarrow Rr_{5\sim7}$

(r = 0~1FH, n = 0~F)

イミーディエト・データ r (0~1FH) 指定の レジスタ Rr の下位5ビットの内容 $Rr_{0\sim4}$ に イミーディエト・データ n (0~F) を加算して、結果を $Rr_{0\sim4}$ に入れます。5ビット目のキャリー C_5 を禁止して加算されますので レジスタの上位3ビット $Rr_{5\sim7}$ の内容は変りません。

ADIMS Rr, n (ADD Immediate Data to Lower 5 bits or 6 bits of Register, Skip if Carry C_5 or C_6)

If MD_0 (64/32) = 0, ($Rr_{0\sim4} \leftarrow Rr_{0\sim4} + n$, Skip if 5 th Carry C_5; $Rr_{5\sim7} \leftarrow Rr_{5\sim7}$)

If MD_0 (64/32) = 1, ($Rr_{0\sim5} \leftarrow Rr_{0\sim5} + n$, Skip if 6 th Carry C_6; $Rr_{6\sim7} \leftarrow Rr_{6\sim7}$)

(r = 0~1FH, n = 0~F)

もし モードレジスタのうちの MD_0 (64/32) が "0" ならば、イミーディエト・データ r (0~1FH) 指定の レジスタ Rr の下位5ビットの内容 $Rr_{0\sim4}$ にイミーディエト・データ n (0~F) を 2進加算し、$Rr_{0\sim4}$ に入れます。5ビット目のキャリーは禁止して加算されるので、レジスタ Rr の上位3ビット $Rr_{5\sim7}$ は変りません。そして 5ビット目のキャリー C_5 があれば スキップします。
もし モードレジスタ MD_0 (64/32) が "1" ならば、レジスタ Rr の下位6ビットの内容 $Rr_{0\sim5}$ に イミーディエト・データ n (0~F) 2進加算し、$Rr_{0\sim5}$ に 入れます。6ビット目のキャリーは 禁止されて加算されるので、レジスタ Rr の 上位2ビット $Rr_{6\sim7}$ は変りません。そして6ビット 目のキャリー C_6 が あれば スキップします。
素片(トーン)波形アドレス歩進(+1,+2,+4,+8)に使用し、32分割か64分割かにより進数変換します。

TADI5 Rr, n (Test ADD Immediate to Lower 5 bits of Register. Skip if Carry C_5)

$Rr + n$, Skip if 5 th Carry.

(r = 0~1FH, n = 0~F)

イミーディエト・データ r (0~1FH) 指定の レジスタ Rr に イミーディエトデータ n (0~F) を 試し 加算し もし 5ビット目のキャリー C_5 があればスキップします。