UDC 681.325/.326：621.3.049.774'14

マイクロプロセッサアーキテクチャの一設計

正員　飯塚　肇†　　正員　古谷　立美†

An Architectural Design of a Microprocessor

Hajime IIZUKA† *and* Tatsumi FURUYA†, *Regular Members*

あらまし　本論文は新しいＬＳＩマイクロプロセッサ（μＰ）のアーキテクチャとその評価について報告している．このμＰは5,000ゲートという制約の下で，はん用エミュレーションなどの新しい世代の応用に適した機能を多数盛りこんだ高性能プロセッサであって，特に，はん用性，フィールド処理，スタック処理などに新しいアイデァや機能が盛りこまれている．基本処理語長は１６ビットであるが，外部とのインタフェースは３２ビット幅であり，３２ビット長のマイクロ命令によって，ハードウエアの細部まで柔軟に制御される．又，レジスタがはん用，専用あわせて４４個あり，内部記憶量も多い．昭和５１年度末までにＬＳＩサンプルの試作を予定しているが，評価の結果は半導体技術的にも処理能力上も初期の目標を十分満たすものであった．

1. まえがき

１～数個のＬＳＩチップにＣＰＵの機能を集積したいわゆるＬＳＩマイクロプロセッサ（以下μＰＵと略）は一般に1971年に発表されたＭＣＳ－４がその嚆矢（こうし）とされている．以来，その技術は集積度，速度共格段の進歩をとげたが，現在の技術レベルはなおその巨大な可能性から見れば氷山の一角にすぎないと一般に信じられている．

このようなμＰＵの用途には，(1)制御装置などどちらかといえば性能より低価格性が重視される分野と，(2)一般計算機のＣＰＵ，あるいは複合して本格的な計算機システムを構成する，という２つの代表的分野がある．これまで開発されたアーキテクチャは主に前者を指向するものであったが，これは現在の技術水準，またＬＳＩが産業的には大量生産を基礎に成立するという事実からして当然のことであった．しかしながら，２～３年後には後者の応用に適合できる可能性は極めて大きく，その実現の暁には計算機システムの構造は大きな影響を受け，μＰＵアーキテクチャの良悪は非常に重要なものとなろう．

電子技術総合研究所（以下，電総研）ではこのような状況をふまえ，新しい高性能μＰＵの開発研究を進めているが，本論文ではそのアーキテクチャ設計について報告する．

2. 基本設計

2.1 半導体技術目標

半導体グループが設定したこのμＰＵの技術目標は表１に要約したとおりである．我々の目的はこのような制限条件の下で，少なくともミニコンピュータ程度の処理能力を持ち，且つ，広範囲の応用に適合できるはん用性を有するμＰＵアーキテクチャを開発することにあるが，5,000 ゲートという条件がその設計に大きな影響を与える．すなわち，この値は市販品の２倍以上ではあるが，目標からすればなお十分とはいえず，

表１　半導体技術の目標性能

半導体形式	Ｎチャネル ＭＯＳ
集　積　度	5,000 ゲート以上
チップ内遅延時間	2ns／ゲート以下
チップ間伝送遅延時間	10ns 以下
消費電力	0.2mw／ゲート以下

†電子技術総合研究所電子計算機部，東京都
Computer Division, Electrotechnical Laboratory, Tokyo, 100 Japan

論文番号：昭 51-159[D-30]

内部へ含めるべき機能の選択が重要な設計課題となる．

次に，集積度と並ぶ制限条件として，表1に表れていないピン数の問題がある．現在は50ピン以下のものが普通であるが，この程度ではインタフェースピンの時分割使用が不可避となり，内部速度に見あった処理ができない．又，短い命令長となって細かい制御が不可能になるであろう．そこで，このμPUでは製作側の事情も考慮して，ピン数の限界を約80として設計することにしたが，この条件を満足する高性能インタフェースの研究開発が第2の設計ポイントである．

2.2 基本設計

我々はこのμPUのアーキテクチャの設計に当って，半導体技術と計算機システム技術の水準，ならびに応用側の要求を考慮して以下のような考え方を設計の基礎に採用した[1]．

(1) 大きいはん用性（シーケンス制御部の分離）

LSIμPUは少種，大量生産を基礎に成立する技術であるからそのアーキテクチャには大きなはん用性が必要である．一般にプロセッサの機能（命令）は

① F形(Functional)：算術論理演算機能
② P形(Procedural)：シーケンス制御機能（ブランチ，割込など）
③ E形(External)：外部制御機能（メモリアクセス，入出力など）

の3種に分類されるが，はん用性との関連はそれぞれ異なる．まず，F形はマイクロプログラムレベルでならどの応用でも差は少なく，はん用性のある機能を抽出することは比較的容易である．ところが，P形は特に割込みの制御に関し，システムごとに異なる要求が多く，標準化は難かしい．又，インタフェース線が多いこととハードウエア量の関係で，LSI中に含めると，ごく簡単な機能しか組み込めないのが通例であって，このμPUの目的とする用途では不満足なものになる可能性が大きい．最後に，E形であるが，これもシステムごとの特殊性はあるが，もともと外部におかれる機能であるから，そのインタフェースのはん用性，拡張性にさえ注意しておけばよいであろう．

以上の考察と集積度，ピン数の制限から，我々はP形命令に対するシーケンス制御部をμPUチップ内に含めないことにして，チップ内の機能はF形に集中し，P，Eの機能に対してはしかるべきものを外部に設定できるようにインタフェースのはん用性，拡張性に配慮した．

(2) 基本語長　16ビット

この種のμPUにとっては一度に処理できるデータ長の決定は重要である．1つの方法は4，8ビットなどのビットスライスによって拡張性を持たせることであるが，状態情報の取扱いなど必ずしもうまくいかない面もある．又，集積度から考えて8ビットでは少ないであろう．そこで16ビットを基本長に採用したが，インタフェース部分は32ビットとし，アーキテクチャ的にも倍長データの取扱いの簡単化に配慮した．これは与えられた条件からすれば最も適切な設定であると考えている．

(3) エミュレーション向き

ここでいうエミュレーションとは既存計算機の機械命令を実行するということに限らず，もっと広く，新しく考えられたシステムアーキテクチャや，高級言語をコンパイルした中間言語のエミュレーションのことを指しており，今後のプロセッサの重要な機能の一つである．そのためには各エミュレーションにおける環境情報をレジスタにセットアップするいわゆるレジデュアル制御を多用し，効率を上げることが望ましいが，ハードウエア量の制限から，1語中のフィールドの抽出に有効なマスクレジスタと若干のモードビットの使用にとどめた．マスクレジスタはマルチビットシフタと組合せることにより，（マクロ）命令のデコードなどの部分フィールド処理を容易にするだけでなく，通常の処理にも有効性を発揮し，一般性がある．

(4) スタック機能

最近の応用ではプッシュダウンスタックがしばしば利用され，その有効性はよく認識されている．そこでその高速化のためにスタックの上部を常に内部レジスタに残すハードウエアサポートを取り入れた．

(5) ハードウエア細部の柔軟な制御

一般に，機能モジュールのレベルが高まると，より低レベルでの機能の実現や変更が困難になって，柔軟性が減る．これをモジュール化による“透明度の減少”と呼ぶ[2]が，このμPUのようにかなりの高レベルモジュールであり，なお，はん用性が重要な要素である場合にはこの透明度の減少を極力小さくしなければならない．そのために，ここではμPUのできるだけ細部までプログラム制御できるように厳しいピン数の制限の中で従来になく長い（32ビット）命令によるマイクロプログラム制御を採用した．

(6) 規則構造

この種のμPUはいわゆるスタティックマイクロプ

ログラミングの場合と異なり，必ずしもハードウエアの専門家ではない一般ユーザによってプログラムされる場合も多いと予想される．そこで，命令体系，内部構造はできるだけわかりやすい規則的構成とした．

(7) 半導体技術に適したアーキテクチャ

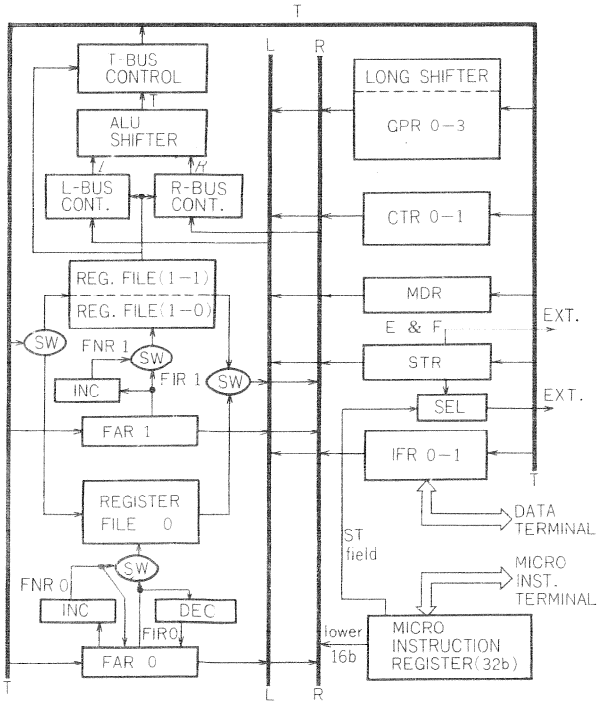
現在の半導体技術で実現が容易な内部構造となるようアーキテクチャに配慮した．レジスタファイルのアクセスが一度に2語以下におさまるようにしたのなどはその一例である．

3. アーキテクチャ

このμPUは図1のブロック図に示されているように，基本的にはALUとシフタを中心にした16ビットの3バス構成であるが，通常ALU入力の一方(L)と出力バス(T)には同一のレジスタが接続される[1]．

3.1 機能ユニット

3.1.1 主レジスタ 表2に示した，常に直接アクセスできる16個のレジスタで，以下に述べるように大部分は専用の用途が割当てられている．



GPR:General Purpose Register, CTR:Counter, MDR:Mode Register, STR:Status Register, IFR:Interface Register, FAR:File Address Register, FIR:File Indirect Register, FNR:File Next Register, INC:Incrementer, DEC:Decrementer, SEL:Bit Selector, SW:Switch

図1 マイクロ処理ユニットのブロック図
Fig. 1 — A block diagram of microprocessing unit.

表2 主レジスタ

ニモニック	名称	長さ(ビット)	数	ハードウエア
GPR	はん用	16	4	yes
FAR	ファイルアドレス	4	2	yes
MDR	モード	16(4)	1	yes
STR	状態	16	1	yes
FIR	ファイルインダイレクト	16	2	no
IFR	インタフェース	16	2	yes
FNR	ファイルネクスト	16	2	no
CTR	カウンタ	16	2	yes

(1) はん用レジスタ(GPR0～3)

4個の独立したはん用のレジスタである．

(2) モードレジスタ(MDR)

μPU内の制御モードを記憶する．RF, SM, SB, SEの4ビット(具体的内容は後述)が定義されている．

(3) 状態レジスタ(STR)

μPU内の16個の状態ビットをまとめたレジスタである．

(4) カウンタ(CTR0～1)

通常のレジスタとしても使えるが，マイクロ命令で指定して，命令の実行と並列にデクリメントすることができる．

(5) ファイルアドレスレジスタ(FAR0～1)

レジスタファイルのアドレスレジスタである．

(6) ファイルインダイレクトレジスタ(FIR0～1)

それぞれFAR0, 1のさすレジスタファイル中の語が対応し，ハードウエアとしては存在しない．この間接指定方式はエミュレーションにおいて，ターゲットマシンのレジスタイメージをレジスタファイル中に取るときなどに有効である．

(7) ファイルネクストレジスタ

FAR0, 1の内容に1を加えたアドレスのレジスタファイル中の語が対応する．FIRとFNRを用いるとFARの変更なしにレジスタファイル中の連続した2語を取扱えるので，1語の長さが32, 64ビットなどのシステムアーキテクチャのエミュレーションに便利である．なお，MDRのSMビットをセットすると**3.1.6**に述べるようにFNR0はスタック動作となり，上記とはやや異なる動作をする．

(8) インタフェースレジスタ(IFR0～1)

μPU外部とのインタフェース用レジスタであって，その入出力は両方向バス形式でインタフェース端子に出ている．この端子を通しての入出力は外部から制御

表3　IFR0ビット転送制御信号

信号値 (1)	(0)	意味 1	0
0	0	hold	hold
0	1	set	hold
1	0	hold	reset
1	1	set	reset

され，内部のマイクロ命令では行われない．又，IFR0は表3に示した2本の制御線を用いて外部からビット単位でもセット／リセットできるようになっており，システムによってはこれを外部状態レジスタとして用いることもできる．

3.1.2　レジスタファイル　16語のレジスタファイルが2組用意されているが，一時期に直接指定でアクセスできるのはMDRのRFビットで定まるいずれか一方のファイルのみである．

(1)　レジスタファイル（RF0）

ターゲットマシンのレジスタ群などの用途を想定したはん用レジスタ群であるが，MDRのSM＝1のときはスタックの上部の記憶として動作する．

(2)　レジスタファイル1（RF1）

各8語のサブファイル1－0と1－1に分かれている．1－1の最初の1語を除く7語はマスクレジスタとして利用できる．

3.1.3　算術論理演算ユニット（ALU）　2進並列加算器，AND，IOR，XOR，NOR の論理回路，それに1けたの10進加減算器などからなる．ここで10進加減算器を1けたとしたのは，その必要度，使いやすさ（10進演算は主メモリ中の情報に対して行われるのが一般的であるから，そのフィールドには符号が混在することもあり，4けたを一度に取扱うのは必ずしも好都合ではない）などを考慮した結果である．

3.1.4　マスキング　マイクロ命令のMKフィールドで指定されたときは，L，Rバスのデータは ALU に入力される前に指定のマスクレジスタと論理積が取られる．又，出力もマスクレジスタと論理積が取られてからTバスに置かれる．これによってレジスタ中の部分フィールドの抽出，処理を効率よく行うことができる．

3.1.5　シフタ

(1)　単語長マルチビットシフタ

単語長データに対し1~15ビットを1マシンサイクルでシフトするシフタを用意した．一般に，マルチビットシフタはハードウエア量が増加するので，LSI向きではないが，これとマスクレジスタを組合せると，効率よいフィールド抽出が行えるうえ，一般的有効性も大きいので，導入する価値は十分あると判断した．

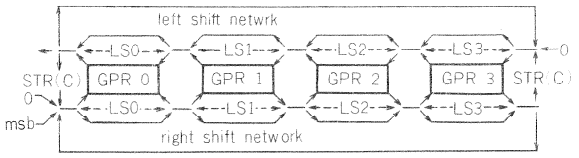
(2)　1ビットロングシフタ

はん用レジスタGPR0~3には1ビットのロングシフタが付属しており，論理的には図2に示すような関係で，32，48，64ビット長の1ビットシフトが行え，乗除算などに有効である．

3.1.6　スタック操作　独立したスタックを持つことはハードウエア量の制限上許されないので，レジスタファイル0にスタックの上部を常に残せるような特殊なハードウエアサポート（リングスタックと呼ぶ）を用意した．すなわち，MDRの SM＝1のときは，RF0はFNR0　を用いて次のようにアクセスされる．

①　命令のTフィールドにFNR0を指定すると，その実行の初めにFAR0に1が加えられる．そして命令実行終了時にMDRのSEビットをリセットし，更にこのときFAR0＝15（MDRのSB＝0のとき）／7（SB＝1のとき）であれば，スタックフルを示すSTR中のFビットをセットする．

②　RフィールドにFNR0が指定されたときは，実行の初めにSE＝0であれば 実行終了時にFAR0から1がひかれ，FAR0＝15（MDRのSB＝0 のとき）／7（SB＝1のとき）であればスタック空を示すSTRのEビットがセットされる．但し，初めにSE＝1であればEビットがセットされ，この命令は無効



LS1: long shift control for GPR1

図2　ロングシフタの論理構造

Fig.2－Logical structure of the long shifter.

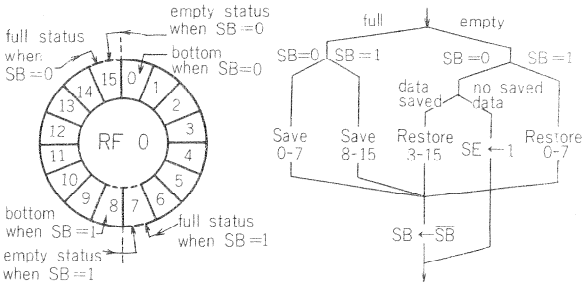


図3　リングスタック機構の論理構造

Fig.3－Logical structure of the ring stack mechanism.

になる．

容易に分かるように，上記の操作で①はスタックの“プッシュ”に，②は“ポップ”に対応する．

ＳＥはスタックが完全に空のときマイクロプログラムでセットしておくビットであり，これによりスタックのアンダフローが検出される．又，ＳＢはスタックの底を示し，図３のようにＳＴＲのＥ，Ｆビットのセット条件を定めるから，Ｆがセットされたときは８語を外部記憶に退避し，Ｅがセットされたときは退避してある情報を８語だけＲＦ０にもどせば，スタック上部が常にＲＦ０中に残ることになる．

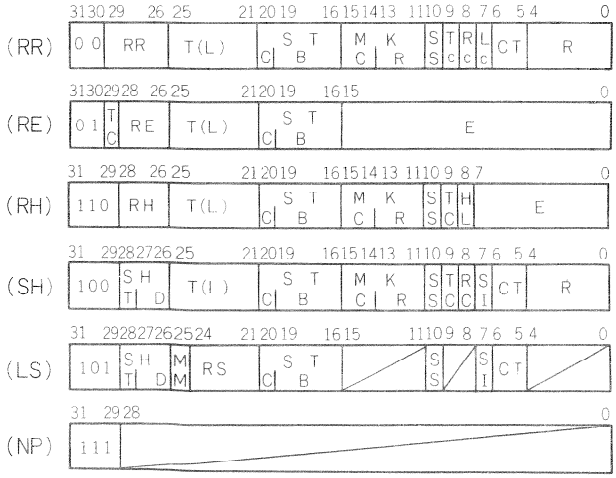
3.2 マイクロ命令

マイクロ命令は使いやすく，且つ，細かい制御ができるように配慮して設計した．図４に示した６個の形式がある．

3.2.1 RR形命令 レジスタとレジスタ，又はレジスタと５ビットのリテラル間の操作であり，全命令の基本形である．

(1) 操作（ＲＲ）

論理演算４種，２進加減算６種，10進加減算４種，それにＡＤＣとＳＢＣと呼ばれる特殊操作２種を加え，合計１６種の操作を指定する．ＡＤＣとＳＢＣは操作はそれぞれ２進の加減算であるが，演算結果をレジスタにセットすることと，状態ビットの変更（ＳＢＣのみ）がキャリー（Ｃ）ビットによって定まる点が特殊である．それぞれ，普通は乗除算ステップに用いることを意図したものである．



RR:Op.code, T(L):T & L bus, ST:Status, control, MK:Mask control, SS:Status save, TC:T bus control, RC:R bus control, LC:L bus control, CT:Counter, R:R bus, RE:Op.code, E:Emit, RH:Op.code, HL:High-low control, SHT:Shift type, SHD:Shift direction, SI:Shift-in control, MM:Multimode, RS:Register select.

図４ マイクロ命令の形式
Fig.4 − Microinstruction formats.

(2) 状態指定（ＳＴ）

ＳＴＢでＳＴＲ中の１ビットを指定し，その０，１（ＳＴＣによる）によって，次の命令を自動的に無効命令にする．いわゆるスキップ操作である．

(3) マスク制御（ＭＫ）

3.1.4に既に述べた．ＭＫＲでマスクレジスタを選択し，ＭＫＣで適用するバス（Ｌ，Ｒ，Ｔのいずれか，又は全部）を選択する．

(4) 状態格納（ＳＳ）

命令実行結果による状態情報をＳＴＲにセーブするかどうかを指定する．この制御によって過去の状態情報をＳＴＲに残しておくことができるようになり，制御の柔軟性が増している．

(5) バス制御（ＴＣ，ＬＣ，ＲＣ）

ＴＣはＴバス上データのレジスタへのセット指定，ＲＣはＲフィールドをリテラルとするか，その番号のレジスタと解釈するかの指定，ＬＣはＬバスに指定のレジスタを接続するか，“0”を入力するかの指定であって，これによりマイクロ命令の力が増強される．

(6) カウンタ制御（ＣＴ）

２個のカウンタのいずれか一方，または両者のデクリメントを指定するフィールドで，主にループ制御に使われる．

3.2.2 RE形命令 レジスタと１６ビットのリテラル間の操作である．命令の下位１６ビット（Ｅ）はリテラルと解釈され，Ｒバスに接続されるのでＲＲ形命令の下位側に割当られた制御はきかない．この形の命令はレジスタへの定数のセット，レジスタ中のビットのテストなどに有効である．操作の種類は８種で，４種の論理演算と２進加減算，バス（リテラル（Ｅ）をそのままTバスへ出す）のほかにパターンマッチと呼ぶ特殊命令がある．これはＬ入力の下位４ビットをデコードした１６ビットの情報とリテラル（Ｅ）の論理積をとってＴバスに出力する命令であって，コードの分類，１０進演算の符号の検出などに便利である．

3.2.3 RH形命令 レジスタと８ビットのリテラル間の操作である．操作の種類はＲＥ形のバスとパターンマッチを除いた６種である．カウンタとＬ，Ｒバスの制御はできないが，ＨＬフィールドでリテラルをＲバスの上位側におくか，下位側におくかを指定できる．この命令ではマスクを用いることができるので，

文字ストリングなど8ビットデータの取扱いに有効に利用できる．

3.2.4　SH形命令　Tフィールドで指定したレジスタの内容をR入力が指定する量（最大15）だけシフトする命令である．この命令ではLバス制御はなく，代りに空いた場所に0を入れるか，STRのCビットを入れるかの制御（SI）が用いられる．シフト形式は算術，論理，循環と通常どおりであるが，他にソースレジスタとしてGPR0をインプライドに指定する特殊論理シフトがある．これはレジスタ中に記憶された部分フィールドを逐次処理するとき，基のレジスタの内容が破壊されないようにしたもので，有効な場合がしばしばある．

3.2.5　LS形命令　GPR0～3に対するロングシフト命令で，その機能は**3.1.5**に既に述べた．なお，この命令に限りMMフィールドの指定によりカウンタが0になるか，スキップ条件が成立するまで同一命令を繰返させることができる．

3.2.6　NP形命令　上位3ビットが“111”の命令は内部的には無効命令である．すなわち，NP形はシーケンス制御と外部制御命令としてシステムごとに設定され，外部で制御される．

4．評　価

これまで述べてきたμPUアーキテクチャにつき，半導体技術と応用側から簡単な評価を試み，その妥当性を検討した．この章ではその主な結果を報告する．

4.1　LSI製作のための検討

まず，このアーキテクチャがあらかじめ設定された半導体技術目標（表1）に適合しているかどうかを調べたが，その結果，レジスタファイルを4ゲート／ビット，その他のレジスタを6ゲート／ビットとすると丁度5,000ゲート，チップは7mm角程度になり，実現可能であるという結論に達した．又，マシンサイクルは200ns前後と推定された．この値は現在のミニコンピュータなどのそれとほぼ等しく，新しいプロセッサとしては多少遅いようにも思われるが，1マシンサイクルに行われる機能からすれば，十分高性能といってよいであろう．

4.2　アーキテクチャの評価

アーキテクチャの良悪を判定することは極めて難しい．このμPUのようにはん用に設計されたものはどんなに優れたものでも，専用に設計されたものに比して，その用途に限っては劣る結果となる．ここで報告したアーキテクチャでもそれは成り立つが，与えられた条件の下でのはん用プロセッサとしては最適でないにしても，一応満足できるものと考えている．以下に2，3の評価結果を示す．

4.2.1　スタック機能　図5は，“プッシュ”又は“ポップ”が連続して出される回数を乱数で発生させ，それにより起こるハードウエアスタックと外部記憶とのデータ転送回数を，リングスタックのある場合と，ない場合につきシミュレーションによって調べた結果[3]である．但し，リングスタックの1回の入出力量は8語であるのに対し，通常のスタックでは，16語であるため，図では，リングスタックの転送回数を2倍に換算してある．この結果から“プッシュ”又は“ポップ”が連続して出される回数が，およそスタックの語数（ここでは16）より少ないとき，リングスタック機構は，非常に効果のあることが分かる．

4.2.2　フィールド処理能力　フィールド処理能力を調べるため，図6(a)，(b)に示したような1語中のデータフィールドを再配置する2つの問題（C. Churchが用いたもの[6]）を，このμPU（シーケンス制御部は標準的なものを仮定した）とHP2100，それにフェーズ0アーキテクチャ（**5.**参照）の3つのマイクロ命令でコーディングしてみた．その結果，このμPUではHP2100に比べ，プログラムステップ数で約1/2，実行ステップ数で1/3～1/4　になっており，フィールド処理能力が優れていることが分かる．これは，強力なシフト能力と使いやすいマスク機能によるところが大きい．

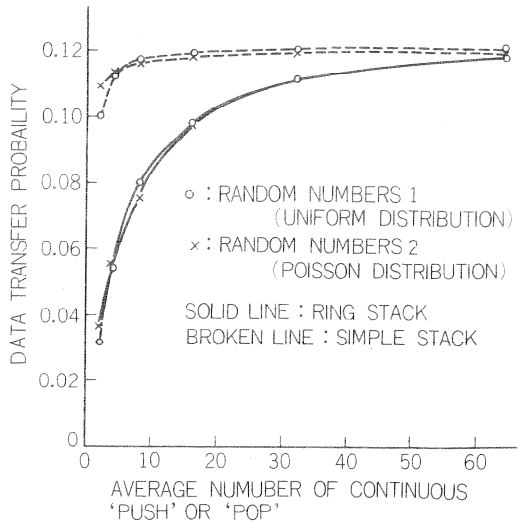
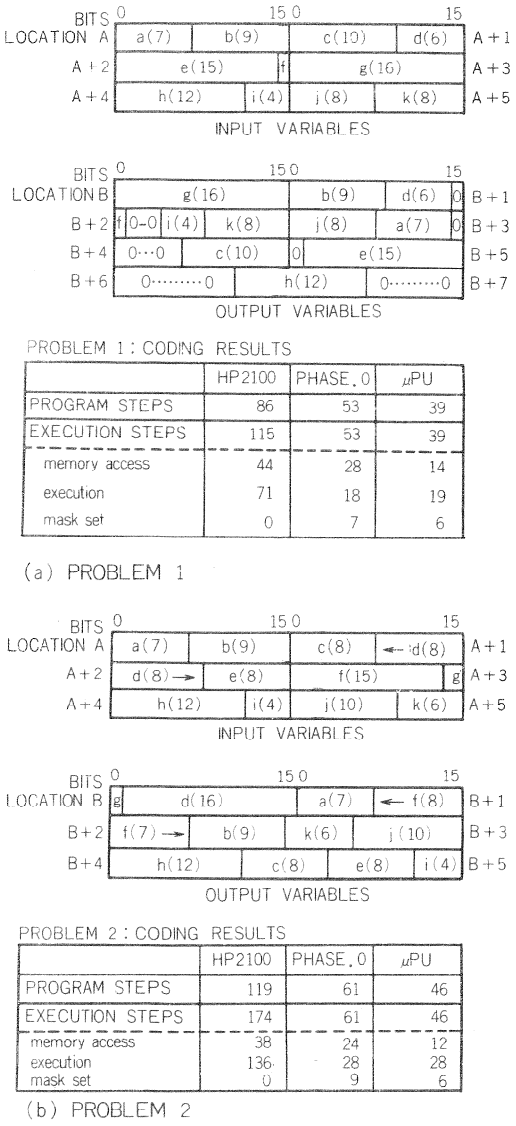


図5　リングスタックの性能評価
Fig.5 – Performance evaluation of ring stack.



PROBLEM 1 : CODING RESULTS

	HP2100	PHASE.0	μPU
PROGRAM STEPS	86	53	39
EXECUTION STEPS	115	53	39
memory access	44	28	14
execution	71	18	19
mask set	0	7	6

(a) PROBLEM 1

PROBLEM 2 : CODING RESULTS

	HP2100	PHASE.0	μPU
PROGRAM STEPS	119	61	46
EXECUTION STEPS	174	61	46
memory access	38	24	12
execution	136	28	28
mask set	0	9	6

(b) PROBLEM 2

図6　データフィールドの再配置
Fig. 6 — Variable length field reformatting.

4.2.3　算術式インタプリンタ　スタック操作を含むと一般的問題として，加減乗除とかっこを含む算術式のインタプリタをマイクロプログラムで実現してみた．これは，主記憶に記憶されたインフィックス形式の算術式を，先頭から1語（16ビット）ずつ処理し，"="がきたとき結果を主記憶に返すものである．ここでオペランドは，16ビットの符号付整数であり，オペレータとの区別は，各オペレータとオペランドに1語の識別記号を用いている．表4は，このインタプリタで2つの例題を行ったときの結果であり，このμPUは，HP2100 に比べ，プログラムステップ数で約1/2，実行ステップ数で約 3/4 になっている．HP 2100が乗除算用ハードウエアを持っているにもかかわらず，

表4　算術式インタプリタによる性能評価

		HP2100	PHASE.0	μPU
プログラムステップ数	式操作部	148	49	53
	演算部	50	40	54
	計	198	89	107
実行ステップ数（例題1）	式操作部	348	152	172
	演算部	76	66	138
$a+b*c-d/e=$	計	424	218	310
実行ステップ数（例題2）	式操作部	542	287	305
	演算部	82	69	143
$((a+b)*c-d)/(e-f)=$	計	624	356	448

表5　Nova エミュレータコーディングステップ数

	μPU	HP2100
初期設定	5	0
命令フェッチ	4	3
プリデコード	11	16
実効アドレス	26	38
シーケンス制御命令	16	19
ロード／ストア命令	10	13+16*
乗除算命令	26	29
算術，論理演算命令	74	121
合　計	173	255

＊　算術，論理演算命令でも使われるサブルーチン

このような差が出る原因は，μPUの強力なフィールド処理能力，エミットデータとの演算能力，スタック機能などによる．又，HP2100 では，レジスタの数が少ないので，1つのレジスタを多目的に使う結果，マイクロプログラムの誤りを起こしやすく，且つデバッグが容易でない．このμPUのレジスタファイルは，この点でも非常に有効であった．

4.2.4　ミニコンピュータのエミュレータ　ミニコンピュータNovaの入出力命令を除く基本命令セット（乗除算は含む）のエミュレータをコーディングしたところ，表5に示すような結果が得られた．フェーズ0の場合もほとんど同じ結果が得られているが，このμPUとHP 2100 で差がつくのはほとんど命令をデコードするフィールド処理の部分であって，フィールド処理能力の重要性がよく分かる．実行速度はこのμPUでもNova 自身よりかなり落ちそうであるが，これはNovaの機械命令自身がかなりマイクロ命令的で機能が小さく，そのデコードステップが実行ステップに対し相対的に大きくなることによっている．外部にデコード用ハードウエアを付加することによって救える部分もあるが，はん用プロセッサの宿命のようなものでもあり，このような用途はこの種のμPUのねらうべき応用でないということでもある．又，HP2100

ではフィールド処理能力が弱体のため，この場合の更に何分の1かの速度になると考えられ，実用にはならないであろう．

5．むすび

以上，新しい高性能マイクロプロセッサとその評価につき報告した．このμPUは50年中に通常IC によるプロトタイプを製作し，ACEシステム[4],[8]のプロセッサモジュール[5]などに利用するつもりであるが，パターン情報処理プロジェクトでは昭和51年末までにLSIサンプル製作を行い，52年度以降相当量を研究に使用することを予定している．先にも述べたように我々はその時点でのはん用マイクロプロセッサとして一応満足できるアーキテクチャであると考えている．しかしながら，最終的LSIとなる前に，サンプル製作時点で機能と必要度の見直しは当然行われなければならない．

最後に，このμPUアーキテクチャの研究，開発過程につき一言付け加えておきたい．研究は昭和48年初頭に電総研を中心に大学，関連メーカの研究者を集めたワーキンググループ（以下WG）を設置し，設計のための調査，検討[7]を行うことから出発した．そして筆者ら電総研の研究者が最初のプロトタイプアーキテクチャ（以下フェーズ0*）を設計しWGに示した．本論で報告したものはWGメンバーの意見を参考に筆者が新たに設計しなおしたものである．従って，ここに盛込まれたアイデアの中にはWGメンバーの示唆によるものも含まれている．特に，箱崎勝也氏と田丸啓吉氏の意見は参考になったことを記し，謝意を表したい．

又，御討論，御協力頂いたWGのメンバー諸氏と，電総研の石井治，藤井狷介，大表良一の各氏，アーキテクチャの評価に協力頂いた坂村健氏（慶大），ならびに本研究の遂行に御助力，御指導頂いた西野博二，黒川一夫両部長，元岡達教授（東大），相磯秀夫教授（慶大）を初めとする関係諸氏に感謝する．

* このアーキテクチャ[3]は機能中心に設計したもので，約8,000ゲートになっている．これはACEシステムのプロセッサモジュールで使われている．

文　献

(1) 飯塚：“高性能マイクロプロセッサのアーキテクチャ設計”，信学会計算機研資，EC75-3（1975-04）．
(2) D. P. Siewiorek, et al.: “Some observations on modular design technology and the use of microprogramming”, CMU report (July 1974).
(3) 飯塚，ほか：“ACEマイクロプロセッサユニットのアーキテクチャ”，情処学会アーキテクチャ研究会資料74-3（1974-10）．
(4) 飯塚，ほか：“モジュール形複合計算機ACE”，同上，74-4．
(5) 飯塚：“ACEプロセッサモジュールのアーキテクチャ”，同上，74-5．
(6) C. C. Church: “Computer instruction repertoire - Time for change”, Proc, SJCC., p.343 (1970).
(7) 石井，ほか：“マイクロプロセッサー設計のための予備的検討”，PIPS-R-No.3 電総研（1973-10).
(8) H. Iizuka, et al.: “ACE-A new modular computer architecture”, Proc. 2nd US-J Comp. Conf., p.36 (Aug. 1975).

（昭和50年6月5日受付）