モジュール型複合計算機 ＡＣＥ

飯塚 肇　大表良一　藤井猶介　石井 治　古谷立美
（電子技術総合研究所）

1. はしがき

計算機アーキテクチャはその時点でのハードウェア技術やコスト、それに応用側の要求に適合したものでなければならないが、現在の最も顕著な傾向はＬＳＩに代表されるハードウェアロジックやメモリの低価格化と応用の複雑，多様化，ソフトウェアコストの増大であることは論をまたないであろう。一方、汎用の単一プロセッサのアーキテクチャにおいては主なアイデアは出つくした感があり、今後のアーキテクチャとしてはあまり期待できない。

こうした認識のもとに、数年来、ポリプロセッサ、ミニコン網、複合計算機等という名称で、複数のプロセッサを含む計算機構造がアメリカの大学を中心に研究されてきたが[1]、特殊の応用に制限した小規模な場合を除いて、今のところ成功に至ったシステムは殆んどない。どのシステムでもプロセッサ等、システム中のハードウェアリソース相互の関係は通常固定であるため、これらを各応用において有効に使うことはソフトウェアに大きく依存し、一般性のあるシステムをつくり得ないことがその理由のひとつと考えられる。複合計算機は確かに現在の技術や要請によくマッチしたアーキテクチャではあるが、真に有効な計算機構造として利用されるためには、この問題が解決され、応用への高い適応性を身につけねばならない。

筆者らはこのような考察のもとに、ＬＳＩ等の新しいハードウェア技術を利用した適応性のある複合計算機構造、ならびにプロセッサ等のその構成要素のアーキテクチャについて、実験研究を行なってきた。本論文ではその考え方の概要と応用に適したＰＭＳ構造を得るための新しい要素間結合方式について説明し、こうした実験システム向きな機能を持たせたマイクロアセンブラ方式についてふれる。

2 ＡＣＥの概要

ＡＣＥ［Adaptive（Advanced）Computer Experiment（of ETL）］システムは1章に述べた考察のもとに、新らしく設計した本格的複合計算機の実験であって、プロセッサ本体を始め、全て新らたに設計し、試作を進めている。

ＡＣＥにおいて、適応機能を与えるために採用した代表的技術は以下の3種である。

① モジュール構成を採用し、モジュール間通信に一般性のある新しい方式を採用した。
② プロセッサにダイナミックマイクロプログラミングを採用し、エミュレーション向きアーキテクチャとした。
③ デスクリプタ方式をマイクロプログラムレベルで採用した。

注　本研究はパターン情報システムプロジェクトの一部として行なわれたが、本論文中の見解は全て筆者のものであって、プロジェクトの公式見解を示すものではない。

この内、②，③についてはプロセッサモジュールのアーキテクチャに関する別の論文[5]で詳しく報告するので、ここでは主に①を中心にＡＣＥの概要を紹介する。

2.1　モジュール型複合計算機

複合計算機の構成上の分類の中にモジュール型と呼ぶことができるものがある。[3] これは相互に論理的にも、物理的にも融通性のある結合が可能なように設計したプロセッサモジュールや記憶モジュールやＩ/Ｏモジュール等、いわゆるＰＭＳレベルのモジュールを用意し、それらを適当に結合して、それぞれの応用に適したシステムを構成しようというものである。

モジュール構成はレジスタ，転送ゲート等のいわゆるＲＴレベルでは古くから実験研究が行なわれ、これで専用マシンを組立てれば場合によっては大型機なみの性能を得られることが報告されているが[7]、最近のＬＳＩ技術の発展からすればＰＭＳレベルにモジュールのレベルをあげた複合計算機を考えることは全く正当なものといえる。

さてＰＭＳレベルのモジュールの研究は比較的新らしい。ミニコンピュータのＰＤＰ－１１やＳＵＥに或程度こうした考えが取り入れられているようにも見受けられるが、複合計算機にまでは至っていない。実際的立場で、モジュール間の通信を一般的に検討したのは現在のところカーネギメロン大学のＳ．Ｈ．Ｆｕｌｌｅｒ等が行なっているコンピュータモジュール[6]ただひとつで、通信方式等に参考になる点はあるが、今のところ実現は試みられていない。

2.2　モジュール間結合方式

モジュール型複合計算機ではモジュール間の論理的結合の融通性、一般性が最も重要な点である。ＡＣＥでは全てのモジュールの結合をひとつの論理的にも物理的にも統一したインタフェースのバス（Ｃ－バスと呼ぶ）で行なっている。ここで'ひとつの'というのは同一仕様のという意味であって、必ずしも、'一本のバスに'という意味ではなく、システム中に複数本のＣ－バスが存在する方がむしろ標準的である。バス方式を採用したのはひとつのバスに接続するモジュール数に対する論理的制限がほとんどなく、融通性が極めて高い方式だからである。また通信バスを通常行なわれているように、メモリバス，プロセッサ間バスのようにそれぞれの用途に合わせて、別々に用意することをせずに、一種に統一したのもモジュール型複合計算機として最も重視しなければならない柔軟性のためである。これによって、通信のためのハードウエアの若干の増加が生じるが、融通性が著しく増し、そのメリットはコスト上の欠点を補い得るものである。

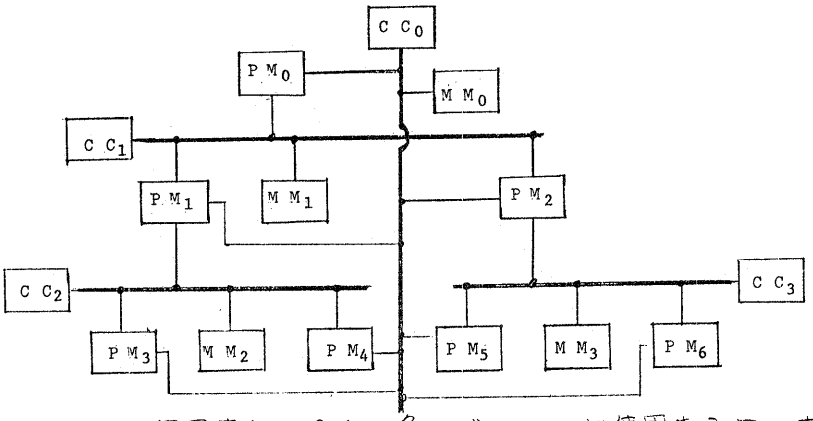
現在試作中の標準プロセッサモジュール[5]はＣ－バスとの接続口（ポート）が最大４個（実装２個）用意されており、その融通性を利用し、アレイ接続やハイアラーキ接続（図１）をとったり、また、メモリ用バスとプロセッサ間通信バスとを分離したりすることができるし、一本のバスでの転送能力が不足ならマルチバス構成によって転送バンド巾を高めることもでき、多種多様の結合方式を状況にあわせて採用し得るようになっている。

Ｃ－バスの実際的検討やその設計については３章に詳述するが、ここでＡＣＥの設計思想を反映している論理的特徴を説明しておく。

(1)　グローバルアドレスの採用：モジュール間の通信にはモジュールの物理番号（アドレス）を用いず、そのＣ－バス上での通信にのみ共通なアドレスである

図1　ハイアラーキ接続の例

PM：プロセッサモジュール
MM：メモリモジュール
CC：C-バスコントローラ



グローバルアドレス（ＧＡ）を使用する。ＧＡは各モジュールが使用するローカルアドレス（ＬＡ）とは全く独立であって、Ｃ－バス上に送り出される前にＬＡから変換され、受信側では必要に応じてこれを自分のＬＡに再変換し、通信路を確立する。送信側から相手のモジュール番号は指定されないので、受信側がＧＡを監視し、自分が通信の対象として関与すべきアドレスであれば応答する。つまり、ＧＡは実はアドレス（その形成されるプロセスから便宜上アドレスと呼ぶ）ではなく情報の一種の名称であるから、その情報は認識するＧＡとＬＡの関係が確定している限り、どのモジュールのどこの記憶にあってもよいわけで、通信の自由度が著しく高まっている。モジュールアドレスを指定する普通の通信方式を電話とすれば、通信関与の決定権を受信側に与えたこの方式は送信側の電波に自分でダイアルを同調させる無線通信に対応する。

(2)　1対多の通信：この通信方式では1対多の転送を簡単に実現できる。すなわち、あるＧＡは全てのモジュールが認識するように認識アドレスを調整しておけばそのアドレスを用いて、全てのモジュールへブロードキャスト転送ができる。もちろん相手はＣ－バス上の全モジュールではなく、そのサブセットであってもよい（図２）。

(3)　グローバビリティ：各モジュールがそのポートで認識するＧＡにはそれぞれグローバビリティ（ＧＢ）という２ビットのデータが付属している。送信側のモジュールは要求の際、ＧＡの他にＧＢを転送するので、ＧＡが一致したモジュールは自分のＧＢと転送されたものを比較し、等しいか自分の方が小さいときだけ通信に関与する。このグローバビリティによって、同一のＣ－バスに接続されているモジュール内でもハイアラーキーをつけることができ、通信の融通性は更に高まる。

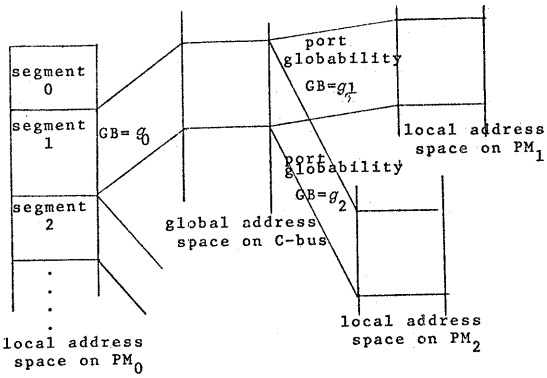


図２　アドレス変換の例

2.3　モジュールの種類

Ｃ－バスに接続されるモジュールは現在のところ、プロセッサモジュールとメモリモジュール、入出力モジュールを考えているが、インタフェースさえととの

っていればどのようなモジュールでも接続できるわけである。メモリモジュールは要求を発生しないで、常に受身であり、GAの認識ロジックと、ブロック転送に対する用意（後述）が必要な以外、特に一般の場合と異なるところはない。入出力モジュールはさしあたりは市販のミニコンピュータにC-バス接続アダプタをつけて用いる予定である。プロセッサモジュールはこの研究の中心部分であり、ACEの思想に適した標準プロセッサモジュールを新らしく設計した。[5]

2.4 ACE 0.1

現在試作中の最初のACEシステム[8]は図3のような構成であって、プロセッサモジュールは一台だけでI/Oプロセッサはあるが、まだ完全な複合計算機にはなっていない。各部分の概略仕様は次の通りである。

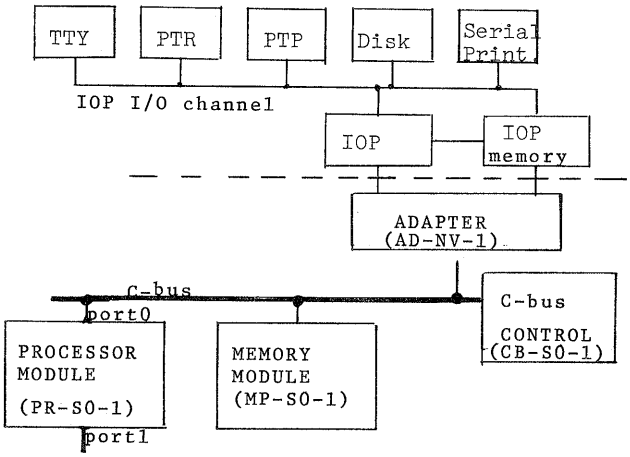


図3 ACE 0.1の構成

(1) プロセッサモジュール：アーキテクチャは別論文[5]参照。マシンサイクル約250ns、TTLのMSI／SSIで構成されたマイクロプログラム制御のプロセッサ。

(2) メモリモジュール：容量2^{19}ビットのICメモリで、サイクル時間は約800ns、32ビットのデータ16個までを5MHzで転送できる。C-バスポートは1個（2まで拡張可）。GAの上位9ビットが全て'0'のアドレスを認識し、ポートグローバビリティは0である。

(3) I/Oモジュール：ミニコンピュータ（Nova 800）にアダプタをつけて、C-バスへ接続。認識アドレスはGAの上位9ビットが'0……01'の固定とする。

(4) C-バスコントローラ：C-バスの制御を集中的に行なう。C-バス一本ごとに必要。詳細は次章参照。

3 C-バス制御方式

システムを構成する各モジュール間の結合方式と通信方法に対する検討は、各モジュールの機能の検討と同様に重要な課題である。論理的な結合、通信方法は前章で述べたとおりであり、物理的な結合、通信方式の分類、検討は文献2に述べてある。ここではそれらを基礎にACEシステムのモジュール間の物理的な結合、通信方式を検討し設計を行なった概略を報告する。

結合要求の発生率、データ転送時間の分布、データ転送速度等をパラメータとしたトラフィック解析（又はシミュレーション）は充分でないので、こうした解析のための基礎データを得ることも考慮した。ACEシステムは実験システムであり、モジュールの結合の自由度、拡張性は重要なので、物理的な結合、通信方式もこの点を配慮したことはすでに述べたとおりである。

3.1 C-バス

モジュール間の結合方式には表1に示したように、種々の方式が考案されてお

り、一長一短があるが、ＡＣＥの設計思想からすれば、環状バス方式か単一バス方式がＣ－バスに向いているようである。両者の相異点は遅延時間と拡張性にあるが、遅延時間の差を重視して、単一バス方式を採用した。ＡＣＥ全体としての特性は、どのような結合形式をとるかに左右されるが、バンドや、拡張性、フェイルソフト特性に若干の特性改善が期待される。

結合方式 / 特性	単一バス	多重バス	専用バス	専用バス・カプラ	環状バス	マルチポート	マトリクススイッチ	ACE
バンドや	△	○	◎	○	△	◎	◎	○
オーバヘッド	△	△	○	△⁻	△⁺	○	○	△
遅延時間	○⁺	○	○⁺	△	△	○⁺	○	○⁺
フェイルソフト	×	○	○	△	×	◎	△	○
拡張性	○	○	○	○	◎	△⁻	△⁺	○⁺
結合自由度	◎	◎	△	◎	◎	△⁺	△	◎
初期投資	◎	○	○	○	◎	△⁺	△	△
ケーブル数	1	n	P	P	1	P×m	P+m	n

表１　各種結合方式の比較

P：プロセッサの数，m：メモリの数，n：バスの多重度

また、特にどれかの特性を重視し、それを最大限に改善するような結合形式をとることも、ポートごとの多重化により可能であることは言うまでもない。初期投資、制御ロジック、ケーブル数の面での不利は、結合の柔軟性などの利点から見れば、小さいものと言えよう。

Ｃ－バス上でのデータ転送速度は、マイクロプログラムが転送されるので、可能なかぎり上げる必要があり、5 M words/sec を予定している。

3.2　Ｃ－バスコントローラ

Ｃ－バスの制御は各バスごとに設けたＣ－バスコントローラによる集中制御方式を基本とするが、前述したように通信の相手の決定が各モジュールによって行なわれるので、一部の機能がポートに振分けてある。集中制御方式は信頼性、拡張性に乏しいが、マルチポート化によってかなり改善されるものと思われるし、制御が簡単になる利点を持っている。

結合路の確立方法は各モジュールがバスコントローラとリクエスト線、セレクト線で独立に接続される独立要求方式を採った。この方式は制御線やコネクタが多数必要になる反面、モジュールの要求状況を並列的に判断できるため、結合路の確立に要する時間が短かくなる利点を持つ。リクエストにはプライオリティ情報（２レベル）が付属し、バスの優先使用が可能である。Ｃ－バスコントローラは、各リクエストのプライオリティ、接続位置（同一プライオリティのリクエストがある場合のみ）を調べ最も優先度の高いリクエストを受け付け、セレクト線により、モジュールへ知らせる。リクエストを受付けられたモジュールは、ＧＡ、ＧＢ等の制御情報をデータバスに送出し、通信の相手が決定される。Ｃ－バス上のモジュールは通信に関与するか否かの返答をし通信路が確立される（図４）。

データ転送はブロックのバースト転送を採用し、独立要求方式とあいまって、オーバヘッドの割合を小とする一方、ブロックサイズを可変長としバースト転送方式の欠点である待ち時間が長くなる問題を軽減しデータバスの使用効率を向上させることとした。ブロックサイズは、1～16 ユニットとし、1 ユニットは 16

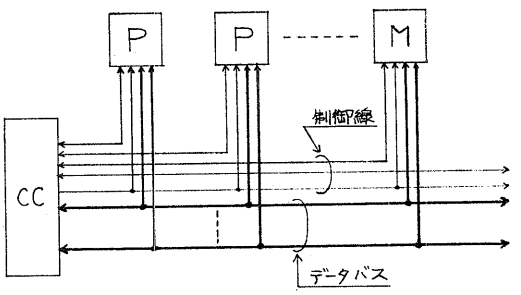


図４

または32ビットでバスを要求するときにモジュールによって指定される。制御線はモジュールとバスコントローラとのスター結線が原則であるが、いもづる式の制御線も存在する（図4）。

以上の制御シーケンスで最も重要な点は、各モジュールの誤動作等によって、一本のC－バスが使用不能になるのをいかにして防止するかにある。リクエストの競合は優先度によって回避できるので良いとして、問題は通信路の確立時の返答が無い場合の対策がオーで、データ転送時に相手の準備が整っていない場合の対策がオニである。前者については、時間により弁別する方法と、返答が無いときは通信に関与しないものと見なしてしまう方法が簡単で良い。ACEでは、モジュールが実装されていなくて返答が無い場合に対して、バスコントローラの入力コネクタに常に返答が'通信に関与しない'となるようにするとともに、故障等の場合には時間で弁別することにした。後者については、バスの使用効率を低下させないように、転送シーケンスを一担ホールドし、準備が整いしだい再開する方式を採用した。転送シーケンスの再開はホールドの返答をしたモジュールがコンティニュ信号を送出することでなされる。また再開に必要な情報（リクエストを送出したモジュールの物理番号）はバスコントローラ中に設けてあるキューに貯えられる。キューは各モジュールごとに1レベルで、具体的には3ビットのレジスタと二重書込み防止用のフリップフロップからなる。キューが1レベルなので同一モジュールが二度続けてホールドすることは許されず、二度目のリクエストはなかったものとされる。

現在試作中のC－バスコントローラは8台までのモジュールを制御できるように設計されている。

4　汎用マイクロアセンブラ（EGMA）

4.1　EGMAの設計思想

EGMAは、マイクロプログラムを記述するアセンブラ型言語のうち、フィールドセンシティブなものを対象とし、この種の言語なら目的計算機に依存しないでアセンブルすることが可能な汎用クロスマイクロアセンブラである。[9] 汎用性を持つマイクロアセンブラを作成することにした目的は、

(1) 今後開発する、または普及すると予想される異なったアーキテクチャを持つ種々のマイクロプロセッサに対し役立てたい。
(2) 特に、ACEの設計、製作とは独立に作成し、ACEの方式設計の段階においてマイクロ命令セットを決定する時など一助として利用したい。
(3) ACEの調整時に、テストプログラムをアセンブラ言語で記述したい。
(4) マイクロアセンブラの作成と使用経験に基づき、拡張と補充、中級または高級言語の設計など次へのステップとしたい。

と言った要求を満たすことであった。

これらの要求を満たすため、EGMAに次の機能を持たせることにした。

(1) ACEの設計段階における変更に対しアセンブラを柔軟な構造にし、アーキテクチャの相違を吸収できるようにし、汎用性を持たせる。
(2) HP2100Aの使用経験から、単純なマクロ機能を持たせる。
(3) 汎用性とマクロによるアセンブラの複雑化を機能の単純化で避ける。

通常のアセンブラにおいても汎用化の試みが行なわれているが、[10] いずれも機械のアーキテクチャなどの相違を反映することが困難で複雑なものとなっている。

しかし、ＥＧＭＡでは次の手段によって汎用性を持たせ得ると判断した。

(1) 各フィールドごとに記憶用マイクロコードを書く記述形式に限れば、記述上の形式とマイクロ命令との対応づけを容易に行なえる。

(2) 複雑なデータ形式を記述することがないため、データ定義のアセンブリ命令を省略でき、アセンブラを簡略化できる。

(3) マイクロプログラムは構造の規則性が大きく、あまり大規模なプログラムをアセンブルしないので、アセンブラを簡略化しても十分に役立つ。

(4) 共通化困難な部分はユーザの助力を求めるかプリプロセッサにより処理する。

4.2 ＥＧＭＡの特徴

ＥＧＭＡの特徴は、ＦＯＲＭＡＴおよびＣＯＤＥと呼ばれる二つのコントロール命令によって達成される汎用性である。ＦＯＲＭＡＴとＣＯＤＥの形式を図５に示す。ＦＯＲＭＡＴはマイクロ命令の形式ごとに一組づつ与えるもので、カード上の記述形式とマイクロ命令形式の対応づけを定義する。また、ＦＯＲＭＡＴは各フィールドで許される記憶用コードのセットやアドレス参照フィールドなどフィールドの性格も指定する。ＣＯＤＥは、ＦＯＲＭＡＴで指定した記憶用コードのセットを定義し、マイクロコードとの対応づけをも定義する。コントロール命令はこの２種のみであり、記述上の形式とマイクロ命令形式との対応づけを容易に行なえるようになっている。

```
.FORMAT. j,f,w
..i(a:b)(c:d,e:d,f:g)(#)...
 i:field number
 j:format number
 f:fields
 w:micro word length
 a:b : 開始および終了カラム番号
 c:d : 開始および終了ビット番号
 #:code table number
.CODE. #,b,c
...code(i).....
 #:code table number
 b:フィールドのビット数
 c:記憶用コードの文字数
 code(i):mnemonic
```

図５ コントロール命令の形式

またＥＧＭＡでは、表２に示すようにアセンブリ命令を６種に限定し、アセンブリ命令のシンタックスの定義およびセマンテックスの記述など複雑化する点を避け、簡略化を達成している。さらにマクロ機能も単純な置換型マクロを導入しマクロのネスティング、記号パラメータの結合など全く行なっていない（図６）。

STRT	アセンブルの開始を指示。
END	アセンブルの終了を指示。
EQU	定義するラベルに値を与える。
DC	16進数の定義。
LIST	リステイングのオン/オフを指示。
EJCT	ページの切りかえを指示。

表２ アセンブリ命令の種類

4.3 ＥＧＭＡの処理手順

ＥＧＭＡに入力する原始プログラムの構成は、コントロール命令、マクロ定義、マイクロプログラムの順になっていなければならない。最初にコントロール命令が処理されコア内にＦＭＴ、ＦＩＴ、ＣＤＴと呼ぶテーブルを作る（図７）。次にマクロ定義があれば処理されコア内にライブラリを作る。最後にマイクロプログラムが処理される。つまり、マイクロ命令なら８０欄の形式番号から上記の各テーブルを引きオブジェクトを作る。マクロコールなら展開した後で通常の処理をする。オブジェクトは紙テープに、

```
        MACD
&0      M-name .....&8,&9
   microinstructions
        MEND
```

図６ マクロ定義形式

りスティングはLPに出力される(図8)。

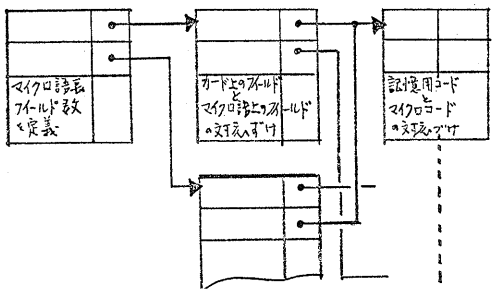


図7 FMT、FIT、CDTの構成

4.4 結論と展望

EGMAはHP2100AとACEのために利用されており、ともにその効果を現わしている。ACEは現在の方式に至るまでアーキテクチャの大きな変更が一度行なわれた。その後も設計上の理由からアセンブラの構造に影響を与えるような変更が2度ばかり行なわれている。いづれの場合にもEGMAの特徴により、コントロール命令を書きかえるだけですんでおり、アセンブラ自体の変更はせずに初期の目的を十分に果している。またマクロ機能はHPにおいて、同一の表現でレジスタ指定に相違があるようなシーケンスをマクロとし、プログラミングやパンチの容易性に役立っている。

しかしながら、EGMAの使用経験から、

(1) 形式番号をユーザが書くのは非常にめんどうである。
(2) 同一フィールドに2種以上の記憶コードを許したい。
(3) 全フィールドに対し命令形式と無関係に自己規定項を書けるようにしたい。
(4) キイ型の記述形式も許したい。
(5) アドレス参照フィールドに単純な式を許したい。
(6) リアセンブルのたびにコントロール命令セットをつけたくない。
(7) マクロにシステム可変記号を導入したい。
(8) オブジェクトをリロケータブルにしたい。
(9) デバッガを作りたい。
(10) レジスタ名はマイクロプログラム時に指定したい。

等の改良が必要と判明した。

現在、これらの反省に基づいてEGMA自体の改良を行なうと同時に新しいEGMAの作成にかかりつつある。しかしながらマイクロプログラム記述用言語として効率を重視した機械依存の中級言語は必要であり、現在検討中である[11]。

```
R   OBJECT COD                STNO M     SORUCE  STATMENT
000                           0001         MACD
000                           0002         OPGT &1,&2
000                           0003                 JSB     &1          PUT ADD OF 1ST WORD IN S1          1
000                           0004                 JSB     &2          PUT 1ST WORD IN S2                 1
000                           0005         MEND
000                           0006         MACD
000                           0007         CW   &1,&2,&3,&4
000                           0008              &1   IOR M  CW  NMPV   START CW CYCLE IF AF/BF N SET      0
000                           0009              &2   IOR AAB     UNC  LOAD 1ST WORD IN A/B IF AF/BF      0
000                           0010              &3   IOR T       &4   SET OR SEND 2ST WORD TO MEMO        0
000                           0011         MEND
000                           0012   SWAP  STRT '0'
000                           0000         OPGT GETA,OPGE
000  FD0F 0F                  0013  M              JSB     GETA        PUT ADD OF 1ST WORD IN S1          1
001  FD0F 0F                  0014  M              JSB     OPGE        PUT 1ST WORD IN S2                 1
002  F695 FF                  0015              S1  INC  S3          PUT ADD OF 2ND WORD IN S3          0
003  F3F1 EF                  0016              S3  IOR  M  RW       FETCH 2ND WORD/SET AF/BF           0
004  AFF4 FF                  0017         AAB COND IOR  S4         PUT 2ND WORD IN S4                 0
005  FC98 FF                  0018              P   INC  P           INCREMENT P PAST DEF               0
000                           0000         CW   S3,S2,S2,
006  F3F1 CA                  0019  M           S3  IOR  M  CW  NMPV  START CW CYCLE IF AF/BF N SET      0
007  F5FC FE                  0020  M           S2  IOR  AAB    UNC  LOAD 1ST WORD IN A/B IF AF/BF      0
008  F5F2 FF                  0021  M           S2  IOR  T           SET OR SEND 2ST WORD TO MEMO       0
009  F7FF 6F                  0022              S1  IOR      AAB     RESET AF/BF ACCORD TO 1ST ADD      0
000                           0000         CW   S1,S4,S4,EOP
00A  F7F1 CA                  0023  M           S1  IOR  M  CW  NMPV  START CW CYCLE IF AF/BF N SET      0
00B  F1FC FE                  0024  M           S4  IOR  AAB    UNC  LOAD 1ST WORD IN A/B IF AF/BF      0
00C  F1F2 FD                  0025  M           S4  IOR  T      EOP  SET OR SEND 2ST WORD TO MEMO       0
00D  FFFF FF                  0026               IOR              END OF ROTINE                      0
```

図8 EGMAのリスティングの例

5 むすび

以上、現在試作中のモジュール型複合計算機ＡＣＥについて、その考え方と概要、ならびにバス制御方式と基本処理ユニットで用いるマイクロアセンブラの方式について報告した。ＡＣＥの方式設計と基本論理設計は既に終了しており、試作１号機は昭和５０年初めに完成の予定である。しかしこれを利用するソフトウェアについては言語以外、また検討があまり進んでいないので今回の報告では省いた。今後はソフトウェアの検討や１号機の評価を進める一方、ハードウェアもプロセッサモジュール３台程度の本格的複合計算機システムの試作を考慮中である。

最後に、ＡＣＥの研究は多数の方々の直接、間接の御援助のもとで進められたことを記し、感謝の意を表わしたい。特に次の諸氏の御協力は大であった。

相磯秀夫、坂村健（慶大）、元岡達、勝又裕（東大）、西野博二、黒川一夫（電総研）

参考文献

(1) 飯塚、藤井、弓場、島田：“新しい計算機システムの調査と評価”、電総研調査報告　１７９号、昭和４９年２月

(2) 石井、飯塚他：“マイクロプロセッサー設計のための予備的検討一”　ＰＩＰＳ・Ｒ－No. 3　電子技術総合研究所　昭和４８年１０月

(3) 飯塚：“コンピュータモジュール”、電気学会　全国大会　昭和４９年４月

(4) 飯塚、古谷、坂村：“ＡＣＥ　マイクロプロセサ　ユニットのアーキテクチャ”、情報処理学会第２回アーキテクチャ研究会資料、昭和４９年１０月

(5) 飯塚：“ＡＣＥ　プロセッサモジュールのアーキテクチャ”、情報処理学会第２回アーキテクチャ研究会資料、昭和４９年１０月

(6) S.H. Fuller et al：“Computer Modules：An Architecture for Large Digital Modules”, Proc. First Annual symp. on Computer Architecture, pp. 231～237, Dec. (1973).

(7) S.H. Fuller et al：“Some Observation on Semiconductor Technology and the Architecture of Large Digital Modules”, Computer vol. 6, No. 10, pp. 15～21, Oct. (1973).

(8) H. Iizuka：“The Configuration of ACE 0.1” Document ACE-4.1, Apr. (1974).

(9) 藤井、飯塚：“ミニコンピュータのための汎用マイクロアセンブラ”、昭和４８年電気学会全国大会予稿, pp. 1651～1652

(10) Y. Nitta, A. Nozaki & T. Uehara：“On An Efficient Assembler Building System — NETAS Meta-Assembler —”、First USA-JAPAN Computer Conference, pp. 442～447 (1972)

(11) G. R. Lloyd & A. Van Dam：“Design Considerations for Microprogramming Languages”, NCC. pp. 537～543 (1974)