マイクロプロセッサを用いたミニコンエミュレータの試作

今瀬眞[+] 萩原兼一[+] 豊高雄二[+] 細見輝政[+] 都倉信樹[+] 岡本卓爾[++]

[+]大阪大学基礎工学部　[++]岡山大学 工学部

1. まえがき

現在，大阪大と岡山大共同のグループでマイクロプログラム計算機(HOP)システムのハードウェアの設計を終り，その試作とソフトウェアの開発を行なっている。

HOPは，複数個所で稼動される予定であるが，その使用環境をまとめると次のようになる。

(1) 各ユーザがそれぞれの使用目的に向いた計算機のエミュレートプログラムを作る。

(2) とくにPDP11のエミュレートを行ない，既存のソフトウェアを利用することが予定されている。

(3) HOPだけを所有し，他の計算機のサポートを利用できないシステムがある。

このような使用環境を予定して，主に次のような目標をもって設計・試作を行なっている。

(1) 既存のいくつかのミニコンのエミュレートだけでなく，問題向き言語(高級言語をコンパイルした中間言語など)のエミュレートなどを効果的に行なえる。

(2) ユーザ・マイクロプログラムが可能で，しかもマイクロプログラミングが容易に行なえる。

(3) 時間的効率にも配慮する。

(4) スタンド・アロンでも，マイクロプログラムが可能。

本稿では，そのアーキテクチャとソフトウエア(マイクロプログラム)開発の方針について報告する。

2. アーキテクチャ

2.1 システム構成の概略

HOP試作システムの構成は，図1に示すように小規模なものである。2台のフロッピーディスク，キーボード，シリアルプリンタを付け，それにいくつかの入出力装置が付加可能である。

2.2 バス構成

アドレス空間　バスは17ビットのアドレス線を持ち，アドレスはバイト単位にふってある。アドレス空間はEとCに区分され，E空間は番地 000000_8 ～ 177777_8 C空間は番地 200000_8 ～ 377777_8 を占める。E空間には主記憶16KB(最大56KB)，機能レジスタ，および作業用記憶の一部(0.25KB)。C空間にはマイクロプログラム格納用記憶CS(30ビット×1K語；最大4K語)，作業用記憶(1.75KB)，各種デバイスのレジスタがそれぞれ含まれる。

E空間とC空間は別系統の入出力命令を用いてアクセスする。また，ワード(16ビット)またはバイト単位でデータの転送が行える。後述のようにEモードではC空間へのアクセスは禁止される。

バスユントローラ　このシステムではバス権をとりうるのがCPU内のRALUとハードコンソールおよびフロッピーディスクのDMA部だけなので，バスの制御は単純な方法を採用している。すなわち，先のミニマイクロ複合体で用いたアービタ[5]を主体に少し変更を加えたものを用いている。

2.3 コンソール[2]

これはPDP11程度の機能(EXAMINE, DEPOSITE, STOP/START, SINGLE STEP)を有しマイクロプログラムのデバッグに用いうるハードコンソールと，コンソールプログラムによってより高度のデバッグ，

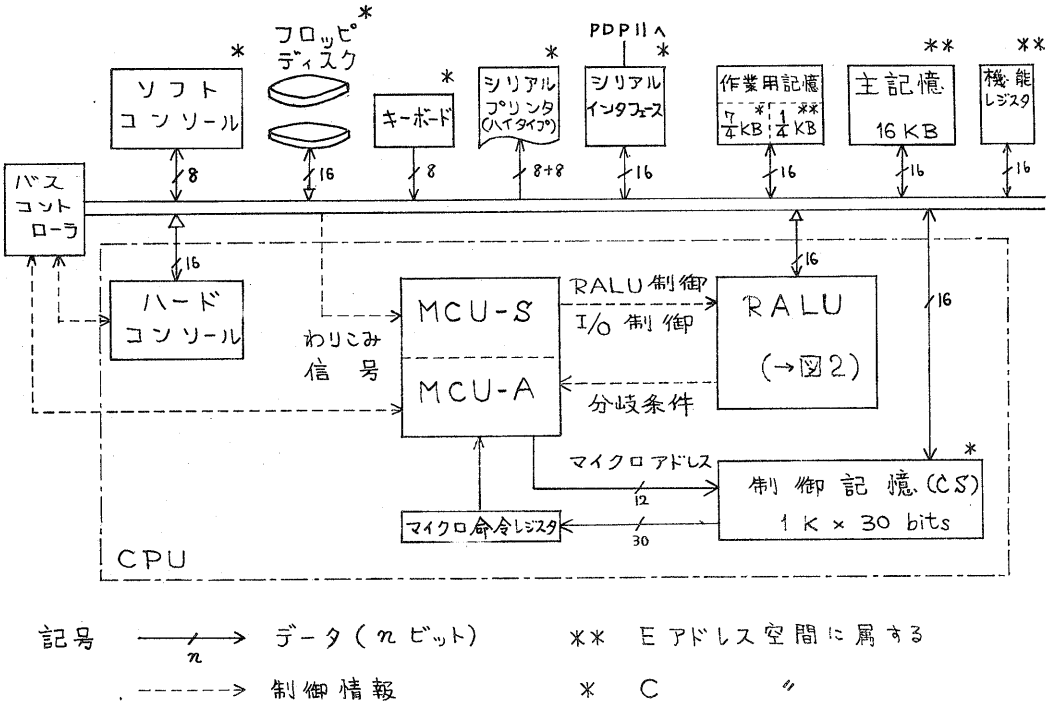


図1　構成概略図

実行制御，ユーザとのインタフェイス機能を実現するソフトコンソールとに機能上分けられているが，同一のコンソールパネルを共用しており，モードを切り換えることで機能を切り換える．入力操作をできるだけ単純にするという目的でテンキー式の入力を用いる半面，出力はできるだけ見易いものとする目的で2行一行の英数示管を採用している．

2.4 CPU

CPUは，図1のように構成されている。

CS(マイクロプログラム格納部) ユーザマイクロプログラムを可能とするため，IPL(Initial program load)用の一部のROM(Read only memory)を除いてCSはRAM(Random access memory)で構成している．書き換えは，バスを介して行う．マイクロ命令(30ビット)を一度にフェッチするために，命令フェッチ用専用バスが設けられている．このためバスと専用バスからのアクセスを排他制御している．

MCU(マイクロプログラムコントロールユニット) MCUのAユニットは，次に実行する命令の番地(シーケンス)を決定する部分である．かなり豊富なシーケンス指定命令を用意している．これは，研究教育用ということで，どのような命令が有効かということを，種々のプログラムを書いた上で判断しようというねらいがあったためである．以下に，本システムでの特徴的と思われる部分について述べる．

(1) ターゲットマシンのエミュレーションにおいて，命令の解読(デコード)をいかに効率よく行うかが大きな問題であろう．特に，HOPのように各ユーザが自分の命令を作ることを許す場合は，デコーダに汎用性を持たせる必要がある．汎用性がありしかも効率のよいデコードの方法として多分岐機能(後述)を設けている．

(2) 2分岐については，CPU内のある状

能ビットが1か0かで分岐する方式ではなくて，4変数の論理式が真か偽かによって分岐する形にしている．特にレジデュアル制御を用いて論理式を任意に指定できるので色々の応用が考えられる．

(3) 強力なサブルーチンコール，コルーチン命令を用いてマイクロプログラムの短縮を図った．

MCUのSユニットは，割込みの受付け及び命令の実行の延期や中断などを行なっている．割込みは，3重まで可能である．また，レベル0をEモード，レベル1～レベル3をCモードと呼び，Cモードのみで実行できる命令（特権命令）が存在する．

RALU（算術論理演算部）　図2にRALUの概略を示す．RALUはMMI社のM6701[6]と補助回路で構成されている．M6701は，4ビットスライスのバイポーラチップである．これを4個用い，基本語長を16ビットとしている．しかし，アドレスはバイトアドレシングを行なっており，バイト演算も語(16ビット)演算と同様に行える．また，4ビットフィールドの取扱いも，

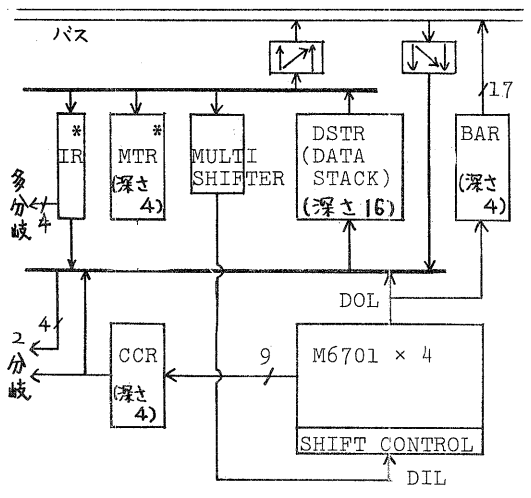


図2．RALUの概略

4ビット単位のシフト機能，4ビットについての判定（2分岐機能）などで比較的容易に行なえる．

M6701の基本サイクルタイムは16ビットの場合205 msecであるが，演算前後の処理（これは，異常状態を検知してすぐ割込みを起こす，implied wait* などソフトの負担を減らすための処理による．）のため現在280 msecに設定されている．

RALUの詳細については，次章にゆずる．

2.5 シリアルインタフェイス

このシステムのマイクロプログラムは，まずPDP11上のマイクロアセンブラなどを用いて作られるが，これをこのシステムに移すのに単純なシリアルインタフェイスを介して行う．

DOSなどのシステムプログラムの移植もこれで行う．

2.6 機能レジスタ[3]

一般に計算機において，バスからアクセスするレジスタには特殊な機能を持ったものがある．（PDP11の入出力装置レジスタ，プログラム状態語など）このような機能をエミュレートしようとするとき，レジスタへアクセスすることで特定のマイクロルーチンを起動させる必要がある．アクセスすることでマイクロ割込みを引き起こす機能レジスタを導入しエミュレートの能率向上を図った．また機能拡張用としても用いうる．

3. 命令体系

マイクロ命令は，一命令30ビットで構成されている．[表1]にその命令体系を示す．各命令コード（可変長）と以下のような各種指定フィールドからなる．

(1) 演算指定フィールド(19-0)：RALUに対する指定（表中，右下りハッチ）

(2) 入出力指定フィールド(22-0)：入出力動作の指定（表中，左下りハッチ）

* implied wait については次章入出力動作の項で述べる．

(3) その他：シーケンス制御の指定と推指定

なお，一命令中いくつかの並列実行可能な指定のあるときは並列実行する．以下，各命令を中心にCPUの動作を説明する．

Table.1 Instruction Format

				MNEMONIC
	000 001	OFFSET	RALI-1	JMPS JSRS
	0100 0101 0110	C	not used / ADDR	JMP/JMPC JSR/JSRC PUSH/PUSHC
*	0111		RALI-2	CONST
	1000 1001 1010	F		SKIPW SKIPB SKIPD
*○	1011	L		CH-MTR
	1100 1101 1110	C	I/O	MEXT/NEXTC RTS/RTSC COR/CORC
	1111000 1111001 1111010 1111011		RALI-1	JMPI PUSHI SKIPS LD-IR
*○	1111100			CSR/W
*○	1111101	i		RE-PTR
*	111111000			CH-SU
*○	111111001	j		CH-MF
*○	111111010			CH-SUMF
	111111011			not used
	1111111000			RESET
*	1111111001			PAUSE
*○	1111111011			RTI
*	1111111100			TRAP
*	1111111101			ISE
	1111111110			not used
	1111111111			POP

*：*以外はMCU-Aに対する命令
○：特権命令でCモードのみで使用できる．

3.1 MCU-A

MCU-AはMCU-Sのもとに実行順序制御を行う．MCU-Aには，次のようなレジスタが存在する．

プログラムカウンタ(PC) PCは現在実行中の命令の番地を示す．もし，命令中で飛び先の指定情報がない時はPC+1が次の実行番地となる．

エントリポイントレジスタ(EPR) 割込みの時には，EPRの示す番地へジャンプする．

マクロ命令レジスタ(IR) 多分岐機能で用いるレジスタで，通常マクロ命令が代入される．

PCスタック(PCSTR) 深さ16のスタックで，サブルーチン，割込みなどの際のリンケージ情報の退避場所となる．

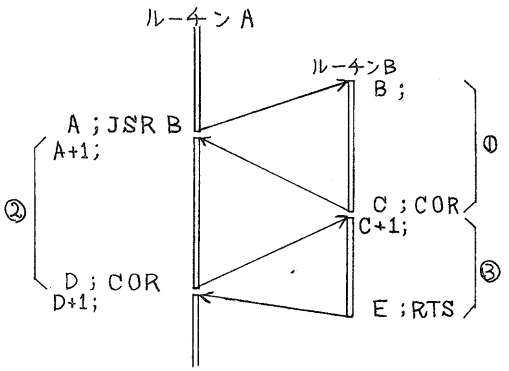
3.2 実行順序制御命令

サブルーチン JSR/JSRC/JSRS はいずれもサブルーチンコール命令でもどり番地をPCスタックに退避して指定番地へとぶ．JSRは12ビットで指定される任意の番地へとぶことを許し，JSRCではCASE修飾をさらに許す．このときは演算指定ができないが，JSRSではとび先をoffsetで，JSRS命令から相対的にPC-61～PC+64の範囲でえらべ，同時に演算指定も行える．*

RTSはサブルーチンからもどるための命令である．RTSCはもどり先をCASEで修飾する．

ジャンプ命令 JMP/JMPC，JMPSはそれぞれJSR/JSRC，JSRと対応するジャンプ命令である．また，JMPIはデータスタックのトップの内容をとび先とするものである．

ユルーチン COR(ユルーチン)命令は，PCスタックのトップの内容をとび先番地と



①の区間；PCスタックのトップにA+1がある．
②の区間；〃 C+1 〃
③の区間；〃 D+1 〃

図3．ユルーチンの使用例

*以下，Cをニーモニックにもつ命令はCASE修飾をする場合である．(これについては多分岐機能の項参照)

する．その際PCスタックのトップがPC＋1に書き換えられる．この命令は，サブルーチン命令，リターンサブルーチン命令と組み合わせて，2つのルーチンで互いに呼びあうシーケンスを実現できる．図3にその使用例を示す．

プッシュ命令　これは，PCスタックに飛び先をプッシュダウンする命令で，命令中のADDR部(12ビットの定数)をプッシュダウンするPUSHとデータスタックのトップの内容をプッシュダウンするPUSHIがある．これは，RTSと組み合わせるとジャンプ命令，CORと組み合わせるとサブルーチンコール命令となる．

2分岐命令　指定された条件が成立する場合は1命令スキップする．HOPでは，レジデュアル制御を用いて条件自体に汎用性をもたせた．図4にその機構を示す．SRに8種類の4変数の論理関数を真理値表の形で書き込んでおく．今，それを $f_i(a_1\ a_2\ a_3\ a_4)$ $(0 \leq i \leq 7)$ とする．分岐命令中のF部(25-23)でiを指定する．$(a_1\ a_2\ a_3\ a_4)$ として $(N_W\ V_W\ Z_W\ C_W)$ をとるSKIPW，$(N_B\ Z_B\ V_B\ C_B)$ をとるSKIPB，DOL(図2参照)の上位4ビットをとるSKIPDがある．ただし，$N_W\ Z_W\ V_W\ C_W$ は16ビットデータの演算結果についての条件ビットで順に，負．零．オーバフロー．キャリを示す．$N_B\ Z_B\ V_B\ C_B$ は8ビットデータの演算結果についての条件ビット．これらは，CCR内のビットである．分岐命令を行なった場合，f_iが真ならばPC＋2，偽ならばPC＋1が次の実行番地となる．

また，シフトキャリビット(CCR中のSビット)が1ならスキップするSKIPS命令がある．

多分岐機能(CASE)　多分岐機能を理解するために，以下にその使用例を示す．

JMPC I, A　　$1 \leq I \leq 4$*　A：アドレス

を実行した場合，次の実行番地は

$$NA(\text{Next address}) = \sum_{i=1}^{I} IR(16-i) \times 2^i + A$$

となる．(ただしIR(i)は，IRレジスタのiビットを意味する．)　つまりIRの上位Iビットをベースアドレス(CASE指定のない時の飛び先)に加えることにより，多分岐を実現している．その際，同時にIRの内容は左へシフトされる．

IRにはデータスタックのトップからデータを転送する．(LD_IR命令)　また，割込み時などの退避のために，CONST命令でデータスタックのトップにプッシュダウンすることができる．

この機能を使えば，IRにマクロ命令を代入した後は，デコード作業はすべてMCU_Aで行うことができる．また，RALUの動作と並列してデコードを行うことができるので，時間的効率もよい．

この方法では，命令の上位ビットから順にデコードしなければならないが，サブルーチン，コルーチンなどをうまく使えば，その制限も支障はない．

3.3 MCU-S

MCU-Sは，CPU全体の制御を行うユニットで次の2つの動作を行っている．

(1) 割込み

割込みは，その受付けと実際の割込み動作の2つのステップがある．

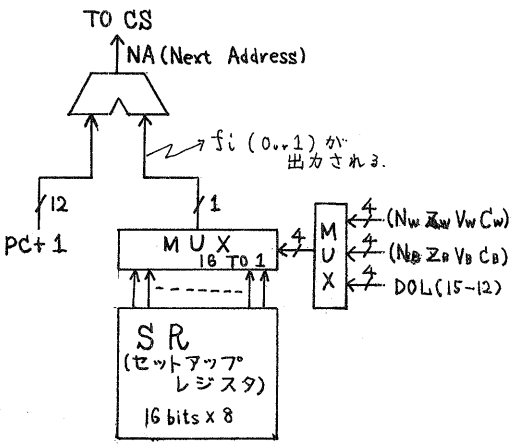


図4．2分岐機能の機構

* Iは命令中のC部(25-23)で指定する．C部＝000_2の時は，CASE指定がない場合である．

・ステップ1（割込みの受付け）；割込み要求が起きた場合，すぐに割込みが受付けられるとは限らない．マスクレジスタで禁止されている割込みについては，割込み禁止が解除されるまで割込みの受付けは延期される．

マスクレジスタには，マスクビットレジスタMTR，マスクフラグMF/SUMFがある．MTRは8ビットレジスタで，異常状態，ソフトコンソール割込み，機能レジスタ及びTRAP命令（後述）による割込み，デバイスiグループ割込み（i=1,2,3）の6つのマスクビットがあり，要因ごとに割込みを禁止している．また，マイクロトレース*，マクロトレース**の2つのトレース要求ビットがある．MF，SUMFはそれぞれ1ビットで全割込みの禁止，部分的な割込みの禁止を表示する．

・ステップ2（割り込み動作）；割り込みが受付けられた時には，1サイクルタイムで次の動作を行う．

(a) 状態の退避；CPU内で状態の退避の必要なものは，4段のレジスタファイルで構成して，MCU-Sがそれらを切り換えている．従って割込みは，3重まで可能である．BAR,MTR,CCRがこのようなレジスタファイルで構成されている．

(b) シーケンスの変更；シーケンスの変更はサブルーチンコールと同様の動作を行う．割込みが起こるとEPRへ飛び，割込み処理を行う．

(c) マスクレジスタの変更；全面的に割込み禁止となる．（MFをセットする）

3.4 割り込みとその他の実行制御に関する命令

CH-MTR，CH-MF，CH_SUMF命令は，マスクレジスタの変更に用いる．

RTI命令は，割込み処理ルーチンからもとへもどる時に使用する．

TRAP命令は，EモードからCモードへの割出し命令である．

ISE命令は，TM（マクロトレースビット）と共に使用する．EモードでISE命令を実行した際TM=1であれば，Cモードに割込みがかかる．ISEは（TMによる）条件付の，TRAP命令は無条件のEモードからCモードへの割出しに用いうる．

PAUSE命令は命令の実行を中断し，コンソールパネルはハードコンソールモードになる．

3.5 RALU（算術論理演算部）

図2にRALUの概略の構成を示した．M6701中には16個の汎用レジスタGRがあり，A，Bの両ポートから任意のレジスタにアクセスできる．また，もう一つのQレジスタQRがあり，GRと同様の使い方以外に，GRと（シフトコントロール回路を介して）連結し32ビットレジスタを構成することもできる．M6701での各演算後，N（負），V（オーバフロー），Z（ゼロ），C（キャリ）の信号をコンディションコードレジスタCCRにとりこむ．バイトとワードについて8ビットのデータを同時にとり2分岐判定がどちらについてもできる．

データスタックDSTRは，深さ16である．M6701の出力DOL およびREAD操作でバスからとりこまれたデータはDSTRのトップに転送される．またCCR，IRをとりこむこともできる．DSTRのトップのデータは，マルチシフタを通してM6701（DIL），WRITE操作でバス上，MTRおよびIRに転送できる．

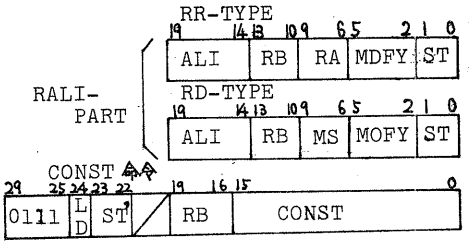
マルチシフタは命令によって4，8，12ビットの左巡回シフトとデータの下バイトの符号ビットを上バイトに広げるサインエクステンションの機能を果す．

シフトコントロールは5ビットとよぶ1ビット（CCR内にある）を語，バイトあるいは倍長語データに連結して，左右へ1ビットすることと，左右の1ビット巡回シフトを許している．上記の2つの

---

* マイクロトレースビットが1であれば，マイクロ一命令ごとに割込みをかける．

** マクロトレースはマクロ一命令ごとに割込みをかけるための機構であり，ISE 命令の中で説明する．



ALI:Arithmetic and logic operation instruction
RB,RA:GR(general registers)select
MS:Multi shift control
MDFY:Load,1bit shift, and DOL control
ST:DATA STACK control
LD:Load constant

Fig.5 INSTRUCTION TO ALU

Table.3 MDFY-PART Instruction

OP.CODE 5-2	DOL Control	1 bit shift and load control
0000	DOL ← F	BR ← F
0010	DOL ← AR	BR ← F
0100	DOL ← BR	BR ← F
0110	DOL ← F	BR ← F(14-0)·S
0111	DOL ← F	S·BR ← F·S
1000	DOL ← F	BR ← S·F(15-1)
1001	DOL ← F	BR·S ← S·F
1010	DOL ← F	If SU=1 then BR·QR←F(14-0)·QR·S else BR(7-0)←F(6-0)·S
1011	DOL ← F	If SU=1 then S·BR·QR←BR·QR·S else S·BR(7-0)←F(7-0)·S
1100	DOL ← F	If SU=1 then BR·QR←S·F·QR(15-1) else BR(7-0)←S·F(7-1)
1101	DOL ← F	If SU=1 then BR·QR·S S·F·BR else BR(7-0)·S←S·BR(7-0)
1110	DOL ← F	QR ← F

F:ALU output, S:Sbit in CCR
BR:GR(RB) , AR:GR(RA)

Table.2 ALI-PART INSTRUCTION

BR:GR(RB-PART),AR:GR(RA-PART)

OP.CODE <19:15>	TYPE	ALU OUTPUT (F) IR<14>=1	ALU OUTPUT (F) IR<14>=0	OP.CODE <19:15>	TYPE	ALU OUTPUT (F) IR<14>=1	ALU OUTPUT (F) IR<14>=0
00000	RR	Force 11...1	Force 00...0	10000	RD	DIL	DIL + 1
00001	RR	AR ∧ BR	AR ∧ BR	10001	RR	BR	BR + 1
00010	RD	DIL ∧ BR	DIL ∧ BR	10010	RR	QR	QR + 1
00011	RR	AR ∨ BR	AR ∨ BR	10011	RR	AR + 11..1	AR
00100	RD	DIL ∨ BR	DIL ∨ BR	10100	RD	DIL + 11..1	DIL
00101	RR	AR ⊕ BR	AR ⊕ BR	10101	RR	BR + 11..1	BR
00110	RD	DIL ⊕ BR	DIL ⊕ BR	10110	RR	QR + 11..1	QR
00111	RR	$\overline{AR}$ + 11...1	$\overline{AR}$	10111	RR	AR + RB	AR + NR + 1
01000	RD	$\overline{DIL}$ + 11...1	$\overline{DIL}$	11000	RD	DIL + RB	DIL + BR + 1
01001	RR	$\overline{BR}$ + 11...1	$\overline{BR}$	11001	RR	AR + QR	AR + QR + 1
01010	RR	$\overline{QR}$ + 11...1	$\overline{QR}$	11010	RD	DIL + QR	DIL + QR + 1
01011	RR	$\overline{AR}$	$\overline{BR}$ + 1	11011	RR	AR - BR - 1	AR - BR
01100	RD	$\overline{DIL}$	$\overline{DIL}$ + 1	11100	RR	BR - AR - 1	BR - AR
01101	RR	$\overline{BR}$	$\overline{BR}$ + 1	11101	RD	DIL - BB - 1	DIL - BR
01110	RR	$\overline{QR}$	$\overline{QR}$ + 1	11110	RD	BR -DIL - 1	BR - DI
01111	RR	$\overline{AR}$	$\overline{AR}$ + 1	11111	RD	DIL - QR - 1	DIL - QR

シフト機能でフィールド処理はかなり容易になっている.

3.6 演算指定

JMP/JMPC, JSR/JSRC, PUSH/PUSHC, CONST の7命令を除いて, 命令の第19ビットから第0ビットの20ビットフィールドをALI-1と呼んでおり, 図5のようにRR型とRD型にわけられる. ALI部はM6701に対する演算指定部で, RB部とRA部で指定される汎用レジスタGR(これをBR, ARと呼ぶことにする), DIL, およびQRの内容のうちどれかまたは1つのオペランドに対する演算を指定できる. MDFY部はM6701の出力DOLへ出力すべきデータ(BR, AR, あるいは演算結果F), 演算結果の左右一ビットシフトなどの指定とシフトコントロールの指定を含む. STはDSTRの指定である. ALI, MDFY, ST部については表2, 3, および4に示す.

Table.4 ST-Part Instruction

OP.CODE <1-0>	Meaning
00	no effect
01	pop up DSTR
10	push down DOL to DSTR
11	push down DOL to DSTR after pop up DSTR

RA, RBは通常GRを選択するが, EモードでRBに15hを指定すると, IRレジスタの上位3ビットでGR0~7の一つを選択する. これはマクロ命令中のレジスタ指定をそのまま用いることを可能にする.

RD型の時は, DSTRからマルチシフタを通してM6701に送られる. MS部はマルチシフタに対する指定で表5に示してある.

Table.5 MS-Part Instruction

OP.CODE <9-7>	Meaning
000	not shift
001	4bits cyclic shift
010	8bits cyclic shift
011	12bits cyclic shift
100	SE(Sign Extention)
101	SE after 4bits cyclic shift
110	SE after 8bits cyclic shift
111	SE after 12bits cyclic shift

CONST命令　CONST命令は図7の形であり命令の下16ビットを定数データとして, LD部が1のときにRB部で指定されるGRに転送する. LD部が0のときは転送せず, ST'部で指定されるスタック操作のみが行われることになる.

Table.6 ST'-Part Instruction

OP.CODE <24-23>	Meaning
00	Push down CCR to DSTR
01	Push down IR to DSTR
10	no effect
11	EPR ← CONST

3.7 入出力指定

入出力動作とは, CPUとバス上のデバイス(メモリ, デバイスレジスタ等)とのデータ転送をいう.

入力動作(read)を行う時は, BARに読むべきデータの番地を代入し, read指令を出す. 読みとられたデータはDSTRにプッシュダウンされる. 出力動作(write)を行う時は, BARに番地を代入しwrite指令を出せばよい. これにより, 所定の番地に所定のデータが書込まれ, DSTRのトップがポップアップされる.

Table.7 I/O part Instruction

OP.CODE <22-20>	Meaning
000	No effect
001	Load BAR
010	Write Word
011	Write Byte
100	Read Word after load BAR
101	Read Byte after load BAR
110	Write Word after load BAR
111	Write Byte after load BAR

表7にI/O部の命令を示す. CSR/W命令の時にはCアドレス空間にアクセスし, それ以外の時はEアドレス空間にアクセスする.

入出力動作は, 他の指定と並列して指定されるが, 他の指定の動作完了後CPUとは非同期に行なわれる.(次以降の命令と並列して行われる.)同期をとる(入出力動作の完了を待つ)必要がある時は, MCU-Sが命令をみて判断し, 入出力動作完了まで命令の実行を延期する. これをimplied wait の機能とよんでおり, ユ

ーザは，wait命令を書く必要はない．
また，バス上のデバイスを初期化するRESET命令がある．

4．マイクロアセンブラ[4]

HOPのマイクロプログラムを開発するために使用されるマイクロ・クロス・アセンブラ(μCA)の2つの主な設計目標を述べる．

01．従来のアセンブラの備えていた機能から，HOPのように対象とするマイクロプログラムが小規模なものゆえ削減可能な部分を切出し，μCA自体を小規模に作成する．

02．HOPのマイクロ命令は，あるフィールドが他のフィールドに依存する場合が多いので，それらのフィールドの無矛盾性の検査やアセンブル時に判別可能な制限事項の検査を可能な限りμCAで行い，デバッグを容易にする．

設計目標01.の達成手段としては，次に示す3つを実行した．

01-M1. 対象とするマイクロプログラムの規模は小さいので，リンカは作成せずソースプログラムの段階でリンクする．従って，リロケータブル・アセンブラのように複雑な処理を行わなくてよい．

01-M2. ソースプログラムで使用可能なシンボルの長さを英数字3文字以内と短くし，シンボルテーブルの大きさを小さくする．

01-M3. J.R.Bellの提案したThreaded Codeの手法[7]を用いてμCAを作成する．

設計目標02.の達成手段としては，次に示す3つを実行した．

02-M1. 使用するシンボルの属性を，ソースプログラム上で陽に指定する．たとえば，サブルーチン名，飛び先のラベル名，グローバル変数名あるいはローカル変数名を，それぞれ，S.XXX，E.XXX，G.XXX，L.XXX(ただし，XXXは英数字3文字あるいはスペース)と記述する．このことにより，サブルーチンコール命令のオペレーションフィールドとサブルーチン名フィールドとの対応関係や，違法な変数参照などの検査が行える．

02-M2. オペレーションを示すニーモニックに，そのオペレーションに合法なオペランドの属性を示す情報も含ませる．このことにより，ユーザの思い違いによるオペランドの違法な組合せなどを発見できる．

02-M3. その他アセンブル時にチェックできることはできるだけチェックする．

以上のような方針でアセンブラの設計を行っている．この方法でたとえば，02-M1.の方法で記号表が小さく，かつ記号表を属性別につくることで検索もはやくできる利点がある．また，Threaded Codeはいわば中間言語レベルでありプログラムも比較的くみやすい．また，各サービスルーチンは純手続き的に作れるので，各サービスルーチンをマクロ命令として直接実行するファームウェアを作製して，アセンブラマシンをHOP上につくることも可能である．

5．ファームウェアシステム[4]

HOPのファームウェアシステムはモジュール構成で作られており，以下に示すようなモジュールからなっている．

スケジューラ　割込みあるいは割出しが起きたときに全物的にここに制御が移り，割込み要因あるいはスーパバイザコールの解析を行い，対応する処理プログラムへ制御を移す役目を持つ．

ソフトコンソール　図1に示したように，HOPには概念上ハードコンソールとソフトコンソールがある．ハードコンソールは，文字通り従来のコンソール機能を全物的に持つ．しかし，RALU内のバス上の番地を持たないレジスタなどを全物的にコンソールから調べるためには，価格/性能比の悪い全物を備えなくてはならない．HOPでは，このようなものは全物でなくファームウェアにより実現している．さらに，ターゲットマシンレベルのコンソールもファームウェアにより実

現している．

トレーサ　マイクロトレース割込み処理プログラムであるが，マイクロプログラムのデバッグに利用する．すなわち，マイクロ命令を実行ごとにトレーサに制御が移るので，そのときのプログラムなどの状態を監視できる．以下に，トレーサにより実現可能なデバッグ機能の例を示す．以下の条件によって，ソフトコンソールに停止条件のメッセージが表示され，ユーザからの入力受付け状態になる：

D1. アドレスストップ：マイクロプログラムの制御があらかじめ指定された番地に来たとき．

D2. マイクロ命令ストップ：実行したマイクロ命令の（部分的な）ビットパターンが，あらかじめ指定されたものと一致したとき

D3. ステップカウンタストップ：あらかじめ指定された数だけマイクロ命令が実行されたとき

D4. 条件ストップ：あらかじめ指定された条件（たとえば，「ある番地の内容がある値に等しくなった」）が満されたとき

マクロ割込み処理　マクロトレース割込み処理プログラムで，ターゲットマシンの割込み機構のエミュレートなどを行う．

エミュレートプログラム　ターゲットマシンの命令をエミュレートする．ユーザは，エミュレートプログラムとマクロ割込み処理プログラムだけを変更することにより，HOPを望みのターゲットマシンのエミュレータとすることができる．

スーパバイザコール処理　ユーザに種々のサービスを行う．使用頻度の低いものはオーバレイプログラムにすることによりCSの使用効率が高められる．

エミュレートプログラムを書く各ユーザにとって入出力装置の制御部などの部分は分離されているので比較的容易にエミュレートプログラムをかけ，またデバッグの手段も有効なサポートとなりうる．

従来，マイクロプログラムのデバッグ法としてはシミュレータなどがよく用いられているが，ここではHOPのみを用いるユーザもあるのでかなり豊富強力なデバッグ手段を提供することとしている．

6. あとがき

現在ハードウェア製作中であるが，PDP11/20，NOVAについてエミュレートプログラムを評価のために作った．PDP11/20については，プログラムサイズは450ステップ程度で，一命令の実行時間は3μsec～6μsec程度であった．また，NOVAについては，プログラムサイズ200ステップ程度で，実行時間は3μsec～6μsec程度であった．作製の過程でJSRC，JSR，CORなどの有効性が明らかとなった．

最後に，日頃よりいろいろ御指導，御鞭撻いただいている嵩忠雄教授に厚く御礼申し上げます．

参考文献

[1] 細見，萩原，今瀬，豊高，岡本：マイクロプロセッサを用いたミニコン汎用エミュレータ，信学技報，EC76-10，1976.

[2] 馬場，岡本，今瀬，都倉：デバッグの能率向上を志向したコンソール，電気四学会中国支部連大，72320，昭和51年11月

[3] 今瀬，萩原，豊高，都倉：ある汎用エミュレータにおける機能レジスタ，昭52信学会統合全大（予定）．

[4] 豊高，萩原，今瀬，都倉：あるマイクロコンピュータシステムにおけるソフトウェア作製，同上

[5] 岡本，酒井，本藏，細見：非同期式リングアービタの一方式，信学論 59-D(8) 582-583，昭和51年8月．

[6] Monolithic Memories：4bit Expandable Bipolar Microcontroller 5701/6701，1976

[7] J.R. Bell：Threaded code，CACM16(6)，370-372，July 1973.