YOHPAC-2100 A におけるマイクロプログラミング技術の実際

馬　目　洋　一†

1. アセンブラ言語とマイクロプログラム言語

表1はコンパイラ言語による足し算で，これがアセンブラ言語とマイクロプログラミング言語では，どのように対応するかを表わしている．

アセンブラ言語例は，X番地の内容にY番地の内容を加算して，これをZ番地に格納することを示す．

YOHPAC-2100 A には，アキュムレータが2個あり，それぞれAレジスタ，Bレジスタと呼んでおり，表1の例では，このAレジスタに関するものを取り上げた。

表1

コンパイラ言語	アセンブラ言語	マイクロプログラミング言語						
Z=X+Y	LDA X	(0) NOP	P	CFLG	M	RW	NOP
		(1) AAB	COND	IOR	IR	NOP	EOP
		(2) NOP	ADR	IOR	S1	AAB	NOP
	ADA Y	(164_8)NOP	S1	IOR	M	RW	NOP
		(165_8)AAB	COND	IOR	S2	NOP	EOP
		(166_8)NOP	S2	IOR	CAB	NOP	NOP
	STA Z	(0) NOP	P	CFLG	M	RW	NOP
		(1) AAB	COND	IOR	IR	NOP	EOP
		(2) NOP	ADR	IOR	S1	AAB	NOP
		(144_8)NOP	S1	IOR	M	RW	NOP
		(145_8)AAB	COND	IOR	S2	NOP	EOP
		(146_8)CAB	S2	ADDO	CAB	NOP	NOP
		(0) NOP	P	CFLG	M	RW	NOP
		(1) AAB	COND	IOR	IR	NOP	EOP
		(2) NOP	ADR	IOR	S1	AAB	NOP
		(134_8)F	S1	DEC	M	CW	NMPV
		(135_8)CAB	RRS	IOR	AAB	NOP	UNC
		(136_8)CAB	RRS	IOR	T	NOP	EOP
		(137_8)NOP	NOP	IOR	NOP	NOP	NOP

アセンブラ言語例は，オペランドアドレスをMとしたとき，次のような意味を持つ．

LDA M (Load(M) into A)

ADA M (Add (M) to A)

STA M (Store (A) to M)

マイクロプログラミング言語例は，表1に示すように，それぞれアセンブラ言語例に対応している．

アセンブラ言語例が3ステップから成っていることに対応して，マイクロプログラミング言語例も3群からなり，これらをさらにフェッチフェーズとエクスキュートフェーズを実行するものに分けると6グループになる．

このアセンブラ言語例において，具体的な数値としてX番地に1，Y番地に2が入っているものとする．

1と2が加算される状態をよく理解するためにLDA X)は，すでに実行されているものとして(ADA Y) を中心にして説明していくことにする．

† 横河・ヒューレット・パッカード(株)システム課

2. YOHPAC-2100 A のブロック図

YOHPAC-2100 A のブロック図を図1に示す．情報は通常1語長並列16ビットで処理される．

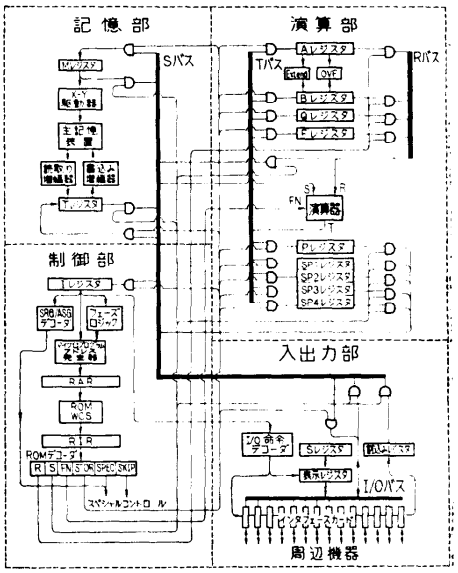


図1　YOHPAC 2100 A のブロック

図1の記憶部と演算部をフェッチフェーズ（図2）とエクスキュートフェーズ（図3）に分けて示す．制御部はその働きを理解するためにIレジスタと制御順序の表だけを示してある．図2および図3における各レジスタは次のように用いられている．

Mレジスタ：メモリアドレス　レジスタ

Tレジスタ：メモリデータ　レジスタ
Iレジスタ：インストラクション　レジスタ
Aレジスタ：アキュムレータ
Pレジスタ：プログラム　カウンタ
SP1レジスタ：ユーティリティ　レジスタ
SP2レジスタ：ユーティリティ　レジスタ

また，同図における太線は16ビットを伝送するバスを表わし，それぞれRバス，Sバス，Tバスと呼ぶ．

図から分るように各バス間の基本的な関係は，Rバスに乗った情報とSバスに乗った情報が演算器 ALU によって演算された後，Tバスに乗るように構成されている．

各バスと各レジスタの間を形式的に AND ゲートで結合しているが，これは原則として，このゲートが ON になったとき，16ビットの情報がレジスタからバスへ，あるいはバスからレジスタへ移動するものと約束する．

実際の各バス間の関係では，その他にRバスからSバスへ直接，情報を伝送したり，Sバスと I/O バスとの間で情報のやりとりをすることなどが自由にできる．

ブロック図の下にある制御部順序表の R, S, F, S_t, S_p, S_k の呼称と代表的な働きを次に示す．

R (Rバスフィールド)：指定したレジスタの内容をRバスへ乗せる．

S (Sバスフィールド)：指定したレジスタの内容をSバスへ乗せる．

F (ファンクションフィールド)：指定した算術または論理演算を行なう．

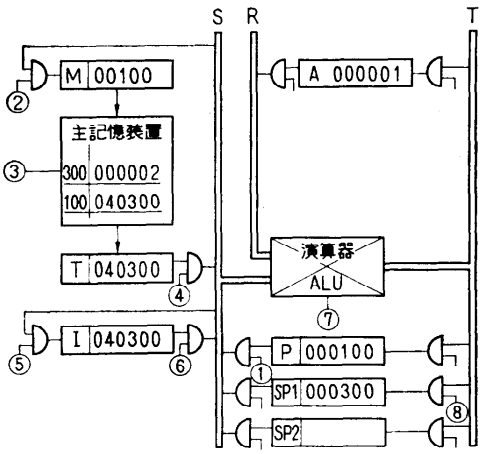
S_t (ストアフィールド)：指定したレジスタへ対応するバスの内容を入れる．

S_p (スペシャルフィールド)：メモリサイクルを起したりマイクロジャンプ命令の上位ターゲットアドレスやSバスに乗せる定数を定めたり様々な特徴をもつ．

S_k (スキップフィールド)：色々な条件を判定したり，マイクロジャンプ命令の下位ターゲットアドレスやSバスに乗せる定数を定めたりなどする．

3. フェッチフェーズの概略

情報の16ビットを下位から3ビットずつ区切って8進数として読むことにして，(ADA Y) を次のようなマシンコードで表わすことにする．



制御部順序表

フェッチフェーズ

R	S	F	S_t	S_p	S_k
	(1)		(2)	(3)	
	(4)		(5)		*
	(6)	(7)	(8)		

図2　フェッチフェーズ・ブロック図

ADA Y⟶040300_8
(ADA⟶04_8, Y⟶0300_8)

そしてこの加算命令自体が100_8番地に入っているものとして図2のフェッチフェーズを順に追っていくことにする．

前述のように (LDA X) は，すでに実行されたものとして考えるので，Aレジスタには000001_8が入り，Pレジスタの内容は1つ進み現在100_8を示しているものとする．この状態からフェッチフェーズは制御部順序表に従って，次のように行なわれる．

① Pレジスタの内容100_8をSバスに乗せる．
② Sバス上の情報100_8をMレジスタへ入れる．
③ メモリサイクルを起こさせ100_8番地の内容を読み出させる．
④ Tレジスタに読み出された情報040300_8をSバスに乗せる．
⑤ Sバス上の情報040300_8をIレジスタへ入れる．
⑥ Iレジスタの内容のうちオペランドアドレスに相当する300_8をSバスに乗せる．

⑦ 演算器にRバスとSバスの論理和演算を行なわせる．この時，Rバス上の情報は全て0なのでTバスに，そのままSバスの情報 300_8 が乗る．

⑧ Tバス上の情報 300_8 を SP 1 レジスタへ入れる．

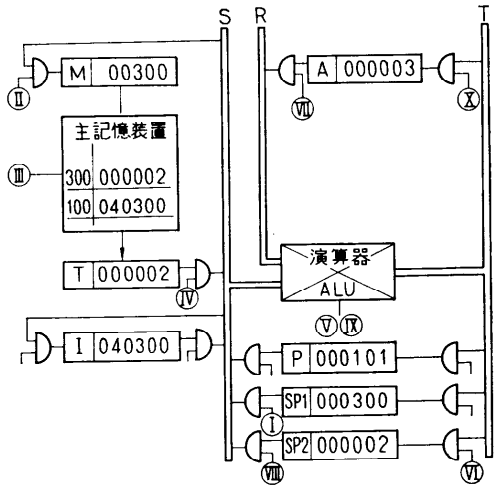
4. エクスキュートフェーズの概略

図3を参照しながら順に見ていくことにする．

フェッチフェーズによってIレジスタに入った情報 040300_8 は加算命令であると解読され，ブロック図の下に示すエクスキュートフェーズの順序表を選択され次のように実行される．

(I) SP1レジスタの内容 300_8 をSバスに乗せる．

(II) Sバス上の情報 300_8 をMレジスタへ入れる．



制御部順序表

エクスキュートフェーズ

R	S	F	St	Sp	Sk
	(I)		(II)	(III)	
	(IV)	(V)	(VI)		*
(VII)	(VIII)	(IX)	(X)		

図3 エクスキュートフェーズ・ブロック図

(III) メモリサイクルを起こさせ 300_8 番地の内容を読み出させる．

(IV) Tレジスタに読み出された情報 000002_8 をSバスに乗せる．

(V) 演算器にRバスとSバスの論理和演算を行なわせ，TバスにSバス上の情報 000002_8 を乗せる．

(VI) Tバス上の情報 000002_8 を SP 2 レジスタへ入れる．

(VII) Aレジスタの内容 000001_8 をRバスに乗せる．

(VIII) SP 2 レジスタの内容 000002_8 をSバスに乗せる．

(IX) 演算器にRバス上の情報とSバス上の情報との加算を行なわせる．その結果，Tバスに情報 000003_8 が乗る．

(X) Tバス上の情報 000003_8 をA レジスタへ入れる．

以上で前節と合せてフェッチフェーズとエクスキュートフェーズを完了する．

制御部順序表の内容は，そのままマイクロプログラムに相当するものである．

実際のマイクロプログラムでは，フェーズを終了させる EOP (End of Phase) マイクロ命令をスキップフィールドに含んでいる．(図2と図3における＊印)

5. マイクロプロセッサ

前節までは，マイクロプロセッサの働きを仮定した上で，制御部順序表を用いてフェッチフェーズとエクスキュートフェーズを説明したので，ここでは，前節のエクスキュートフェーズを例にとって，マイクロプロセッサの働きの概略を述べることにする．

マイクロプロセッサのブロック図を図4に示す．

図はフェッチフェーズが終了してエクスキュートフェーズが開始される状態を示す．

この時，Iレジスタにはフェッチフェーズが終了した事により情報 040300_8 が入っており，この情報は，マイクロプログラムアドレス発生器により，加算命令であると判断されている．

YOHPAC-2100 A の基本命令のマイクロルーチンを全て納めてあるリードオンリメモリ (ROM) では，144_8 番地が加算ルーチンの先頭番地となっているのでROM アドレスレジスタ (RAR) には，マイクロプログラムアドレス発生器より情報 144_8 が与えられている．

説明に先立って，マイクロ命令とマイクロインストラクションの意味を約束しておく．

Rバスフィールドからスキップフィールドまでの各フィールドに該当する個々の命令をマイクロ命令と呼

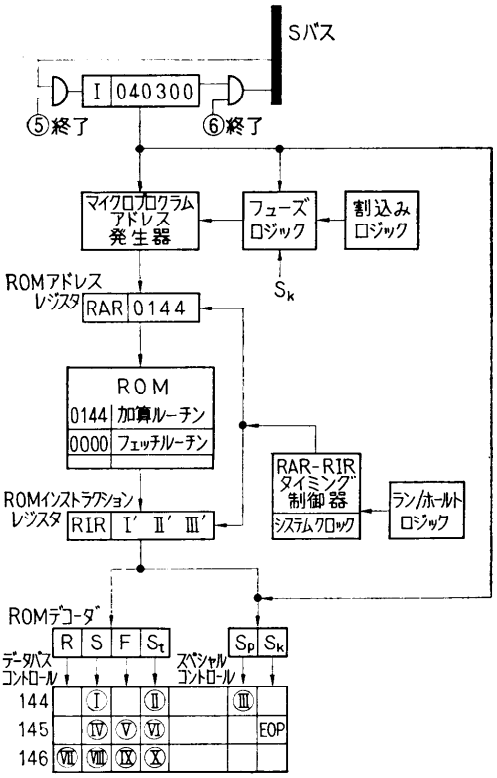


図4 マイクロプロセッサ・ブロック図

ぶことにする．これは正しくは図4におけるROMインストラクションレジスタ (RIR) の内容の I′, II′, III′, などに相当するが説明の便宜上，同図下方のROM デコーダでデコードされた個々の信号 (I), (II), (III) なども同じように扱うものと約束する．

そして同図の各フィールドに沿って行方向に並ぶマイクロ命令を6個まとめてマイクロインストラクションと呼び，あるフィールドに該当するマイクロ命令がないときは，非実行マイクロ命令 NOP (No OPration) があるものとする．

たとえば 144_8 番地のマイクロインストラクションは (NOP I′ NOP II′ III′ NOP) とする．

以上の約束のもとで，マイクロプロセッサの働きを説明すると次のようになる．

a．ROM の 144_8 番地より，最初のマイクロインストラクション (NOP I′ NOP II′ III′ NOP) が RIR に読み出される．

これは RIR-RAR タイミング制御器によって行なわれ，このとき RAR の内容は 145_8 となる．

b．RIR に読み出されたマイクロインストラクションは ROM デコーダでデコードされ，前節の (I), (II), (III) の仕事を行なう．

同一マイクロインストラクションに含まれる全てのマイクロ命令は，同一基本単位時間内に実行開始される．

c．RIR-RAR タイミング制御器により，RAR の示す 145_8 番地の内容が RIR へ行き RAR は 146_8 番地を示す．

d．RIR の内容は ROM デコーダでデコードされ前節と同じ (IV), (V), (VI) の仕事をする．スキップフィールドの EOP 命令は，マイクロルーチンの最後から1ステップ手前のマイクロインストラクションに入れられ，最後のマイクロインストラクションが実行されると，そのフェーズを終了させる．

e．RIR-RAR タイミング制御器と EOP 信号を受けたフェーズロジックによって，RAR は0番地を示し，RIR は ROM の 146_8 番地の内容を受ける．

f．RIR の内容は ROM デコーダでデコードされ前節と同じ (VII), (VIII), (IX), (X) の仕事をして，EOP 信号により，エクスキュートフェーズを終了する．

以上がマイクロプロセッサの動作の概略である．

表1では，この例の (I)～(X) などをマイクロアセンブラ記号にして表わしている．

6. マイクロプログラムとタイミング

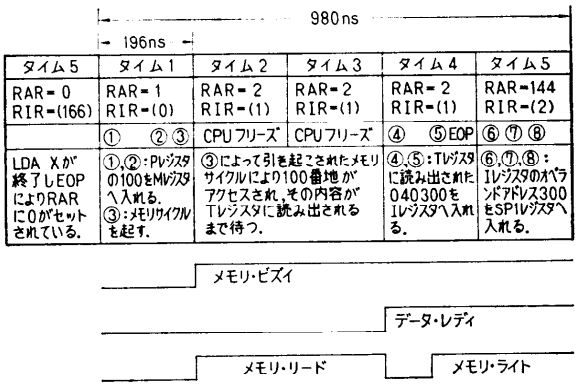
マイクロインストラクションは6フィールドのマイクロ命令から成り，これらは全て 196 ns の時間内に実行開始される．

これを ROM サイクルと呼び，通常，マイクロインストラクションは ROM サイクルごとに1ステップずつ実行されていく．ただし特殊なマイクロ命令を含んでいる場合は，自動的に CPU フリーズを起し，待ち時間を設ける．これは簡単にいえば図4における RAR-RIR タイミング制御器などのクロック信号を一時凍結することである．たとえば RW (Read/Write) マイクロ命令がこれにあたる．

表2は，フェッチフェーズとタイミングの関係を示したものである．図2を参照しながら説明することにする．

表1の (LDA X) に相当するエクスキュートフェ

表2 フェッチフェーズ・タイミング表



タイム5	タイム1	タイム2	タイム3	タイム4	タイム5
RAR=0 RIR=(166)	RAR=1 RIR=(0)	RAR=2 RIR=(1)	RAR=2 RIR=(1)	RAR=2 RIR=(1)	RAR=144 RIR=(2)
	① ②③	CPUフリーズ	CPUフリーズ	④ ⑤EOP	⑥⑦⑧
LDA Xが終了しEOPによりRARに0がセットされている.	①,②:Pレジスタの100をMレジスタへ入れる. ③:メモリサイクルを起す.	③によって引き起こされたメモリサイクルにより100番地がアクセスされ,その内容がTレジスタに読み出されるまで待つ.		④,⑤:Tレジスタに読み出された040300をIレジスタへ入れる.	⑥,⑦,⑧:Iレジスタのオペランドアドレス300をSP1レジスタへ入れる.

ーズのマイクロルーチンは，ROM アドレスの 164_8 番地から 166_8 番地に入っており，このエクスキュートフェーズが終了した時点から (ADA Y) に相当するフェッチフェーズが表2のようなタイミングで実行される．YOHPAC-2100 A の基本命令は，通常，1.96 μs で実行されていることになる．

7. コントロールストアの拡張

YOHPAC-2100 A の基本命令群を実行するマイクロルーチン群を全て格納してあるコントロールストアをモジュール0 (M0) と呼ぶ．

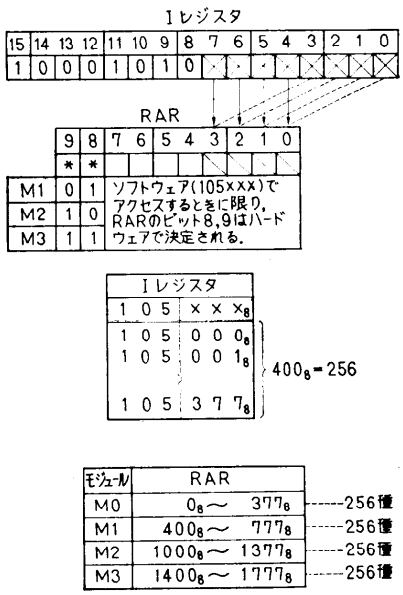
この ROM 中のマイクロインストラクションは1語 24 ビットで構成され，256 語の容量をもつ．

基本命令群の他に，独自に新しい命令を作って付加することができるように，コントロールストアは，モジュール1(M1)，モジュール2 (M2)，モジュール3(M3) まで 256 語単位で拡張することができる．M0～M3として WCS (Writable Control Store) を用いることもできる．

8. ユーザマイクロプログラミング

ユーザが作ったマイクロプログラムが M1，M2，M3に書き込まれている場合は，ユーザのプログラム

表3 マクロ命令と先頭番地の関係



でその先頭番地を RAR に与えなければならない．

そのため，

$105\times\times\times_8$

の形式のマクロ命令が用意されている．この×××は RAR へ入る先頭番地を決めるためのものであり，そのビット構成を表3に示す．またこの命令コードはアセンブラ記号で RAM と書かれる．

この擬似命令 RAM はコード 105 に相当するもので，×××に当る部分は別に定義した英文字で表わすことができる．

マクロ命令 ($105\times\times\times_8$) は，アセンブラ言語ばかりでなく，YOHPAC-2100 A のアルゴル，フォートラン，ベーシックなどのコンパイラ系統の言語からも簡単に呼び出すことができるので，この特徴を利用すれば，かなりダイナミックな興味あるプログラムを作成できるものと思われる．

(昭和 48 年 3 月 28 日受付)