55—R 007

16ビットマイクロコンピュータの動向

—その応用分野と高位言語を展望する—

昭和56年3月

JIPDEC

財団法人　日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて，昭和55年度に実施した「マイクロコンピュータの応用に関する調査研究」の一環としてとりまとめたものであります。

序

最近のマイクロコンピュータを中心としたマイクロエレクトロニクスの進展は，８０年代に新たな産業革命を引き起こす勢いを見せており，産業構造審議会の情報産業部会においては，「情報化の社会全般に及ぼす影響は１８世紀の産業革命にも匹敵する広がりと深さをもち，点から面への広がりをもった第二次情報革命の段階に来ている」と指摘している。産業界においては，生産性の向上，省エネルギー，機電一体化等のかけ声の下に，全ゆる産業にマイクロコンピュータの応用が進んでいる。

当協会では，昭和５３年度にマイクロコンピュータ振興センターを設置し，わが国のマイクロコンピュータの普及，産業の振興を目的として，調査研究を行っておりますが，本年度は，今後１６ビットマイクロコンピュータの普及に伴いそのソフトウェアが大きな問題となるとの認識から，その応用分野の分析，システム記述言語の評価と展望およびエンドユーザにおける諸問題の分析を中心に調査研究を行い本報告書としてとりまとめました。

ここに本調査研究の実施に際し，業務ご多忙中にもかかわらず有益なご意見とご協力をいただきました関係各位に対し厚くお礼申し上げます。

最後に，本書が広くマイクロコンピュータの応用に関係する方々に利用され，マイクロコンピュータ利用技術が人間の創造力を支援する人類の大いなる財産として発展することによりわが国産業の発展に寄与することを念願とする次第であります。

昭和５６年３月

財団法人　日本情報処理開発協会
会　長　上　野　幸　七

マイクロコンピュータ応用技術調査委員会

（敬称略）

委員長	西川禕一	京都大学工学部教授
委員	田丸啓吉	京都大学工学部教授
〃	五十嵐久芳	日本電気㈱電子デバイス販売事業部 マイコン販売部技術課長
〃	石田敏	パナファコム㈱ソフトウェア開発部 第1ソフトウェア課長
〃	大野侑郎	協同システム開発㈱参与
〃	黒川利明	東京芝浦電気㈱総合研究所
〃	千葉正 （第3回委員会まで）	三菱電機㈱技術管理部技術計画担当部長
〃	松浦卓丈 （第4回委員会より）	三菱電機㈱技術管理部部長代理
〃	栃原一元	アイ電子測器㈱技術開発部
〃	前田英明	ソフトウェアコンサルタント
〃	真弓和昭	松下電器産業㈱中央研究所情報第一開発室長
〃	溝部悦夫	日本電信電話公社技術局データ宅内部門
〃	渡辺坦 （第2回委員会まで）	㈱日立製作所システム開発研究所主任研究員
〃	浜田穂積 （第3回委員会より）	㈱日立製作所システム開発研究所研究員
協力者	太田喜久	日本電気㈱マイクロコンピュータ 応用事業部デザインエイド部主任
オブザーバ	小島彰	通商産業省機械情報産業局情報処理振興課
〃	稲積義登	通商産業省機械情報産業局情報処理振興課
〃	勝山治夫	通商産業省機械情報産業局電子政策課
〃	長岡久人	通商産業省機械情報産業局電子政策課
〃	佐藤昌彦	通商産業省機械情報産業局電子政策課
〃	辻義信	通商産業省機械情報産業局電子機器電機課
〃	島戸常喜	情報処理振興事業協会産業プログラム課
〃	古沢章	㈱日本電子工業振興協会システム担当

応用市場小委員会

委員	真弓和昭	松下電器産業㈱中央研究所 情報第一開発室長
〃	五十嵐久芳	日本電気㈱電子デバイス販売事業部 マイコン販売部技術課長
〃	溝部悦夫	日本電信電話公社技術者局データ宅内部門
〃	野中孝弘	東京芝浦電気㈱半導体事業部 半導体営業推進部部長附

システム記述言語小委員会

委員	大野侑郎	協同システム開発㈱参与
〃	石田敏	パナファコム㈱ソフトウェア開発部 第1ソフトウェア課長
〃	前田英明	ソフトウェアコンサルタント
〃	栃原一元	アイ電子測器㈱技術開発部
〃	浜田穂積	㈱日立製作所システム開発研究所研究員
協力者	黒川利明	東京芝浦電気㈱総合研究所
	太田喜久	日本電気㈱マイクロコンピュータ応用事業部 デザインエンド部主任

エンドユーザ問題小委員会

委員	田丸啓吉	京都大学工学部教授
〃	黒川利明	東京芝浦電気㈱総合研究所
〃	千葉正	三菱電機㈱技術管理部技術計画担当部長

目　　次

序　　論

序　論

マイクロコンピュータの歴史ももはや１０年に近い歳月を経ようとしている。１０年といえば現代技術の変遷の速さ，めまぐるしさを考えるとき，一昔というに十分長い期間である。そして驚くべきことに，この長い期間の間，マイクロコンピュータは当初予想されたよりもはるかに順調にかつ急速に伸展を続けてきた。ＰＭＯＳ４ビット中心の第１世代からＮＭＯＳの第２世代へ，そして４ビット１チップを中心としながら，ＣＭＯＳ，８ビット，ビットスライス，そして１６ビット高性能など，様様な機種が華々しく登場した第３世代も，もう既に終りをつげたとされる。その間，大勢において基礎技術の発達も応用の量，質両面における拡大も，一度もつまずきをみせなかった。つまずくどころか，現実は常に予想を超えて発展し続けたのである。これは技術史の上でも類い稀な出来事であり，それゆえに，第一次産業革命における蒸気タービンにも比すべき，第２次産業革命技術の核をなすものだともいわれている。

本委員会においては，昭和５３年及び５４年の両年度にわたってマイクロコンピュータ応用の立場から，素子，周辺機器，ソフトウェアなどの現状と各種応用の関連を体系的に探ってきた。その中から将来に向けてのさらなる発展の方向を予測するとともに，技術開発を進めるについての基本的な考え方を提言してきた。そしてまた，発展の障壁となっているいくつかの課題を整理して明示することも行った。それらの詳細は，昭和５３年度の報告書「応用からみたマイクロコンピュータ技術の現状と課題」及び昭和５４年度の報告書「マイクロコンピュータ応用上の課題と展望」に述べられている。

その中には，ハードウェアとソフトウェアの両面にわたって様々な課題が指摘されているが，その要点を整理すれば次のようにいうことができるだろう。

(1) ハードウェアとソフトウェアの標準化，規格化，モジュール化を，ある程度の範囲内であっても推し進めることが必要である。

(2) 応用システム規模の拡大と機能の高度化の要求に伴い，システム開発サポート・ツールの整備・充実が要望される。特にソフト開発に要する手間と時間，従って経費は増加する一方であり，マイクロコンピュータにおけるソフトウェア・エンジニアリングの強化が強い要望となっている。

(3) ソフトウェアの問題解決のため，高位言語の利用が必要不可欠のものとなろう。その際，ユーザの立場からすれば，コンピュータのハードウェア，アーキテクチュアとかかわりなく，しかもあまり特殊な素養と訓練を要さずに使えるような言語が望ましい。その他，いわゆるマイクロコンピュータ用の軽装ＯＳの開発，普及が要望される。

(4) 周辺回路のＬＳＩ化，標準化，低価格化が望まれる。これはある程度順調に進展しているとみることができる。また入出力機器の価格低減，信頼性の向上もなお努力を要する点である。

(5) センサやアクチュエータには機能，性能，サイズ，信頼性，価格など，多くの点で課題が多い。しかしこれはマイクロコンピュータ応用に特有の問題ではなく，また範囲も広いので，技術体系全般の基本的な課題と考えるべきであろう。

(6) その他ユーザ向け情報提供，ソフトウェア流通，訓練・教育体制などにも改善乃至検討を要する

いくつかの点がある。

以上の諸課題のうち，(1)~(3)は特にマイクロコンピュータ技術に固有の，しかも最も基本的な課題であって，早急に具体的な方策を立案し実行に移すべきものである。標準化については，ユーザの立場からマイクロコンピュータ応用の初期の頃から，絶えずいわれ続けてきたことである。互換性のなさはユーザの立場を甚だ窮屈なものにし，またときには弱いものにしている。これはメーカの立場からすれば本来放置できることではないはずである。しかし現実には，既に膨大な数と種類の素子，回路，それに応用商品群が系列化されて存在している。ソフトウェアもまたそれに応じて相互流通性を失ったものが多い。

この現状を乗り越えるには，第一にはすべての関係者の協力が必要である。ユーザはもちろん，メーカだけでもできない。メーカとユーザ，そしてリーダーシップをとるべき公共機関（政府，学界など）の一致した知恵と熱意と忍耐がなくてはとてもできない仕事である。そして第二にはいきなり全面的な標準化は無理，無謀であって，一歩ずつ可能性を確かめながら進むべきである。その第一歩として既に多くの商品群を抱えてしまった機種についておさらいの努力を払うよりは，今後のことにまず目を向けるのがよいではないか。

以上に述べたような経緯と認識が，本委員会の今年度の調査研究の基盤となった。ときあたかもわが国の市場には，１６ビット・マイクロコンピュータが本格的に登場しようとしている時である。メーカの努力による製造技術の進歩によって，１６ビット機種の機能．性能は８ビット以下のそれらをはるかに凌駕するものとなり，また価格も実用に耐えるものになってきた。一方，ユーザ側の要求はコンピュータ，情報処理機器，制御用機器など，知識集約型技術の核心的な分野において，ますます高度なものになりつつある。ナショナル・レベルで考えてみても，量産品的なものから脱却して，より知識・技術集約化が進んだもの，そしてより付加価値の高い先進的なものを重視していく必要がある。

そこで本年度は１６ビット・マイクロコンピュータとその応用に焦点をあてることにしたわけである。まず，マイクロコンピュータの技術・利用体系について概観し，１６ビット機出現への流れを把えた後，現在乃至極めて近い将来市場に現れる国内外の代表的な１６ビット機７種（いわゆるミニコン系１６ビットマイコンも含む）を選び，ハードウェアの構成と特徴，ソフトウェアの特徴，システムの構成，開発サポートの各項目について，詳細に調査した。その際特に，各コンピュータの特徴を比較し，わかり易く整理することに心がけた。その成果は本書の第２及び３章にまとめた。

次に委員会の中に，「応用・市場小委員会」，「システム記述言語小委員会」及び「エンドユーザ問題小委員会」を設け，それぞれ課題を分担して調査にあたった。これらの小委員会は，それぞれ，１６ビット・マイクロコンピュータの応用分野と利用の動向，マイクロコンピュータ用システム記述言語の評価と展望，及びマイクロコンピュータのエンドユーザにおける問題意識と課題ならびにその解決策について分担した。それらの成果は，本書の第４章から第６章までに詳述した。

まず応用については，１６ビット機の特徴と市場ニーズとの結びつきによって，導入の気運が熟した背景を概観した後，各応用分野別に将来の動向を詳細に分析した。シーズとニーズの動向による長

語長機種への移行の必然性を明確にするとともに，具体的な開発例を数多く拾い上げた。

次いでシステム記述言語については，その利用が必要とされる問題の背景を，１６ビット機に即してハードとソフトの両面から考察し，言語に要求される機能を総合的に論じた。その後現在実用されている１３種の言語について，それぞれの仕様及び機能を調査し，その上で一定様式の評価の枠組みを設定して，各言語を多角的に評価し比較する試みを遂行した。このような比較評価の体系的な作業は，国内外を通じてほとんど例をみないものと思われる。このような作業は，多様な言語がめまぐるしく出現し，ユーザにともすればとまどいと混乱のみられる現状において，貴重な資料というべきであろう。もちろん各言語はそれぞれに優れた特徴と用途・市場をもつており，いずれか一つを良しと断定しようとしたものではない。この報告書の中で敢えて総合評価を試みているが，その結論よりも資料作成の経過と努力をおもんばかって，それぞれの判断の材料として頂きたい。また現存する言語は決して完成されたもの，いずれかが最終的なものとみるべきではなく，長短の諸点を踏まえてより優れた言語体系を開発する方向を見出したいと願うものである。

最後にエンドユーザの問題について，調査と考察を加えた。マイクロコンピュータ応用開発において，エンドユーザは極めて大切な存在である。すべての技術はいうまでもなく，究極的にはユーザのためにある。従来のコンピュータ技術は，ともすればその点をおろそかにしてきたきらいがある。すなわちメーカ技術主導型であった。マイコン応用でそのようなことがあるとすれば，これはまさに死活にかかわる。マイクロコンピュータの発達とともにユーザはどう変わってきたか，ユーザの実状，認識，受けとめ方はどのようなものか。ユーザは何を望み，何を問題としているのか。このような問いにいささかでも答えようとしたのが，第６章である。そのために，数名のユーザとの対話の機会が設けられ，その中から活きた問題把握と解決策の提案を試みた。ユーザは実は極めて多様であり，今回の試みが意を尽くしたものとはいい難いが，これまた一つのユニークな成果であるといえよう。さらにユーザ言語の興味ある一例についても述べている。

本報告書は以上のようなかなり多面的な調査結果をまとめたものである。全委員による５回の共同討議の他，各小委員会でも数回ずつかなり長時間の討議と作業が行われた。その成果は極めてユニークな内容を含み，諸方面の参考になるところが多いと自負している。各部分は委員の分担執筆になるもので，委員会の討議を経たものとはいえ，各著者独自の見解も必然的に表われていよう。読者諸氏の間に活潑な議論を呼び起こし，種々のご批判を賜われば幸甚である。

用語の多少の不統一もご寛容願いたい。マイクロコンピュータは単にマイコンと記したところも多いが，その他はなるべく勝手な用語は避けるように努めたつもりである。

第１章　マイクロコンピュータの定義と分類

第１章　マイクロコンピュータの定義と分類

1.1　マイクロコンピュータの定義

マイクロコンピュータ(micro computer)は基本的には一般の汎用コンピュータの中央処理装置CPU(Central Processing Unit)に相当する部分、メモリ、および入出力装置用制御回路のそれぞれを半導体のＬＳＩ化技術を高度に用いて１チップ又は数チップのLSI 回路として構成したものである。特に，ＣＰＵに相当する装置，即ちデータの算術・論理演算を行う演算部(ALU, Arithmetic aud Logic Unit)，データを一時蓄えるレジスタ群，およびこれらの動作を総合的に制御する制御部を１チップ又は数チップのＬＳＩ上に実現したものをマイクロプロセッサ(micro processor)と呼んでいる。したがって，マイクロプロセッサは外見上は小さな部品(群)であるが、プログラムカンタとストアドプログラム(stored program) 制御方式をもち，高度のデータ処理機能を潜在させたプログラマブル(programmable)なＬＳＩ素子(群)であり，機能的・性能的制限を別にすれば従来から存在する一般のコンピュータのＣＰＵと等価である。

マイクロプロセッサは半導体の大規模集積回路技術の産物であるが，非常に速くめざましい進歩を続けているＬＳＩ技術でも、現在マイクロコンピュータとして要求されている機能性能を１チップのマイクロコンピュータにまとめ上げることは一般には困難である。

現在のマイクロコンピュータは図１－１に示すように基本的には次の４部分より構成されている。

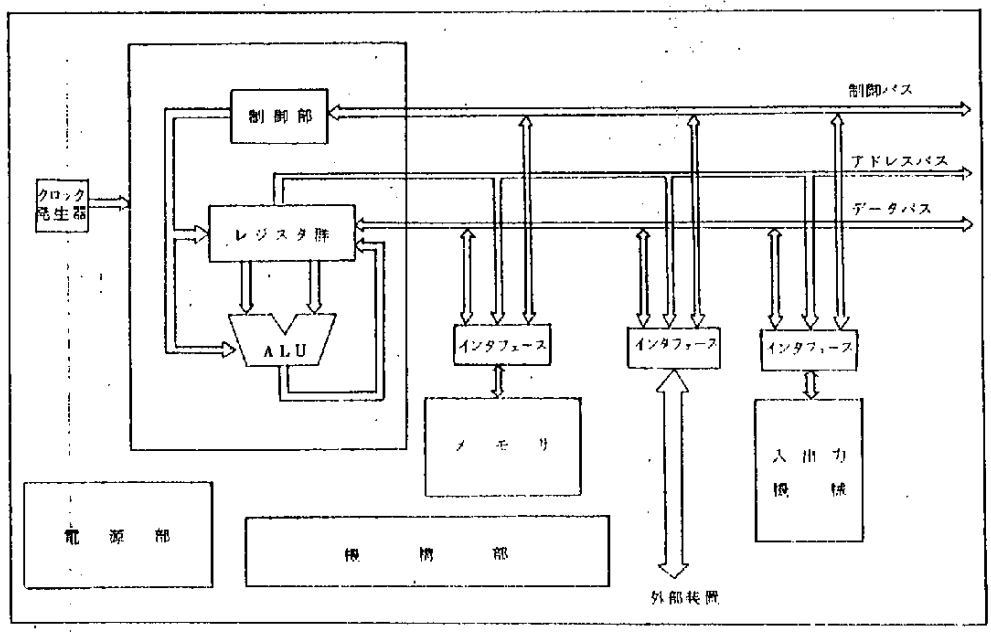


図１－１　マイクロコンピュータの概念構成図

(1) マイクロプロセッサ

制御部はプログラムの構成要素である機械語命令の実行を制御するシーケンス制御部とメモリや入出力機器を制御するためのインタフェース制御部を含む。命令の実行には，マイクロプログラム制御方式と直接配線方式がある。

(2) メ モ リ

マイクロコンピュータのメモリは、ミニコンピュータ以上のコンピュータの場合とその構成にかなり差違があり，使用上の性格によりRAM(Random Access Memory)とROM(Read Only Memory)に大別される。

(3) インタフェース

マイクロコンピュータには多種多様な機能をもつ機器を接続する必要がありインタフェース部は重要である。最近はインタフェース部にもＬＳＩ化が進み，汎用性の高いプログラマブルな周辺ＬＳＩが開発されている。

(4) バ ス

ＣＰＵにメモリや入出力装置を接続するための情報の共通線路であり， 一般に次の３種類がある。

① アドレスバス：メモリのアドレスや入出力機器の番号をＣＰＵから送り出す。

② データバス：データや命令の送信を行う。

③ 制御バス：接続されている装置を制御する。

以上の他にソフトウェアが必要であり，ソフトウェア開発用の各種ツールも不可欠である。16ビットの代表的マイクロプセッサであるi8086の内部構造を図１－２に示す。

1.2 マイクロコンピュータの分類

マイクロコンピュータは，半導体技術の集大成と言われており，急速な技術革新による高性能化，多量生産によるコスト低下，応用技術の多様化を特徴としている。そのためマイクロコンピュータの分類にもいろいろな考え方がある。ここでは応用側から見てみる。

1.2.1 マイクロプロセッサのアーキテクチャによる分類

マイクロプロセッサでもアーキテクチャは一般の計算機と原理的に同じであり，半導体技術の制限，応用側の多様な要求，経済性などの要因への重点の置き方により種々のアーキテクチャが実現できる。

半導体技術からくる主な制限は，集積度とピン数である。そのためレジスタの多用（論理回路の規則性），バス構造（配線面積の減少），演算機能の簡略化（簡単な論理回路）等を特徴としているが現在この制約は緩和されつつある。代表的マイクロプロセッサの集積度とピン数を図１－３に示す。

(1) 汎用型／専用型

応用側からの要求が多様化してくると，まずハードウェアは汎用のものを用い，ソフトウェア

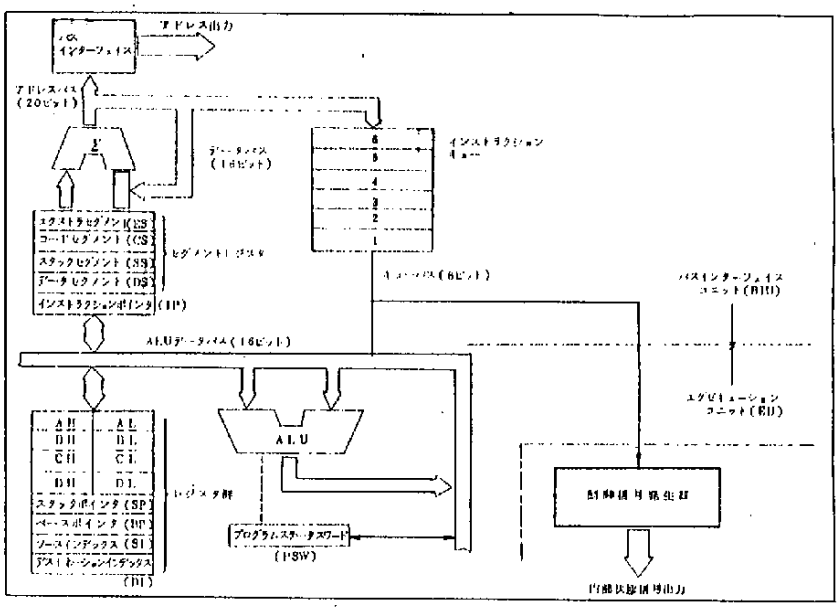


図１－２　１６ビットマイクロプロセッサｉ８０８６の内部構造

で個別化・専用に対処することになるが，さらに高性能なものや低価格なものを求める場合は専用のアーキテクチャを求めることになる。

アーキテクチャを専用にしてより一層の低コスト化が可能なのは，家電製品への応用のように少品種多量生産の場合であって，開発費の投入に経済的なメリットがなければならない。このような例としては４ビット型に多い。

一方，汎用型より一層高性能なものを必要とする場合は，ＣＰＵ機能の縦型分散によるマルチプロセッサ方式と高速バイポーラ素子群による基本語長分割のビットスライス方式がある。ビットライス

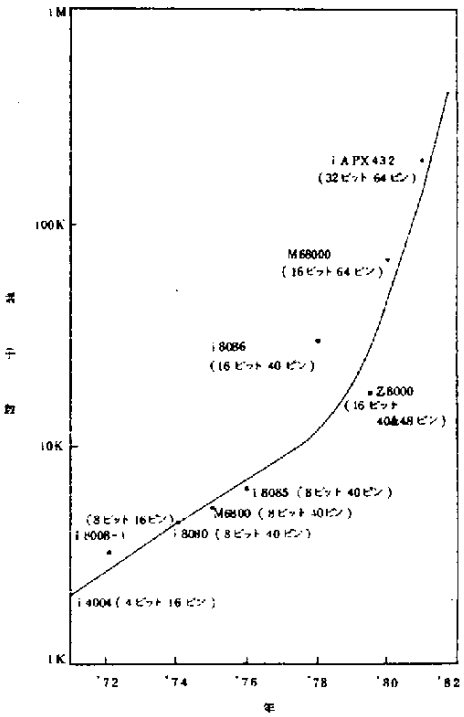


図１－３マイクロプロセッサの集積度

方式のねらいは，高速化と柔軟性にあり，ＣＰＵの制御方式が機械語命令自体を応用目的に合わせて構成することができるマイクロプログラム制御になっている。

ＣＰＵアーキテクチャを専用化するとその程度にもよるが一般に周辺回路，ソフトウェア，開発用サポートツール等にも独自の体系が必要になることが多い。これらの問題には性能と経済性に対する総合的判断が必要であるが，特に応用製品のライフサイクルが長い場合のメンテナンス体制が課題である。

(2) 基本語長

基本語長はマイクロプロセッサのアーキテクチャを決める最も基本的な事項であり，応用側でマイクロプロセッサの機種選定を行うとき演算速度とともに重要な選択要因となっている。そして語長の決定は，多くの場合応用側から要求される処理能力と経済性のバランスの問題である。

現在市販されているマイクロプロセッサの語長は，４ビット，８ビット，１２ビット，１６ビットの固定語長とビットスライス型の可変語長であり，語長が長くなる程高性能・高価格である。又マイクロプロセッサの開発動向も４ビットから順に１６ビットと発展し，さらに３２ビット機が出現している。

語長はさらに，命令形式やアドレス方式にも直接関係し，その結果ソフトウェア開発の生産性にも大きな影響を与える。

表１－１　マイクロプロセッサの開発動向[(1)]

世代（年）	ＬＳＩ技術	ビット数（実装ゲート数）	基本命令処理時間(μs)	ピン数	特　　徴
Ⅰ（1971～1974）	Ｐ－ＭＯＳ	4／8（300～1,000）	10～20	16／24	演算機構指向
Ⅱ（1974～1977）	Ｎ－ＭＯＳ Ｃ－ＭＯＳ ＳＴＴＬ	4／8／12／16（500～3,000）	1～10	40／64	マイクロコンピュータ指向基本ソフトウェア開発
Ⅲ（1977～？）	Ｎ－ＭＯＳ Ｃ－ＭＯＳ SOS-MOS ＳＴＴＬ I^2L ＥＣＬ	4／8／16／32（3,000～20,000）	0.2～1	40/64/100	マルチプロセッサ指向，冗長技術導入高レベルソフトウェア開発，サポートシステムの利用

マイクロプロセッサの開発動向を表１－１と図１－４に示す。又マイクロプロセッサの性能を表１－２に，その位置付けを図１－５に示す。さらに，各語長によるマイクロプロセッサの特徴と応用分野を表１－３に示す。

(3) そ　の　他

アーキテクチャ面から分類する場合の重要な項目としては次の諸点がある。

(a) 命令形式と機能

語長と無関係でなく，１６ビットなど比較的語長の長いものではミニコンピュータに類似の形

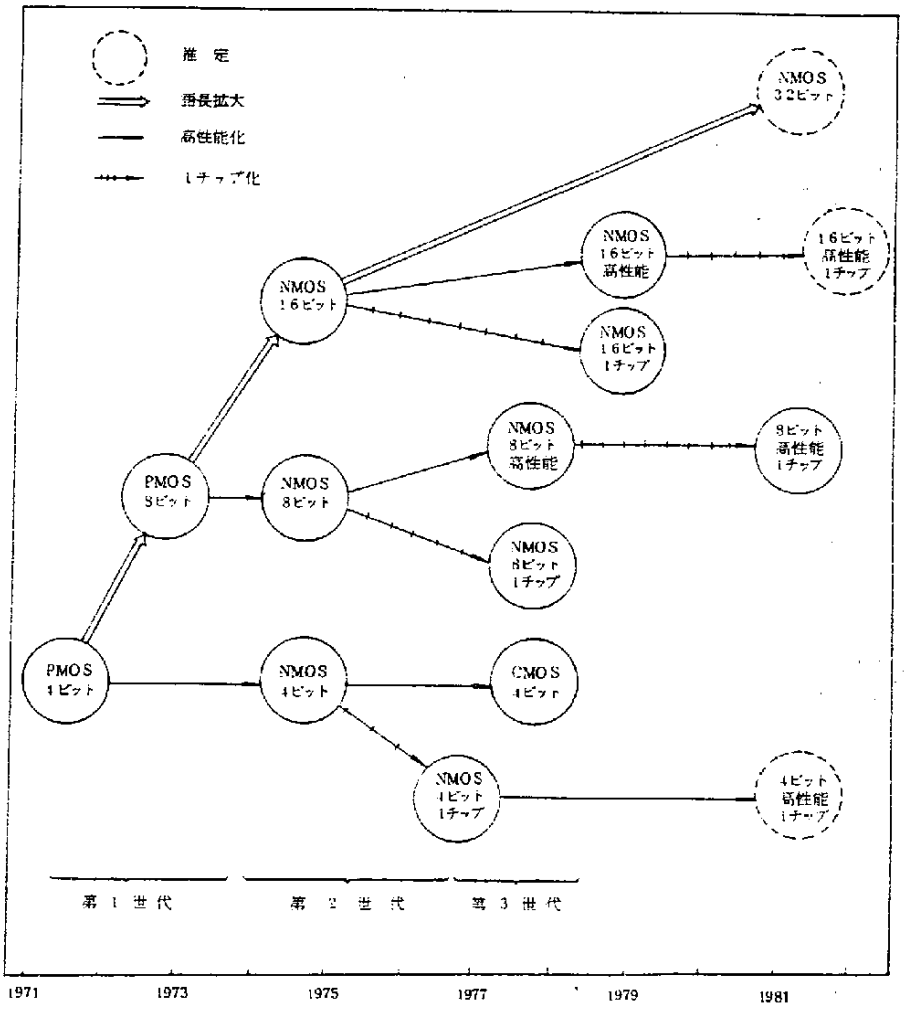


図1－4　マイクロコンピュータの発展[3]

表１－２　マイクロプロセッサーの性能表

ビット数	4	8					16			4（1チップ）	8（1チップ）		4ビット ビットスライス
プロセッサ名称	4040	8080	8085	Z−80	6809	F8	8086	Z8000	68000	TMS1000	8049	6801	Am2901
メーカ名称	インテル	インテル	インテル	ザイログ	モトローラ	フェアチャイルド	インテル	ザイログ	モトローラ	テキサスインスツルント	インテル	モトローラ	アドバンストマクロディバイス
プロセス	PMOS	NMOS	NMOS	NMOS	NMOS	NMOS	HMOS	NMOS	NMOS	PMOS	NMOS	NMOS	ローパワーショットキー
電源電圧（V）	+5, −10	+12,+5 −5	+5	+5,0	+5	+5V/45mA (TYP)	+5	+5	+5	−1.5	+5	+5	+5
消費電力（mW）	900	780 (TYP)	1.5W	1W (max)	1.0W	12V/12mA (TYP)	2.5W	1.5W		400	1.5W		1.4W
クロック（MHz/相）	750K/2	2/1	3/1	2.5, 4/1	1/2	2/1	5, 8	4	4	400K	11/2		6.6
ピン数	24	40	40	40	40	46	40	48	64	28/40	40	40	40
命令数	60	78	113	158	59	76	105	116	59	54	96	82	マクロインストラクション（31）
メモリー容量	8KB	64KB	64KB	64KB	65KB	64KB	1MB	8MB	16MB	ROM 1KB RAM 64×4ビット	ROM 2KB RAM 128B	ROM 4KB RAM 128B	
レジスタ数	21	11	11	17	9	16	14	16	16	7	9	28	
レジスタ加算速度（μs）	10.8	1.3〜2	0.8〜1.3	1.6	2	2	0.25〜0.6	0.75	0.5	（15）	平均実行時間(1.36)	（2）	（97ns）
I/Oインターフェイス	メモリR/W	I/O命令	I/O命令	I/O命令	メモリR/W	I/O命令	I/O命令	I/O命令	メモリR/W		I/O命令		4
スタックレベル	8	∞	∞	∞	65KB	2	∞	8MB	16MB	1	8	65KB	マイクロプログラム
DMA接続	−	可	可	可	可	可	可	可	可		−	−	

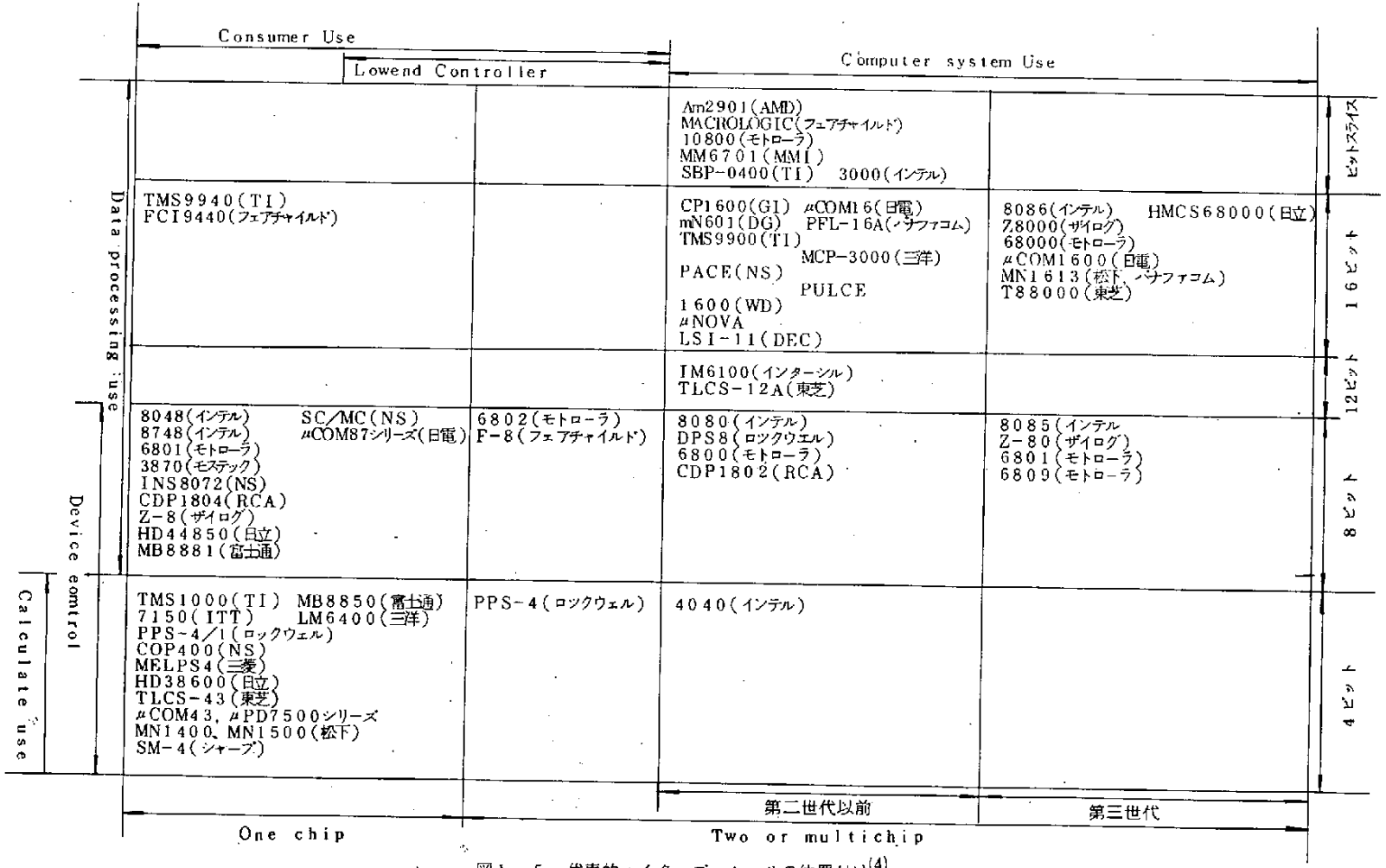


図1－5　代表的マイクロプロセッサの位置付け[4]

表１－３　語長によるマイクロプロセッサの特徴と応用分野

ビット数	特　　徴	応　　用	備　　考
4ビット	1.低価格だがコンピュータ能力に限界あり。 2.演算速遅くメモリが小さい，専用的用途。 3.1ワード4ビットで10進1ケタの演算に強い。 4.命令機能弱くプログラム開発効率低い。 5.プログラム規模小さく開発ツールに高度のものは不要 6.1チップ化	1.事務機械等の簡単な制御と電卓 2.家電機器の制御	1.機器組込み 2.必要最小限の機能をもつ専用機種で特注多量生産
8ビット	1.現在の標準品でコンピュータ機能も一応のレベルに達す。 2.1ワード8ビットでキャラクタマシンに適す。 3.1ワードビットはデータ伝送用として利点が多い。 4.周辺ファミリー素子が充実。 5.命令機能に制能に制約がある。 6.数値計算には一般に精度不足	1.コンピュータ周辺・端末機器 2.通信制御機器 3.POS，インテリジェント端末，キャシュレジスタ，ファクシミリ等，事務用機器 4.一般産業機器の低機能制御（プロセスコントロールには不十分）	1.応用範囲が非常に広い 2.セカンドソースが多い
12ビット	1.1ワードで10進3ケタの精度が得られる。 2.A／D変換器の精度と同程度で計測制御データの処理効率が良い。	1.計測制御用	1.機種は少ない
16ビット	1.ハードウェアが複雑化 2.1ワードで10進4ケタ 3.高速演算 4.命令機能が強化されアドレス空間は大きくコンピュータ能力は高い。 5.ソフトウェアが大規模複雑化するため開発にはサポートソフトウェアが必要 6.ソフトウェアの高度化によりミニコンと対抗できる。	1.ミニコン／オフコンのCPU 2.コンピュータ周辺端末機器 3.N／C 4.プロセスコントロール 5.通信制御	1.8ビットの上位機種と独自のアーキテクチャーをもつ機種がある。 2.ソフトウェアの諸問題が顕在化
ビットスライス	1.バイポーラデバイスによる高速演算MOS系より1ケタ速い。 2.応用目的対応の語長，命令体系を構成。 3.チップは低集積のCPU高性能部品	1.ミニコンピュータのCPU 2.応用別の専用計算機 3.産業用高性能コントローラ 4.ミニコンの低価格エミュレータ 5信号・画像処理	

式と機能が見られるのに対して，４ビット，８ビットなど語長の短い場合は特殊性が顕著である。

(b) アドレス方式

ミニコンピュータの場合命令の中にアドレス形式の指定フィールドをもつため，あらゆる命令において各種のアドレス形式を使用することができるのに対して，マイクロプロセッサでは命令の種類にアドレス形式を混在させて命令コードを定義する場合が多く，アドレス形式も不充分にならざるをえない。しかし，語長の長い機種ではこの点が改良されているものもあり，アドレス空間も１Mバイト以上のものが多い。

(c) レジスタ

マイクロプロセッサでは複数のアキュムレータ方式又は比較的数の少ない汎用レジスタ方式が多いが，長語長の機種には本格的な汎用レジスタをもつものがある。そしてマイクロプロセッサでは比較的レジスタ間の処理が強い。特にスタックが本格的に導入されているのが特徴である。

(d) 内部バス

データやアドレス等の情報の共通路である。一般に配線面積がチップ面積の５０%以上を占めるためバス構造をできるだけ減らしたい。しかしそうすると性能が低下するので種々の工夫が必要である。

(e) 割込み制御

外部割込み制御は，ＣＰＵに対して非同期的に発生する外部事象と同期をとるため非常に重要である。割込みの要因，優先度，さらにソフトとハードでの機能分担等について種々の扱い方がある。

(f) 入出力装置とのデータ転送

入出力装置とのデータ転送方式は，プログラム制御によりデータ転送をＣＰＵがすべて行う方式（プログラム制御）とデータ転送の開始手続きだけをＣＰＵが行うが，その後はＣＰＵが介在せずメモリと入出力装置の間で独立にデータ転送を行う方式（DMA：Direct Memory Access）がある。ＤＭＡ方式はＣＰＵの速度に制約されずに高速でデータを入出力するときに適している。（１０倍以上）　ＤＭＡ転送を行うには外部制御回路としてサポート用ＬＳＩを用いる。

(g) 制御方式

一般のコンピュータのハードウェアの制御方式には，制御順序をすべて固定的な 配線ロジックで行う直接配線方式とハードウェア各部の詳細な行動を逐一制御するマイクロプログラム（各機械語命令は数ステップ以上のマイクロプログラムから構成される）をもつマイクロプログラム方式がある。その特徴は，直接配線方式の高速性と経済性，マイクロプログラム方式の柔軟性である。４ビット，８ビットでは経済性の面から直接配線方式が，１６ビット以上の高

性能機特にビットスライス型ではマイクロプログラム方式が多く用いられる。

(h) ワンチップ／マルチチップ

ＬＳＩ技術の制約のため３つの方向がある。１つは性能より価格の低下に重点を置きＣＰＵ，メモリ（RAM, ROM），入出力インタフェース等を１チップに集積したワンチップマイクロコンピュータで，２つは高性能と柔軟性をねらったビットスライスマイクロプロセッサである。３つ目は複数のマイクロプロセッサ／マイクロコンピュータを結合したマルチマイクロコンピュータである。これは機能分散による性能向上，モジュール化による拡張性の向上，処理の多重化による信頼性の向上等をねらったものである。

1.2.2 ＬＳＩデバイスによる分類

コンピュータに使用される論理回路ＩＣは，バイポーラ系とＭＯＳ系（Metal Oxide Semiconductor）に分類される。バイポーラ系は高速であるが高集積化が困難で消費電力も多い。しかし最近集積度と消費電力の面でＭＯＳに近いI^2Lが出現している。

ＭＯＳ系はバイポーラ系に比べて動作速度は遅いが集積度が高く消費電力が低い。そのため電卓やマイクロコンピュータに多用されている。低価格品には構造も製造法も簡単なPMOSが，８ビット以上のものにはPMOSより速く高集積可能なNMOSが，さらに低消費電力で雑音に強いものが必要なときにはCMOSがそれぞれ用いられる。図１－６と表１－４に論理回路ＩＣの特性と集積度を示す。

表１－４　主な論理回路ＩＣの特性[5]

（$n=10^{-9}$）

種	類	スイッチング時間（ｎ秒）	消費電力（mW）
バイポーラ	LCML	0.5～ 1	1～5
	ECL	0.5～ 2	15～45
	TTL	10	10
	S-TTL	3	20
	LS-TTL	10	2
	IIL	＞10	＞0.05
MOS	PMOS	50～200	0.1～2
	NMOS	10～ 50	0.05～0.2
	CMOS	10～100	0.001～0.1

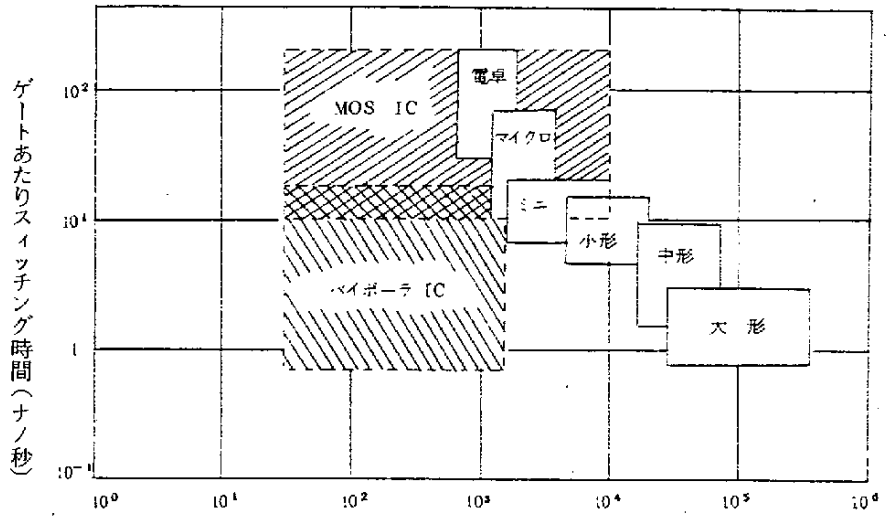


図１－６　論理回路規模（ゲート数／プロセッサ）(5)

1.2.3　メモリによる分類

マイクロコンピュータでは一般のコンピュータと多少異なり単目的の使用が多いから，多種類のプログラムをリロードすることは少ない。そのためマイクロコンピュータのメモリでは読出し専用メモリＲＯＭ（Read Only Memory）の多用，応用目的に応じてRAMとROMの混用，実装されたメモリにおけるアドレスの不連続等の特徴が見られる。

メモリ用ＬＳＩの開発動向も急速で，大容量化（低価格化）は特に著しく，近い将来1Mビット／チップの出現も予想されている。

メモリの大容量化は，ＣＰＵの高性能化と相まってアドレス空間の拡大とソフトウェアの高度化・大規模化を可能にしている。

マイクロコンピュータに用いられるメモリは，RAM（Random Access Memory）とROMに分類できる。電源切断時に記憶内容が消滅する揮発性のもの（例えば，ダイナミックＲＡＭ）と電源切断後も記憶内容が保持され再使用ができる不揮発性のもの（例えば，ＲＯＭや一部のＲＡＭ）がある。さらに，ＬＳＩデバイスからは，ＭＯＳ系とバイポーラ系がある。

マイクロコンピュータで使用する各種のメモリの分類とその特性を図１－７に示す。

以上の通りメモリ用ＬＳＩには種々のものがあるが、実際にマイクロコンピュータを構成するときにはこれらの特性を十分活用する。

マイクロコンピュータをメモリのハードウェア面から見て分類すると，ＣＰＵチップ内にメモリを内蔵させたものと，メモリを別チップに持つもの（１～数十チップ）がある。またメモリの機能面すなわちシステム的側面から見るとＲＯＭとＲＡＭの容量比が大きな分類上の特徴となろう。

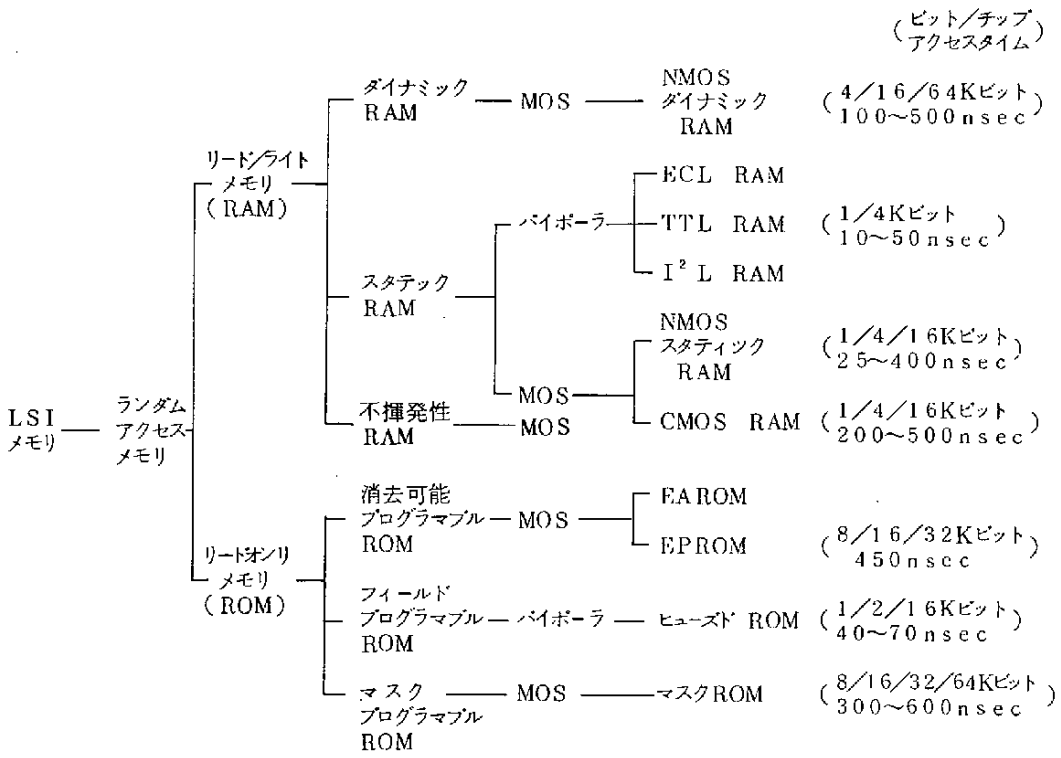


図1－7　LSIメモリの分類

1.2.4　形態による分類

マイクロコンピュータは最終的には完成されて応用側で種々の形態で稼動することになるが，応用側がメーカ側より受けとる形態には種々のレベルがあり，それに応じて応用側で行うシステム完成までの作業範囲が大きく変る。

(a)　ビットスライス用チップ

チップはCPUの構成部品であるからCPUを構成することからはじまりソフトウェア（マイクロプログラムから応用ソフトまで）を含めてシステムが稼動するのに要する作業はほとんどすべて応用側で行う。

(b)　マイクロプロセッサ

(a)と同様チップレベルであるがCPU自体は完成しているから，メモリ，インタフェース等ボードレベル以後の作業が残されている。

(c)　ワンチップマイクロコンピュータ

マイクロコンピュータとしての基本機能はすべて内蔵されており，プログラム記憶用ROM（1

～２Ｋバイト）にプログラムを書込んで使用する。

(d) ボードコンピュータ

機能的にはワンチップマイクロコンピュータと同じであり，マイクロプロセッサ，ROM, RAM, プログラマブル入出力制御用ＬＳＩ，バッファ回路等を一つの基板上にまとめたものである。構成や規模の選択範囲は広く，産業用には標準化されたコンピュータユニットとして便利で応用範囲が広い

(e) 応用製品組込みコンピュータ

利用者側からみればハードウェア的には応用製品組込みのブラックボックス的存在であり，その形態は応用分野により多種多様であるがシステムの中で比較的独立性の強い構成要素となっているものである。ユーザプログラマブルなものとそうでないものがあり，前者では応用分野向きの専用言語をもっているものが多い。このタイプのものとしては産業用コントローラ，事務用インテリジェント端末機等がある。

(f) 自立型コンピュータ（スタンドアロン型）

規模又は性能において差はあっても基本的機能は一般のコンピュータと同じものである。これに属するものとしては，パーソナルコンピュータ，さらにＣＰＵをマイクロプロセッサで構成し従来のミニコンピュータとソフトウェアの斉合性をもたせたものなどがあるが，こうなると従来のミニコンピュータとの境界が不明確になってくる。

1.2.5 ソフトウェアによる分類

語長が長くなり高性能化してくるとマイクロコンピュータの利用技術の拡大にとってソフトウェアの開発整備が大きな課題であり，特に応用範囲が拡大して多種類のソフトウェアが必要になる場合の生産性と信頼性の向上が大きな問題である。このためには良いソフトウェアを蓄積してその流用・流通を図ることとソフトウェア生産用の高度なサポートツールを活用することであろう。サポートツールには，そのマイクロコンピュータ自身をソフトウェア開発にも使うセルフ系のものと開発用に別のコンピュータ（多くの場合汎用ミニ／大型コンピュータ）を使うクロス系のものがありそれぞれ長所短所がある。

マイクロコンピュータのソフトウェアの規模は現在，４ビットでは１～２Ｋバイトのものが多く最大で４Ｋバイト程度，８ビットでは５～１０Ｋバイトのものが多く大きなものでは４０Ｋバイト以上のものもある。これらのソフトウェアはほとんどアセンブラで開発されており，開発工数も１～２人月程度から数十人月のものもある。１６ビット以上の高性能マイクロコンピュータでは，ソフトウェアの規模は従来型の１６ビットミニコンピュータと同程度になるが，問題はマイクロコンピュータの応用技術が非常に広く拡大することによりソフトウェアニーズが質量ともに急増することであろう。そのため優れた高位言語の開発と標準化・実用化，汎用性が高くかつマイクロコンピュータの長所を発揮した高性能なオペレーティングシステムの開発が緊急の課題になっている。

応用側でマイクロプロセッサ／マイクロコンピュータの機種選定を行う場合，応用側から見たソフトウェア開発環境に十分に注意することが必要である。

1.2.6 開発支援装置による分類

マイクロコンピュータすなわちハードウェア／ソフトウェアを開発する場合どうしても使用するＣＰＵ対応に開発支援装置が必要であり，使用する開発支援装置の機能・性能により開発効率が大きく変る。したがって，開発支援装置によりマイクロプロセッサを分類することができる。

開発支援装置としてはレベルの低い方から次のようなものが使われている。

(a) 一板のプリント基板にマイクロプロセッサ，メモリ，タイプライタや表示灯などのためのインタフェースを実装し小規模な管理プログラムを内蔵しているもの。

(b) ミニコンに似た操作パネルをもち筐体内がＣＰＵ基板やＲＡＭ基板など基板単位の構成体系になっていて，アセンブラやテキストエディタなどのサポートソフトウェアを内蔵するもの。

(c) ディスク（フロッピー／ハード）など多くの周辺機器をもちその上に高位言語などのレベルの高いサポートソフトウェアを稼動させプログラム開発効率を高めようというもの。さらに，プログラムを同種のＣＰＵで走らせて検査する（in-circuit emulation）機能をもっている。

(d) 以上の３つの他に専用装置ではないが，大型コンピュータにクロスサポートソフトウェアをのせてプログラム開発をする場合があり，多量にプログラムを開発する場合などに極めて有用である。

1.2.7 セカンドソースによる分類

Ａ社がａという製品を市場に出し，Ｂ社が後からａと使用上は全く同一の製品ｂを発売した場合，Ｂ社をＡ社のａに関するセカンドソースという。セカンドソースは機種の標準化，供給の安定化，低価格化等を促進するわけでマイクロプロセッサ市場では大きな役割を占めており，セカンドソースの有無・数等によりマイクロプロセッサを分類することができる。

〔参考文献〕

(1) 相磯：マイクロコンピュータとその応用，電子通信学会
(2) 森：マイクロコンピュータハンドブック，丸善
(3) 真弓：マイクロコンピュータの家電製品への応用，電四学連大会，３５－２，昭和５４年
(4) 保安装置に対するマイクロプロセッサの適用の研究報告，信号保安協会
(5) 石黒：ハードウェアー素子技術とアーキテクチャ，電気学会雑誌，１００巻１１号

第2章　マイクロコンピュータの技術・利用体系

第2章　マイクロコンピュータの技術・利用体系

マイクロコンピュータあるいはマイクロプロセッサという言葉は，今日ではほとんどあらゆる分野の技術の専門家のみならず，一般市民の間でも極めて耳馴れたものになってきた。いわく，マイコンつき家電製品，マイコン・コントロールとチェック機能を具えた乗用車云々，といった種類のパンフレットがあちこちで配布されている。それにオフコン，パソコンといった言葉も，毎日のように極くふつうのマスメディアを賑わせている。

これらは，マイコンによる技術イノベーション，あるいはより広く社会イノベーションを象徴する現象であるといってよいだろう。つまり，特定の分野に限定された技術，あるいは特別な訓練を受けた専門家だけに利用される技術という，従来型の技術の概念と枠を超えて，極めて広い分野の中に，そして市民の生活の中に直接入り込んでくるのが，マイコン関連技術の大きな特徴だからである。もちろん，マイコンは工業生産及び商用活動に関した多種多様の機器の中に基礎的な技術体系として，いわばプロフェッショナルの技術としても活躍の道を開いている。その場合も，あるときはコンピュータつまり情報処理の専用マシンとして用いられ，あるときは情報処理・制御の機能を付与する素子として，一般マシンの高度化，インテリジェント化に役割りを果たしている。

ひとくちにいって，こういった機能の一般性，用途の多様性が最大の特徴であり，従ってマイクロコンピュータあるいはマイクロプロセッサの技術・利用体系を要領よく概観するのは，容易なことではない。特に半導体の材料及び製法に関する技術は，まさに日変りの速さで進展していくので，先を見通した議論は偏見と誤ちをおかす覚悟なしにはできないだろう。以下の論述も一つの見方として受けとめて頂きたい。

2.1　素子技術の進歩と分化

マイクロプロセッサは，最初１９７１年末にPMOS4ビットのものが世に出て以来，集積度，語長，製造技術の点で，この１０年足らずの間に急速な進展を遂げ，機能の高度化と機種の多様化の道を歩んできた。すなわち語長別でみれば，４ビット，８ビット，１２ビット，１６ビット及びビットスライスに分かれ，内部はさらに細分できる。４ビット機はワンチップ，低価格，（準）専用の傾向をたどり，数量の上では断然大きな比率を占める。用途として最も多様化を発揮している８ビット機ではワンチップ，低価格の低位機種（Intel8048/8748，8021，Motorola 6700，Zilog Z－8など）と，高位機種（Intel 8085，Motorola 6800，Zilog Z-80 など）への分化の傾向が顕著である。また１６ビット機（Intel 8086，Motorola 68000，Zilog Z-8000 など）やビットスライス機では，ひとくちにいって，ミニコンとのソフト互換性を目指した高性能化（ミニコン化）が特徴である。

製造技術としては，ＰＭＯＳ（第１世代）から出発して，次第にＮＭＯＳ（第２～第３世代）が主流となり，７０年代終りにはＣＭＯＳも登場し，さらに１チップ化も目立つようになった。その間，数量的には少いが，バイポーラもビットスライス機用に使われて，順調な発展をみせている。

そして基本的なアーキテクチャが同一の，いわゆるファミリー機種の内部においても，利用目的に応じてメモリ種別，メモリ容量，命令数，入出力機能などを変えて，多様化をはかっている。

またＣＰＵ，メモリと比べてＬＳＩ化の遅れていた入出力インタフェースにもＬＳＩ化の努力が重ねられ，汎用入出力インタフェースＬＳＩ，専用入出力機器インタフェースＬＳＩ，システム・サポートＬＳＩなどの開発が盛んである。

８０年代に入る頃から第４世代と呼ばれる時代に入ったとされるが，４～１６ビットの高性能機種の１チップ化，そしてやがて３２ビット機種の本格的な登場も予測されるに至っている。今後の動向を正確に占うことは難しいとしても，およその傾向として次のようなことがいえそうである。まず，全体としての需要は，今後数年間，金額ベースで少くとも年率２０程度の伸びを示すであろう。省エネルギー，省力，それにインテリジエント化に対する産業界ならびに民生面からの要求が，需要の増大を生み出し続けるとみられるからである。

そのうち主流はＭＯＳ系，特にＮＭＯＳ系が占めよう。ＰＭＯＳは最も安価な用途，例えば玩具，ゲームなど一部の民生用機器に限られ，集積技術の進歩とともにＮＭＯＳはＰＭＯＳよりもむしろ安価となって，全面的に取って替わることも考えられる。ＣＭＯＳの機能も向上し，少電力消費の特徴を生かした部門に用途を拡げるであろう。それに対し，現在ビットスライス機に利用されているバイポーラ系は，８０年代半ばをピークとして，その後は漸減の傾向を示すであろう。高速度になった１６及び３２ビットＭＯＳ機にその活躍の座を譲るようになると考えられるからである。

次いで語長別ではどうだろうか。現在のところ４ビット機が数量的には優位を占めているものの，金額的には８ビット機の伸びが目立つようになっている。この傾向は今後の需要によって拡大され，やがて金額ベースで８ビット機が全体の半ばを占めるようになり，それとともに１６ビット機も伸びて，８０年代半ばには½以上の占有率となるものとみられる。そしてその頃から３２ビット機もかなりの比率を占めるようになり，次第に１６ビット機の用途に浸透していくであろう。その結果，８０年代の終り頃には金額ベースでの比率は８，１６，３２，４の順位となることが予想される。

この間，１チップ機種については４ビットの優勢は変わらないが，８ビットが相対的に増し，１６ビットもかなり現れよう。またマルチチップの価格は低下し続け，その傾向は特に１６，３２ビットについて著しく，３２ビットの対８ビット価格は恐らく１桁増し以内の程度におさまるものと思われる。

年代や価格，比率の絶対値はいろいろの予想でかなり幅があり，あまり確かなあてになるものはない。しかし大勢としては，１６ビット及び３２ビット機の低廉化が進み，８０年代の後半にはかなりの部門で８ビットから１６ビット，そしてさらに３２ビットへと長語長機種への移行が行われるという点では，多くの予想が一致した見方を示している。

それでは，どのような条件に基づいて長語長機種の選択（移行）が行われるのだろうか。一般的基準として考えられるのは，処理速度（特にオンライン性が要求されるとき），演算精度，メモリ空間の量，データ幅，割込の機能，外部通信機能などであろう。さらにソフトウェアの生産性，生産期間短縮，保守・拡張性なども考慮される。

その結果，オフィス・コンピュータ，パソナル・コンピュータなどでは，メモリ空間，割込，通信の点などが考慮され，１６ビットが最も優勢となろう。また機械制御，プロセス制御といった工業制御分野，ＭＥや一般分析などの計測分野では，上記のほとんどすべての項目が選択基準として考慮され，用途に応じて８，１６あるいは３２ビットのいずれかが採用されることになろう。その他自動車や高級事務機器の制御や診断，高級電卓やワードプロセッサ，通信・画像端末，通信機器，工業試験システムなどの中にもかなり高い比率で１６ビット機が浸透するとみられる。

2.2　利用の分類と体系

マイクロプロセッサは今日でも既に極めて多くの利用分野を実現しており，今後それはほとんどあらゆる産業，民生の機器ならびにシステムに及ぶことになるだろう。利用の形態を分類する仕方は，必ずしも一通りではない。最も基本的な分類としては，情報処理を目的とした利用と（広い意味での）制御を目的としたものとに大別することが考えられる。情報処理関連は下は電卓から上はかなり大形の汎用までのコンピュータ，それに通信情報処理，画像処理などといった専用の情報処理機器に分かれよう。また制御は民生・商業，機械制御，工業プロセス制御など，制御機能の種類とレベルに応じて分けることができる。そして計測・検査関連は情報処理と制御の中間あるいは両者にまたがるものといえよう。それらの概略をまとめたのが表２－１である。

表２－１　マイクロプロセッサの利用分類

情報処理的利用	汎用コンピュータ	一般電卓，科学技術用電卓
		（超）小形（パーソナル，オフィス，ビジネス）
		小，中形（ミニコン，スーパミニコン）
		中，大形（マルチプロセッサ・システム）
	専用機器	通信端末，画像端末，音声入出力装置，データロガ，会計機ＥＣＲ，ＰＯＳ，業務管理，生産管理，教育機器など
中間的利用	計測・検査	工業計測，実験計測，医用計測，交通計測，自動検査など
制御的利用	機械	シーケンス制御，数値制御，機械組立，ロボット，自動倉庫，ビークル制御など
	プロセス	化学プロセス，鉄鋼プロセス，電力系統，原子炉，送配水，廃水処理など
	交通・通信	空間的広がりをもつネットワーク・システムの制御
	民生・商業	簡単な機械またはプロセス制御，あるいはその組合せ，時には情報処理も含む。

ただし，これはかなり大づかみにした分類であって，たとえば自動車関連でもエンジンの燃焼制御は機械及び化学的な制御であり，安全チェックや警報は情報処理的な性格のものであり，細かくいえばひとまとめにはできないものである。また専用の情報処理装置乃至中間的とした装置では，まったく制御の機能を欠くわけではないし，逆に制御装置は情報処理の機能なしでは目的を達し得ないものである。よってこれらの分類は、機能のうち主たるものに注目して行ったものということができる。

現在わが国でよく引用される分類に，日本電子工業振興協会の報告書によるものがある。*）本報告書の後の章（第４章及び第５章）でもそれに準拠しているところがあるので，ここで触れておこう。その分類は，次の表２－２のようになっている。これは機能というより，利用の分野及び用途によった分類であるといえよう。従って機能分類の点ではやや異種の項目が混在した結

表２－２　マイクロコンピュータの応用分野と用途例

記号	応用分野	用途例		
A	民生・家電	1. 自動車	3. 電　卓	5. 教育（個人的）
		2. 家　電	4. 玩　具	6. その他
B	事務・商業	1. 事務・会計計算	3. ワード処理	5. 金融，端末
		2. 販売・在庫管理	4. ＥＣＲ，ＰＯＳ	6. 事務機，その他
C	工　業	1. 生産機械装置	3. プロセス制御	5. 生産管理
		2. 機械制御	4. データロガー	6. その他
D	交通・輸送	1. 信号制御	3. 駅務自動化	5. その他
		2. 運行制御	4. 航空機器	
E	計測・試験・監視	1. 測定器	3. 試験検査機	5. 医用電子
		2. 分析器	4. 監　視	6. その他
F	通　信	1. 有線通信	3. 画像通信	5. 放　送
		2. データ通信	4. 無線通信	6. その他
G	データ処理	1. 汎用コンピュータ	3. 科学技術計算	
		2. 周辺端末機器	4. その他	
H	その他	1. ビル管理	3. システム管理	
		2. ホテルシステム	4. その他	

＊）　日本電子工業振興協会；マイクロコンピュータに関する調査報告書―応用動向・ＩＥＣインタフェース・試験評価，５５－Ａ－１７１分冊　昭和５５年３月

果になっているのは否めない。ただし利用分野別は実用的にわかり易いという利点がある。

現在，製品の数量でいえば民生・家電分野が圧倒的に多い。これは直接のユーザの数が多いということから，当然ともいえよう。そしてそこで使われているプロセッサは，４ビット機がこれまた圧倒的である。しかし製品の種類の多さでは工業分野それに計測等の分野が多く，これらが技術集約型の利用分野であるといえる。また民生・家電以外の分野では８ビット機の占める比率が断然他を圧倒しており，現在のところ技術の主流は８ビット機にあることも歴然としている。

しかし2.1節でも述べたように，１６ビット機も特にデータ処理（コンピュータ）分野及び工業（制御）分野において着実な進出を果たしつつある。これらの分野における製品機能の向上に対する要求が強まったこと，あるいは低位機種のミニコンより経済性が優れることなどがその理由である。近来１６ビット機種の性能が安定したものとなり，価格面でも十分実用可能となってきた。さらに周辺機器や開発サポート体制も整えられてきたので，この一，二年の間に増加の傾向は急速に増幅されるすう勢にある。性能，機能面では，メモリアドレス能力が強化されたこと，命令体系が強化されたこと，特に乗除算などの命令による演算速度の向上，割込機能及びマルチプロセッサによる分散処理の命令が付加されたこと，システム・クロックも高速化されたことなどが，現在市場に出現しつつある１６ビット機の特徴である（代表的機種についての詳細は第３章参照のこと）。

端的にいえば，４ビット機利用分野の技術開発は，少くともマイクロプロセッサの側からみる限りあらかた終局をつげたのであり，現在の８ビットからさらに１６ビットへと利用技術開発の中心は移りつつあるといつてよい。すなわち資源，エネルギーの制約と経済の成長率低下という厳しい社会経済環境，さらにはますます競争とコンフリクトの激化する国際環境の中で，わが国の産業では今後とも生産及び事務・流通部門の省力化，合理化，情報化が求められる。それゆえ，プロセス計装や組立生産ラインの自動化，自動倉庫など，従来からわが国の得意としてきた分野にもますます拍車がかかり，さらに今までは比較的立遅れていた高度の生産管理，自動設計，オフィス・オートメーション，流通システムなどの管理・流通部門に対する技術要求が増す。さらに社会的部門ともいうべき通信，交通・輸送，社会情報サービス，医療，教育などの分野にも，多様な要求が生まれる。要するにより高度な産業・経済・社会的要求が，内生的（選好の継続的発展）ならびに外生的要因によって醸成され，それが簡便，低価でしかも質の高い情報処理，制御技術の開発を促すのである。

過去の例にもみられたように，低位量産品のみではやがて中進国，発展途上国に席をゆずらざるを得ず，わが国としてはより高度な技術による高付加価値製品の開発へと進まねばならない。しかも本当は，ハードとソフトを含めた独自のイノベーションを生みだすことが必要であり，いつまでもセカンドソースの地位に甘んじていてよいのかどうか，真剣に考えるべき時点にさしかかっている。なお１６ビット・マイクロプロセッサの応用分野と利用動向のより具体的な内容については，第４章を参照されたい。

2.3　利用技術における問題点と展望

マイクロプロセッサの利用が量，質ともに拡大するにつれ，技術上解決すべき問題点もいくつか出て来ている。もちろん素子自体の処理速度，使用温度条件なども利用上の限界を規定していることは否めない。しかし，より多くの" システムとしての問題点 "があることに注目しなければならない。

システムの問題点は，ハードウェアとソフトウェアの２面に大別できる。ハードウェアの問題点としては，まず第一に規格，仕様の統一化，標準化の必要性があげられる。プロセッサや周辺機器が多様化するにつれて，ユーザ側からはハード及びソフトの互換性の要求が強まるのは当然の成りゆきといえよう。メーカの系列ごとに機種の系列化が進展しつつはあるが，部品的機能素子として使われるためには，一層の標準化が進められねばならない。しかし各系列のメーカがそれぞれに独自設計思想と技術をもち，またかなり膨大な応用商品群を既に開発しているという現実がある。しかも技術内容自身が日進月歩的に変化しつつある状況下においては，画一的な標準化は極めて多くの困難を伴うであろう。強引な標準化が混乱と開発意欲の停滞をもたらすようでは，効果はマイナスになってしまう。

そこで次のような方策が考えられる。一つは，あらかた技術開発のピークを過ぎたような機種，現状でいえば４ビット機種について，まず規格の標準化をはかることである。技術内容が一応の定着をみたものでは標準化の可能性が高く，また数量的にも依然として優位が予想されるので，必要性も高いといえる。４ビットの次は８ビットへと漸次進める方策が考えられるわけである。二つには，将来開発と利用の進展する機種についても，全面的な標準化でなく，基本的なところからそれを進める方策を考えることである。例えば１６ビット機種といえども，４ビット，８ビット機種の延長上に展開されるわけであるから，ピン数，ピンの種類，配置などについても，ある範囲内で仕様の共通化をはかることは不可能ではないと思われる。それには，将来機種は主としてどういう利用形態の可能性が高いかというように，使われ方の予測について議論を交わし，その認識の共通部分を洗い出すことが必要であり，これも不可能なことではない。

ハードウェアの第二の問題点として，周辺機器技術の相対的な立遅れがあげられる。周辺機器といってもその範囲は広く，まずプロセッサチップ周辺の外部記憶装置，インタフェース，次に人間とのインタフェースである入出力機器群，さらに外部環境とのインタフェースとしてのセンサ及びアクチュエータの３種に大別される。そのうち，外部記憶（ディスク，フロッピーディスク，カセットテープなど）からインタフェース関係は最も進んだ部分であり，あまり問題はない。入出力機器（キーボード，ＰＴＲ；プリンタ，ＣＲＴなど）も最近かなり低廉化が進み，今後利用の拡大につれてこの傾向は加速されよう。最も問題が残されているのはセンサ及びアクチュエータ技術である。これらはいわばコンピュータ外の技術であるから，一概には論じられないが，民生・家電，工業，交通・運輸，計測・試験・検査，医用など，広範囲の応用分野においてボトルネックになっているのが現状である。

システム・センサの出力は電気信号であるが，多くのセンサでは他の物理・化学量，すなわち位

量，ひずみ，速度，加速度，圧力，流量，熱，温度，光，放射線，超音波，磁気，質量，組成などの諸量が，電圧，電流あるいは周波数に変換される。その際，原理的には基本的な物理・化学変換原理または変換効果が用いられ，具体的には半導体，磁性体，誘電体などの比較的新しい固体素子の物性変化を利用するものがふえている。従って，感度がよく雑音の少いセンサ用の固体素子及びその材料の開発が基本的課題である。

センサ技術が最も要求されるのは制御的利用においてである。プロセス制御でもセンサには精度，信頼性，価格，操作性，保守性に優れた特性が要求されるが，機械制御では加えて特に小形，軽量，即応などの要求が厳しい。プロセス・システムはそれ自体が大きく，また比較的大きな時定数をもっているが，機械システムは精密な機構をもち．即応性が重要なポイントとなることが多いからである。最も有望なのはやはり半導体センサである。その他，光センサと光伝送を用いた光計装システム，そして光コンピュータの将来にも注目すべきであろう。さらに興味のある課題としては，生体における感覚器官，特にその複合機能，パターン計測機能，センサ・フィードバック機能などの解明と利用があげられる。

ハードウェアに比べてソフトウェアにはより多くの問題点が存在する。応用商品開発の観点からは，ハードとソフトの機能面と価格面とのバランスのとり方が設計上の一つのポイントであるが，利用の拡がりを考えると，ソフトの比重は将来ますます高まるであろう。大形コンピュータの場合でもソフトの生産性の伸びの低さ（現状ではせいぜい年３～４％といわれる）は大きな問題であるが，システム開発者及びエンド・ユーザに多くの負担がかかるマイクロプロセッサ利用の場合，事情はより深刻である。ソフトの標準化，モジュール化，高位言語利用，その標準化・簡易化と普及，生産工程管理の改善，開発サポート・システムの整備，そして教育・訓練体制の見直しなど，数々の問題が解決を迫られている。

１６ビットから３２ビットといつた高位機種がかなりのウエイトを占めるとみられる８０年代において，わが国の立場はいかにあるべきか。まずソフトの開発体制の整備と改善，そのための標準高位言語開発の可能性を検討すべきであるが，マイクロコンピュータ技術においてはソフトとハードはかなり強固に結びついていることを考えるとき，既存のチップ体系とのコンパティビリティ，独自のチップ体系開発の可能性をも含めて，より深い検討を迫られることになるかもしれない。

第3章　16ビットマイクロコンピュータの現状

第3章　16ビットマイクロコンピュータの現状

3.1　16ビットマイクロコンピュータ出現の背景

語長16ビットの計算機分野は，マイクロコンピュータが出現する以前から，ミニコンピュータの技術が確立していた分野である。新しく半導技術の進歩により，16ビットのマイクロプロセッサが開発され，16ビットマイコン分野が出現してきたが，同じマイコンでも4ビットマイコンなどとは性格に大きな差異がある。それは4ビットマイコンと電卓の関係と16ビットマイコンとミニコンピュータの関係を比較すればわかる。4ビットマイコンは知能部品としてコントローラ分野への展開に成功し，母体となった電卓LSIの枠を破って新天地を開拓していった。従って現在においては4ビットマイコンと電卓とは直接的には何の関係も無くなっている。これに対して16ビットマイコンはまだミニコンコンピュータの枠を破ることができず，むしろミニコンピュータに置きかわる傾向を見せている。従ってミニコンピュータ技術は非常に強い影響を与えている。

16ビットのLSIプロセッサには2つの考え方の系統がある。1つはマイコンとして設計開発され，LSIが販売されているもので，16ビットマイコンである。マイコンの歴史は1971年に4ビットの4004がインテル社（Intel Corp.）から発表された時に始まる。その後1972年に最初の8ビット機種である8008が発表され，1973年には8080が発表されてはやくも第2世代に入った。16ビットマイコンは8ビット機種の次のステップとして1974～76年ごろにかけて，GI社（General Instrumeuts Corp.）のCP1600，ナショナルセミコンダクタ社（National Semiconductor Corp.）のPACE，パナファコム社のPFL－16A，日本電気のμCOM16，TI社（Texas Instruments Inc.）のTMS9900など多数の機種が発表された。この中ではTMS9900がミニコンピュータと系列を構成し，ソフトウェアの互換性をとっていることを特徴として成功している。それ以外の16ビットマイコンは需要量もあまり伸びず，一部の専用用途を別にして一般市場から消えた。したがって16ビット機種は，この第2世代の時期に8ビット機種のように標準的機種が固まるまでに至らなかった。このような状態になっている理由として考えられることは，この時期に開発された16ビット機種は，若干の改良はあるにしても基本的には8ビット機種と性能的に同クラスのマイコンであったことによる。このため特徴が十分発揮されず，需要が開拓できなかった。

これに対して1978～80年にかけて，新設計の16ビットマイコンが相ついで発表された。これらの機種は以前の16ビット機種の性能レベルをやぶったもので，第3世代機種と言えるものである。これらの機種はLSI技術の進歩と従来のマイコンの性能限界をこえるアーキテクチャの採用により，性能的にはミニコンピュータと同レベルの能力をもつことが特徴であり，8ビット機種との間には明確な性能上の差がついたので，16ビット機種としての特徴がはっきりしてきた。性能上の特徴の例としては

① 高速であること，例えばレジスタ間加算時間は1μs以下になっている。

② 大容量メモリ空間をもつこと，従来の64Kの枠をやぶり，1Mないし16Mバイトまでア

クセスできる機構を備えている。

③ レジスタ数の増加，命令の強化など演算能力が強化されている。

④ アドレシングモードが充実している。

⑤ マルチプロセッサシステム化の考慮が払われている。

⑥ 命令先取りなど高度の制御方式が採用されている。

⑦ 高性能ＭＯＳ技術が使用されていて，高集積度ＬＳＩである。

などがある。

１６ビットマイコンの良否の問題の一つにソフトウェアの問題がある。ソフトウェアの役割の重要性は８ビット機種でも言われるが，１６ビット機種ではさらに重要であることはミニコンピュータを見ればわかる。ミニコンピュータ用ソフトウェアが使用できるＴＭＳ９９００のような場合はよいが，ソフトウェアが揃わないために実用にならなかった機種もいくつかある。新機種もハードウェアの性能としてはミニコンピュータに匹敵する性能のプロセッサになっているが，ソフトウェアについては未だ十分なものを持たないものもある。これから先のソフトウェアの充実ぶりが普及の鍵を握っているとも言える。

以上に述べた新機種の１６ビットマイコンには，日本電気μＣＯＭ１６００，インテル社8086／８０８８，ザイログ社（Zilog Inc.）Ｚ８０００，モトローラ社（Motorola Inc.）ＭＣ６８０００，松下電器ＭＮ１６１３，東芝Ｔ８８０００などがある。設計思想的にはこれまでのマイクロコンピュータの上位機種という考えのものと，ミニコンピュータ対抗機種（性能的に同等以上を狙ったもの）という考えのものがあり，さらに後者は従来のミニコンピュータとは独立のものと，ソフトウェアの互換性のある系列のものがある。前者の系列のものは8086／8088，μCOM１６００，ＭＮ１６１３などであり，後者の系列のものにはZ8000，MC68000，T88000などがある。これらの機種の内容については３．２節で説明される。

１６ビットＬＳＩプロセッサにはもう一つの系統として，既存のミニコンピュータのＣＰＵをＬＳＩ化したものがある。これらは通常ミニコンピュータのファミリの下位機種としてボードコンピュータなどの形で販売されている。１９７５年～７７年にかけて開発されたＤＥＣ社（Digital Equipment Corp.）のＬＳＩ－１１シリーズ，データジェネラル社（Data General Corp.）のMicro　ＮＯＶＡ，東芝のＴＯＳＢＡＣ－４０Ｌなどはこの例である。この種のプロセッサは技術的にはマイクロプロセッサと全く同じであるが，ＬＳＩとして販売されていないので，一般的な意味でのマイクロコンピュータには含めないことが多く，ＬＳＩミニコンという特別の名前で呼ばわれることもある。代表的なものについては３．２節で説明する。

3.2　代表的１６ビットマイクロコンピュータの特徴

3.2.1　μＰＤ８０８６／８０８８

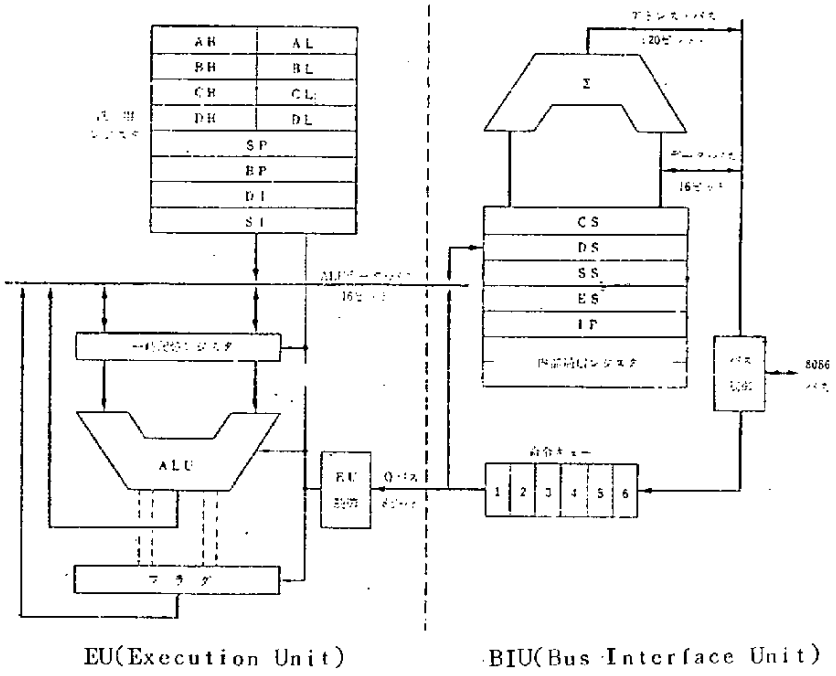
８０８６は，高性能の１６ビットマイコンで，８ビット，１６ビットの両方のマイコンの機能

を備えており，メモリ，Ｉ／Ｏに対して１６ビットのデータバスを備えている。

これに対し８０８８は，内部の構成は８０８６とほとんど同じで，メモリ，Ｉ／Ｏに対して８ビットのデータバスを備えた外部８ビット，内部１６ビットのマイコンである。

(1) ハードウェアの構成と特徴

８０８６／８０８８の内部ブロックは大きく２つのブロックから構成される。１つは，データレジスタ，ＡＬＵ，命令デコーダ等を含んで，実際に命令を実行するＥＵ（Execution Unit）であり，もう１つは，セグメントレジスタ，命令キュー等を含み，ＥＵで必要となる前にあらかじめ命令をプリフェッチしたり，命令実行上必要になるメモリ，Ｉ／Ｏとのデータ転送の物理アドレスを計算して，データ転送を行うＢＩＵ（Bus Interface Unit）である。８０８６と８０８８の違いは，後者のＢＩＵが１６ビットデータバスとなっているか，８ビットデータバス



EU(Execution Unit)　　BIU(Bus Interface Unit)

図３－１　８０８６内部ブロック図

となっているかだけである。

(a) レジスタ構成

８０８６／８０８８は，ＥＵ内部に，演算論理処理に利用する１６ビットの汎用レジスタを４つ（ＡＸ，ＢＸ，ＣＸ，ＤＸ），後述するスタックセグメントレジスタと組合せてメモリアドレスを指定する１６ビットのポインタを２つ（ＢＰ，ＳＰ），後述するデータセグメントレジスタと組合せてメモリアドレスを指定するインデックスレジスタを２つ（ＳＩ，ＤＩ）および算術，論理演算の結果を記憶したり，ＣＰＵの動作を制御したりする９ビットのフラグレジスタを持っている。

ＢＩＵ内部には，後述するコードセグメントレジスタと組合せて，ＢＩＵがつぎに読出すべき命令のストアされているアドレスを示す１６ビットのインストラクションポインタ，読出した命令をＥＵが必要になるまでの間ストアしておく６バイトの命令キューを持っている。また

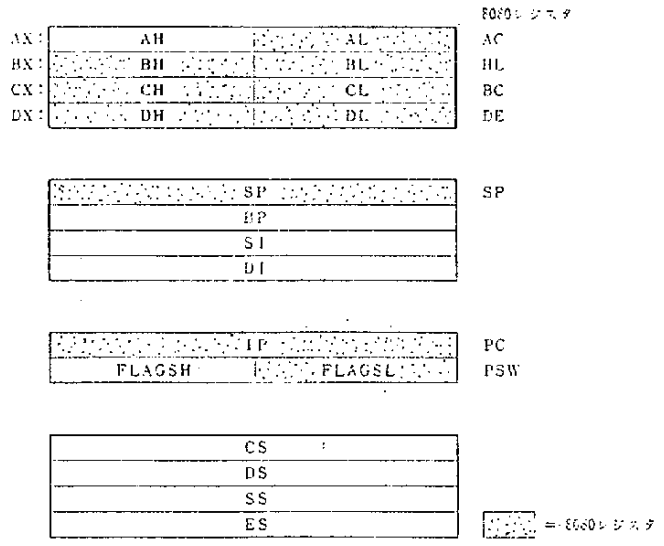


図３－２　８０８６内部レジスタ

ＢＩＵ内部には，前述の汎用レジスタ，ポインタ，インデックスレジスタ，インストラクションポインタを組合せて２０ビットのアドレスを指定するための１６ビットのセグメントレジスタを４つ（ＣＳ：コード，ＤＳ：データ，ＳＳ：スタック，ＥＳ：エキストラ）持っている。セグメントレジスタの内容を４ビット上位にシフトし，組合せるレジスタの内容と加算を行って２０ビットのアドレスを計算する。

(b) メモリ構成

８０８６の場合，メモリは，ＣＰＵのアドレスラインＡ１９～Ａ１によって並列にアドレスされる５１２Ｋバイトの上位バンク（Ｄ１５～Ｄ８）と下位バンク（Ｄ７～Ｄ０）により構成される。

そしてＡ０，$\overline{\text{ＢＨＥ}}$により，上位バンク，下位バンクまたは両方に対して選択的にリード／ライト動作を行う。

８０８８では，アドレスラインＡ１５～Ａ０でアドレスされる１Ｍバイトの単一バンクにより構成される。

(c) ＭＩＮモードとＭＡＸモード

８０８６／８０８８は，２つの動作モードを持っている。１つはＣＰＵを１つしか使用しない場合に用いるＭＩＮモードであり，この時にはメモリ，Ｉ／Ｏへの制御信号はＣＰＵから直接出力される。もう１つは複数のＣＰＵを使用する場合あるいは補助プロセッサ（８０８９，８０８７）を使用する場合に用いるＭＡＸモードであり，この時にはメモリ，Ｉ／Ｏへの制御信号は，バスコントローラ８２８８から出力される。

(2) ソフトウェアの特徴

８０８６／８０８８のソフトウェア上の大きな特徴は，相対アドレスジャンプやダイナミックリロケートが可能となったことである。

また，１Ｍバイトまでのメモリは，各セグメントレジスタにより最大６４Ｋバイトのコード，データ，エキストラデータ，スタックの各セグメントとして扱うことができる。

１６ビットＣＰＵにありがちな１６ビットデータのアドレス配置の制限は８０８６にはない。これは，ＢＩＵが必要に応じて１回あるいは２回のリード／ライト動作を行うことにより可能となっている。その他ストリング機能が強化されたことと，ＢＣＤ，ＡＳＣＩＩの各種演算機能も追加されている。

(a) アドレシングモード

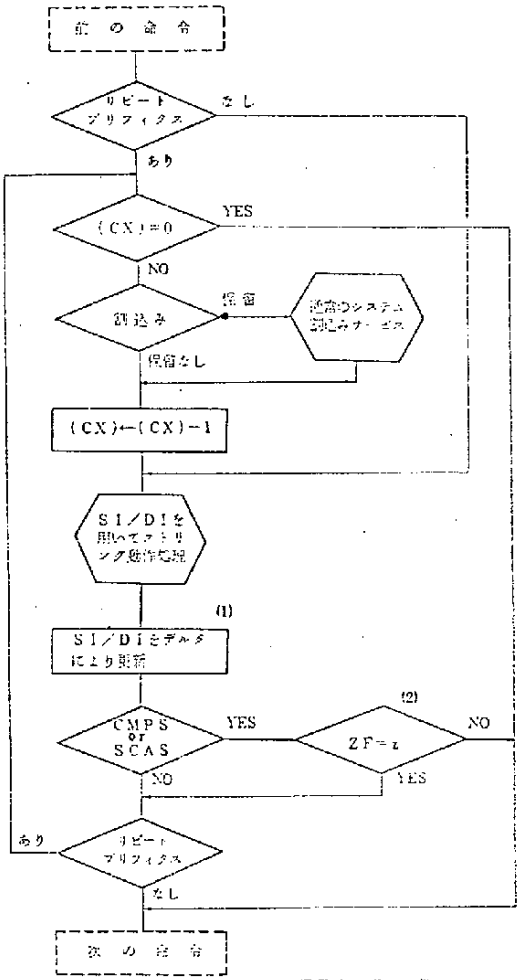
８０８６／８０８８には，アドレスを直接指定するダイレクトアドレシング，レジスタにより間接指定するレジスタ・インダイレクトアドレシングに加えて，レジスタＢＸ，ＢＰの内容と命令の後に付加されているディスプレイスメントの和でアドレスを指定するベーストアドレシング，インデックスレジスタＳＩ，ＤＩの内容とディスプレイスメントの和でアドレスを指定するインデックストアドレシングがある。また，レジスタＢＸ，ＢＰと，インデックスレジスタＳＩ，ＤＩおよびディスプレイスメントの３つの和でもってアドレスを指定するベースト・インデックストアドレシングもある。

どのアドレシングモードもこれらの方法で与えられた１６ビットのデータに，セグメントレジスタの内容を４ビット上位にシフトしたデータを加算して物理的な２０ビットアドレスを生成する。

(b) ストリング動作

８０８６／８０８８には，ＳＩ，ＤＩを用いてある動作を繰返し行うストリング機能がある。

MOVS, CMPS, SCAS, LODS, STOSなどのストリング命令の前にREP, REPZ, REPNZなどのリピートプリフィックスを付加することにより，ＣＸレジスタを－１していき，０になるかあるいはＣＭＰＳ，ＳＣＡＳの場合には条件が満足されるまで指定された動作を繰返す。ストリング動作の場合，ソースアドレスは，ＳＩ，ディストネーションアドレスはＤＩにより指定される。またＳＩ，ＤＩの内容はＤＦ（フラグレジスタ中のディレクションフラグ）の状態およびデータのタイプ（バイトまたはワード）により，自動的に±１あるいは±２される。



備考1. デルタ値

ストリング	DF	デルタ
バイト	0	1
バイト	1	−1
ワード	0	2
ワード	1	−2

備考2.

プリフィクス	z
REPE	1
REPZ	1
REPNE	0
REPNZ	0

図３－３　ストリング動作

(3) システム構成

８０８６／８０８８には，クロックを発生するクロックジェネレータ８２８４，８ビットラッチ８２８２／８２８３，８ビット双方向性バスバッファ８２８６／８２８７，ＭＡＸモード時メモリ，Ｉ／Ｏ等に制御信号を出力するバスコントローラ８２８８，マルチＣＰＵで動作させる場合バス利用の優先度をコントロールするバスアービタ８２８９等の周辺チップがある。

また，８０８６／８０８８と組合せて使用する補助プロセッサとして，主に入出力の管理を行いＤＭＡの制御機能も持った８０８９，浮動小数点演算等，演算動作を高速で行う８０８７等がある。

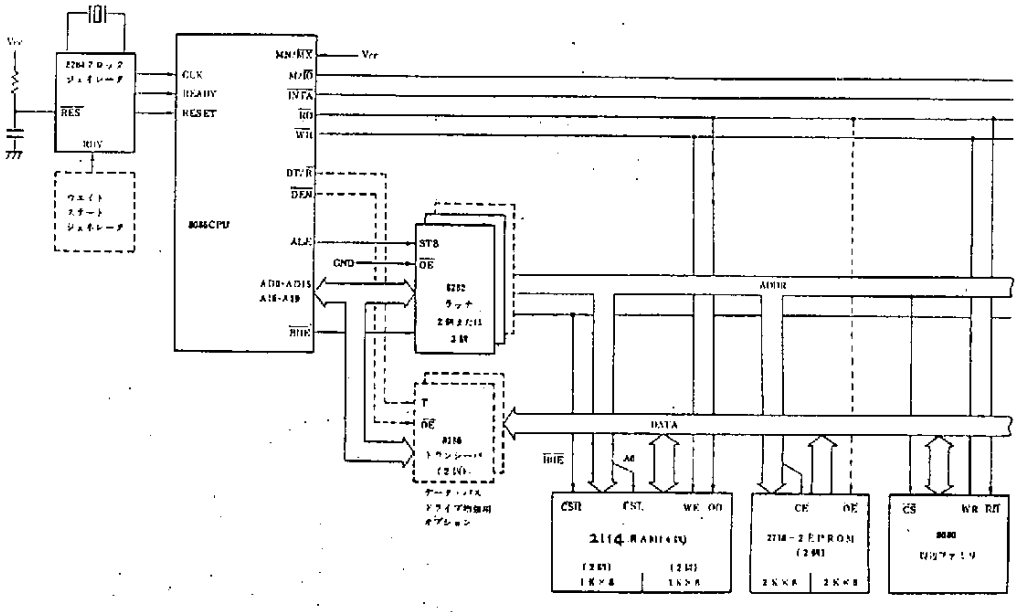


図３－４　ＭＩＮモードシステム構成例

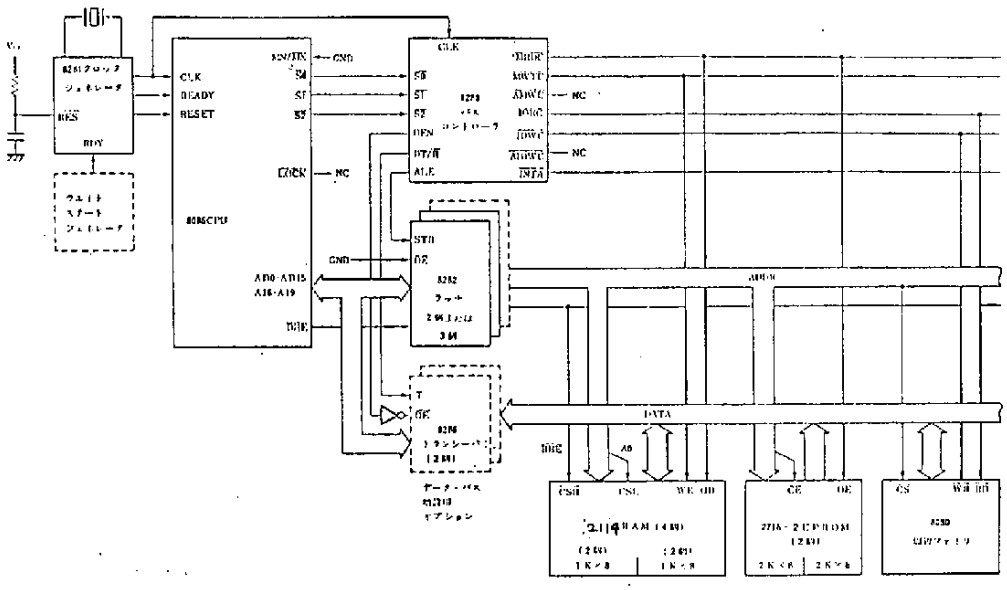


図３－５　ＭＡＸモードシステム構成例

(4)　開発サポート

８０８６／８０８８のソフトウェアの開発は，フォートランベースあるいはＰＤＡ－８８０ベースのアセンブラ，リンカーにより行う。

８０８６／８０８８のニーモニックコードで書かれたソースプログラムは，アセンブラにより，リロケータブルなオブジェクトプログラムに変換される。そして複数のリロケータブルなオブジェクトプログラムを，リンカーにより１つのアブソルートなプログラムに変換し実際にプログラムを実行させる。

なお，８０８６／８０８８のファースト・ソースメーカのインテル社からは，ＭＤＳベースのアセンブラＡＳＭ－８６，コンパイラＰＬＭ－８６，リンカーＬＩＮＫ－８６，ロケータＬＯＣ－８６，ハードエミュレート用としてＩＣＥ－８６，ＩＣＥ－８８等が供給されている。

〔参考文献〕

(1) μCOM86ユーザーズ・マニアル

(2) μCOM86ペーパーマシン

(3) The 8086 Family Users Manual

3.2.2 HMCS68000

(1) ハードウェアの構成と特徴

(a) レジスタ（図3－6，図3－7参照）

- 32ビットのデータ・レジスタ　8個

  8ビット，16ビットのデータの処理も可能

- 32ビットのアドレス・レジスタ　9個

  16ビット・アドレス処理も可能

  7番のレジスタは，スタック・ポインタとして使われるが，ユーザ用とスーパバイザ用とで別々に用意されている。

- 24ビットのプログラム・カウンタ　1個

- 16ビットの状態表示レジスタ　1個

(b) 命令形式とアドレシング（図3－8参照）

実効アドレス・フィールドで指定される3つのモードによる次の12種のアドレシングが可能である。実効アドレスはバイト・アドレスを指す。

- レジスタ直接モード
- 次の5種のメモリ・アドレス・モード
  - ① レジスタ間接
  - ② ポスト・インクリメント・アドレス・レジスタ間接
  - ③ プリ・デクリメント・アドレス・レジスタ間接
  - ④ ディスプレースメント付きアドレス・レジスタ間接
  - ⑤ インデクス付きアドレス・レジスタ間接
- 次の6種の特殊アドレス・モード
  - ① 短絶対アドレス（1語で指定）
  - ② 長絶対アドレス（2語で指定）
  - ③ ディスプレースメント付きプログラム・カウンタ相対
  - ④ インデクス付きプログラム・カウンタ相対
  - ⑤ イミディエート・データ
  - ⑥ CCR／SR

(c) データ

- 1語のビット・ストリング
- 1バイト，1語，2語の整数
- 2進化10進数
- 文字列

なお，高位言語（FORTRAN等）によって，2語，4語の浮動小数点表現（IEEE標準案に準拠）による実数値計算も可能である。

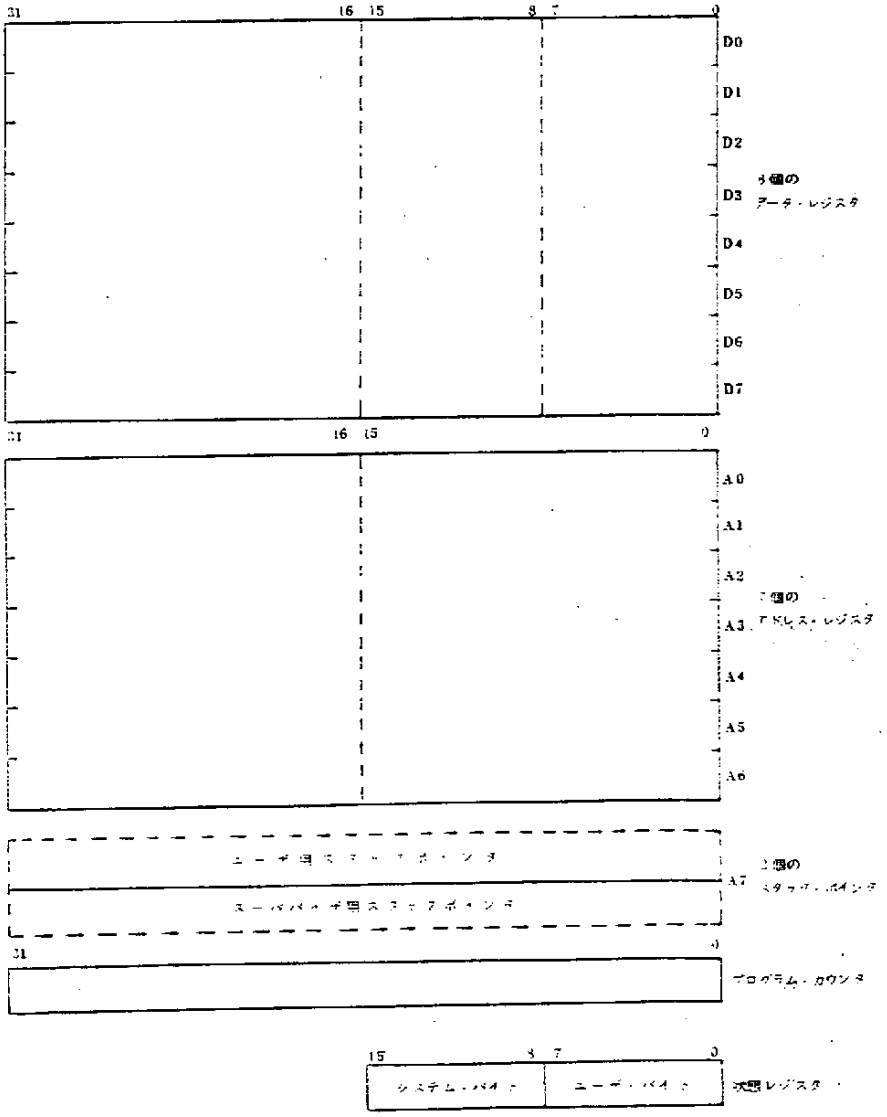


図3－6　HD68000のレジスタ構成

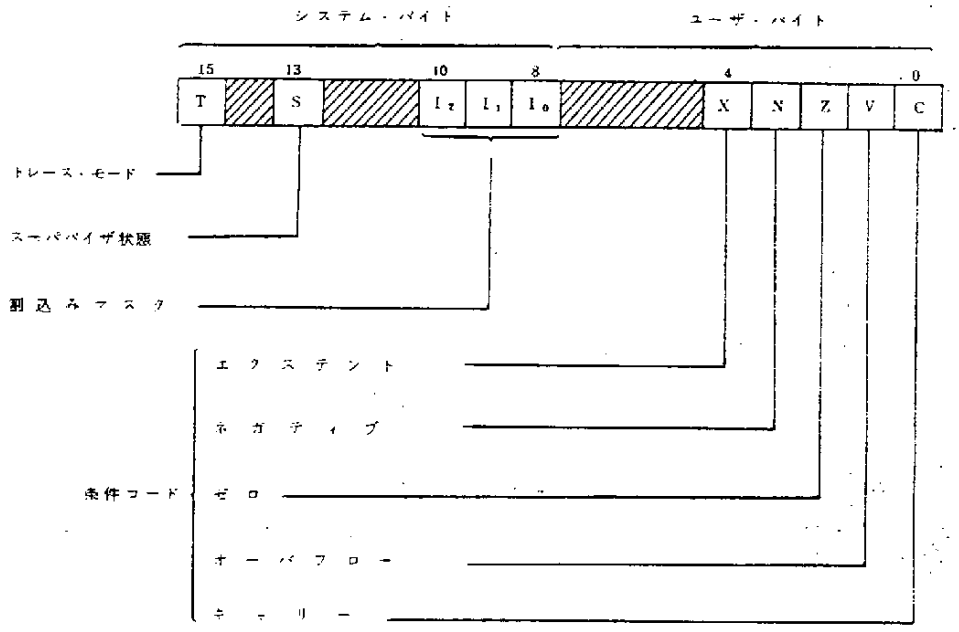


図３－７　状態レジスタ

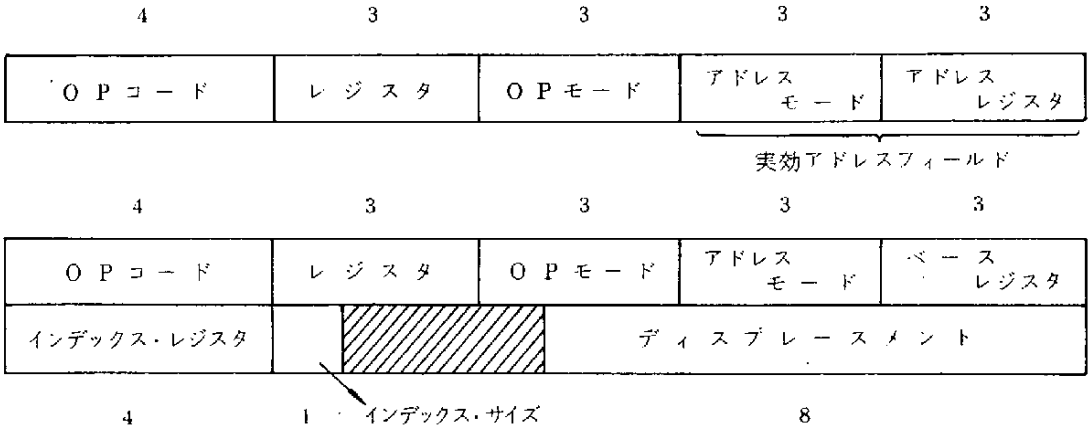


図３－８　命令語構成例

(d) ハードウェアの特徴をまとめると以下の通りである。

- 1～8MHz クロック
- 24ビット・アドレス
- 豊富な命令セット（76種）
- 32ビット・データ・レジスタ（8個）
- 32ビット・アドレス・レジスタ（9個）
- 豊富なデータ型
- 多様なアドレス方式
- 7レベルの割込み，255個の割込みベクタ
- 拡張性（浮動小数点，文字列，ユーザ・コード等）

(2) ソフトウェアの特徴

(a) クロス・システム

次の，HITACMシリーズによるクロス・システムがある。

- Pascalコンパイラ
- スーパPL／Hコンパイラ（PL／M86＋α）
- マクロ・アセンブラ
- リンケージ・エディタ
- シミュレータ

(b) レジデント・システム

- Pascalコンパイラ
- FORTRAN
- スーパPL／H
- マクロ・アセンブラ
- リンケージ・エディタ
- シンボリック・デバッガ
- CRTエディタ
- ユーティリティ（ファイル変換等）

(3) システム構成

MPU：HD68000（16ビット・マイクロ・プロセッサ）

DMAC：HD68450（ダイレクト・メモリ・アクセス・コントローラ）

MMU：HD68451（メモリ・マネージメント・ユニット）

IPC：HD68120（8ビット・インテリジェント・ペリフェラル・コントローラ）

その他，次のHMCS６８００用周辺ＬＳＩが接続可能である。

ＰＩＡ：ＨＤ４６８２１

ＦＤＣ：ＨＤ４６５０３Ｓ（フロッピーディスク・コントローラ）

ＣＲＴＣ：ＨＤ４６５０５Ｒ

ＡＣＩＡ：ＨＤ４６８５０

ＳＳＤＡ：ＨＤ４６８５２

ＣＰＧ：ＨＤ２６５０１

(4) 開発サポート

次の特徴を有するシステム開発装置Ｈ６８０ＳＤ３００が用意されている。

- ＨＤ６８０００ＭＰＵ用の完全なシステム開発装置
- １４インチディスプレイコンソール
- 両面単密度フロッピーディスク装置（０.５ＭＢ）２台標準装備
- １８０字／秒，１３２桁シリアルプリンタ
- モトローラ社標準ＶＥＲＳＡバス使用
- 外部システムとの交信用ポート有
- フロントパネル上にシステムの状態／エラー情報を表示
- マルチプロセッサ用のバス管理
- ＣＲＴディスプレイ上で編集ができるＣＲＴエディタ
- リロケータブルマクロアセンブラ
- 高位言語Ｐａｓｃａｌ，ＦＯＲＴＲＡＮ，スーパＰＬ／Ｈ
- 自己診断機能をファームウェアに内蔵

ＳＤ３００のＨＤ６８０００用ソフトウェアパッケージは次のものから構成される。

- 標準パッケージ

(a) ＦＤＯＳ

フロッピーディスク・オペレーティングシステム

(b) マクロアセンブラ

リロケータブル（再配置可能）なオブジェクトを出力する。

(c) リンケージエディタ

(d) ＣＲＴエディタ

ソースプログラムの作成，変更をＣＲＴ画面上で行う。ページモードとコマンドモードに使い分けられる。

- オプション

(e) ＦＯＲＴＲＡＮコンパイラ

ＦＯＲＴＲＡＮ７７のサブセット仕様にビット処理機能を追加したもの

(f) スーパＰＬ／Ｈコンパイラ

システム記述言語であり，能率のよいプログラムを短時間に作成できる。

(g) Pascalコンパイラ

標準Pascalに，次の拡張を行った。

① 変数の絶対アドレスによる割付け

② 名前形式のラベル

③ ストリング型

④ ＥＸＩＴ文

⑤ ２進，１６進定数

⑥ 実行時エラーのチェック

⑦ 実行時ファイルの割当て

⑧ 分割コンパイルとリンク

3.2.3 Ｚ８０００

(1) ハードウェアの構成と特徴

Ｚ８０００は高機能１６ビット・マイクロプロセッサの代表的なものであり，次のような特徴を持つている。

全体のアーキテクチャの設計思想として，次の点が特筆される。

① １６ビット・ミニコンのアーキテクチャをベースとして，３２ビットへの今後の拡張をも配慮している。

② ＩＢＭとＤＥＣのアーキテクチャの特徴が取入れられている。

③ 汎用システムや高位言語をベースとするシステムへの適合性が配慮されている。

これらを具体的に列挙すると下記のような特徴があげられる。

① Ｚ８０００にはノンセグメントタイプのもの（Ｚ８００２）とセグメントタイプのもの（Ｚ８００１）の２種のＣＰＵが用意されており，ソフトウェア・コンパチブルである。

② アドレス空間が大きい。８メガバイトまで直接アドレッシングが可能で，工夫により，さらに拡張が可能である。

③ １６ビットの汎用レジスタ１６個を持っている。

④ 整然とした割込みやトラップ処理のアーキテクチャを持っている。

⑤ システムモード／ユーザモードの区分，特権命令の機能があり，上述③，④の特徴と合わせて，リアルタイム，マルチプログラミング・システムを構成するのに適している。

⑥ ８種のアドレスモードを持っている。

⑦ 命令が豊富である。（基本１１０種，アセンブラのニーモニックで１９１種）

⑧ ７種のデータタイプを持っている。（ビット，バイト，デジット，ワード，ロングワード，バイトストリング，ワードストリング）

⑨ 倍精度の乗除算命令をハードウェアで持っている。

⑩ ブロック転送およびストリング処理の命令群を持っている。

⑪ Ｚ８０１０メモリマネジメントユニットと組合わせ，ダイナミック・メモリ・リロケーションが行える。

⑫ ５Ｖ単一電源

⑬ ４ＭＨｚ～６ＭＨｚ 単相クロックなど。

Ｚ８０００のＬＳＩは，シリコンゲートＮＭＯＳ技術により，１７５００ゲート，チップサイズ約7㎜角となっている。Ｚ８０００のピン構成は，図３－９のようになる。枠内に示したように，Ｚ８００１とＺ８００２のピンの違いは，セグメント番号およびセグメントトラップの7本である。

レジスタ構成は図３－１０のようになっている。１６個の１６ビット汎用レジスタ（Ｒ０～Ｒ１５）は，命令の語長の指定により１６個のバイトレジスタ（RH0~RH7，ＲＬ０～ＲＬ７），8個の32ビットレジスタ（ RR0~ＲＲ１４），１４個の６４ビットレジスタ（ＲＱ０～ＲＱ１２）としても使用することができる。ただし，バイトレジスタとして使用できるのは，前半8個の汎用レジスタ（ Ｒ０～Ｒ７）のみである。

また，Ｒ１４とＲ１５はスタックポインタとして設計されており，Ｒ１４がセグメント番号を，Ｒ１５がセグメント内のオフセットを示すようになっている。さらにユーザ用（ＮＯＲＭＡＬ）とシステムとの2セットが用意されている。

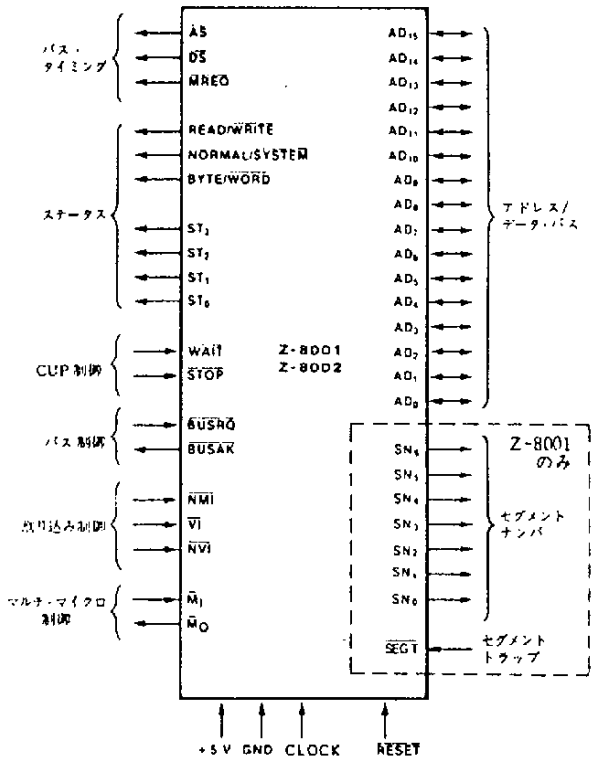


図３－９ Ｚ８０８０のピン構成

プログラム・ステータスは４個の１６ビットレジスタで表わされており，プログラムの状態を示すフラッグ・レジスタと，プログラム・カウンタを示すＰＣレジスタ（共に２個づつ）で構成されている

新プログラムステータス領域ポインタは，７ビットのセグメント部と８ビットの上位オフセッ

トから成り，２５６バイト単位で新プログラム・ステータス領域を設定できる。リフレッシュ・レジスタを用いてリフレッシュの間隔をソフト的に設定できる。

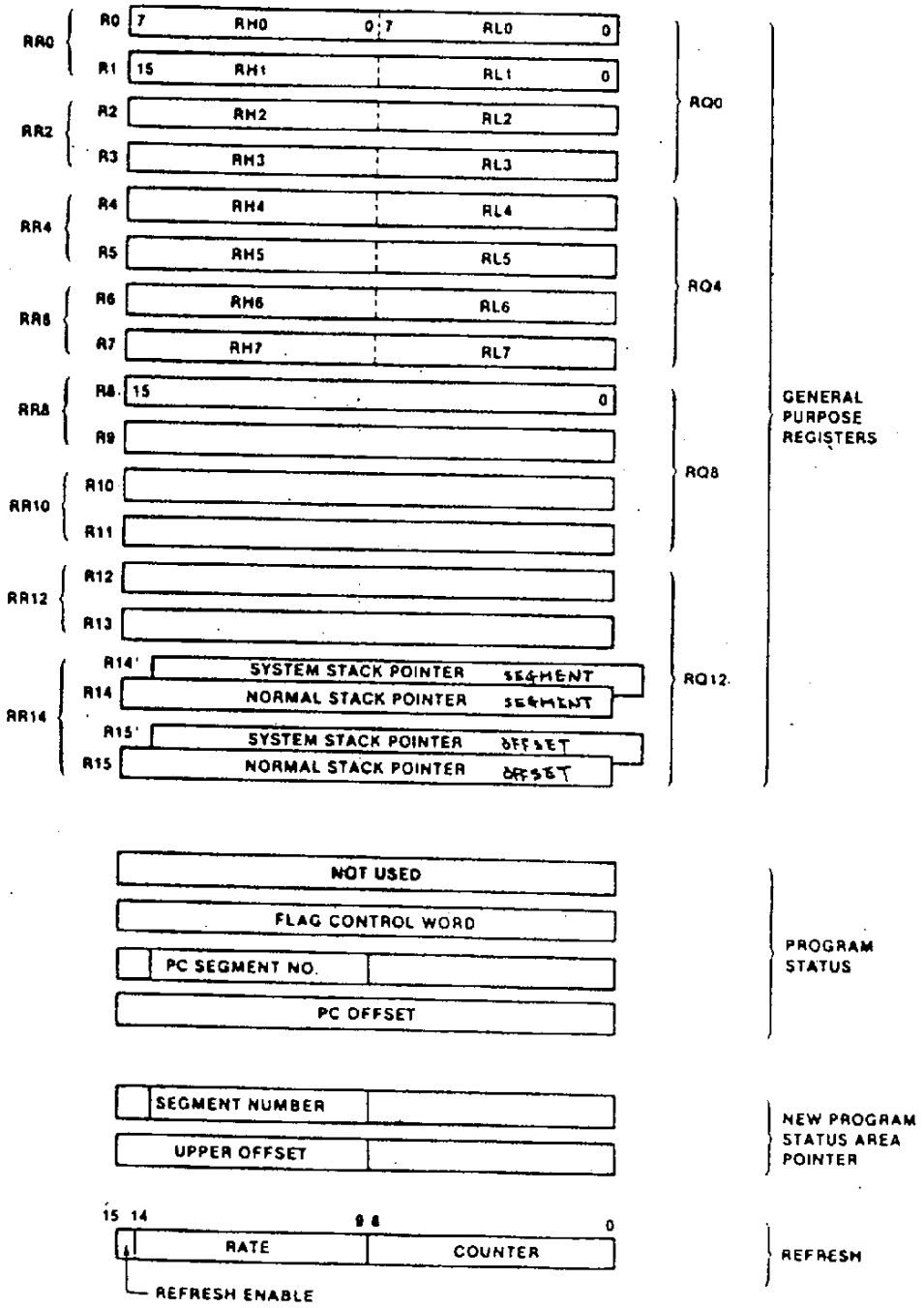


図３－１０　Ｚ８０００のレジスタ構成

メモリアドレスは7ビットのセグメント部と16ビットのオフセットから成り，全体として23ビット（8メガバイト）が直接指定できる構成となっている。

Z8000の命令アドレスには次のような8種類のアドレス・モードがある。

①	レジスタ	R
②	インダイレクト・レジスタ	IR
③	ダイレクト・アドレス	DA
④	インデックス	X
⑤	イミディエイト	IM
⑥	ベース・アドレス	BA
⑦	ベース・インデックス	BX
⑧	相対アドレス	RA

この各アドレスはZ8000の命令語のsrc(ソース) とdst(デスティネーション）として用いることができる。

アドレス指定はセグメントを指定する場合は32ビット（2語長），セグメントを指定しないで64Kバイトのオフセットで済ます場合は16ビット（1語長）となる。

例えばインダイレクト・レジスタの場合，セグメントを用いる時はレジスタ対が必要ないが，セグメントを使わない時は1つのレジスタで充分である。（ただし，R0やRR0は除く）

ダイレクト・アドレスにおいても，1語で済ます場合と2語で済ます場合がある。

インデックス・モードの場合にはR0以外の16ビット・レジスタが用いられ，1語または2語のアドレス値に加算される。

ベース・アドレスの場合には，R0とRR0を除くレジスタまたはレジスタ対に1語のディスプレイスメントが加算される。

ベース・インデックスではベース・レジスタ（またはレジスタ対）にインデックス・レジスタの値が加算される。

相対アドレスではプログラム・カウンタに対して1語の相対アドレスを加える形で，現在命令で実行しているアドレスを基本にした相対的アドレス処理ができる。

(2) ソフトウェアの特徴

Z8000のソフトウェアは関係する各社で開発が進められており，今後の展開については未だ明確でない面があるが，前項で紹介したすぐれたアーキテクチャをベースとしているので，今後汎用16ビットミニコン同様の経過により，すぐれたソフトウェアが出揃うものと期待される。

例えばシステム記述用の高級プログラミング言語として，CおよびPascalの開発が発表されており，Ada なども標準的に整備されるだろう。ユーザの応用プログラムの記述 には当然COBOL，FORTRANを始めとして，LISP，SNOBOL など各種プログラミング言語が揃えられるだろう。

Z8000のOSとしては現在8ビットマイマン用標準OSとなっているCP／M* 相当のレ

ベルのものから（CP/MはDigital Research社の登録商標である。），リアルタイム，マルチユーザ型のＵＮＩＸまたはＴＨＯＴＨもしくはこれらの拡張版のレベルのものまで，開発着手され話題にのぼっている。例えばOnyx社のONYX，マイクロソフト社のXENIX，アドバンストマイクロソフト社のＥＰＯＳなどがその例としてあげられる。これらはC，Pascalをはじめとして，ＯＳも移植性に富むものが多い点が特に注目される。

また，これらの他にチップメーカその関係会社などで比較的軽量のターゲット・システムを主対象とするＯＳも開発されて行くであろう。

(3) システム構成

Ｚ８０００によるシステム構成例を図３－１１に示す。Ｚ８０００に対しては周辺チップが各種開発（または開発中）されており，このようなシステムを構成するに際して，新たにＴＴＬで回路を組む必要はほとんど無いようになっている。

まずメモリ管理関係ではＺ８０１０メモリ・マネジメント・ユニットがあり，セグメント方式のプログラムのダイナミックリロケーション機能と各種のメモリ・アクセスに対するプロテクション機能を持つ。

Ｚ８１６４ダイナミック・メモリコントローラ，Ｚ８１６５，８１６６ダイナミック・メモリドライバ，Z8161，８１６２メモリ・バス・バッファ，Ｚ８１６０エラー検出および訂正用のＬＳＩ，Ｚ８１６３エラー訂正検出とリフレッシュを同時に行うチップといったメモリ・モジュール作成用のＬＳＩの他に，ラッチやデコーダといったシステム周辺用ＬＳＩ，入出力用のLSIなどが開発されている。

入出力用には，Ｚ８０５２ＣＲＴ制御用ＬＳＩとかＺ８０６８データ暗号用ＬＳＩなども含まれ，システム開発に配慮がなされている。

(4) 開発サポートシステム

Ｚ８０００の開発サポート・システムには，

① ミニコン（主としてＰＤＰ－１１）を利用したクロス開発システム

② ８ビット・マイコン（Ｚ－８０または８０８５）を利用したクロス開発システム

③ Ｚ８０００自身を利用したセルフ開発システム

④ 大型計算機を利用したクロス開発システム

の４種類がある。

しかし，現在のところ１６ビットミニコンをホストとするものや，８ビットマイコンをホストとするクロス開発システムが実使用の主体であろう。

これらのシステム記述言語としてCが中心となっていく気配であり，今後の高機能マイコンのソフトウェア開発費の膨張への対応策としてその成果が期待できる。

もちろん今後の発展としてＵＮＩＸクラスのＯＳが標準的に装備されるようになるとともに，Ｚ８０００自身をホストとする開発システムの比重が，スタンドアロンあるいはこれらと結合されたシステムという形で次第に増大していくものと思われる。

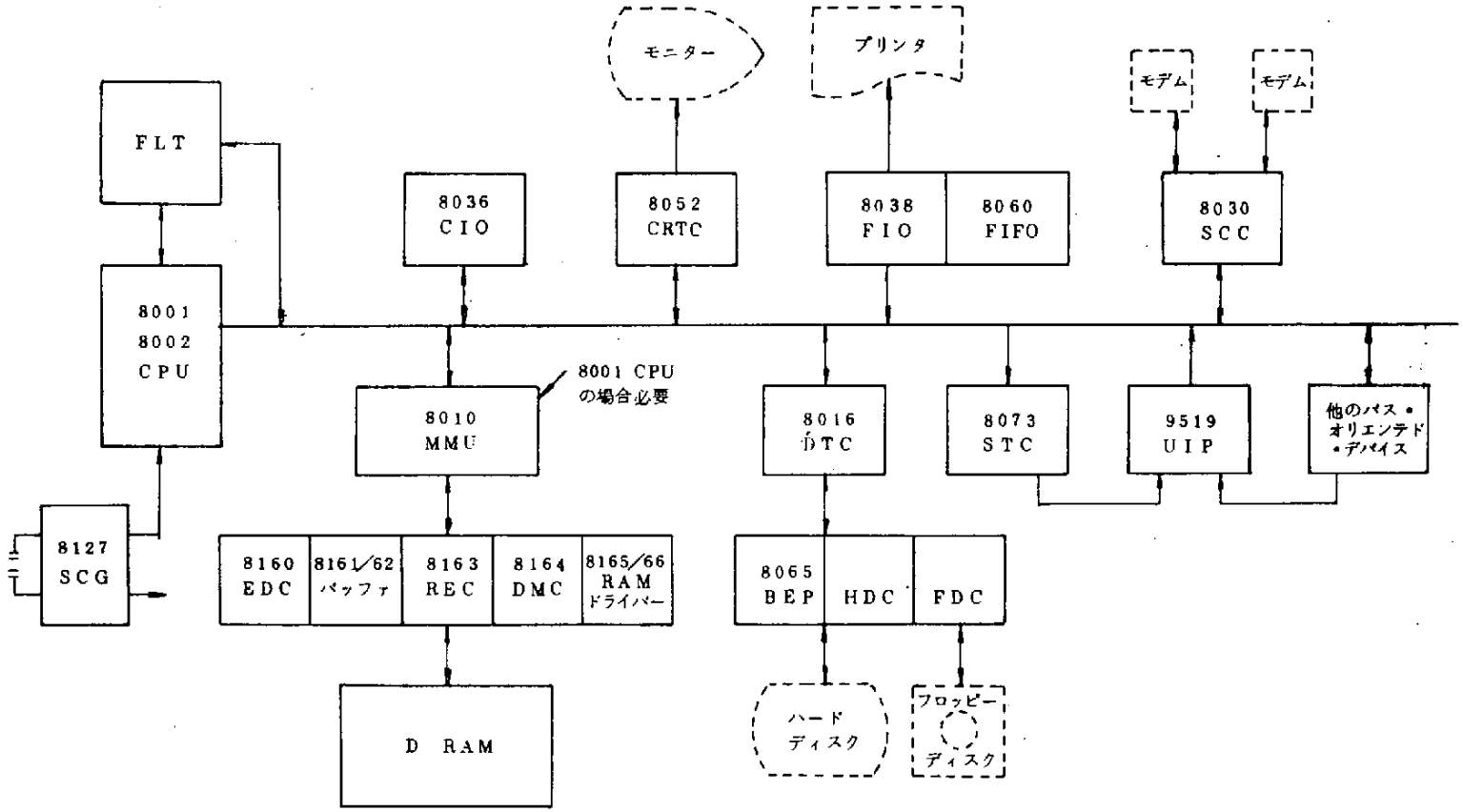


図３－１１　Ｚ８０００システム構成例

3.2.4 T88000

(1) ハードウェアの構成と特徴

T88000は東芝の16ビット系ミニコンピュータ・シリーズと互換性のあるアーキテクチャをもつマイクロプロセッサである。

T88000のLSIはシリコン・オン・サファイヤ(SOS)技術により，高集積度(12000ゲート)，高速(10MHz クロック)，低消費電力(0.7W)を約7㎜角のチップ上に実現している。

電源は5V単一で64ピン標準セラミックDIPパッケージ上に実装されている。

T88000の機能上の特長は以下のとおり。

① 16個のジェネラル・レジスタ。
(このうち15個はインデックス・レジスタとして使用可能)

② 151種を越える豊富な命令群。

③ ファームウェアにサポートされた高速の割り込み処理機能。

T88000はLSI自身が高速，高性能であると同時に次のようなハードウェア上の特色を有している。

① 最大16メガバイトまでアドレス指定可能。

② 命令先取りバッファによるパイプライン処理による高速化。

③ ハードウェアによる高速乗算機構の内蔵。

④ バレルシフタによるシフト演算の高速処理。

⑤ 32ビット幅の共通入出力バスによる時分割多重利用。

⑥ 内部2バス方式による演算実行時間の短縮など

さらにT88000はマイクロプログラム処理機構の外付方式により，浮動小数点演算命令，10進演算命令などを装備することが可能でハードウェア機能の弾力的な活用が可能である。

T88000の論理ブロック図を図3－12に示す。

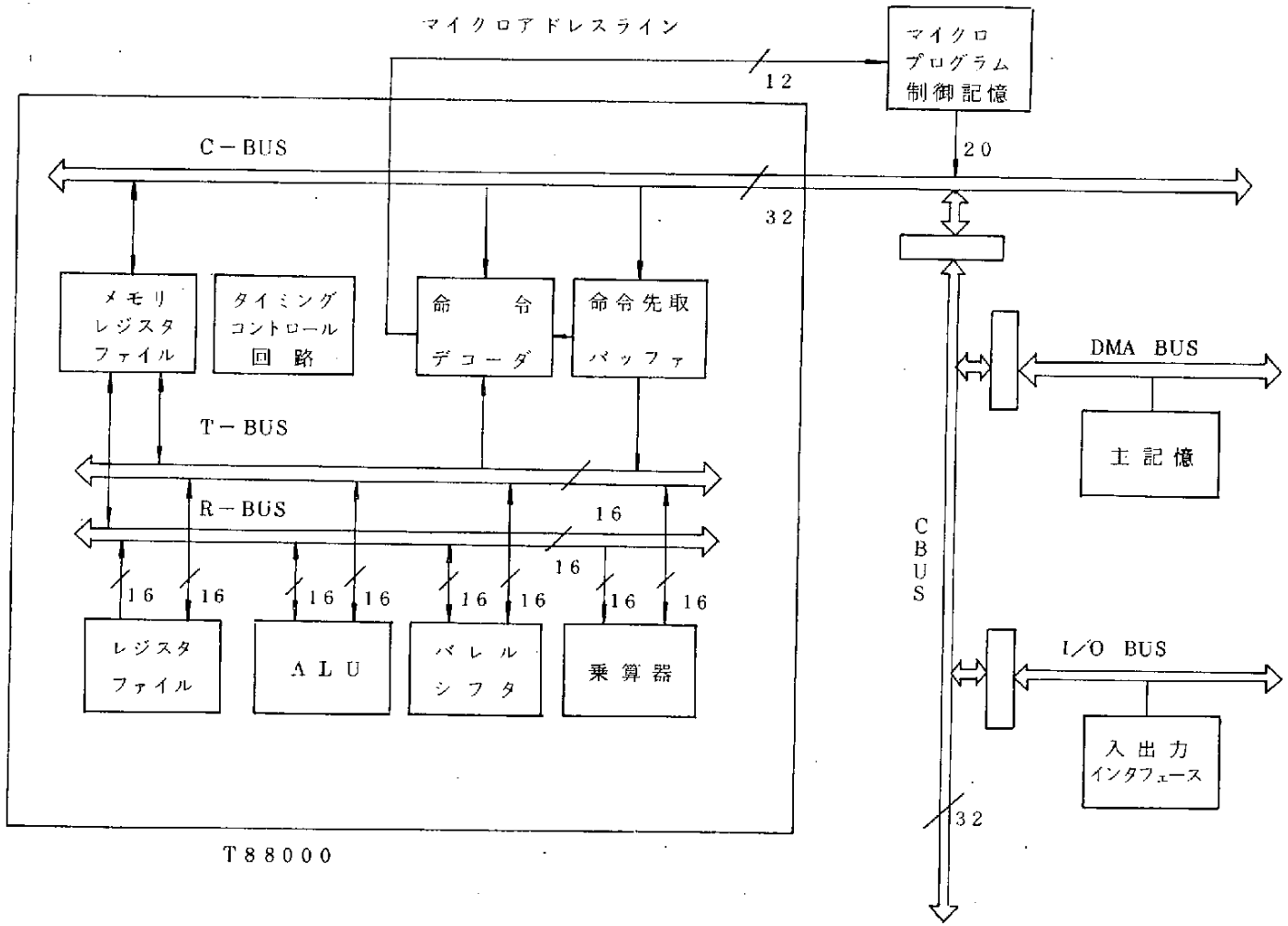


図３－１２．Ｔ８８０００の論理ブロック図

(2) ソフトウェアの特徴

Ｔ８８０００のソフトウェアは１６ビット系ミニコンピュータのシリーズとアーキテクチャ上の互換性を持っているので，基本的に従来から蓄積されてきたソフトウェアを活用することができる。

Ｔ８８０００の処理ソフトウェアはいくつかのレパートリに分れるが，専用指向形のオペレーティング・システムも用意されており，概略次のような特徴をもつ

① 主記憶常駐部のサイズが小さく，小規模の主記憶域でも充分にユーザエリアを取ることができる。

② オートマチック入出力機能をフルに活用した優先度制御による多重処理方式がとられている。

③ 中核部のモジュールはＲＯＭ／ＲＡＭ分離のできる構造となっている。

④ ＯＳの基本モジュール，入出力ドライバ，システムタスクなど，全体がモジュール構造になっており，開発サポートシステムでの編集機能により，必要に応じてこれらを選択してシステムに組込むことができる。

⑤ ＯＳの基本モジュールと入出力ドライバのインタフェースが簡潔であるため，ユーザが開発した入出力装置のドライバを容易にシステムに組込むことができる。

⑥ フロッピーディスクなどを装備したディスクシステムにおいては，高位言語により作成したオブジェクトプログラムの実行も可能であり，より高機能なシステム開発ができる，など

このような手段を用いて作成されたユーザソフトウェアの構成例を図３－１３に示す。

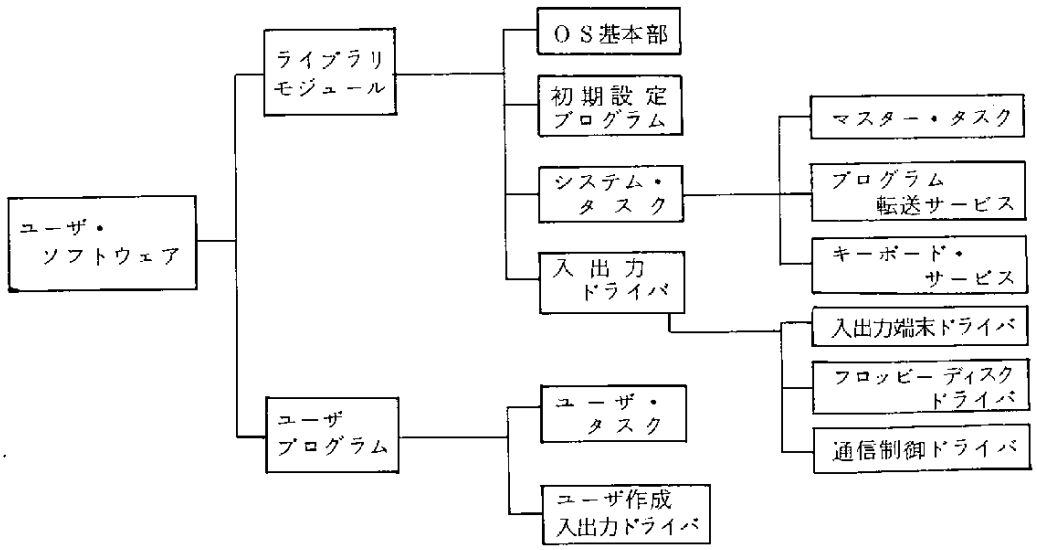


図３－１３　ユーザ・ソフトウェア

(3) 開発サポートシステム

Ｔ８８０００によるシステムの開発はＴ８８０００自身を用いた比較的規模の大きな構造によるセルフ・プロダクション方式や，他のミニコンピュータまたは汎用コンピュータによるクロスサポートを利用した方式などがあり，豊富なバックアップが可能になっている。

3.2.5 ＰＦＬ－１６Ａ

ＰＦＬ－１６Ａ（以下Ｌ－１６Ａと略す）は，１６ビットマイクロプロセッサを中心に構成された高性能マイコンであり，利用者の立場に立ち，数々の創意工夫を施し，設計されている。

(1) ハードウェアの特徴

Ｌ－１６Ａは，ＣＰＵ，ＣＰＵ周辺回路用及び２種類の入出力制御用の４種のＬＳＩから構成されている。

(a) 命　令

Ｌ－１６Ａの命令はすべて１語長であり，体系的に整理された使い易い命令セットを実現している。基本的な命令の数は３３個と少なくし，基本命令にバリエーション情報を付加することによりあらゆるアプリケーションに対処することができる。

(b) レジスタ

命令語内のレジスタ指定部で任意に指定できる５個の演算レジスタを設けている。このため記憶装置との情報授受の回数が減り実行効率を向上させることが可能となる。

(c) ＣＰＵ周辺回路用ＬＳＩ

割込み制御には，プログラムステータスワード切換え方式を採用し，３レベル多重割込み制御を行っている。この多重割込み機能に加えて，インターバルタイマ機能，メモリパリティチェック機能などを一つのＬＳＩに集約し，繁雑なＣＰＵ周辺回路の統一化を図っている。

(d) 入出力制御用ＬＳＩ

２種の入出力制御用ＬＳＩは，各種入出力装置の接続を統一的かつ容易に取扱えるよう共通する機能要素を抜き出したものである。さらに，ユーザ側の設計効率の向上をはかるために，バイト及び語単位で転送する汎用入出力インタフェースカードを始めとして，いくつかの標準入出力カードを用意している。

(e) メモリ

Ｌ－１６Ａがアクセスできるメモリは，最大６４Ｋ語であるが，実装メモリ容量が少なくてすむよう命令形式やデータアクセス方法を統一し，メモリの利用効率を向上させている。この利点をさらに生かすために２Ｋ語を最小単位とするメモリカードを提供しユーザ側の設計効率の向上をはかるとともに統一的設計基準を提供している。

(2) ソフトウェアの特徴

Ｌ－１６Ａのソフトウェアには，コンパクトなモニタプログラムＭＯＮＩＴ，アセンブラ及びリンケージエディタ，ソースプログラム編集，ＲＯＭライト，デバッガなどのユーティリティ群

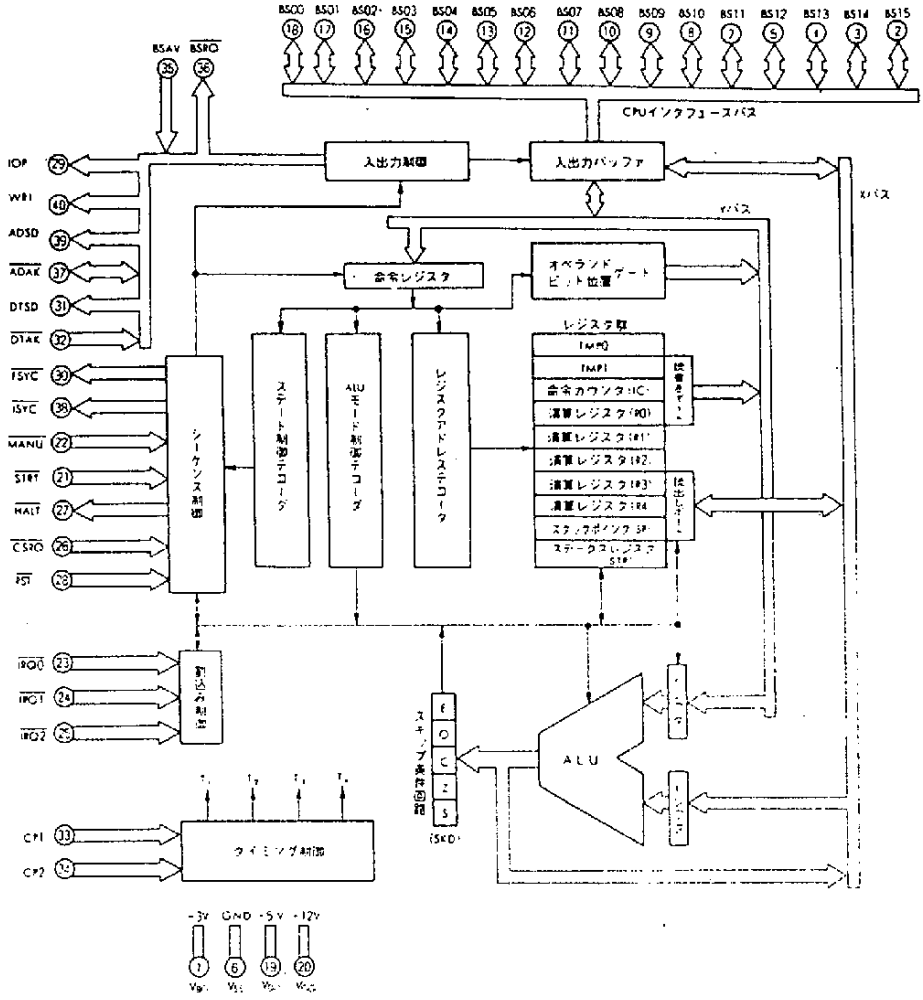


図３－１４　PFL－１６A　CPUブロック図

が用意されている。主なソフトウェアの機能は次のとおりである。

(a)　ＭＯＮＩＴ

ＭＯＮＩＴは，ハードウェア機能を充分に発揮できるリアルタイムモニタであり，その主な機能は次のとおりである。

①　最高１５レベルの多重優先処理と並行処理タスクに最高１５レベルの緊急度を設定で

き，その緊急度に応じた優先処理を行うことができる。また，あるタスクの処理が事象待ちで中断された場合，他の実行可能タスタを動作させる並行処理も可能である。

表3－1　命令一覧表

区　分	略　称	命　令　名　称	形　式
メモリ参照命令	L	Load	RM
	ST	Store	RM
	B	Branch	RM
	BAL	Branch and Link	RM
	IMS	Increment Memory and Skip if Result is Zero.	RM
	DMS	Decrement Memory and Skip if Result is Zero.	RM
入出力命令	RD	Read	RCM
	WT	Write	RCM
直接数値命令	MVI	Move Immediate	RCM
	AI	Add Immediate and Skip on Condition	RC
	SI	Subtract Immediate and Skip on Condition	RC
算術演算命令	A	Add and Skip on Condition	RR
	S	Subtract and Skip on Condition	RR
	C	Compare and Skip on Condition	RR
	CB	Compare Byte and Skip on Condition	RR
論理演算命令	AND	Logical AND and Skip on Condition	RR
	OR	Inclusive OR and Skip on Condition	RR
	EOR	Exclusive OR and Skip on Condition	RR
転送命令	MV	Move and Skip on Condition	RR
	MVB	Move Byte and Skip on Condition	RR
	BSWP	Byte Swap and Skip on Condition	RR
10進操作命令	LAD	Load Adjust Part and Skip on Condition	RR
	DSWP	Digit Swap and Skip on Condition	RR
ビット操作命令	SBIT	Set Bit and Skip on Condition	RC
	RBIT	Reset Bit and Skip on Condition	RC
	TBIT	Test Bit and Skip on Condition	RC
シフト命令／制御命令／他	SL	Shift Left and Skip on Condition	RC
	SR	Shift Right and Skip on Condition	RC
	PUSH	Push Stack	RC
	POP	Pop Stack	RC
	LPSW	Load Program Status Word	RC
	RET	Return	RC
	H	Halt	RC

② 豊富なマクロ命令

タスク制御及びタスク間制御などのために２０種の制御サービス用マクロが用意されている。

③ 豊富な組込みモジュール

利用者の使用頻度が高く，どのようなシステムでも必要となる共通的な機能を持った組込みモジュールが含まれている。組込みモジュールには，リアルタイム系の処理に必要な入出力処理サブルーチン，演算処理などを行う演算パッケージがある。

④ ＲＯＭ化可能な機能別モジュール構成

ＭＯＮＩＴは，詳細な機能毎にモジュール化してあり，個々のモジュールは，ＲＯＭ化に対応できるようプログラム部，定数部，変数部を完全に分離している。これにより，ユーザはシステムテーブルと必要な機能を組み合わせユーザ独自のモニタを構築することができる。さらに，構築したシステムをＲＯＭライトユーティリティによりＲＯＭに焼付けることも容易に行なえる。

(b) アセンブラ

アセンブラには，言語仕様を同じくするＬ－１６Ａ自身で動作するものと他機種で動作するものがある。両者共，アセンブラはソースプログラムを入力し，再配置可能な機械語モジュールを出力する。

リンケージエディタでは，アセンブラが出力した一つあるいはいくつかの再配置可能モジュールを結合し，Ｌ－１６Ａで動作可能なプログラムに編集する機能を持つ。このとき，MONITの組込みモジュールを併せて編集することができる。

(3) システムの構成

ユーザがＬ－１６Ａシステムを構築しやすいよう，ＬＳＩ単体の純粋な部品レベルを始めとして入出力装置を接続したシステムレベルに至るまで，利用者のあらゆる要求に応じられる製品体系をとっている。この製品体系を大別すると，ＬＳＩ単体，標準カード，構成ユニット，コンピュータシステムの４種類に分類される。

(a) ＬＳＩ単体

Ｌ－１６ＡのＬＳＩファミリは，高水準のマイコンシステムのハードウェア／ソフトウェアのサポート及び標準化をできるだけ少ないＬＳＩ構成で構成できるよう，機能の集積化を図っている。これにより，４種のＬＳＩファミリのみで広範囲な応用分野への適用を可能としている。

(b) 標準カード

Ｌ－１６Ａは利用者システムに組み込まれる場合が多く，この中のコンピュータ部分について，設計を短期間に行い，信頼性が高く効率のよいシステム開発を行うことを目的として標準カードを用意している。Ｌ－１６Ａ標準カードは使用目的に応じて自由に選択できるように，各種豊富にとりそろえられている。

(c) 構成ユニット

複数枚の標準カードを組み合せて使用するような場合は，標準カード間を接続するため，バックパネル，筐体，電源，コンソールパネルなどを個別に用意し，利用者のさまざまな要求にこたえられるよう各種構成ユニットをとりそろえている。

(d) コンピュータシステム

さらに，利用者の必要とするシステム構成がある程度固定化できる場合には，コンピュータシステムを構成した形でＬ－１６Ａを提供することができる。

(4) 開発サポート

Ｌ－１６Ａの開発サポートシステムは，マイコン自身で自己のソフトウェアを作成するセルフ系と，他機の開発サポートシステムで作成するクロス系に分れ，より多くの利用者の開発環境に対応できるように考えられている。

① セルフ系開発サポートシステム

フロッピィディスクをベースとしたＦＰＳ－１６を用意し，Ｌ－１６Ａでも効率のよいプログラム開発が行えるよう対処している。ＦＰＳ－１６に含まれる主なプログラムは，アセンブラ，マクロアセンブラ，リンケージエディタなどである。

② クロス系開発サポートシステム

ミニコンピュータによるサポートシステムとしては，ＰＦＵシリーズ及びＭＡＣＣ－７／Ｌを汎用機によるサポートシステムとしては，ＩＢＭシステム３７０系及びＦＡＣＯＭＭシリーズを用いるシステムが用意されている。

サポートしている主なプログラムは，アセンブラ，マクロアセンブラ，シミュレータ，リンケージエディタなどである。また，汎用機サポートにはコンパイラ言語ＰＬ／１６のサポートも行っている。

① アセンブラ

「(2)ソフトウェアの特徴」を参照

② マクロアセンブラ

マクロアセンブラは，システムマクロの他に，ユーザマクロも登録できる機能を持ち，これらマクロ命令より一連のアセンブラ命令に変換する機能を持つ。

③ リンケージエディタ

「(2)ソフトウェアの特徴」を参照

④ シミュレータ

他機種により，Ｌ－１６Ａの実行プログラムを擬似的に実行する機能を持ち，実行に際して有効なデバッグ機能も備えている。

⑤ ＰＬ／１６

ＰＬ／１６はＰＬ／Ｍの機能をほとんど包含しており，さらに，Ｌ－１６Ａアーキテクチャの特徴を充分活用できるよう設計されている。ＰＬ／１６はつぎのような特徴をもっている。

イ．高レベルの言語仕様

言語仕様には，語，バイト，ビット単位のデータ定義，入出力命令，Ｌ－１６Ａ特有の領域指定などを含み，アセンブラ並にきめこまかなプログラミングができるよう配慮されている。ソースプログラムは読み易くなっており，信頼性，保守性の向上を図ることができる。将来，システム機能拡大への対処も容易となる。

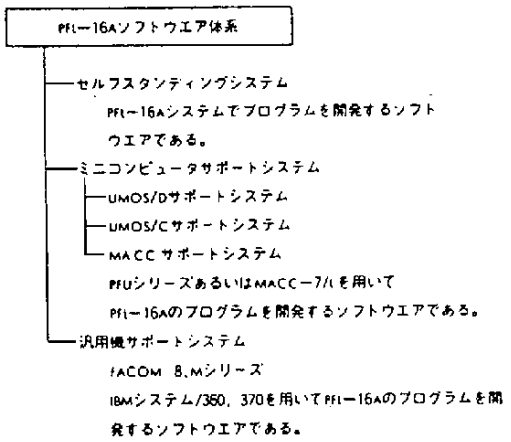
ロ．優れた実行効率

オブジェクトプログラムはＬ－１６Ａで最適に動作するよう作成されている。さらに，オブジェクトコードのＲＯＭ化，アセンブラプログラムとの結合など，柔軟な対処が可能である。

ハ．移行性のよいコンパイラ

コンパイラは，ＦＯＲＴＲＡＮで記述されており，ＦＯＲＴＲＡＮが動作する計算機には，容易に移行できる

なお，電電公社のＤＥＭＯＳ－ＥによるＴＳＳサポートシステムにも，手軽に利用できる開発システムが用意されている。



図－１５　ＰＦＬ－１６Ａのソフトウェア体系

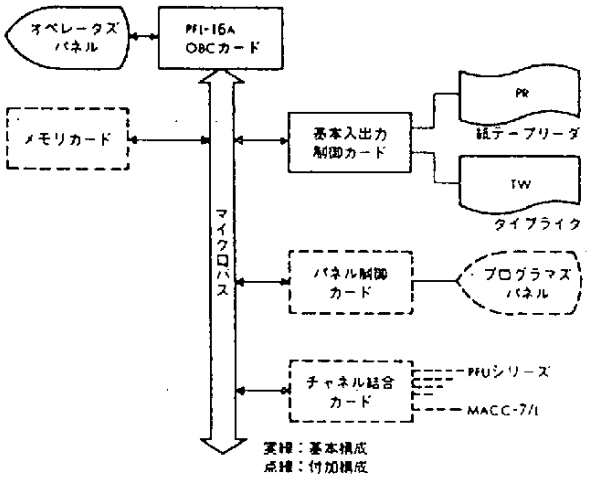


図３－１６　セルフスタンディングシステムの機器構成

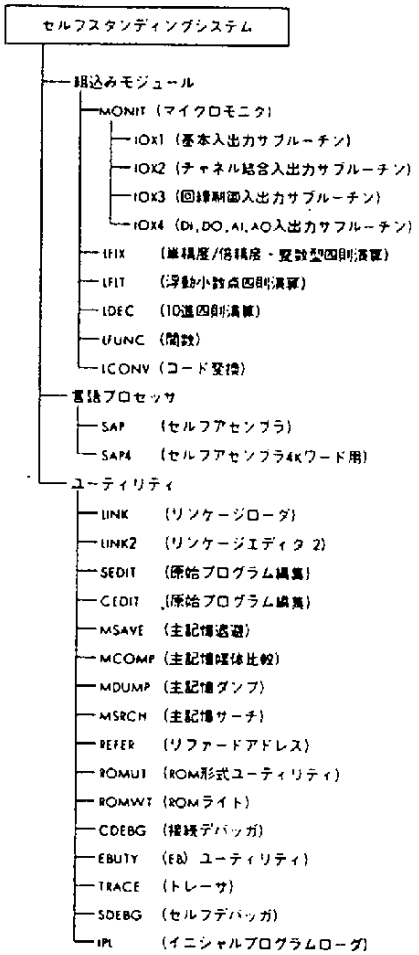


図３－１７　セルフスタンディングシステムのプログラム構成

3.2.6 MN1613

(1) ハードウェアの構成と特徴

MN1613は次の様な特徴を有する16ビットマイクロプロセッサである。

① 高速処理

命令先取り機構を内蔵し，レジスタ間演算は0.3μ秒

② 強力な演算処理能力

乗除算命令（夫々，8.1／17.1 μs），浮動小数点演算命令（加算22.5μs，乗算57.9μs）を持つ。

③ アドレス空間

セグメンテーション方式により最大256K語（512KB）

④ 周辺機能

タイマ，シリアルI／Oを内蔵，バス変換チップを組合せれば6800系8ビット周辺LSIが利用できる。

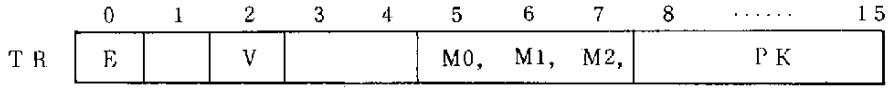
⑤ 自己診断機能

内蔵診断機能により検査時間が短縮可。

MN1613は機械語命令・レジスタ構成・割込み方式およびバス転送シーケンスがMN1610（PFL-16AのCPU）（3.2.5参照）と上位互換性を有するので，MN1610上で動作するプログラム，MN1610に接続される入出力装置はMN1613に無修正で適合する。MN1613の仕様を表3－2に，内部構造を図3－18に示す。内部レジスタは次の5種に分類される。

① 汎用レジスタおよび命令カウンタ

16ビット汎用レジスタR0～R5，ステイタスレジスタSTR，命令カウンタICであり，R3，R4はインデックスレジスタ，R5はスタックポインタとしても使用できる。STRは下図のように演算フラグ（E，V），割込みマスク（M0，M1，M2）およびプログラムキーPKから成る。



② ベースレジスタ

4ビットから成る4本のレジスタで下図のような加算により18ビットの物理アドレスが生成される。

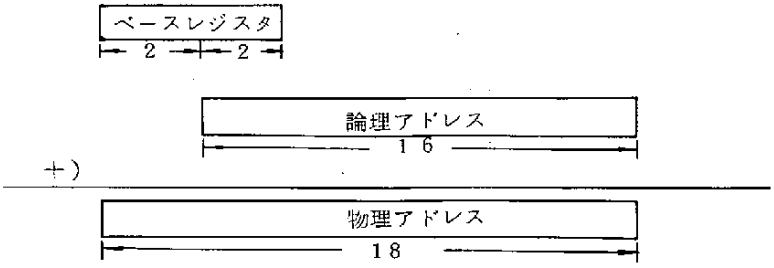


表３－２　ＭＮ１６１３仕様概要

	項　　目	MN1613	MN1610(参考)
1	データ幅	１６ビット	
2	互換性	1610機械語互換	
3	主記憶容量	２５６Ｋワード	６４Ｋワード
4	主記憶拡張方式	セグメント方式	無
5	制御方式	マイクロプログラム方式	配線論理方式
6	命令数	約１００種	３３種
7	Ｒ－Ｒ演算時間	0.3 μs	3.0 μs
8	１６ビット乗／除算時間	8.1μs/17.1μs	365μs/513μs
9	浮動小数点加算時間	15.3～22.5 μs	295～646 μs
10	浮動小数点乗算時間	51.9～57.9 μs	962～1439μs
11	平均命令実行時間	1.124 μs	4.85 μs
12	汎用レジスタ数	５ワード	
13	割り込みレベル	３レベル	
14	タイマー，直列入出力機構	内蔵	――
15	クロック発振器	内蔵	ＲＳＣチップによる
16	クロック周波数	13.3 MHz	2 MHz
17	命令先読み機構	内蔵(２Ｗ)	無
18	素子数(３μルール)	39,000tr	4500tr
19	電源	+5V 単一	+12V, 5V, -3V
20	パッケージ	４０ピンＤＩＰ	

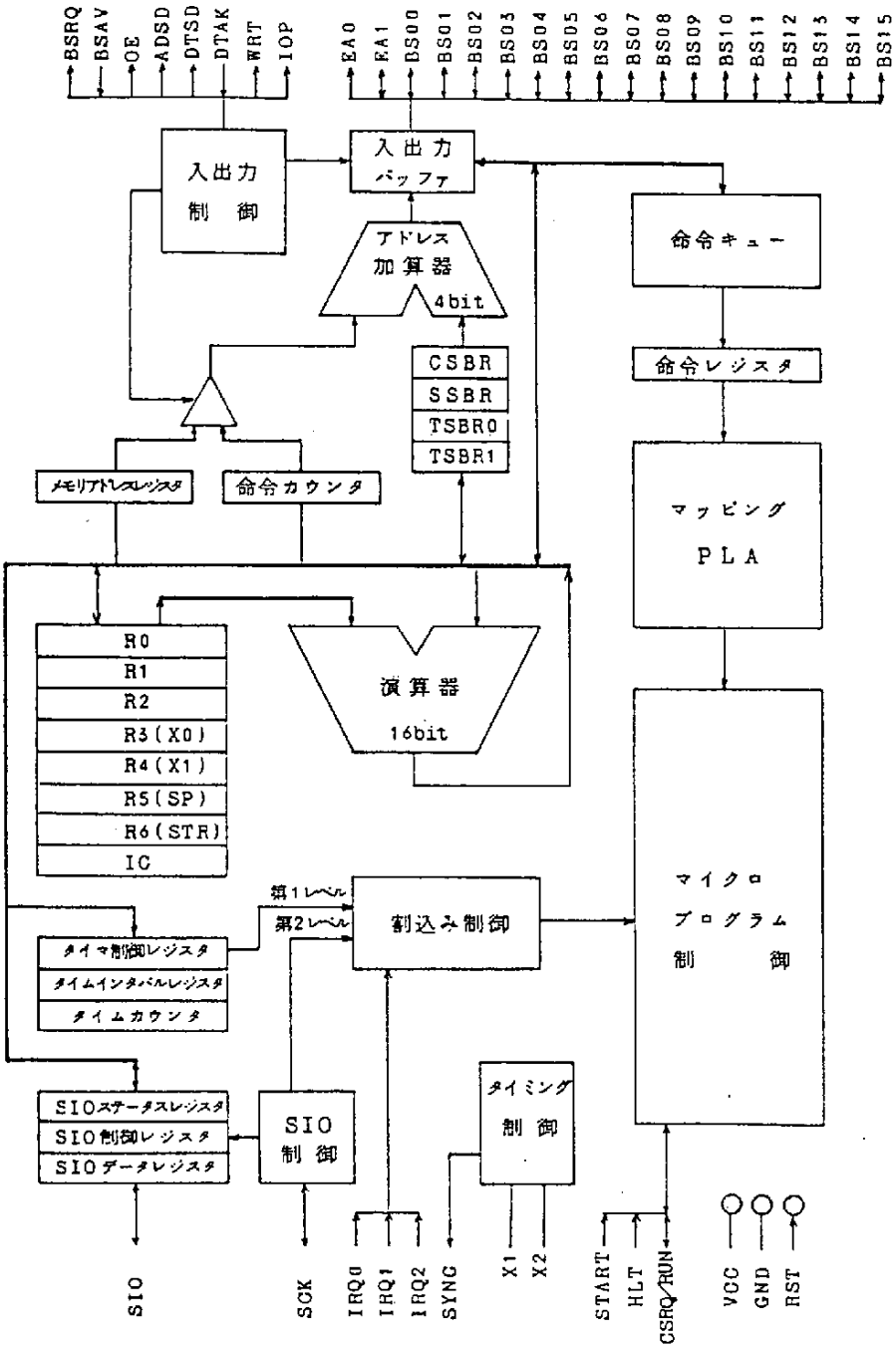


図3－18　MN1613ブロック図

③ ベース退避レジスタ

割込み発生時にカレントベースレジスタの内容が退避されるレジスタで割込みレベルに対応して4本用意されている。

④ ＮＰＳＷポインタ

ニュープログラムスティタスワードＮＰＳＷのメモリ上の位置を指定するレジスタである。

⑤ タイマ・直列入出力レジスタ

内蔵タイマとシリアルＩ／Ｏインタフェースの制御およびデータバッファ用のレジスタである。

内部レジスタを図３－１９にまとめて示す。

ＭＮ１６１３の割込みの要因は各々３優先レベルを持つ内部割込と外部割込みである。

① 内部割込み

レベル０：未定義命令のフェッチ

レベル１：内蔵タイマのオーバフロー

レベル２：内蔵直列入出力における受信完了と送信完了

② 外部割込み

割込み要求の各レベルに対応する。ＭＮ１６１３は図３－２０に示す４０個の端子を有する。各端子は次の意味を持つ。

ＢＳ００～ＢＳ１５：システムバスでアドレス情報とデータが多重化されている。

EA0／FSYC，EA1／CSTP：拡張アドレス端子

ＳＹＮＣ：マシンサイクルに同期したクロック出力

ＧＮＤ：ＣＰＵチップのアース端子

ＲＳＴ：リセット入力

$\overline{\text{ＳＣＫ}}$：直列入出力用クロック端子

ＳＤ：直列入出力用データ端子

ＣＫ１，ＣＫ２：クロック発振用水晶振動子の接続端子で水晶周波数の２倍がマシンサイクルとなる。

ＨＬＴ：マシンを停止せしめる制御入力

ＳＴＡＲＴ：マシンの起動用制御入力

CSRQ／RUN：Ｉ／Ｏアドレスからのプログラム実行指令（コンソール処理指定）

ＩＲＱ0～ＩＲＱ2：割込み要求

ＢＳＲＱ：ＣＰＵのバス使用権要求信号出力

ＢＳＡＶ：ＣＰＵに対するバス使用許可入力

ＯＥ：バスによるデータ転送制御信号

ＡＤＳＤ：アドレス出力時の同期信号

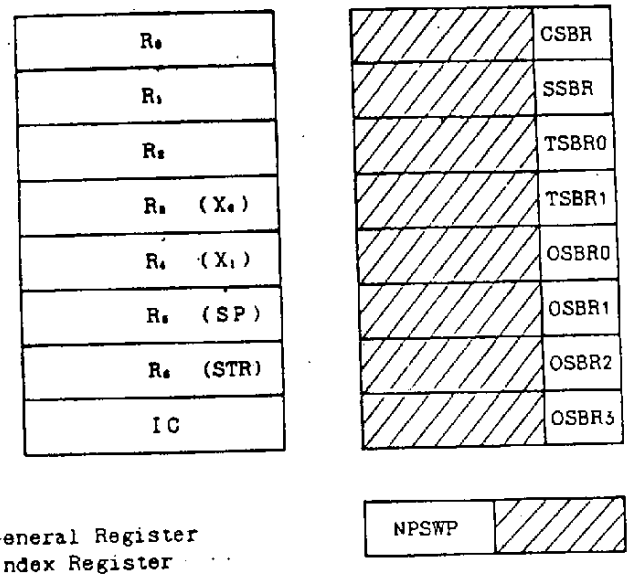
ＤＴＳＤ：データ入出力時の同期信号

ＤＴＡＫ：ＤＴＳＤに対する応答入力

ＷＲＴ：データの転送方向を指定する信号

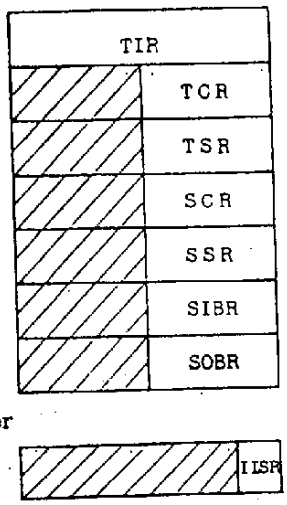
ＩＯＰ：データ転送空間がＩ／Ｏ空間であることを示す信号

Ｖｃｃ：電源供給端子



Ri :General Register
Xi :Index Register
SP :Stack Pointer
STR :Status Register
IC :Instruction Counter
CSBR :Current Segment Base Register
SSBR :Stack " " "
TSBRi:Temporary " " "
OSBRi:Old " " "

NPSWP:New Prog.Status Word Pointer



TIR :Timer Interval Register
TCR :Timer Control "
TSR :Timer Status Register
SCR :Serial Control Register
SSR :Serial Status "
SIBR :Serial Input Buffer "
SOBR : " Output " "

IISR :Internal Interrupt Status Register

図３－１９　内部レジスター覧

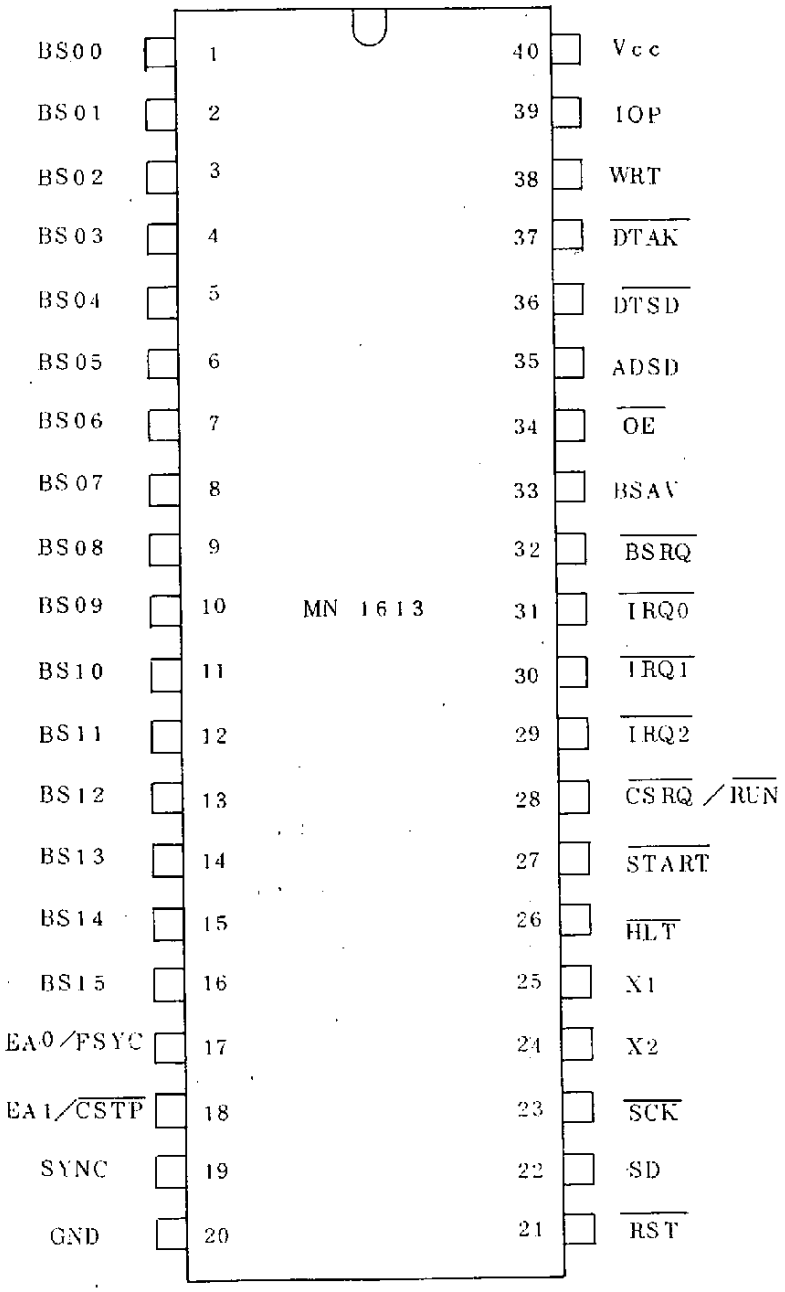

BS00 1
BS01 2
BS02 3
BS03 4
BS04 5
BS05 6
BS06 7
BS07 8
BS08 9
BS09 10
BS10 11
BS11 12
BS12 13
BS13 14
BS14 15
BS15 16
EA0/FSYC 17
EA1/CSTP 18
SYNC 19
GND 20
MN 1613
40 Vcc
39 IOP
38 WRT
37 DTAK
36 DTSD
35 ADSD
34 OE
33 BSAV
32 BSRQ
31 IRQ0
30 IRQ1
29 IRQ2
28 CSRQ/RUN
27 START
26 HLT
25 X1
24 X2
23 SCK
22 SD
21 RST

図３－２０　ピン配置

(2) ソフトウェア

ＭＮ１６１３は表３－３に示す命令セットを有しＭＮ１６１０が持つ全ての命令とは機械語で共通である。

(3) システム構成

ＭＮ１６１３はＭＮ１６１０の周辺チップが接続できる。また図３－２１に示すバス変換チップが用意されている。ＭＮ１６６８はデータ転送が非同期ハンドシェーク方式で行われる。ＭＮ１６１３のバス上に同期転送方式を持つＭＣ６８００系の周辺チップの接続を可能とするものである。このチップを利用したシステムの構成例は図３－２２に示される。

(4) 開発サポート

ＭＮ１６１３用開発システムＦＰＳ－１６１３はＭＮ１６１０用開発システムＦＰＳ－１６にＭＡＣＲＳ（１６１３ＭＡＣＲＯアセンブラ）とＳＤＢＧ１３（１６１３デバッグユーティリティ）を追加して構成される。サポートツールはＦＰＳ－１６のＯＢＣを１６１３用ＯＢＣに，パネル制御を１６１３用に差換えて図３－２３の如く構成される。

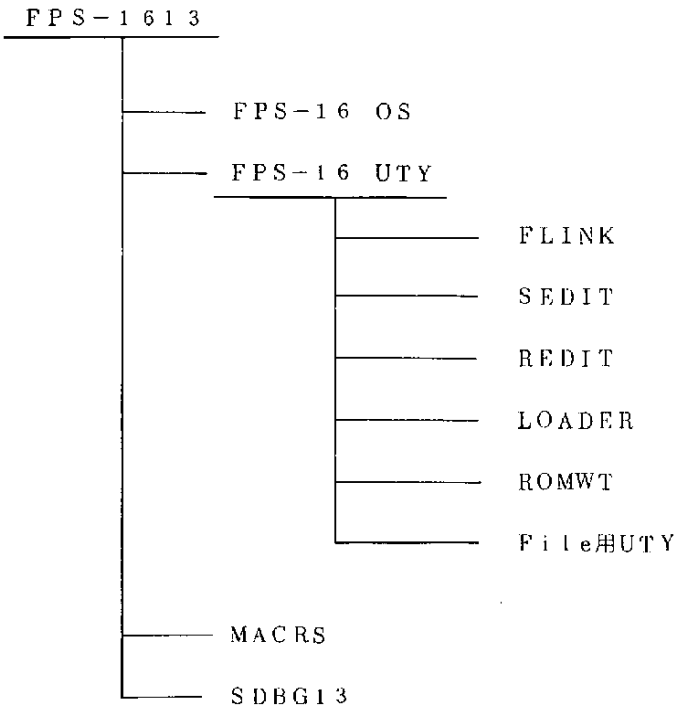
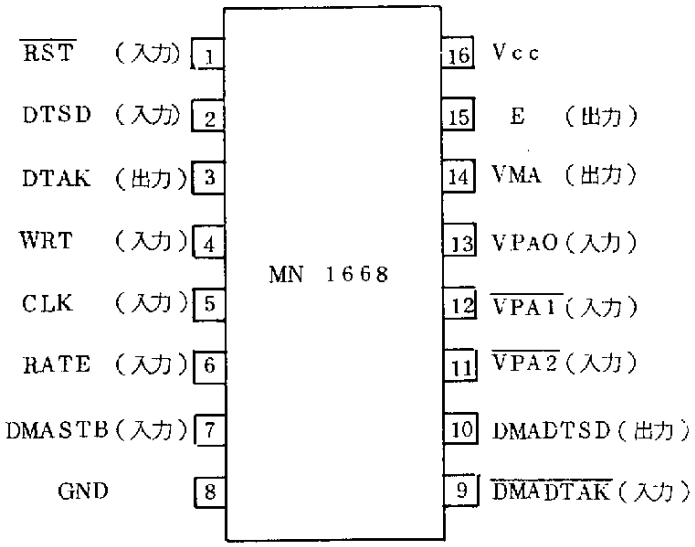


図　サポートツール



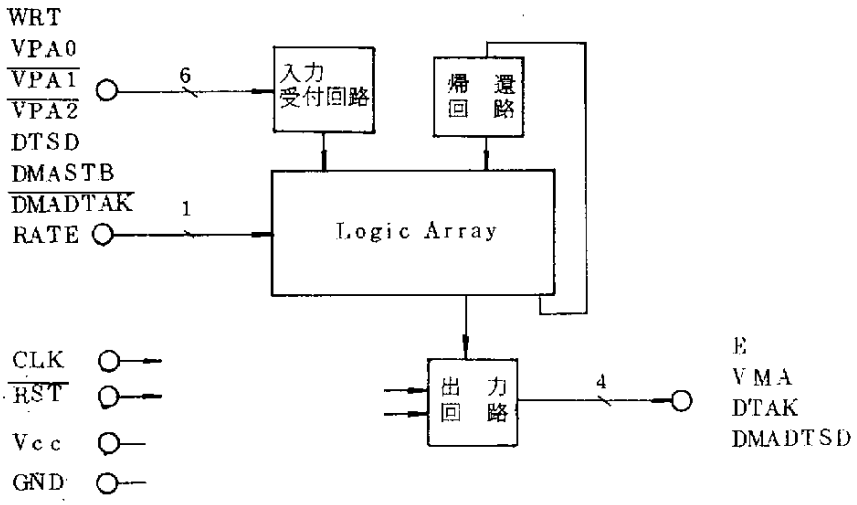


図３－２１　バス変換チップＭＮ１６６８

表３－３　ＭＮ１６１３命令セット一覧

型	記　号	命　令
RM	B	Branch
	BAL	Branch and Link
	IMS	Increment Memory and Skip if Result is Zero
	DMS	Decrement Memory and Skip if Result is Zero
	L	Load
	ST	Store
RR	BSWP	Byte Swap
	DSWP	Digit Swap
	MV	Move
	MVB	Move Byte
	AND	Logical And
	LAD	Load Adjust Port
	OR	Inclusive or
	EOR	Exclusive or
	A	Add
	S	Subtract
	C	Compare
	CB	Compare Byte

型	記　号	命　令
RC	SL	Shift Left 1 bit
	SR	Shift Right 1 bit
	PUSH	Push
	POP	Pop
	RET	Return
	H	Halt
	LPSW	Load Program Status Word
	SBIT	Set bit
	RBIT	Reset bit
	TBIT	Test bit
	AI	Add Immediate
	SI	Subtract Immediate
ROM	MVI	Move Immediate
	RD	Read
	WT	Write
	NOP	No Operation
	SKIP	Skip
	CLEAR	Clear

型	記　号	命　令
RIM	MVWI	Move word Immediate
	ANDI	And Immediate
	LADI	Load Adjust Part Immediate
	ORI	Inclusive or Immediate
	EORI	Exclusive or Immediate
	AWI	Add word Immediate
	SWI	Subtract word Immediate
	CWI	Compare word Immediate
	CBI	Compare Byte Immediate
RORI	MVWR	Move word Register Indirect
	MVBR	Move Byte Register Indirect
	BSWR	Byte Swap Register Indirect
	DSWR	Digit Swap Register Indirect
	ANDA	Logical AND Register Indirect
	LADR	Load Adjust Part Register Indirect
	ORR	Inclusive or Register Indirect
	EORR	Exclusive or Register Indirect

型	記　号	命　令
RORI	AWR	Add word Register Indirect
	SWR	Subtract Word Register Indirect
	CWR	Compare Word Register Indirect
	CBR	Compare Byte Register Indirect
	DAA	Decimal Adjust Addition with Carry
	DAS	Decimal Adjust Subtraction with Carry
	AD	Add Double with Carry
	SD	Subtract Double with Carry
	M	Multiply
	D	Divide
	FA	Floating Add
	FS	Floating Subtract
	FM	Floating Multiply
	FD	Floatirg Divide
RRI	LR	Load Register Indirect
	STR	Store Register Indirect
	RDR	Read Register Indirect

型	記　号	命　令
RRI	WTR	Write Register Indirect
	BR	Branch Register Indirect
	BALR	Branch and Link Register Indirect
RD	TSET	Test and Set
	TRST	Test and Reset
	LD	Load Direct
	STD	Store Direct
	BD	Branch Direct
	BALD	Branch and Link Direct
	BL	Branch Long
	BALL	Branch and Link Long
RCE	NEG	Negate with Carry
	FIX	Fix
	FLT	Float
	PSHM	Push Multi
	POPM	Pop Multi
	RETL	Return Long
	BLK	Move Date Block

型	記　号	
SRD	LB	Load Base Register
	STB	Store Base Register
	LS	Load Special Register
	STS	Store Special Register
SRR	CPYB	Copy Base Register
	SETB	Store Base Register
	CPYH	Copy Hardware Control Register
	SETH	Set Hardware Control Register
	CPYS	Copy Special Register
	SETS	Set Special Register
	DEBP	Decode Bit Position
	SRBT	Search Bit Position

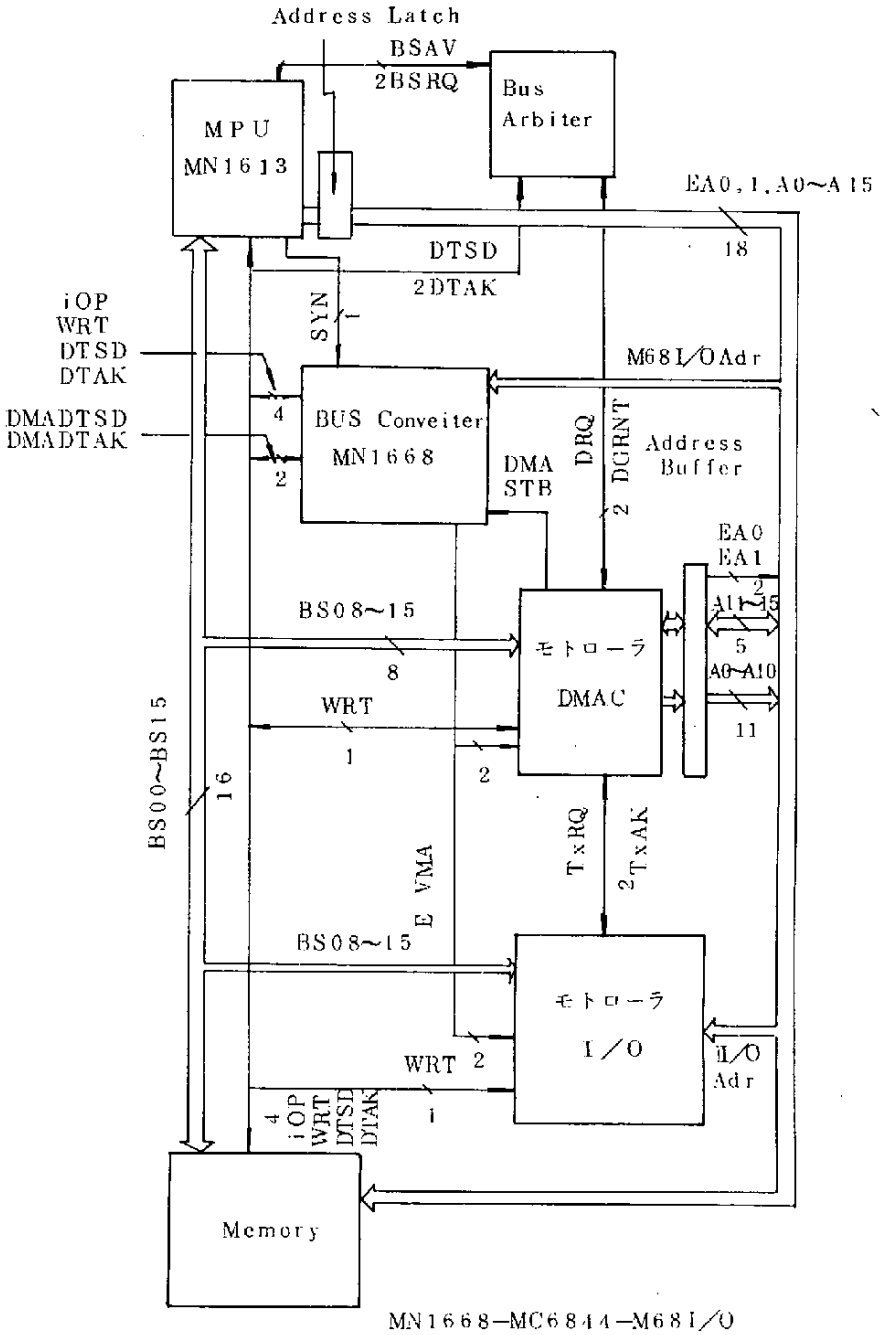
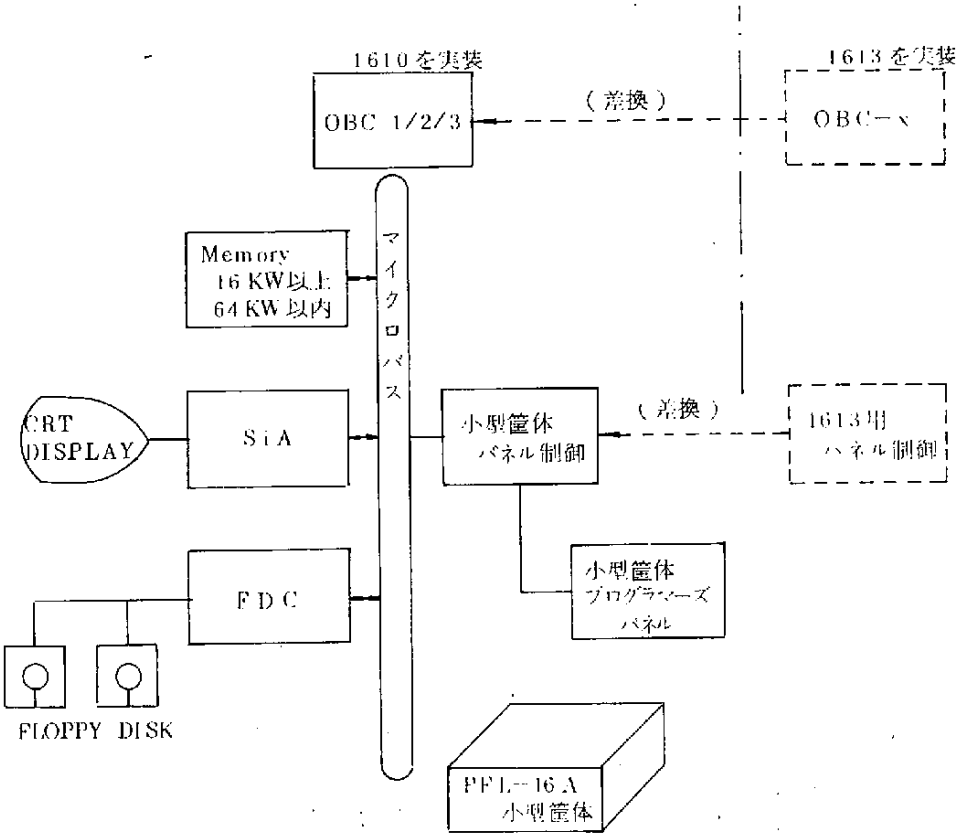


図3－22　システム構成例



FPS－16 現行システム

図３－２３　サポートシステム

3.2.7　ミニコン系１６ビットマイコン

マイクロプロセッサを部品として従来のミニコンピュータとソフトウェアの互換性をもつ１６ビット小型コンピュータが登場している。ここではその一例としてDEC社のLSI11／23をとりあげその概要を説明する。

ＬＳＩ－１１／２３は，ＬＳＩ－１１／２の上位方向互換性を持つ１６ビット・シングルボードコンピュータとして，１９７９年３月に発表された。ＤＥＣミニコンピュータＰＤＰ－１１シリーズの中位機種ＰＤＰ－１１／３４と機能的にコンパチブルな性能を持つ。

その特徴は次のとおりである。

- インストラクションの種類およびメモリーマネージメントはＰＤＰ－１１／３４と完全に互換性を持つ。
- マルチタスク／マルチプログラミング用オペレーティングシステムＲＳＸ－１１Ｍの使用が可能である。
- ＬＳＩ－11，ＬＳＩ－11／2 とバスコンパチブルであり，ＣＰＵボードの交換だけでＬＳＩ／２３システムを作成することが可能である
- 処理速度はＰＤＰ－１１／３４の約９０％程度の性能を持つ。
- ＣＰＵボードの外形は，13.2×22.8*cm* とコンパクトである。

(1) ハードウェアの構成と特徴

ブロックダイヤグラムを図３－２４に，性能仕様を表３－４に示す。

(a) 特　徴

- ４０ピンＭＯＳ／ＬＳＩ３チップ構成
- ベクタード方式４レベルの割込み制御
- 最大２５６Ｋバイトのメモリーサイズ
- ページごとのダイナミックアロケーション
- ８個の１６ビット多目的汎用レジスタ
- ４００を越える命令群
- micro ODT（Octal Debgging Technique）のファームウェア化

(b) 構成の概要

プロセッサは，３つのＭＯＳ／ＬＳＩチップと２つのバスから構成される。前者は，データ／コントロール・チップ，メモリーマネージメント・ユニット（ＭＭＵ），フローティングポイント・プロセッサ（ＦＰＰ）であり，後者は，マイクロインストラクションバス（ＭＩＢ）とデータ／アドレス・ライン（ＤＡＬ）である。

① データ／コントロールチップ

データチップ（ＤＣ３０２）とコントロールチップ（ＤＣ３０３）をパッキングした４０ピンハイブリッドチップである。ＤＣ３０２は，８個のジェネラルレジスタ（ＧＲ），プロセッサステータスワード（ＰＳ）（図３－２５に内容を示す。）数個のワーキングレジスタ，アリスメティックコントロールユニット（ＡＬＵ）と条件ブランチ回路から構成され，演算・論理・判断のすべてを行い，ＬＳＩ－１１バスとのデータアドレスの転送も行なう。また，アドレスのリロケーションを行う場合は，ＭＭＵチップをアクセスする。ＤＣ３０３は，５５２ワードのマイクロプログラムストレージ，制御信号を入力条件によって選択出力するプログラマブルロジック・アレイ（ＰＬＡ）とＰＤＰ－１１インストラクションのデコード，実行を行うためのマイクロコードを格納したＲＯＭから構成され，プロセッサ全体の主制御を行う。

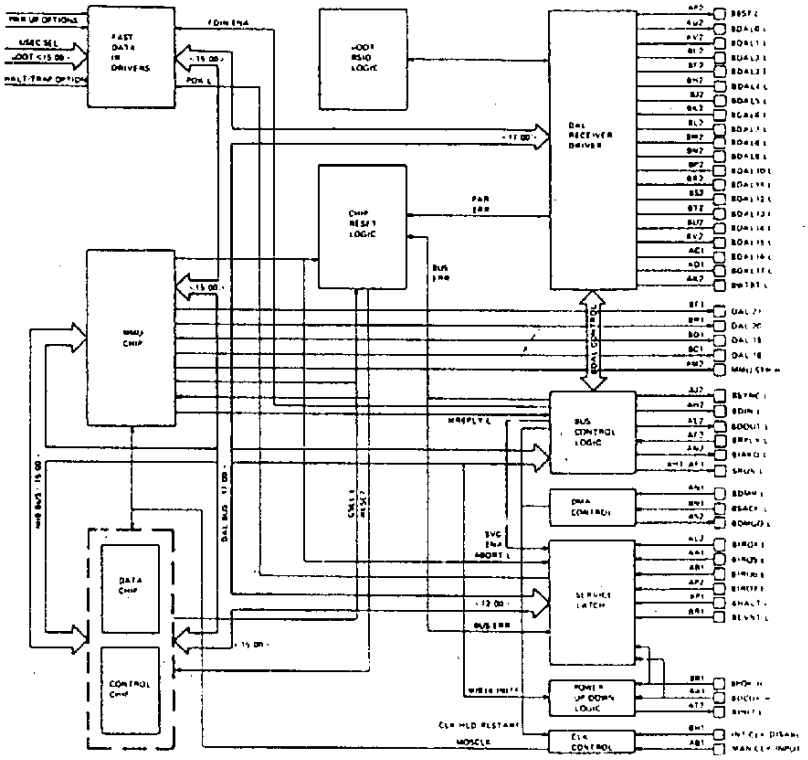
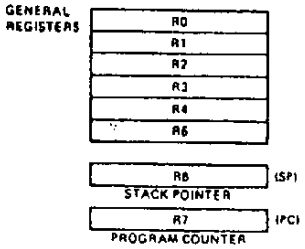


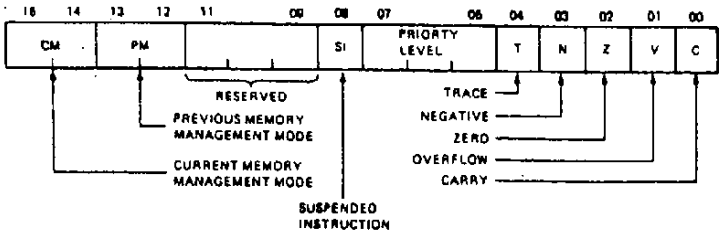
図３－２４　ＬＳＩ－１１／２３ブロックダイヤグラム

表３－４　ＬＳＩ－１１／２３性能仕様

Power Requirements	+5 V ± 5%, 2.0 A +12 V ± 5%, 0.2 A
Bus Loads	ac 2 unit loads dc 1 unit load
Environmental	
Storage	40° C to 65° C (104° F to 149° F) 10% to 90% relative humidity, non condensing
Operating	5° C to 60° C, (41° F to 140° F) Maximum outlet temperature rise of 5° C (9° F) above 60° C (140° F) Derate maximum temperature by 1° C (1.8° F) for each 305 m (1000 ft) above 2440 m (8000 ft).
Timing (Based on 300 ns CPU microcycle time)	
(Refer to Appendix A for detailed listing of instruction times.)	
Interrupt Latency (based on MSV11-D without parity, add 500 ns worst case with parity)	
Worst Case	55.7 microseconds (for infrequently used instructions) 10.8 microseconds (for more frequently used group)
Typical	6.0 microseconds
Interrupt Service Time	8.2 microseconds
DMA Latency	3.49 microseconds (worst case)



General Register



Processor Status Word (PS)

図３－２５　ゼネラルレジスタ（GR）とプログラムステータスワード（PS）

② MMU（Memory Management Unit）

MMUは，１８ビットアドレスのための各種レジスタと，ＦＰ１１浮動小数点演算機能のためのレジスタアキュームレータで構成され，次のような機能を持つ。

- タイムシェアリングに有効な２種のオペレーションモード（KERNEL，USER）を持つ。ＫＥＲＮＥＬ モード下では，モニタあるいはスーパーバイザプログラムが実行され，ＵＳＥＲのモード下ではＫＥＲＮＥＬモード下で設定された条件のもとでプログラムは実行される。
- 各モード下には各々専用の８個の３２ビットアクティブページ・レジスタ（ＡＰＲ）があり，ＡＰＲは１６ビットのページアドレス・レジスタ（ＰＡＲ），ページディレクティブ・レジスタ（ＰＤＲ）から構成され（図３－２６に示す）リロケーション，サイズ，プロテクションのために使用される。
- ＡＰＲ内のＰＡＲの下位１２ビットをリロケーション定数とし，１６ビットのバーチャルアドレス（ＶＡ）の上位３ビットとマッチングさせ，１８ビットのフィジカルアドレス（ＰＡ）にリロケーションさせる。（図３－２７）

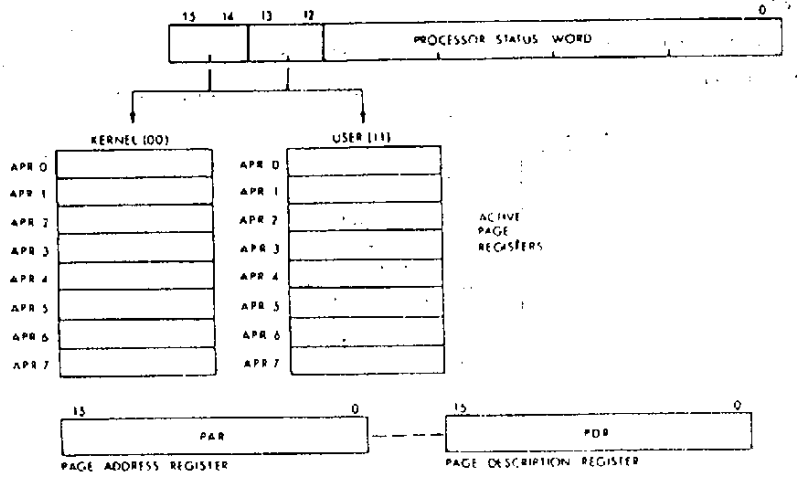


図３－２６　アクティブページ・レジスタ（ＡＰＲ）の構成

- メモリープロテクション機能は，ＰＳ，ＰＡＲ，ＰＤＲによる３種の保護機能がある。（表３－５）

③ ＦＰＰ（Floating Point Processor）

ＦＰＰは２つのＭＯＳ／ＬＳＩチップをパッキングした４０ピンハイブリットチップであり，４６種の命令群を持つ。単精度（３２ビット），倍精度（６４ビット）の演算，固

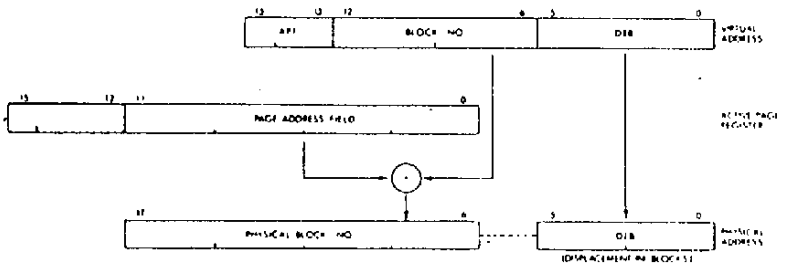


図３－２７　フィジカル・アドレス（ＰＡ）の作成

表３－５　プログラムステータスワード（ＰＳ）にする保護

Processor Status Word Protection

	RTI,RTT		Traps & Interrupts		Explicit PS Access		MTPS		Power Up
PS Bits	User	Kernel	User	Kernel	User	Kernel	User	Kernel	
Condition Code PS<3:0>	Loaded From Stack	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Loaded From Source	Loaded From Source	Cleared
Trap Bit PS4	Loaded From Stack	Loaded From Stack	Loaded From Vector	Loaded From Vector	Unchanged	Unchanged	Unchanged	Unchanged	Cleared
Interrupt Priority PS<7:5>	Unchanged	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Unchanged	Loaded From Source	SET when powered up to bootstrap Cleared when powered up to ODT, loc 24, or microcode
SI PS 8	Loaded From Stack	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Non-Accessible	Non-Accessible	Cleared
Previous Mode PS<13:12>	Unchanged	Loaded From Stack	Copied From PS <15:14>	Copied From PS <15:14>	Loaded From Source	Loaded From Source	Non-accessible	Non-Accessible	Cleared
Current Mode PS<15:14>	Unchanged	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Non-accessible	Non-accessible	Cleared

定小数点から浮動小数点および浮動小数点から固定小数点へのデコートが可能である。チップの内容は，命令実行のためのマイクロコードＲＯＭであり，実際の実行は，ＭＭＵチップ内の浮動小数点レジスタ，ステータスレジスタが使用される。

④ ＤＡＬ（Ｄａｔａ／Ａｄｄｒｅｓｓ Ｌｉｎｅ）

プロセッサ，モジュール内の各チップ間およびＬＳＩ－１１バスのデータとアドレスの転送を行なう１８ビットバスである。

⑤ ＭＩＢ（Ｍｉｃｒｏ Ｉｎｓｔｒｕｃｔｉｏｎ Ｂｕｓ）

プロセッサ内の各チップ間を結合する１６ビットバスであり，１６ビットデータのセットを行なうときのみ使用される。

(2) ソフトウェアの特徴

(a) 特　徴

- ＬＳＩ－１１／２の上方向互換性を持ち，処理速度は２～４倍である。
- ＰＳＸ－１１／Ｍの使用が可能。
- １２種のアドレスモード。
- ４００を越える命令群はすべてＬＳＩ－１１／３４と共通。
- 浮動小数点命令群もすべてＬＳＩ－１１と共通。

インストラクションセットを表３－６に示す。

表３－６　インストラクションセット

(1) ＬＳＩ－11／2 インストラクションセット

オペコード (8進)	ニモニック	
00 00 00	HALT	
00 00 01	WAIT	
00 00 02	RTI	return from interrupt
00 00 03	BPT	breakpoint trap
00 00 04	IOT	input/output trap
00 00 05	RESET	
00 00 06	RTT	return from interrupt
00 00 07	(unused)	
00 00 77		
00 01 DD	JMP	jump
00 02 0R	RTS	return from subroutine
00 02 10	(reserved)	
00 02 27		
00 02 40	NOP	
00 02 41	cond codes (注1)	
00 02 77		
00 03 DD	SWAB	swap bytes
00 04 XXX	BR	branch (unconditional)
00 10 XXX	BNE	br if not equal (to 0)
00 14 XXX	BEQ	br if equal (to 0)
00 20 XXX	BGE	br if greater or equal (to 0)
00 24 XXX	BLT	br if less than (0)
00 30 XXX	BGT	br if greater than (0)
00 34 XXX	BLE	br if less or equal (to 0)
00 4R DD	JSR	jump to subroutine
00 50 DD	CLR	clear
00 51 DD	COM	complement (1's)
00 52 DD	INC	increment
00 53 DD	DEC	decrement
00 54 DD	NEG	negate (2's compl)
00 55 DD	ADC	add carry
00 56 DD	SBC	subtract carry
00 57 DD	TST	test

オペコード	ニモニック	
00 60 DD	ROR	rotate right
00 61 DD	ROL	rotate left
00 62 DD	ASR	arith shift right
00 63 DD	ASL	arith shift left
00 64 NN	MARK	mark
00 67 DD	SXT	sign extend
00 70 00	(unused)	
00 77 77		
01 SS DD	MOV	move
02 SS DD	CMP	compare
03 SS DD	BIT	bit test (AND)
04 SS DD	BIC	bit clear
05 SS DD	BIS	bit set (OR)
06 SS DD	ADD	add
07 0R SS	MUL	multiply
07 1R SS	DIV	divide
07 2R SS	ASH	shift arithmetically
07 3R SS	ASHC	arith shift combined
07 4R DD	XOR	exclusive (OR)
07 50 0R	FADD	floating add
07 50 1R	FSUB	floating subtract
07 50 2R	FMUL	floating multiply
07 50 3R	FDIV	floating divide
07 50 40	(unused)	
07 67 77		
07 7R NN	SOB	subtract 1 & br (if≠0)
10 00 XXX	BPL	branch if plus
10 04 XXX	BMI	branch if minus
10 10 XXX	BHI	branch if higher
10 14 XXX	BLOS	branch if lower or same
10 20 XXX	BVC	br if overflow is clear
10 24 XXX	BVS	br if overflow is set
10 30 XXX	BCC	br if carry is clear
	BHIS	branch if higher or same
10 34 XXX	BCS	br if carry is set
	BLO	branch if lower

オペコード	ニモニック	
10 40 00	EMT	emulator trap (not for general use)
10 43 77		
10 44 00	TRAP	trap
10 47 77		
10 50 DD	CLRB	clear
10 51 DD	COMB	complement (1's)
10 52 DD	INCB	increment
10 53 DD	DECB	decrement
10 54 DD	NEGB	negate (2's compl)
10 55 DD	ADCB	add carry
10 56 DD	SBCB	subtract carry
10 57 DD	TSTB	test
10 60 DD	RORB	rotate right
10 61 DD	ROLB	rotate left
10 62 DD	ASRB	arith shift right
10 63 DD	ASLB	arith shift left
10 64 SS	MTPS	move byte to PS
10 67 DD	MFPS	move byte from PS
11 SS DD	MOVB	move
12 SS DD	CMPB	compare
13 SS DD	BITB	bit test (AND)
14 SS DD	BICB	bit clear
15 SS DD	BISB	bit set (OR)
16 SS DD	SUB	subtract

R：汎用レジスタ (0～7)
XXX：相対アドレス (±128)

(ロ) ＬＳＩ－１１／２３追加インストラクション

オペコード（８進）	ニモニック	
00 02 41	CLC	clear C
00 02 42	CLV	clear V
00 02 44	CLZ	clear Z
00 02 50	CLN	clear L
00 02 57	CCC	clear all condition
00 02 61	SEC	set C
00 02 62	SEN	set N
00 02 63	SEV	set V
00 02 64	SEZ	set Z
00 02 77	SCC	set all es

（注１）　LSI-11/23では，(イ)のインストラクションの（注１）項に(ロ)で示したインストラクションが追加された。

（注２）　ＳＳはソースレジスタ，ＤＤはデスティネイションレジスタを意味する。

(b)　ＯＤＴ（Octal Debuggiug Technique）

１０種類の命令群を持ち，ターミナルからの入力によりプロセッサの実行を制御するコンソール・ルーチンがファームウェア化されている。ＯＯＴ命令群を表３－７に示す。

表３－７　ＯＤＴ命令群

Command	Symbol	Use
Slash	/	Prints the contents of a specified location.
Carriage Return	<CR>	Closes an open location.
Line Feed	<LF>	Closes an open location and then opens the next contiguous location.
Internal Register Designator	S or R	Opens a specific processor register.
Processor Status Word Designator	S	Opens the PS—must follow an S or R command.
Go	G	Starts program execution.
Proceed	P	Resumes execution of a program.
Binary Dump	Control Shifts-S	Manufacturing use only.
	H	Reserved for DIGITAL use.

(3) システムの構成

メモリー一覧表を表３－８に示す。

モジュール一覧表を表３－９に示す。

基本システムの構成例を図３－２８に示す。

表３－８　メモリー一覧表

◦ MMV11−A	4Kバイト・16ビットコアメモリー
◦ MRV11−AA	4Kバイト　16ビットROM
◦ MRV11−BA	LSI11　UV　PROM／RAM
◦ MRV11−C	ROMモジュール
◦ MSV11−B	4Kバイト　16ビットダイナミックRAM
◦ MSV11−CD	16Kバイト　16ビットダイナミックRAM
◦ MSV11−D−E	メモリモジュール
◦ MXV11−A	LSI−11マルチファンクションモジュールRAM

表３－９　モジュール一覧表

モジュールシステム	
◦ KD11-F	CPU モジュール　4K RAM 付
◦ KD11-J	CPU モジュール　4K CORE 付
◦ KD11-R	CPU モジュール　16K RAM 付
◦ AAV11-A	12 ビット D/A コンバータモジュール
◦ ADV11-A	12 ビット A/D コンバータモジュール
◦ KWV11-A	プログラマブルリアルタイムクロックモジュール
◦ KEV11	固定/浮動小数点・乗除算用オプション
◦ RXV11	デュアルドライブフロッピディスク
◦ VT50	12 行×80 桁 CRT ディスプレイ
◦ VT52	24 行×80 桁 CRT ディスプレイ
◦ VT55	グラフィック CRT ディスプレイ
◦ LA36	30 文字/秒ドットマトリックスキーボードプリンタ
◦ PDP-11/03	キャビネット形 LSI-11 マイクロコンピュータ
◦ PDP-11 V 03	フロッピディスクベース LSI-11 マイクロコンピュータシステム（リアルタイムオペレーティングシステム付）

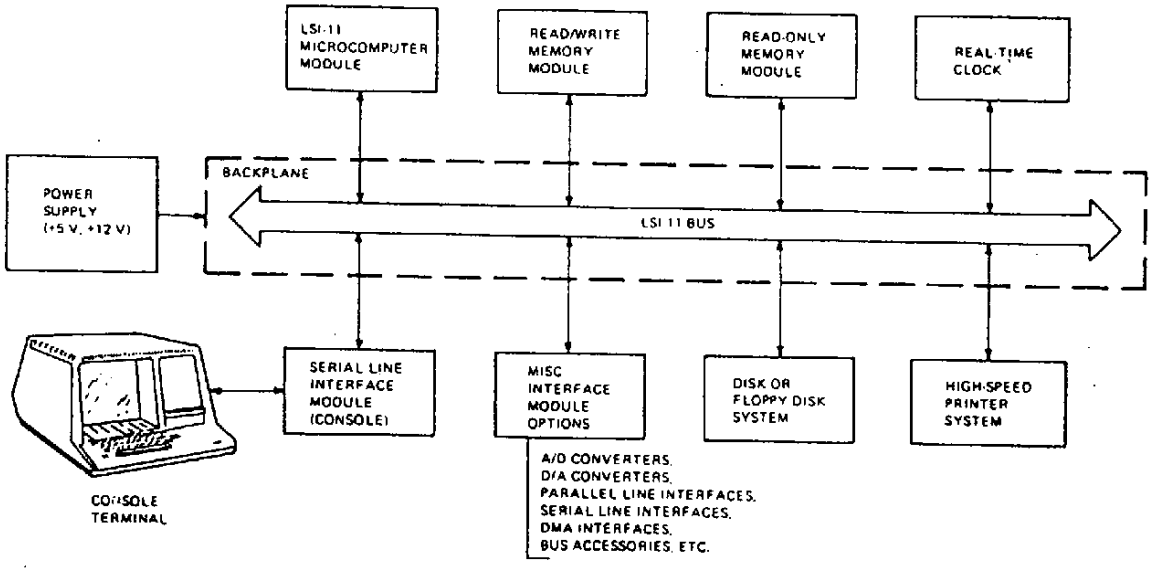


図３－２８　基本システム構成例

(4) 開発サポート

開発サポート一覧表を表３－10に示す。

表３－10 開発サポート一覧表

開発サポート用ソフトウェア

- マクロアセンブラ，ローダ，エディタ，デバッガ，ライブラリア
- ＯＳ：リアルタイム用　ＲＴ－１１
  マルチタスク／マルチプログラミング　ＲＳＸ－１１Ｍ
- コンパイラー：FORTRAN Ⅳ-PLUS, BASIC-PLUS-2, FORCAL, DATATRIEVE, MACRO
- ファイル管理ユーティリティ：Filee-11, RMS-11
- ネットワークシステム：DECNET

〔参考文献〕

(1) ＤＥＣ社：microcmputer processor handbook

(2) 森亮一編：マイクロコンピュータハンドブック，朝倉書店。

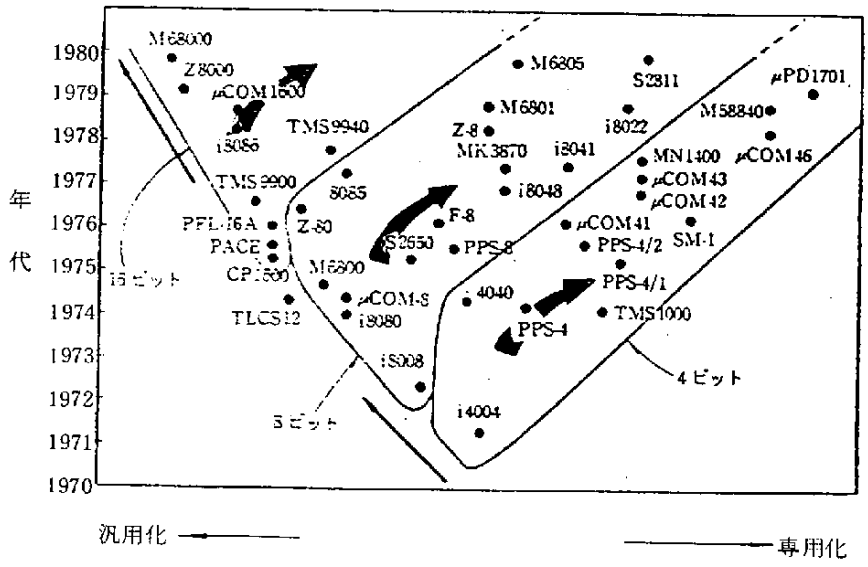
第４章　１６ビットマイクロコンピュータの応用分野と利用動向

第４章　１６ビットマイクロコンピュータの応用分野と利用動向

4.1　導入の背景

4.1.1　マイコンの発展経緯の中の１６ビットマイコン

１９７１年インテルの４ビットマイコン（ｉ４００４）から始まったマイコンの発展経緯を見ると，用途の多様化，特定マーケットの拡大並びにデバイスのプロセス技術（大集積化，高性能化）の発展等により，一方は専用化，他方は汎用化という２方向に発展して来ている。これ等の製品の開発時期と専用化，汎用化の方向を図式化すると図４－１になる。この図からわかるように，４ビットは，当初，ある程度汎用性を持ったマルチ・チップ構成であったものが，年代と共に集積度が上がり，また用途が拡大したことにより，ＲＯＭ，ＲＡＭ，Ｉ／Ｏを限定した，２チップシステムから１チップシステムへと発展し，最近はデジタル・チューニング・システムの様な用途を限定した専用マイコンの出現に到っている。



注）　ＣＭＯＳ化した製品は図４－１の発展経緯よりも数年の遅れが見られる。またバイポーラ・ビット・スライスのマイクロプロセッサは，ミニコンや汎用大型機のコンポーネントとしての性格が強いので本図には載せていない。

図４－１　汎用化，専用化という面から見たマイクロプロセサの発展経緯

一方８ビットにおいても同様で，マルチ・チップシステムからスタートし，集積度と用途拡大により，１チップシステム製品が加えられ，最近は４ビットと同様に信号処理の様な専用マイコンの出現に到っている。ここで注目すべきことは，４ビットは現在殆んどシングル（１チップのこと）のみであるが，８ビットはシングルとマルチが共存していることである。これは用途によりシステム規模が違うからであるが，４ビットのマルチは８ビットのマルチが低価格になったこともあり，全て８ビットに吸収されて来ている。

さて，１６ビットの開発は１９７３年から各社より発表されてきているが，当初は１６ビットシステムであるミニコンをエミュレートするためとか，マイクロプログラミング制御方式の特徴を生かす高速制御装置等の限られた用途のみ使用され，デバイス自身もミニコン志向が強かった。１６ビットマイコンをフルに生かすためにはかなりのソフトウェアの蓄積を要する。この点ミニコンメーカは大きな蓄積を持っており，このソフトウェアを活用するため，ミニコンのアーキテクチャをそのままＬＳＩ化したマイコンが作られた。これはＬＳＩメーカが，このことに気付き作ったものと，ミニコンメーカがミニコンのローエンドの応用分野を８ビット系マイコンに侵蝕されることを恐れ，対抗上作ったものとがある。その後，４ビット，８ビットと同様プロセス技術の発展により１６ビットの第２世代とも言うべき性能，機能が強化されたＣＰＵが各社から発表されて来ている。これ等はより汎用性が高く，従来のミニコン志向を一歩脱却しており，そこに応用の発展性が伺われる。そして今後は８ビットの経緯と同様に，１チップ製品が加えられ，８ビットのマルチの一部は１６ビットのマルチに吸収され，シングルとマルチが共存して行くものと思われる。一方マルチの発展方向としては，より高性能，高機能を持つ３２ビットが開発されて行くことであろう。

4.1.2　１６ビットマイコンの応用と導入理由

前節で述べたようなデバイスの発展により応用分野の変化が予想されるが，現在まだ，第２世代の１６ビットマイコンがあまり市場に出回っていない現状における８ビットおよび１６ビットの応用分野は表４－１の様になっている。しかし，この応用分野の中で，従来８ビットで処理していたものの一部は１６ビットに移行し，また新たな応用分野が出現することが予想される。それは次のような理由によるものと思われる。

(1)　１６ビットＣＰＵ自体，第２世代とも言うべき性能，機能が強化され，より汎用性が高く，ある程度の標準化（即ちハードの固定化）が進み，セカンド・ソースの出現を見たこと。このことは８ビットと同じ状態に到って来ている。

(2)　セカンド・ソースの出現により，入手し易くなり，パフォーマンス／コスト比も高くなりつつある。

(3)　ＲＯＭ，ＲＡＭ，Ｉ／Ｏ等周辺デバイスの価格低下。

(4)　システムの応用上からは，性能，機能の向上を図る必要が生じていること。

(5)　第２世代１６ビットマイコンの性能，機能の特徴の一部または全部を活用した新マーケッ

トの展開。

表４－１　８ビット，１６ビットの応用分野の現状

	８ビット		16ビット/ビットスライム
	Single	Multi	Multi
特　徴	・システム規模　小 ・中速 ・安価 ・大量生産向	・システム規模　中 ・中速 ・普通	・システム規模　大 ・高速 ・高価
用　途	・ＥＣＲ／ＰＯＳ ・電子はかり ・自動車エンジン制御 ・簡単なシーケンスコントロール ・簡単な複写機	・高級ＥＣＲ／ＰＯＳ ・複写機 ・タイプライタ ・オフイスコンピュータ ・パーソナルコンピュータ ・ワードプロセッサ ・バンキングシステム ・データ集収装置 ・シーケンスコントローラ ・自動改札機 ・ファクシミリ ・グラフィックターミナル ・その他，機械制御及び各種制御装置	・ミニコンのエミュレート ・ＮＣ ・プロセス制御 ・医療検査システム ・電話交換機制御 ・高速ファクシミリ ・その他，高速制御装置

この中で性能，機能が強化された汎用面における第２世代とも言うべきプロセッサの共通的な特徴を以下に示す。

(1)　メモリアドレス能力強化

少なくともＭ（メガ）バイトまで直接アクセス可能。

(2)　マルチ・プロセシングが容易に出来る構造を持つ

全ての処理内容を一つのＣＰＵで実行するのが困難な場合，複数のＣＰＵを使用し，仕事の分散化を図る構成を取ることが多い。このように分散処理を行った方が，上位のＣＰＵ（例えばミニコン）を使用するより有利な場合が多い。マルチバス・アーキテクチャもこの一つの方法である。

(3)　命令の強化

メモリがＭバイトと大きくなっているため，アドレッシング向上のためのセグメント化（論理的分割）およびリロケーション機能も考慮されている。また演算スピードを向上させる乗除算命令を持っていること。割込強化およびマルチ・プロセシングを助ける命令群等多くの命令が追加されている。

(4) 高　速

システム・クロックの高速化と命令群の強化により，従来の１６ビットマルチより約３～８倍の処理スピードを持つ。

次に応用上の展開を促進するものに，システムに使用するデバイスのコストが重要である。８ビットＣＰＵが開発当初より極端に価格低下したのも，年々の半導体プロセス技術の進歩と，半導体特有のラーニングカーブによる。このことはＣＰＵより周辺に使用するメモリが一層重要である。何故ならば，プログラム方式であるマイコンシステムは，処理手順をストアしておくためのメモリを多く使用しなければならないからである。このメモリの価格低下の様子を図４－２に示す。メモリのコスト低下が，システム規模の拡大に影響し，４～５年前代表的システムが約３Ｋバイトであったものが，今日では約４０～４５Ｋバイトに規模が拡大して来ている。このソフト量の拡大はコストばかりでなく，システムの機能アップから来ていることでもある。このように，ＣＰＵ，メモリ等のハードウェアはコスト低下となったが，一方規模が拡大するにつれ，プログラムを開発するソフト開発費が大きくなってしまうという現象が現われ始めて来ている。従って，今後ソフト開発を効率よく行う方策が考えられるであろう。その一つの方策としてメーカ側から提供するプログラム開発用ＯＳがある。

以上述べた背景により，１６ビットマイコンが次第に市場に導入されて来るものと思われる。

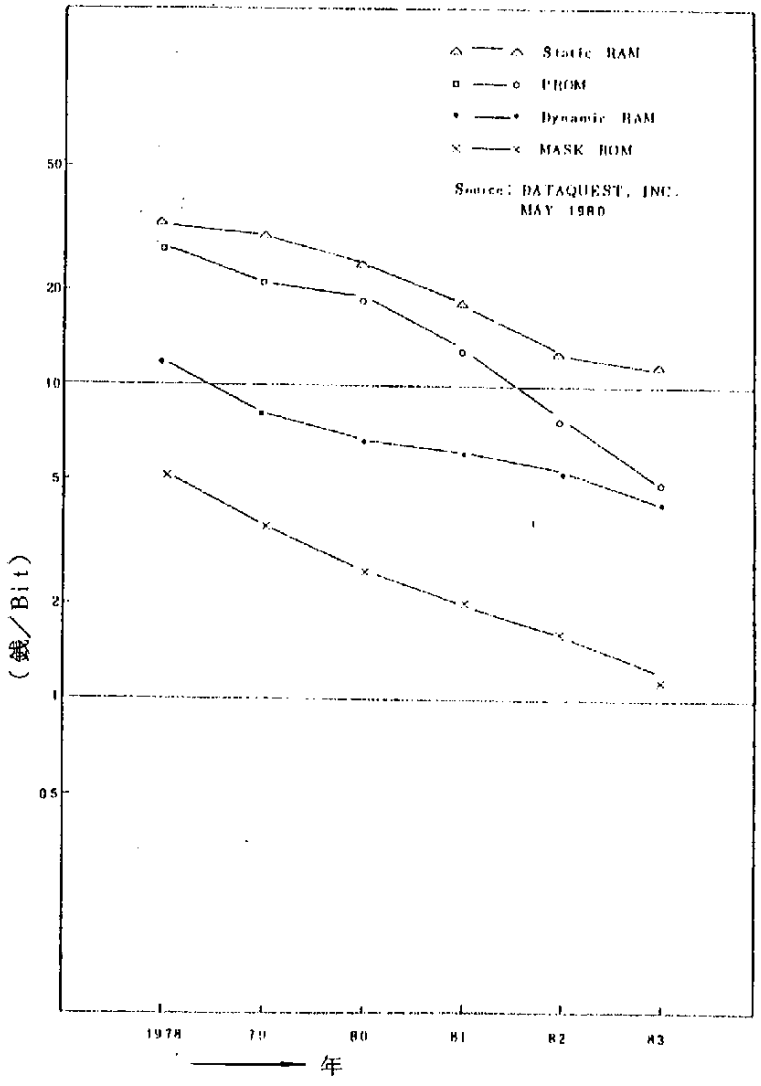


図４－２　メモリ価格推移（ｂｉｔ単価）

4.1.3　シングル・ボード・コンピュータへの応用

１６ビットマイコンが浸透するなかで，注目されるのが，ボード・ビジネスである。先日，日経産業新聞社が，企業に対して，これからのマイコン利用の意識調査を実施したが，それによると，生産の合理化，事務の合理化が１，２位を占めていた。この様な用途である例えば生産機械，計測，計装，制御など少量多品種生産でかつ高付加価値を持つシステムにおいては，技術者不足を補うものとして保守の容易な標準的なシングル・ボード・コンピュータ（ＳＢＣ）を利用した方が得策な場合が考えられる。１６ビットマイコンはこのようなマルチバス，スタンダードに最適なアーキテクチャとなっており，そのＳＢＣを使用する設計者に取っても便利なように，今後はボード上に，益々充実したモニタまたはＯＳが搭載されて来るものと考えられる。

〔参考文献〕

(1) 日経エレクトロニクス「特集・１９８０年代のエレクトロニクス」（２２９号　P120～123）（１９８０－１－７）

(2) 日本電気㈱μＣＯＭ８６ユーザーズ・マニュアル。

(3) 浅田勝彦：１６ビットマイクロコンピュータとその動向（ミニコン・マイクロコン研究分科会資料　１９７９－１０－２４）日本自動制御協会

(4) 日経エレクトロニクス：米インテル社の１９８０年代マイクロコンピュータ戦略（２４１号　P２４３～２５２），（１９８０－６－２３）

(5) 日経エレクトロニクス別刷「コンピュータ」１９７８－７９（P．４５～５７）

(6) 日本情報処理開発協会　マイクロコンピュータ応用上の課題と展望　昭和５５年３月

4.2　応用分野別利用動向

4.2.1　概　　要

最近注目されてきている１６ビットマイコンの利用はおおむね次の方向にあると言える。その一つは，従来は８ビット以下のマイコンが使用されていたがさらに多様な機能の付与または性能向上が必要となり１６ビットに移行するものと，もう一つはミニコンなどの利用から，必要な機能・性能を十分満たし，経済性の有利な１６ビットマイコンへの移行を行うものである。

このような１６ビットマイコンの応用分野を検討していくうえでは，１６ビットマイコンの特徴を明確することによってその動向が浮き彫りになってくると考えられ，性能，機能，経済性の点から考察してみる。

性能面においては命令実行時間とビット幅を勘案すると８ビットマイコンの５～１０倍，従来のミニコンの低位機種と同等以上となっており，機能面ではメモリアクセス能力を評価要素とすれば８ビットマイコン，従来ミニコンの低位機種の１０倍以上となっている。経済性については，現時点ではプロセッサそのものが８ビットに比べて高価なことや，データバスの倍増，アドレスバスの分離による回路増，並びに入出力ポートなどの周辺用ＬＳＩが８ビット用ほど整備されていないことなどから１６ビットマイコンを構成する場合にはかなり割高となっている。また，８ビット以下のマイコンではその使われ方が機器制御用すなわちコントローラ的利用か加減乗除や統計処理用すなわちコンピュータ的利用かが比較的明確であったのに対し，１６ビットでは，コントローラ的に使ってもコンピュータ的に使っても相応の能力を有しているので両方の要素を併せ持った機能が実現できることに特徴がある。

以上のように１６ビットマイコンは機能・性能においては優れてているものの経済性の面から，現時点では８ビット以下のマイコンが主流を占めているが，徐々に１６ビットマイコンが使われ始めている。これらの状況を分野を代表する機器について１６ビットマイコンの利用動向を考察すると表４－２のとおりとなる。

民生・家電の分野では住宅設備機器制御での利用が今後期待される。事務・商業の分野では日本語を扱うワードプロセッサや在庫管理機能を包含した電子レジスタやＰＯＳシステムが登場しこの中に１６ビットマイコンが組み込まれる。工業分野では数値制御の工作機械やプロセス制御用として，交通・輸送では広域運行監視や運行制御，在庫管理と製品の入出庫のための搬送機器制御を組み合せた自動倉庫システムでの利用が進んでいる。計測・試験分野では計測又は試験結果を高精度で統計処理または分析することとなるがこれを連携して行うには１６ビットマイコンが最適であり，従来はミニコンで実現されていたシステムもマイコンに移行してくると考えられる。通信の分野ではデータ通信の分散処理システム用のデータ端末，大容量のＰＡＢＸなどに１６ビットマイコンが利用されており，今後さらに利用が進むものと考えられる。データ処理の分野ではオフィスコンピュータが１６ビットマイコンの利用代表例であるが今後は１６ビットマイコンを複数個使用して従来の中小コンピュータ規模のものを実現する研究も行われている。

その他の分野としては防災を含めたビル管理システム，火力発電所のタービン制御や自動バーナ制御装置などで１６ビットマイコンが使われている。

表４－２　１６ビットマイクロプロセッサの応用分野

分野	応用機器	使用されるプロセッサ			選択される条件						
					処理		演算精度	メモリ空間	割込(多重)	データ幅	通信
		＜１６	１６	＞１６	演算	即時性					
民生・家電	自動車（制御・診断）	○					○		○		
	H.C（住宅設備機器制御）		○								○
	教育機器	○									
事務・商業	ワードプロセッサ	○ →	○			△		○		○	
	店頭機器（ECR，POS）	○ →	○					○			○
工業	機械制御	○	○	○	○	○	○	○	○	○	
	プロセス制御	○	○	○	○	○	○	○	○	○	
交通・輸送	運行制御	○	○					△	○		○
	自動倉庫	○	○					△	○		○
	駅務	○									
計測試験	監視	○									○
	検査		○ ←	○							
	ＭＥ	○	○	○	○	○	○	○	○	○	
	分析		○	○	○		○	○		○	
通信	通信制御（端末）	○ →	○			○			○		○
	画像端末	○ →	○			○		○	○	○	○
	無線装置（自動車電話）	○									
データ処理	Ｉ／Ｏ制御	○				○			○		
	オフィスコンピュータ		○					○	△		
	パーソナルコンピュータ	○	△								

4.2.2 利用動向

(1) 民生・家電

マイクロコンピュータの展開によって，

① 均一化と多様化；大量生産された均一商品の利用と個々の好みを反映した商品の利用とが共に求められる。

② 情報の個別化；テレビ・ラジオ・新聞などのマスコミ情報に加えて，放送や電話の高度利用によって個人的に必要な情報を選択的に利用することが求められる。

③ 資源・エネルギーの高度利用；家庭内のスペースの有効利用，エネルギーの収支バランスの管理による省エネルギー化などの要求がある。

④ 防災・防犯；物的損失や生命の安全に対する意識が高まっている。

⑤ 教養・娯楽；余暇の増加に伴い，知識や趣味に対する要求が大きい。

などに対処できる様になってきた。しかし，民生機器は一般にシステム規模が小さく，しかもコスト要求が極めて強いため，ワンチップマイクロコンピュータが利用されることが多い。16ビットマイクロコンピュータの応用が展望されるのは「ホームコンピュータ」が代表的な例である。ホームコンピュータはより安全，快適，経済的で便利な家庭を満足するための家庭システムであり図4－3に示すホームコンピュータを中心に，次のサブシステムから構成されている。

① 安全のためのホームセキュリティ

② 経済性を確保するためのエネルギー管理

③ 居住性を保証するハウスコントロール

④ 生活を楽しむための家事管理，余暇利用

このシステムはホームコンピュータと3線式情報ライン，サブコントローラから成る。情報ラインは情報コンセントを介して任意の機器と接続できる宅内伝送路システムである。システムの機能は次の通りである。

① セキュリティ，防災防犯のための自動監視，異常時の通報，ドアホン入力と音声応答および自動ドア

② オートダイアル；交際録から番号を検索し，自動ダイアルする。

③ ホームファクシミリ；家庭用ファクシミリの同報通信

④ 電話リモコン；宅外の電話器から指令信号発振器（テレコマンダ）を利用して，家庭内の機器を制御

⑤ エネルギー管理；メータ類の自動検針とオンオフ時刻のプログラム設定，ピークカット

⑥ ソーラCHCシステム；太陽熱利用の冷暖房および給湯システムの制御と監視

⑦ 家計簿；家計簿の作成と予算・実績の対比，統計グラフの表示，交際録の参照および住所氏名の印刷

⑧ ホームクッキング；献立の選択，電子レンジとガスオーブンによる自動調理

⑨ CAI：プログラム学習方式による自己学習

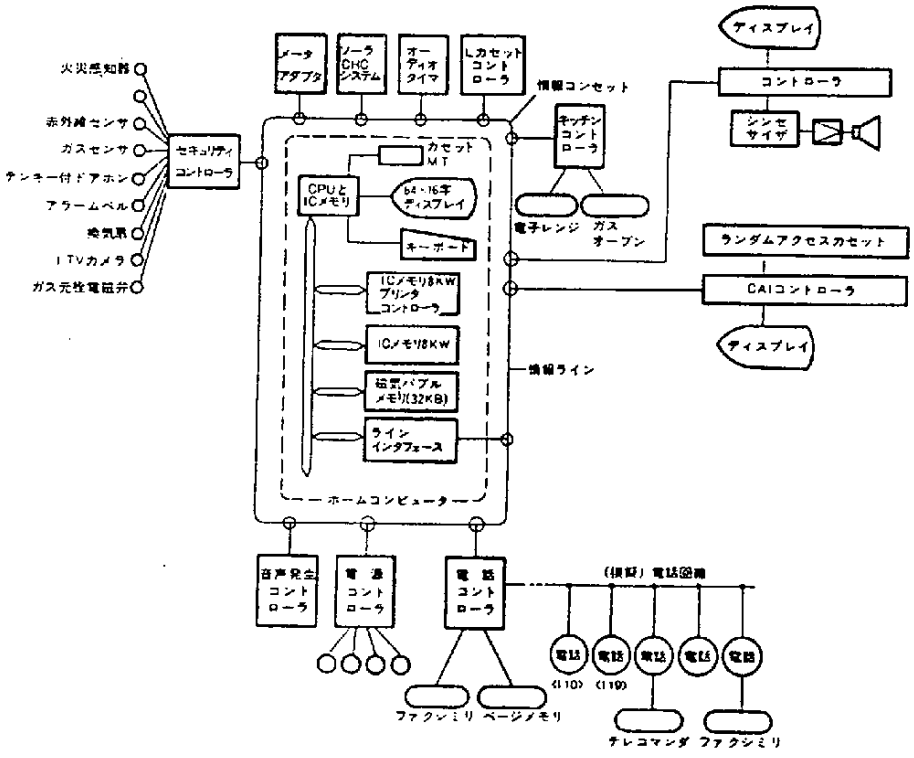


図４－３　ホームコンピュータの構成

これらプログラムは核となる並列多重処理モニタを中心として図４－４の様な構成を採っている。また，このシステム中のセキュリティサブシステムは図４－５の如く構成される。このサブシステムは

① 煙センサ，熱センサ，ガスセンサによる火災とガス漏れの検知

② ガラス破壊センサと赤外線センサ，ドアカメラによる侵入監視および暗証番号とドアホンによる侵入防止

③ 回線の自動診断

を実行し，異常発生時，ホームコンピュータのディスレイ上に異常場所を表示し，合成音声によって警報を発する。留守中であれば，自動ダイアルによって，１１０番，１１９番に自動通報する。ガス漏れの場合は換気扇を作動し，元栓を閉塞して二次災害を防止する。

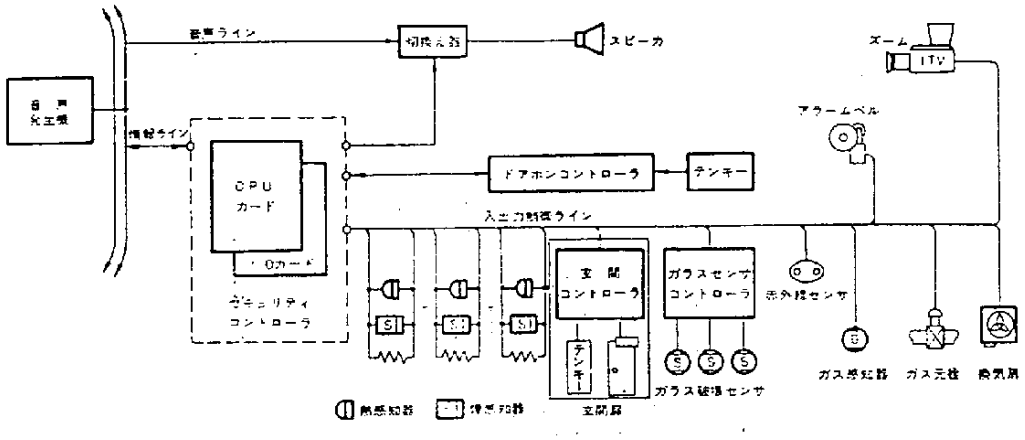


図4－4　ホームコンピュータシステムのソフトウェアの構成

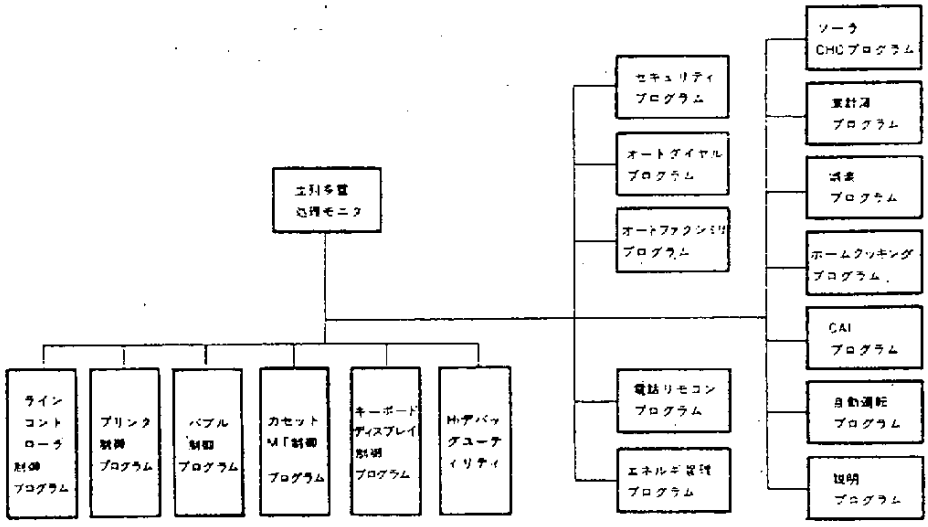


図4－5　ホームセキュリティサブシステム

(2) 事務・商業

小売会計器，金融端末，ワードプロセッサ等で代表される事務・商業分野における応用製品名，応用システム名を表４－３に示す。

表４－３　事務・商業分野におけるマイクロコンピュータの具体的応用製品名，応用システム名 (1)

用　　途	具代的応用機器・応用システム
1　事務計算	会計計算機，病院事務管理
2. 販売・在庫管理	在庫管理機，販売管理
3. ワード処理	タイプライタ，ワードプロセッサ
4. 小売会計	ＰＯＳ，ＥＣＲ
5. 金融端末	端末，金融端末，競馬端末，金融窓口装置
	オンライン・テラーズマシン
6. 事務機，その他	複写機，オフィスコンピュータ，グラフ作図器
	商業用はかり，選果システム　等

(a) 利用動向

事務・商業分野におけるマイクロコンピュータの利用に関して，購入されたビット長別チップの数量，およびそれの全市場に対する占有度を表４－４，及び表４－５に示す。

表４－４　事務・商業分野で購入されたビット長別チップの数量

（単位10^3個）

ビット長	1978	1979	1980（見込）
4ビット	253（81%）	335（86%）	197（67%）
8	55（18　）	50（13　）	94（32　）
10～12	—	—	—
16	—	0.1（0　）	0.1（0　）
ビットスライス	2（　1　）	3　（1　）	3　（1　）
計	311	388	294

表４－５　事務・商業分野で購入されたビット長別チップの全市場に対する比率

（単位％）

ビット長	1978	1979	1980
4ビット	8	7	3
8	30	17	12
10～12	—	—	—
16	—	1	1
ビットスライス	20	18	14

これによると

- 事務・商業分野で使われているマイクロコンピュータは，4ビットが多数を占めている。
- 年毎の傾向としては，4ビット系が減少，8ビット系が増えていく傾向にある。
- 16ビットマイクロコンピュータは，まだ，絶対量としては少いが徐々に使われ始めている。

16ビットマイクロコンピュータの応用の開発状況を見てみる。（図４－６）

これによると，事務・商業分野では半数が開発中，あるいは開発予定の状況にある。

16ビットマイクロコンピュータの採用理由をみてみる。（図４－７）

これによると，16ビットマイクロコンピュータを採用する理由としては，高速，メモリ容量，命令が強力等が多数を占めている。

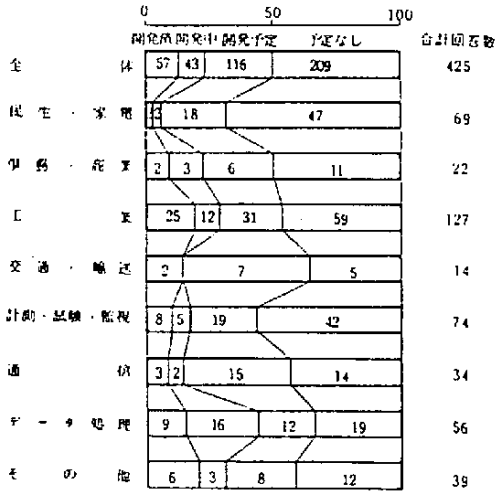
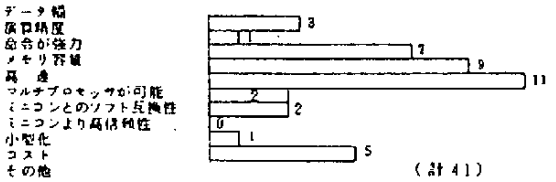


図４－６　開発状況(1)



図－７　事務・商業分野における１６ビットマイクロコンピュータの採用理由(1)

(b) 利用上の特徴

この分野での応用機器は，マイクロコンピュータとして使われる機能の複雑さから見ると，工業分野，計測試験監視分野，データ処理分野と民生家電分野との中間に位置すると思われる。従って従来この分野においては，主として4ビットおよび8ビットマイクロコンピュータを応用したシステムが開発されて来た。16ビットマイクロコンピュータはまだ工業，計測，データ処理分野ほどには使われていない。

しかし，この分野における各機器も分散処理指向に伴い，端末としての機能向上，あるいはホストコンピュータとのネットワーク形成等に16ビットマイクロコンピュータの使用が促進され，インテリジェンシィが一層増強されることになると思われる。

この分野における代表的な機器であるECRについて，その傾向を以下に示す。

ECRにおける使用マイクロコンピュータは4ビットが主体であったが，現在は8ビットが増加傾向にあり，50～60%に達している。

これまでは普及機を形成する単純型ECRが大部分を占めていたが，今後はこの分野にも高機能化されたECRが期待されている。

このECRは，スタンドアロンでも使用出来るが，複数台のECRの集中管理を行ったり，あるいはホストコンピュータとの交信を行うことが出来るいわゆるシステムECR（POSターミナル）である。

このシステムECRは，それ自体高度なデータプロセシング機能をもつスモールコンピュータである。従ってこれらの機器にも更に機能の高いマイクロコンピュータが逐次採用され，そのインテリジェンシーが高められて行くものと思われる。

〔参考文献〕

(1) 「マイクロコンピュータに関する調査報告書」
日本電子工業振興協会　昭和54年3月　昭和55年3月

(3) 工　業

この分野は，生産機械装置，機械制御，プロセス制御，データロガ，生産管理等のアプリケーションを含み，分類分けからすると産業用システムの制御用に含まれるものである。その具体的な応用機器，応用システムを表４－６に示す。

表４－６　工業分野におけるマイクロコンピュータの具体的応用製品名，応用システム名[1]

用　　途	具体的応用機器・応用システム
1. 生産機械装置	数値制御，プログラム・コントローラ，溶接機，工作機の自動化，部品自動生産機　等
2. 機械制御	機械制御，シーケンス制御，自動組立機，産業用ロボット，自動ボンダ，歯車組立機，工業用ミシン，巻鉄心装置　等
3. プロセス制御	プロセス制御，熱処理炉制御，重合プロセス制御，プロセスモニタ，原子炉制御，ドライエッチング装置，連続焼鈍炉　等
4. データ・ロガ	データ・ロガ，テストハンドラー，異物検出装置，試験装置，データ収集機，ガス分析装置，オペレータコントローラ　等
5. 生産管理	出荷コントロール，生産管理，作業管理システム，公害監視システム，ビル管理システム，自動倉庫，車両自動運転　等
6. その他	電源システム，魚労機器，端末制御，回線制御　等

(a) 利用動向

工業分野におけるマイクロコンピュータの利用に関して，購入されたビット長別チップの数量およびそれの全市場に対する占有度を表４－７，および表４－８に示す。

表４－７　工業分野で購入されたビット別チップの数量

（単位10^3個）

ビット長	1978	1979	1980（見込）
4ビット	73.4（84%）	149.3（82%）	226.4（76%）
8	11.0（13　）	27.5（15　）	62.5（21　）
10～12	0.2（　0　）	0.3（　0　）	0.6（　0　）
16	0.2（　0　）	3.0（　2　）	3.8（　1　）
ビットスライス	2.2（　3　）	2.8（　2　）	3.6（　1　）
計	87.0	183.0	296.7

表４－８　工業分野で購入されたビット長別チップの全市場に対する比率

（単位％）

ビット長	1978	1979	1980（見込）
4ビット	2	3	3
8	6	9	8
10～12	17	13	14
16	11	37	27
ビットスライス	21	19	18

この表でみると

- 工業分野で使用されているマイクロコンピュータは依然として4ビットが多数を占めている。
- しかし８ビットおよび１６ビットの数量の伸びも著しいものがある。１６ビット系では１９８０年の見込では全市場に占める比率が２７％と大きな比率を占めている。
- このように，この分野では民生家電分野比べて，８ビット以上のマイクロコンピュータの比率が高いことが特徴の一つである。
- 用途上では，民生家電分野でマイクロコンピュータを少品種の製品を大量使用するのに対して，工業分野では多品種の製品に少量使われるのも特徴の一つである。

１６ビットマイクロコンピュータの応用の開発状況は図４－６によると約半数が開発済，開発中，あるいは開発予定の状況にある。

１６ビットマイクロコンピュータの採用理由をみてみる。（図４－８）

これによると，１６ビットマイクロコンピュータを採用する理由としては，演算精度，高速，命令が強力等が多数を占めている。

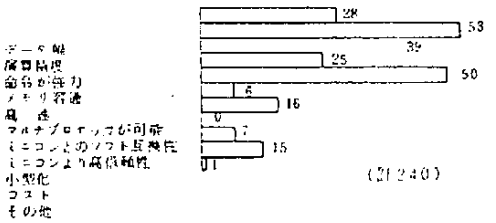


図４－８　工業分野における１６ビットマイクロコンピュータの採用理由(1)

工業分野に於ける代表的なアプリケーションであるプロセス制御についてその動向を次にのべる。

(b)　プロセス制御

プロセス制御システムは，産業界の規模の拡大や，多様化が進んだこと，および社会的な要請，即ち省エネルギー，省力化，低公害などによるシステムが高度化すると共に，複雑さも増すことになった。マイクロコンピュータはこれらの時代の要請に効率よく対処出来

ることから，製品への適用は飛躍的に伸びている。主として，鉄鋼や石油化学，セメント，上水道といったプラントに使われている。これらプラントのオートメーション化を図るための温度か流量を測るものが工業計器であり，これらの計器で測定されたデータをもとにプラントが正常な運転を行っているかどうかを制御監視するのが計装システムである。

マイクロコンピュータがプロセス制御の分野で効果的に使われはじめたのは，１９７５年以降あいついで発表された分散形総合計装システムである。これはプラントの制御が複雑になり，アナログ式では限界が出て来たことと，マイクロコンピュータ技術の進歩とその低価格によるものである。これらには１２－１６ビットマイクロコンピュータが搭載されている。

分散形計装システムの分散度は製品により異っているが，おおむね８～３２ループを１台のマイクロコンピュータで制御し，それら複数台のステーションとマン・マシン・インターフェイス相互間をデータウェイで結合して，総合計装システムを形成しているのが特徴的である。最近は従来アナログ型制御方式で行われてきた調節計にもマイクロコンピュータを活用したシングルループＤＤＣ（デジタル式制御機器）がデジタル計装システムの中核をなすものとして注目を集めている。

デジタル計装にマイクロコンピュータを搭載することのメリットは次のものが考えられる。

① ミニコンによる集中型ＤＤＣシステムに比べ信頼性の高いＤＤＣが実現でき，かつコストダウンができる。

② 従来のアナログ計装に比べ，自己診断，相互診断機能が充実しているので，安全運転が期待出来る。

③ 数種類のハードウェアで小規模から大規模システムまで構成でき，かつ制御用計算機やシーケンスとの結合が容易である。

④ アナログ調節計と同一操作性を保ちつつ，ＤＤＣの良さ，すなわち制御性の改善を容易に行うことができる。

⑤ デジタル化により精度の向上が図れる。

⑥ 従来のアナログ計装で必要な補助計器を集約化できるので，ハードウェアが少くなり，盤内配線も簡素化される。

⑦ 数種類の機種ですべての計装を実現できるので，保守が容易で，かつ予備品も少くてすむ（システムコストダウン）

これらの多くのメリットをもつデジタル型計装システムも，今後普及が広がるにつれ，更に要求される機能は増大し，複雑性，重要性が高まるとともに，システムとしてとらえた信頼性が重要になってくる。こうした高度な要求に答えるためにも，システム上のバランスを考えた高い機能をもつマイクロコンピュータ（１６ビットマイクロコンピュータ）の一層の活用が期待される。

〔参考文献〕

(1) 「マイクロコンピュータに関する調査報告書」
日本電子工業振興協会　　昭和54年3月　昭和55年3月

(2) 「マイクロコンピュータの活用法」　技術評論社　P177

(4) 交通・輸送

この分野でのマイコン利用方法はそれぞれのシステムの一部に組込まれ，コントローラ的に使用されている。現時点でのマイコンの導入目的は概して，機能実現を従来のワイヤードロジックから経済化と高信頼化を目指してマイコンに置換する傾向が見られ，8ビットマイコンが一つのシステムの中でいくつも使われるような場合も見られる。このことは複数から成るサブシステムの一部の故障を手動で補完して安全確保をはかる必要のあるこの分野では却って都合のよいことにもなっているのである。

この分野でのマイコン導入では車両関係の制御，列車運行制御，エレベータの制御やその自動運転，倉庫の貨物搬出入の自動化と管理などがあげられる。ここでは代表例として車両制御と，自動倉庫における16ビットマイコンの導入について考察を行う。

(a) 車両制御

車両へのマイコンの導入は5～6年前から行われ始めている。特に強振動と雑音レベルの高い環境の中で高い信頼性を確保するという厳しい条件を満す必要があるが，列車の自動運転制御装置，車両状態監視装置，自動車内放送，地上または編成車間とのデータ伝送装置などの機能実現のためにマイコンが使用されている。

表4－9　車両制御へのマイコンの適用例

車両制御機器	内　　容
自動運転装置	一人乗務及び無人運転
モニタ装置	事故発生時の措置及び記録
チョッパゲート制御装置	ゲート回路のディジタル化制御
データ伝送装置	地上の信号又は編成車両間のデータ伝送
自動案内装置	メモリに録音した放送文の編集と放送
その他	行先表示灯回路の集中制御

現在はこれらの装置それぞれに8ビットマイコンが使用されており，それぞれの装置は車上で結びつけられてネットワークを形成している。ここではその中心となっている自動運転装置におけるマイコンの実現機能を表4－10に示す。

表4－10　自動運転装置の機能[1]

機　能	内　容
速度検出	速度パルスと車輪径から列車速度を求める。
駅間走行制御	ＡＴＣ信号若しくは中央からの速度指令で走行すべき目標速度を設定し，この目標速度に追従すべく制御する。
駅停止制御	車上で停止パターンを発生させ，この停止パターンに列車を追従させ，駅の所定位置に停止させる。
力行・ブレーキ指令	駅間走行制御系と駅停止制御で決定される力行・ブレーキ指令の二者の低位優先を行ない，出力の力行ブレーキ指令を決定する。
出発制御	出発条件が整ったこと（機器正常，ドア閉，出発指令など）を確認して列車を出発させる。
停止制御	停止検知及び停止時の転動防止ブレーキ制御を行なう。
ドア制御	駅停車時のドアの開閉を制御する。列車が駅のプラットホームの定位置に停止したことを検知してホーム側ドアを開き，運行管理システムからの出発指令を受信したらドアを閉じる。
車内放送	車内自動案内放送の制御を行なう。
情報伝送	運行管理システムと本装置間の情報（運行情報の授受など）伝送制御を行なう。
モニタ	車載機器の動作状態を監視，記録する。
試験	自動試験装置の指令により自己診断を行ない，結果を転送する。
異常処理	機器故障などの異常の発生を検知し，故障機器の切離し，バックアップ処理，非常停止などの処理を行なう

これら車両制御に必要とされる機能は一組の１６ビットマイコンで達成することによりさらに小型化，経済化が実現可能と考えられ今後の導入が期待されるが，列車は人命を扱っていること，また故障時には容易に移動ができないことなどから，マイコンシステムの高信頼性確保が導入上の大きな課題となろう。

(b)　自動倉庫

自動倉庫におけるマイコンの応用は入庫または出庫すべき棚の位置の選択と貨物の収容，取り出し作業を行うクレーンの制御およびベルトコンベアの制御，並びに入出庫する貨物の在庫管理の機能の実現が考えられる（図4－9）。クレーンやベルトコンベアの制御のみを実現するには機能，性能面とも８ビットマイコンで十分である。在庫管理機能では倉庫規模に応じて収容する貨物の種類も入出庫量も多くなり貨物リストや数量の管理のみならず在庫期間や空スペースなどの情報を管理するために補助記憶装置や検索用端末の接続が必要となり１６ビットマイコンが有利になってくる。

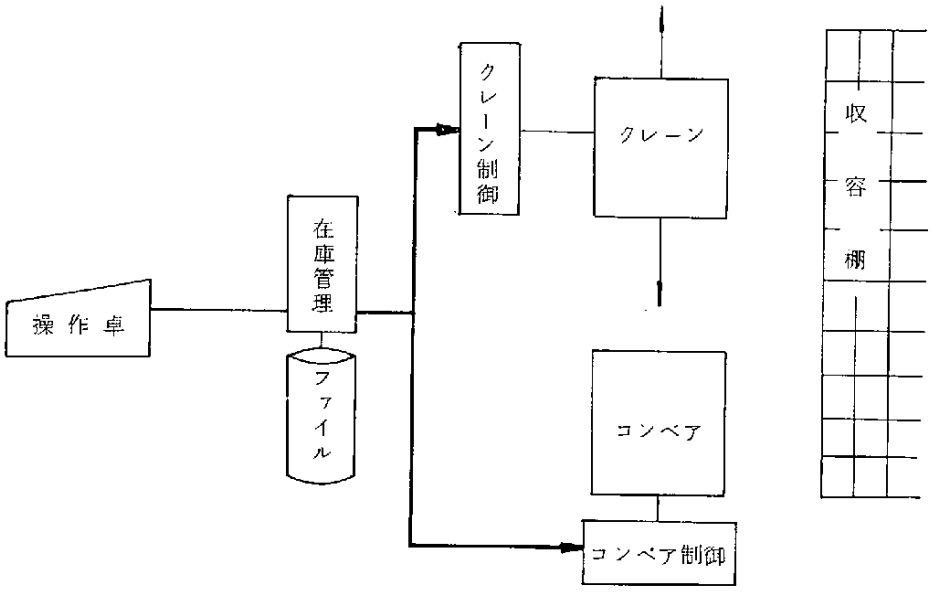


図４－９　自動倉庫システム概念図

〔参考文献〕

(1) 日立評論　Vol61　No. 5（１９７９－５）
マイクロコンピュータの車両制御への適用

(5) 計測・試験

測定器，分析器，試験・検査機，監視，医用電子等が含まれるこの分野は，その技術の進歩，発展は科学技術や産業システムを支えるとともに，省資源，省エネルギー，省力化，公害防止等社会的効果に果たす役割は極めて高い存在といえる。

この分野の具体的応用機器，応用システムを表４－１１に示す。

(a) 利用動向

この分野では機器に対する機能のアップ，信頼性の向上等への要求が極めて強く，マイクロコンピュータがこれらの要求に効率よく対処出来ることから，積極的に採用されている。さらにマイクロコンピュータ使用により多品種製品の宿命である特別仕様に対して柔軟に対処出来るという理由から，マイクロコンピュータを採用する例も多くあり，この分野でのマイクロコンピュータの増加は著しいものがある。

計測・試験分野におけるマイクロコンピュータの利用に関して，購入されたビット長別チ

表４－１１　計測・試験分野におけるマイクロコンピュータの具体的応用製品名，応用システム名[1]

用　　途	具体的応用機器・応用システム
1. 測　定　器	形状測定装置，自動温度圧力計測荷重計，選択レベル測定器，光度計，測光計，硬度計，電子顕微鏡，光電子測定装置　等
2. 分　析　器	X線マイクロアナライザ，イオウ分析計，分析器，ガスクロマトグラフィ分析処理，自動計測データ分析機，血液ＰＨ，PCO_2計　等
3. 試験・検査機	試験検査機，車両自動試験，トランジスタ試験装置，ＩＣチップテスタ，磁気ヘッド特性計測システム，カメラ試験機，故障記録装置，異常診断装置，自動車エンジン試験，メモリテスタ，センサ試験機　等
4. 監　　視	監視装置，総量規制モニタリングシステム，大気モニタ，ガス警報監視システム，電力系統監視システム，電話テレメータ，回転板監視装置，データ・ロガ，無線回線監視，原子炉監視，環境制御　等
5. 医用電子	医用電子，胎児監視システム，X線診断装置，臨床検査装置，患者監視装置，心電図解読，心音計測システム，睡眠モニタ，レーザメス，機能的生体作用装置，体表配電位分布処理システム等
6. そ　の　他	多色ＬＥＤ表示，タッチパネル，気象データ処理装置，ディジタライザ，ワイヤリング装置，データ処理関係，連続炉焼鈍炉　等

ップの数量およびそれの全市場に対する占有度を表４－１２および表４１３に示す。

表４－１２　計測・試験分野で購入されたビット別チップの数量

（単位10^3個）

ビット長	1978	1979	1980
4ビット	180（ 0%）	70（ 0%）	5080（ 4%）
8	41821（93 ）	77245（94 ）	98429（87 ）
10～12	150（ 0 ）	300（ 0 ）	420（ 0 ）
16	1045（ 2 ）	1948（ 2 ）	4965（ 4 ）
ビットスライス	1812（ 4 ）	3024（ 4 ）	4628（ 4 ）
計	45008	82587	113522

表４－１３　計測・試験分野で購入されたビット長別チップの全市場に対する比率

（単位％）

ビット長	1978	1979	1980（見込）
4ビット	0%	0%	0%
8	23	25	12
10～12	15	13	10
16	48	24	35
ビットスライス	17	20	23

これによると

- 使われているマイクロコンピュータは，４ビットからビットスライスまで多岐にわたっている。これは機器本体の価格が数万円から数億円と幅があり，マイクロコンピュータに要求される機能も広範囲にわたっているためと思われる。
- この分野は８ビット以上のマイクロコンピュータの比率が高い。
- １６ビットマイクロコンピュータの使用量は年々着実に伸び，１９８０年の見込では全市場の３５％を占めている。
- 図４－６によればこの分野は４３％が１６ビットマイクロコンピュータを開発済；開発中，あるいは開発予定である。
- この分野における１６ビットマイクロコンピュータの採用理由は図４－１０のとおり。これによると高速，演算精度，命令が強力，データ幅等が理由として多くを占めている。

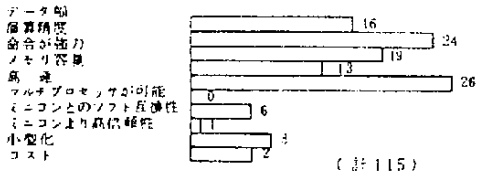


図４－１０　計測・試験分野における１６ビットマイクロコンピュータの採用理由 (1)

(b)　応用上の特徴

多品種少量生産の典型的な存在であるこの分野では，そこに使われるマイクロコンピュータも少量にして種類が多い事を特徴としている。このように多品種少量生産の分野では，多少ハードウェア（ＬＳＩ）の価格が高くても１６ビットマイクロコンピュータを採用してソフトウェアコストの低減を図る方が有利であるというケースも多くある。あるケースでは８ビットマイクロコンピュータではプログラミングに数ケ月を必要とするものが，同じアプリケーションに１６ビットマイクロコンピュータを用いるとプログラミングは数週間で完了したという。このようにソフトウェア開発費削減とか，日程短縮の効果は，８ビットから１６ビットにしたＬＳＩ価格の上昇に余りあるものが出て来よう。

この分野におけるマイクロコンピュータを搭載することのメリットは次のものが上げられる。

①　機能および性能の向上

測定精度の向上のほか，マンマシン・インターフェイスの機能が向上する。

②　信頼性の向上

マイクロコンピュータそのものの信頼性が高い上に，さらに自己診断，相互診断機能を持たせることによる故障率の低下が期待出来る。

③　特注要素の吸収が容易

④　小形化

⑤　開発期間の短縮

⑥　生産性の向上によるコストダウン

上記のとおり各種のメリットがあげられるが，これらをさらに厳しく追及すると８ビットより１６ビットマイクロコンピュータの方がはるかに有利である。従って少量多品種のこの分野においては，広範囲のアプリケーションに１６ビットマイクロコンピュータがより多く搭載されて行くものと思われる。

〔参考文献〕

(1) 「マイクロコンピュータに関する調査報告書」
日本電子工業振興協会　昭和５４年３月　昭和５５年３月

(2) 渡部「マイクロプロセッサの計測・制御への応用」
電気四学会連合大会　昭和５４年

(6) 通　信

通信分野でのマイコン利用方法はコントローラ的応用が強いが機能分散に伴って，コンピュータ的応用も強くなってきている。

マイコンの利用が最も進んでいる部分は画像通信端末（ファクシミリ），データ通信用端末，ＰＡＢＸ，自動車電話など，端末機器が中心となっているが，最近ではコンピュータの通信制御装置やＤＤＸパケット交換網の通信制御部にも使用されている。これらの装置ではほとんどが８ビット以下のマイコンが使用されているが，データ通信用端末ではプリンタやキーボードなどの機器制御に加えて，入出データの編集，軽微な処理などのインテリジェンス機能を実現するために，１６ビットマイコンが本格的に導入されるようになってきた。

ここでは通信分野における代表例としてデータ端末とファクシミリ端末における１６ビットマイコンの導入について考察を行うこととする。

(a) データ端末

データ端末へのマイコン導入はマイコンの誕生と同時期からであり，多くの論理素子を必要とした通信制御や，キーボード，プリンタなどの入出力機器制御にマイコンを導入して小形，高信頼化，経済化に寄与することとなった。これらはすべて８ビット以下のマイコンで機能性能を満たしていたが，データ通信の機能分散化が進み，データ端末でもデータの前置処理，局所的なデータ処理などのインテリジェンス機能を具備するようになってきており，このようなデータ端末には機能，性能を満足する１６ビットマイコンが使われている。

インテリジェント端末に１６ビットマイコンを採用する理由は，プログラムの容量が大きくなるために，数百キロバイトのメモリが必要となるオンライン通信に併行していくつもの業務処理プログラムを同時走行させるために，相応の処理能力が要求される。利用者がプログラムの作成を行うために汎用の高位言語処理プログラムを用意しているものもある。このような機能を持ったインテリジェント端末の構成およびソフトの体系を図4－11と図4－12に示す

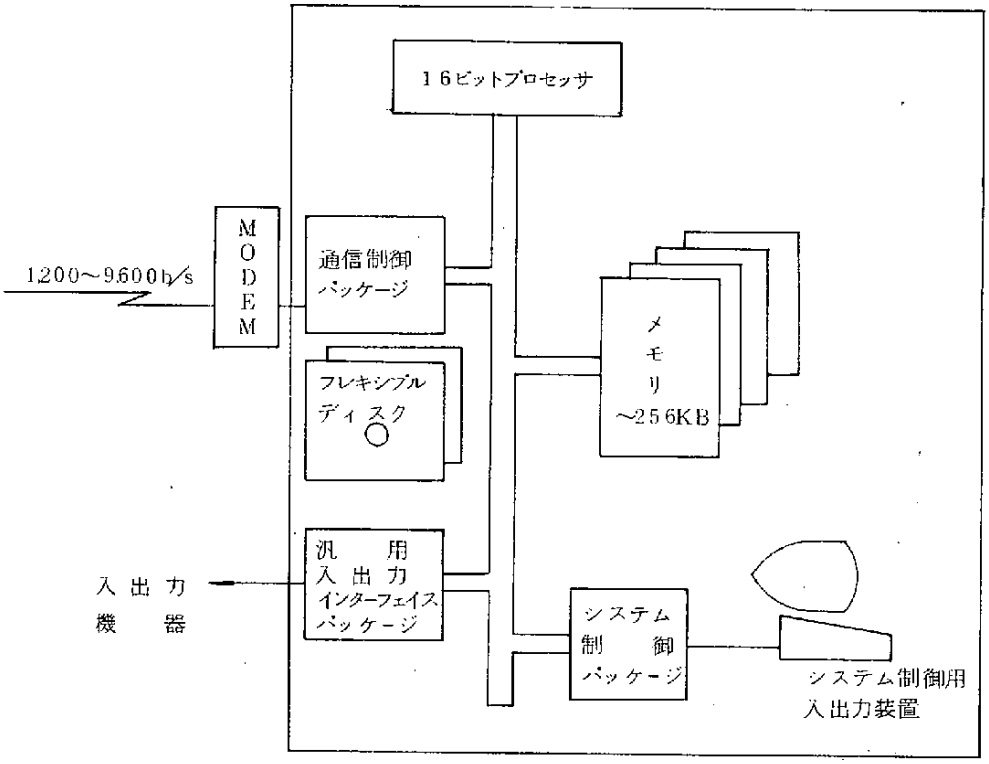


図4－11　インテリジェント端末の構成

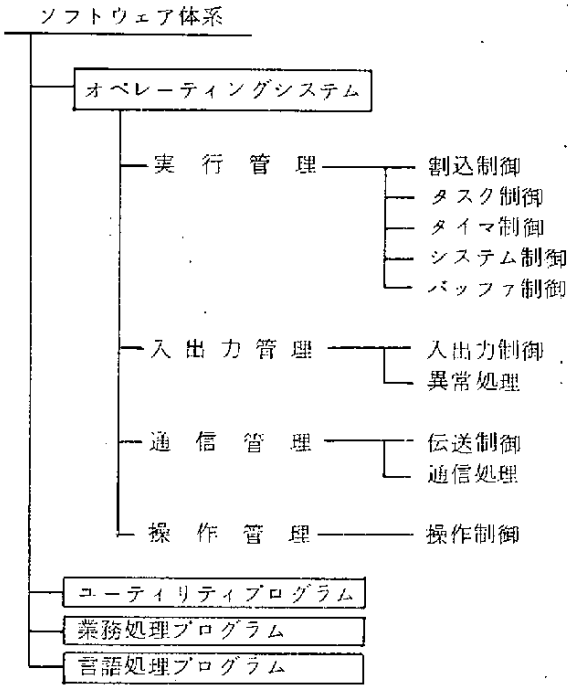


図4－12　インテリジェント端末のソフト体系

(b) ファクシミリ端末

ファクシミリ端末では機器制御と通信制御，信号の圧縮など，装置のコントローラ的な役割でマイコンが使用されている。

現在は上記のような基本的な機能の実現の手段としてマイコンが使用されているので，ほとんど8ビット以下のものとなっている。しかしながら帳票形式のオーバレイ，縮小，拡大，編集などインテリジェンス機能などの具備により，16ビットマイコンが使用されるようになってくるものと考えられる。

(7) データ処理

16ビットマイコンの応用機器の中で，今後最も期待されているものの中にデータ処理分野がある。中でもオフイス・コンピュータ，パーソナル・コンピュータ，汎用マイクロコンピュータ（開発ツール）およびアナログデータを処理するデータ・アクィジッション・システム等は現在既に16ビットマイコンを使用しているものもあるが，遠からず16ビットが主流になるものと考えられる。

オフイス・コンピュータは年々出荷台数があがり，現在オフイス・オートメーションと呼ばれる時勢にも相まって，第三次普及期を迎えている。昭和42年からの出荷台数の伸びの状況を図4－13に示す。この様に成長した理由は次の4つが上げられる。

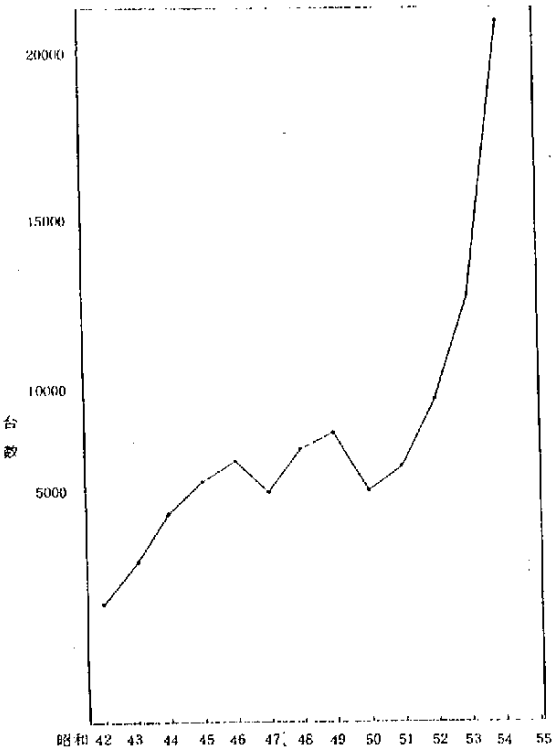


図4－13 オフイスコンピュータ出荷台数
（日本電子工業振興協会調査，ただし，国内メーカ17社）

(a) 機能強化と低価格化

メモリの集積度が４Ｋから１６Ｋに向上し，ビット当たりのコストが低下したことに代表されるように，ＬＳＩ化によって性能対価格比が向上したこと。

(b) 製品の多様化

漢字システムの登場，簡易コード入力装置の普及，フロッピーディスクの大容量化，大容量固定ディスクの採用，大型ＣＲＴの採用，マルチ・ワーク・システムなど製品が多様化してきたこと。

(c) 市場の成熟

新規需要層が十分な製品知識を持つようになったこと，買い換え需要が増えていること。

(d) 分散処理指向の拡大

オフィス・コンピュータの大型クラスが順調に伸びていることなどから，分散処理需要の比重が高まっていると推定されること。

現在のオフコンの価格は，約２００～２０００万円にばらついているが，その中で最も出荷数の多いのが，約５００～７００万円の中クラスのものである。また使用者即ち，出荷先の産業分野別に見ると，卸売業・商事会社が最も多く，次いでサービス業，小売業が続き，この三分野で５０％以上占める。（図４－１４参照）

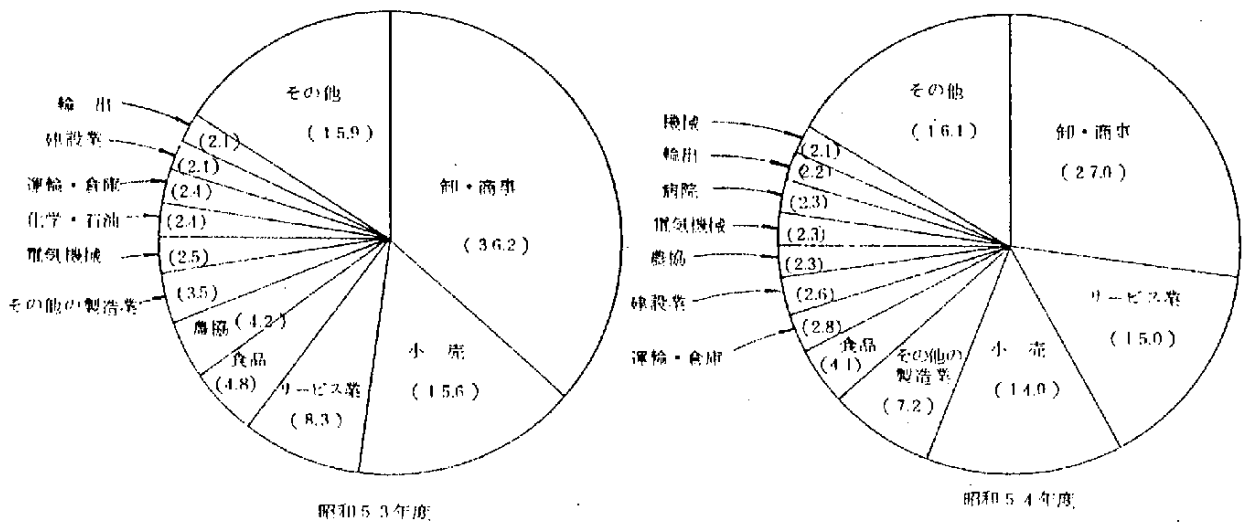


図４－１４　オフコンの産業分野別出荷台数比率

（日本電子工業振興協会調査による）

最近日本電気が発表したオフコンの上位機種であるＮＥＡＣシステム１５０モデル5･5の仕様を見ると表４－１４の如くである。

◦基本処理装置

中央処理装置

演算素子	バイポーラＬＳＩ
メモリサイクルタイム	600ns／2Byte
メモリ容量	512～1024Byte
ＲＯＭ	18K Byte

フロッピィディスク装置

接続台数	２台（１台は内蔵）
容　量	1M Byte／Unit

磁気ディスク装置

接続台数	１～２台目
アクセスタイム	平均３７.３ｍｓ
容　量	64M Byte／台

◦増設ラック

高速磁気テープ装置

接続台数	２台（ラック２台）
トラック数	９トラック
記録密度	800／1600 BPI
長　さ	2400フィート

磁気ディスク装置

接続台数	３～４台目
アクセスタイム	平均３７.３ｍｓ
容　量	64M Byte／台

表４－１４　NEC製オフコン　NEACシステム１５０　モデル５５の性能

このオフコンはバイポーラＬＳＩを使用し，高速かつメモリ容量も1.Mバイト迄拡張可能なもので，しかもシステム機能上も次のような点を特徴とするオフコンの中でも高級機に属するもの

である。

① 大規模マルチワークシステム

② 分散処理

このようにこれらからのオフコンは，マルチワークのための多重処理機能，分散処理を行うデータ通信機能等，低級機または中級機においても少しづつ取り入れていくと考えられる。また最近のオフコンはコンピュータで取り入れられていたデータを整理統合し，ファイル管理を行うデータ・ベース管理システムを導入して来ており，そのための操作言語を基本ＯＳに組込んでやる必要がある。その他，漢字の導入も１つの方法である。以上のことを考えた時，８ビットから16ビット（第二世代）に移行するものと考えられる。

次にパーソナル・コンピュータを見てみよう。パーソナル・コンピータは，当初ホビー産業よりスタートし，アマチュア的個人のコンピュータとして誕生したが，最近は関連製品の開発により，用途にあったシステム構成を取る事が出来る様になった。参考としてユーザの職業別比率を図４－１５に示す。業務用の代表的システム構成を図４－１６に示す。このように業務用に使用されるようになると，オフィス・コンピュータとの境界が無くなり，同一線上に位置して来る。特に業務用のソフトがパッケージにて提供されて来ると，ますますオフコンの低級機の領域を侵蝕して来るものと思われる。また図４－１７のように最も低価格であるため，オフコンと違うところはオフコンより経済的な単能機または専用機として，業務，計測，事務，医療，教育関係およびシステムへの組込等，コンピュータ・システム部品の様な使用法など，オフコンには考えられない，用途の多様化である。

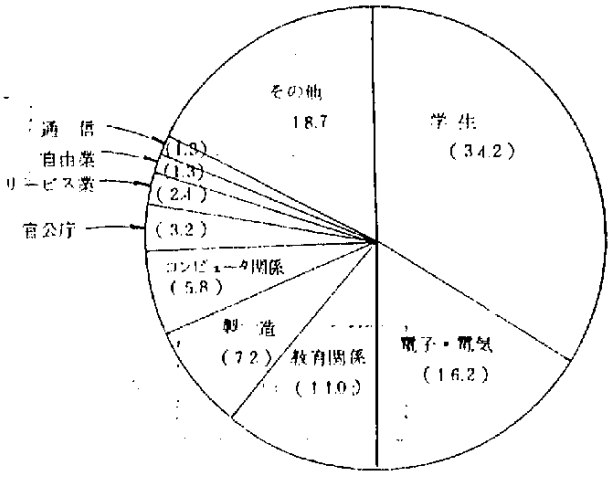


図４－１５　マイコンユーザ職業別比率
（NEC社内資料より）

用途の多様化を考える時，これからのパーソナル・コンピータは，次の様な方向に行かざるを得ない。

(a) 使用者が千差万別なため，ＯＳを強化しなければならない。〔言語，通信，外部メモリ（ファイル）等〕

(b) 高解像度のカラー・グラフィックが必要となろう。

(c) 音声合成，音声認識等インターフェイスの強化等。

以上を考えた時，メモリ空間が６４Ｋ Ｂｙｔｅに制限される８ビットでは，性能を達成するこ

とが出来ず，スピードも向上出来る１６ビットに移行することは必至である。

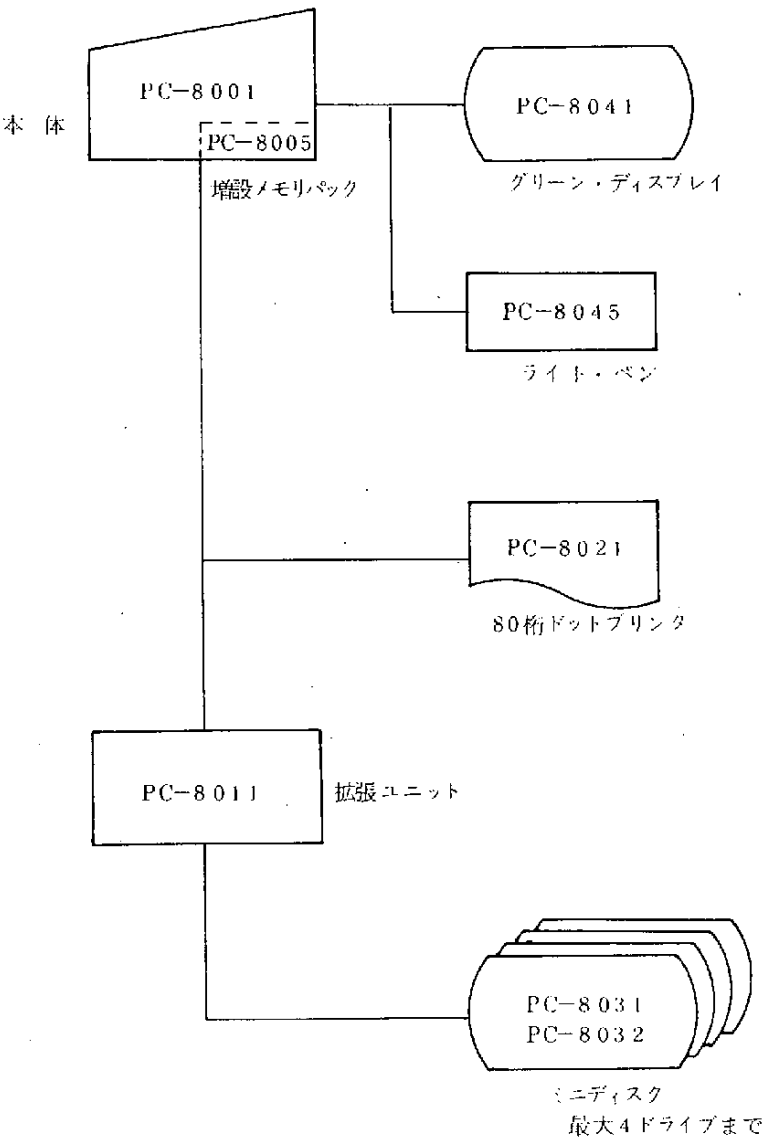
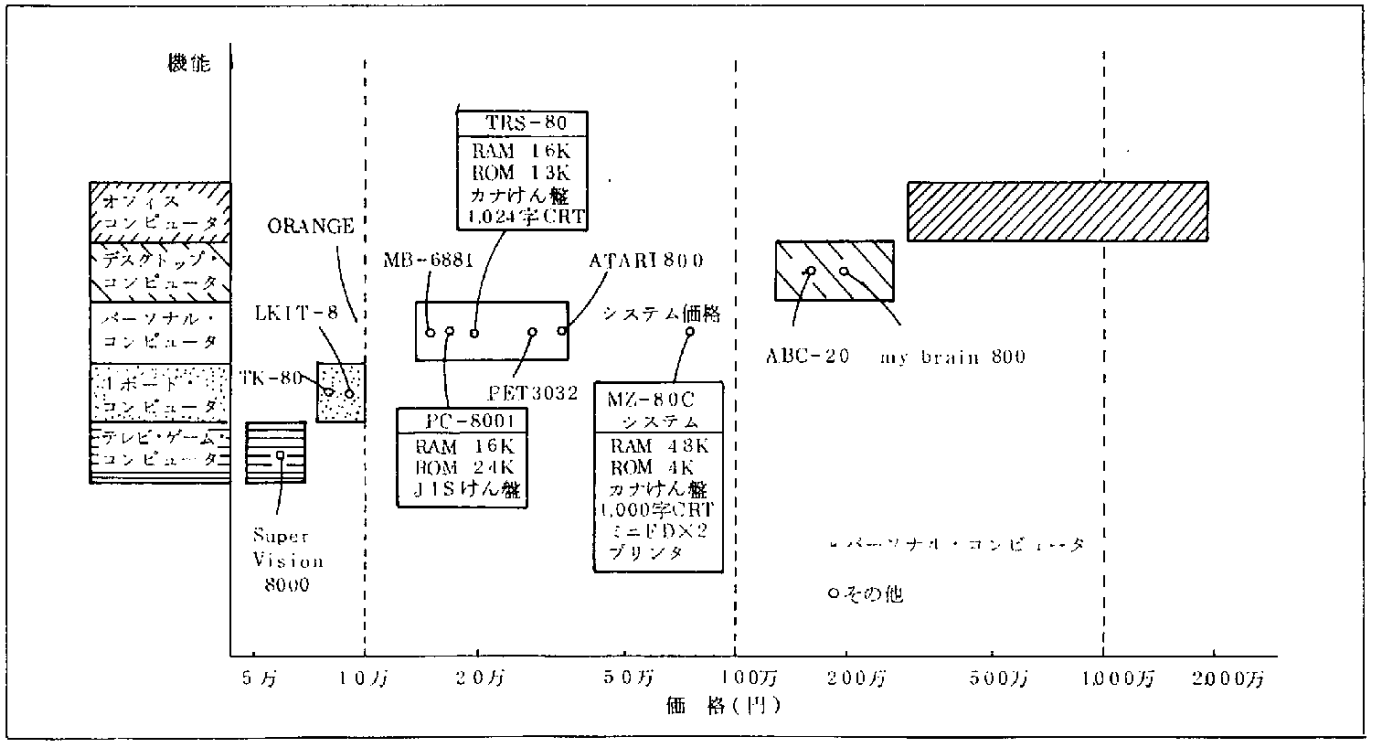


図４−１６　NEC製パーソナル・コンピュータ　PC−8000シリーズによる
業務処理用基本システム例



パーソナル・コンピュータ中心に低価格コンピュータを分類したもの。パーソナル・コンピュータ本体の価格は15～23万円付近に分布する。デスクトップ・コンピュータは150～200万円，オフィス・コンピュータは500～2,000万円程度のものが多い。

図4－17　低価格コンピュータの分類[4]
（日経エレクトロニクス19803-17より）

〔参考文献〕

(1) 日経エレクトロニクス；電子産業の動き（２２４号　P．１９１～１９２）
（１９７９．１０．２９）

(2) 日本電気㈱；ＰＣ－８００１ユーザーズ・マニュアル

(3) 日経エレクトロニクス別刷「コンピュータ」１９７８－８０（Ｐ４８～５０）

(4) 「１２３４　ＭＩＣＲＯＳＯＦＴ　ＱＵＡＲＴＥＲＬＹ」資料
Ｍｉｃｒｏｓｏｆｔ社　１９８１

(5) 日経エレクトロニクス；電子産業の動き（２３４号，Ｐ．１８８～１９３）
（１９８０．３．１７）

4.3　市場動向

図４－１８は日本電子工業振興協会のアンケート調査の結果をまとめたビット別に見た応用分野別市場累計の百分比（１９７８年～１９８０年）を示すものである。４ビットマイクロコンピュータは圧倒的に民生・家電分野で消費されている。８ビットマイクロコンピュータは販売管理・在庫管理・ワードプロセッサ・ＰＯＳ・ＥＣＲなどの事務・商業分野が大半の市場である。１６ビットマイクロコンピュータは生産機械装置・機械制御・プロセス制御・データロガ・生産管理などの工業分野での使用実績が大きく，次いで，汎用コンピュータ・周辺端末機器・科学技術計算などのデータ処理分野での実績が大きい。極言すれば４ビットマイクロコンピュータは民生用，８ビットマイクロコンピュータはオフィス用，１６ビットマイクロコンピュータは工業用として利用されている。

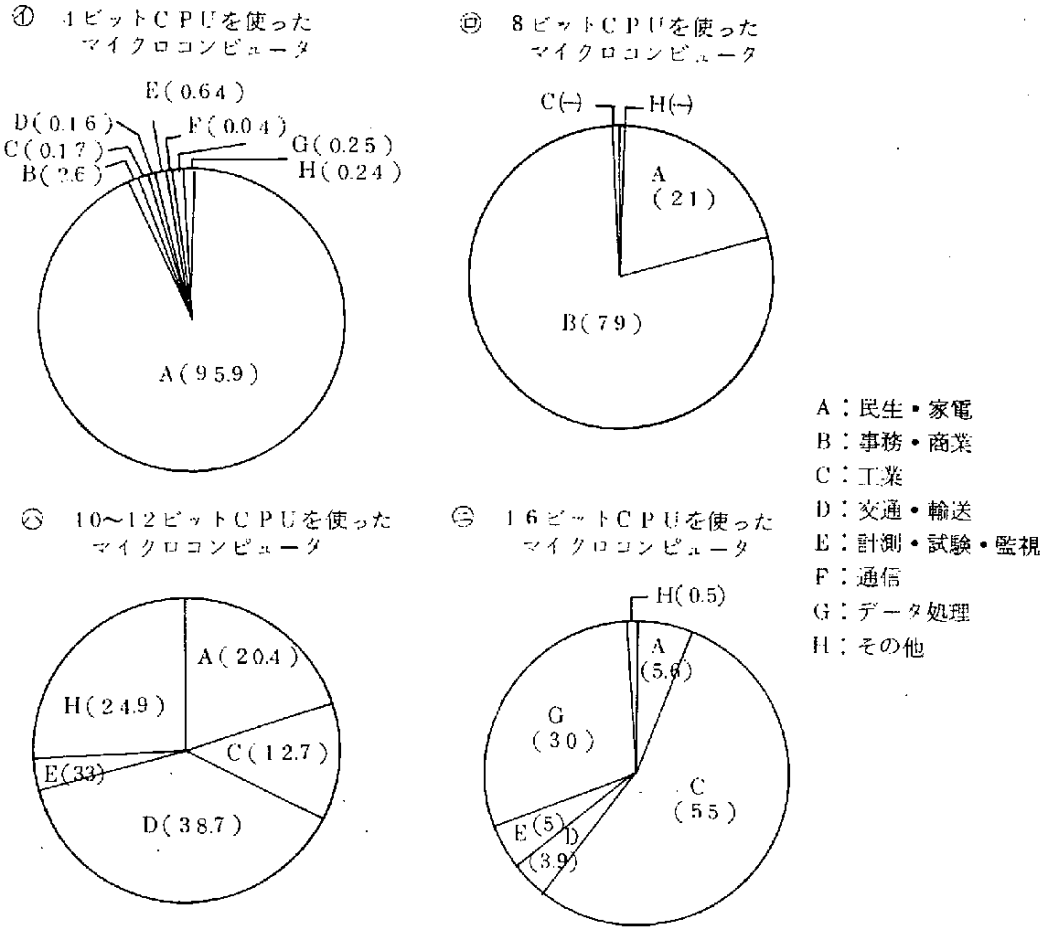


図４－１８　ビット別に見た応用分野
（１９７８～１９８０年累計台数百分比）

データクエスト社の推定データから今後１０年間の市場推移を図示すれば図４－１９，２０，２１，の様になる。これらの図は世界規模でのビット別，形式別の市場動向を金額ベース，数量ベースで示すものである。４ビットや８ビットのワンチップマイクロコンピュータの市場は増大の一途であり，特に４ビットマイクロコンピュータの伸び率が極めて大きいことを物語っている。一方，マルチチップマイクロコンピュータでは，１６ビットマイクロコンピュータの伸びによって８ビットマイクロコンピュータの市場が'８５年をピークに減少に転じている。同じ傾向が３２ビットマイクロコンピュータと１６ビットマイクロコンピュータとの間にも見られ'８８年をピークに１６ビットマイクロコンピュータの市場は減少している。総合的にも，マルチチップマイクロコンピュータの市場は数量的にも金額的にも'８０年代にピークを迎え，減少に移る。これはチップ価格の低下もあるがＬＳＩの規模拡大に伴ってワンチップマイクロコンピュータに代替されるためであろう。

図４－２２はマルチチップマイクロコンピュータが消費するメモリの市場推移を語長別に示すものでデータは前述のデータクェスト社の推定である。この図で顕著なことは，１６ビットマイクロコンピュータに関係して消費されるメモリが極めて大きいことである。換言すれば１６ビットマイクロコンピータはメモリ消費型マイクロコンピュータである。更に数量ベースの市場推移が減少傾向にあって，しかもメモリの消費推移が増大していることは，システム当りのメモリが著しく拡大する傾向にあることを示している。

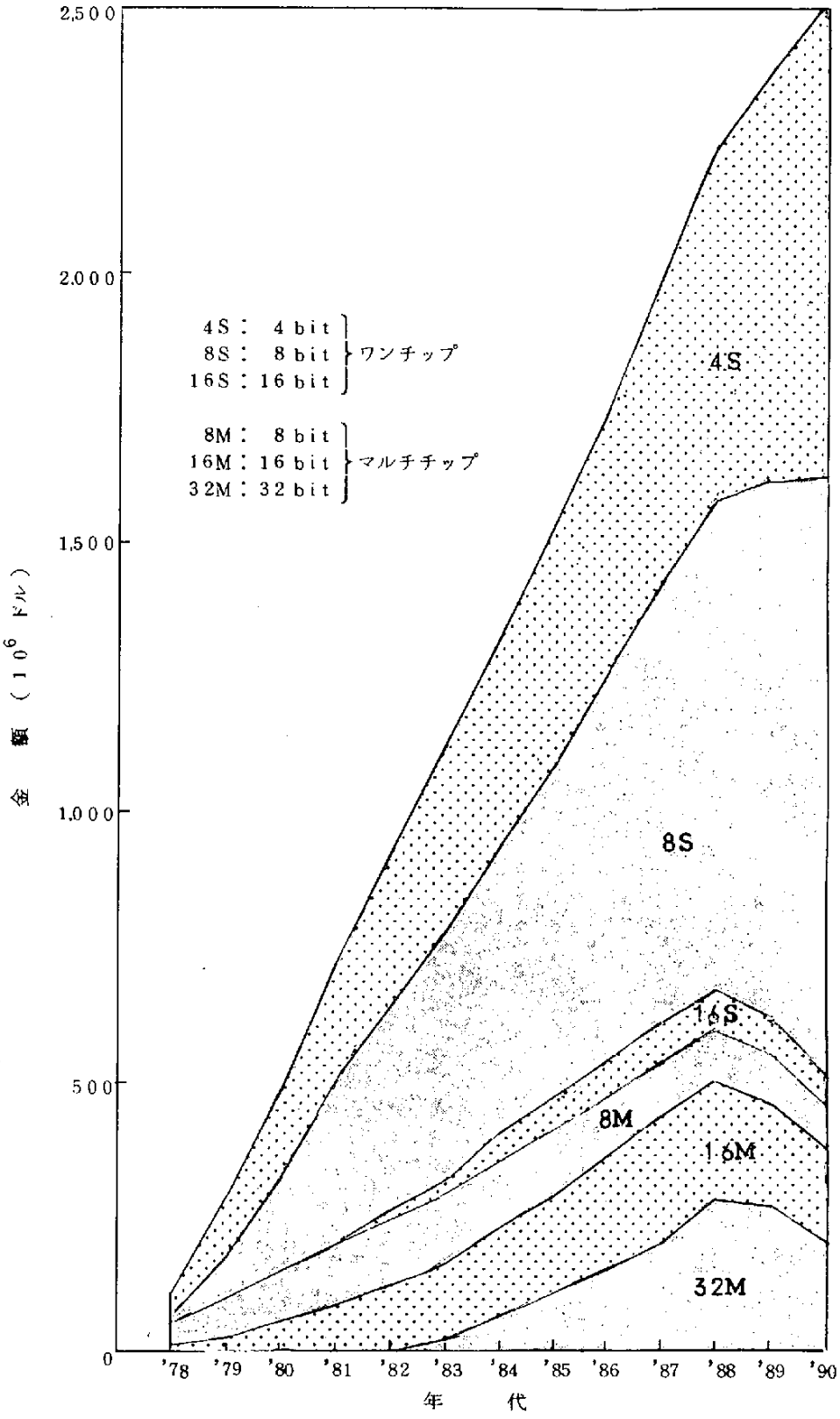


図４－１９　ビット別市場規模

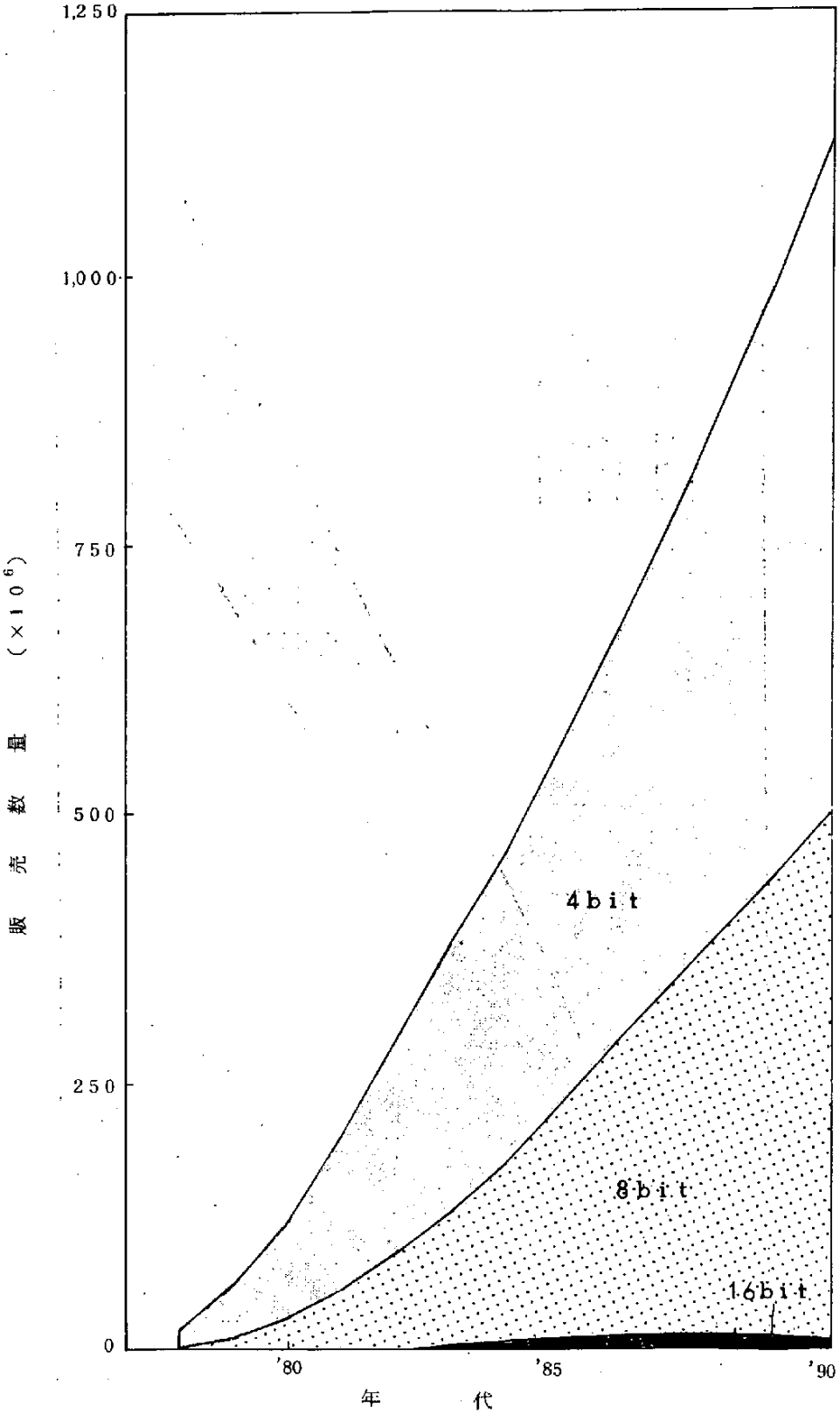


図4－20　ワンチップマイクロコンピュータ市場推移

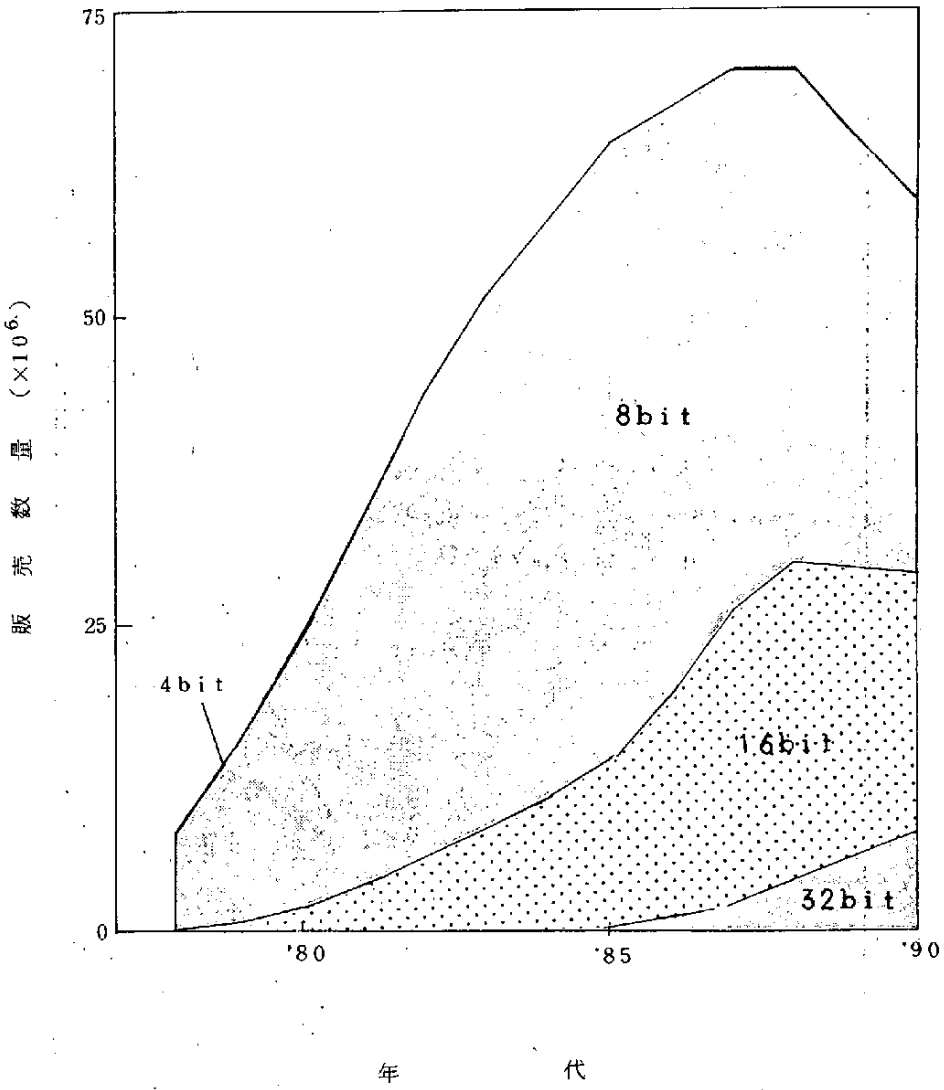


図４－２１　マルチチップマイクロコンピュータ市場推移

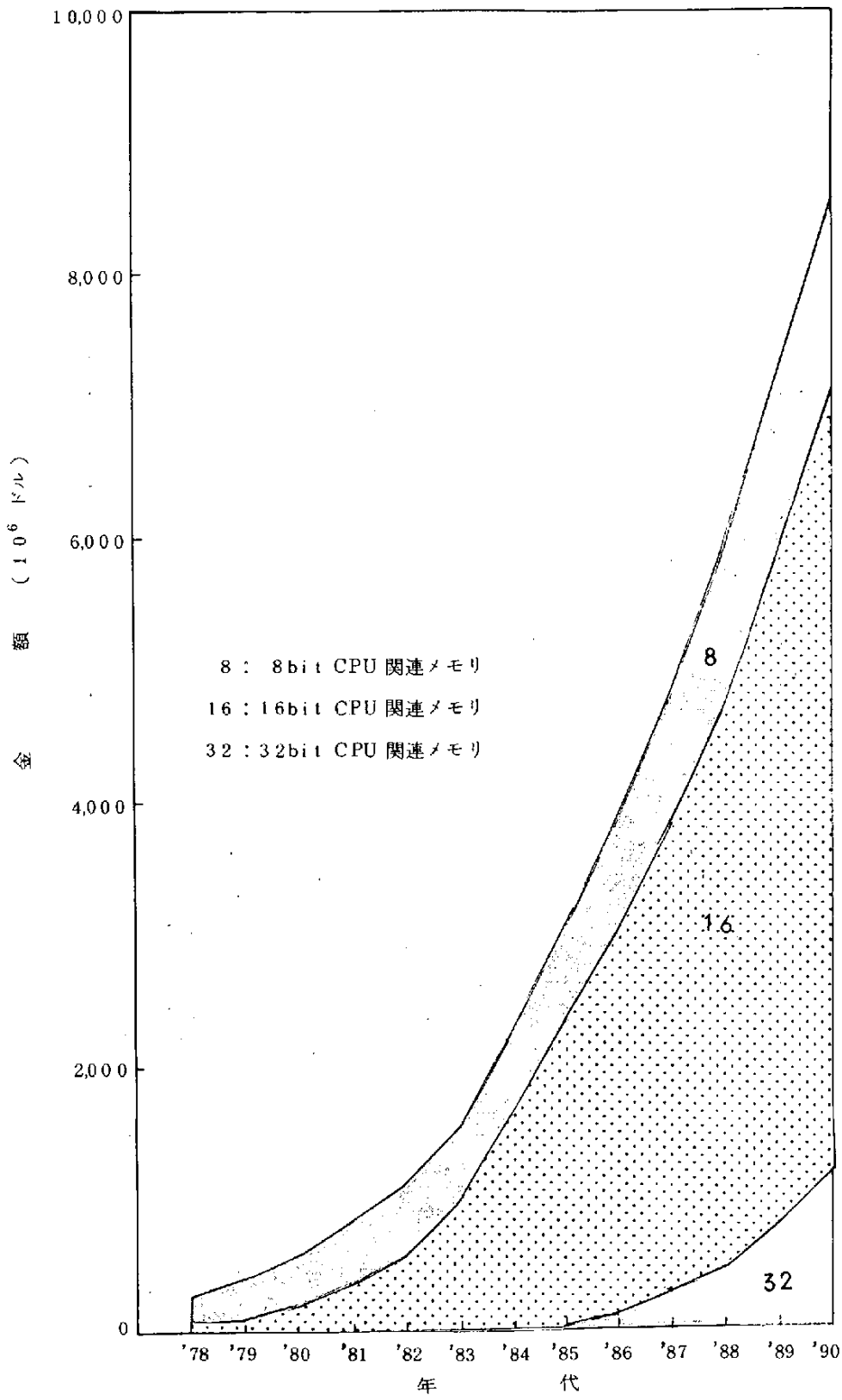


図4－22　マルチチップマイクロコンピュータ市場推移

第5章　マイクロコンピュータ用システム記述言語の評価と展望

第5章　マイクロコンピュータ用システム記述言語の評価と展望

5.1　システム記述言語問題の背景

16ビットマイクロコンピュータのシステム記述言語を考察するにあたって，少なくとも以下の背景に言及する必要がある。

(1)　ハードウェアの性能向上

16ビットマイクロコンピュータになって初めて，64ＫＢのメモリと2連のフレキシブルディスクという壁を超えてメガバイト単位のメモリやハードディスクが利用できるようになった。4ビットや8ビットの時，アセンブリ言語あるいは汎用機や専用開発装置によるクロスコンパイル方式が主流を占めていたのは周知の通りである。16ビットマイクロプロセッサによる標準プログラム開発環境は，例えば次のようであろう。

Ｃ　Ｐ　Ｕ	16ビットマイクロプロセッサ
Ｒ　Ａ　Ｍ．	1ＭＢ～8ＭＢ
ディスク．	１００ＭＢ
プリンタ	３００ＣＰＳ

このとき，この環境は自立した（ stand　alone ）ものとして，自身のオペレーティングシステムや言語処理系の開発を行うに足るリソースをもっている。つまり，自身のシステム記述言語による自身の基本ソフトウェア開発が可能となっている。

クロス方式やＴＳＳ方式による代替手段がないわけではないが，その低価格化とニーズに最も適合したオペレーティング環境を汎用機ほどの手間をかけずに，新規に開発できるという利点によって，「16ビットマイクロコンピュータにおけるシステム記述言語問題」という固有の課題が生じたとしてもよいであろう。

(2)　開発ソフトウェアの多様性

自身でオペレーティングシステムや言語処理系を記述しようとすれば，汎用機の基本ソフトウェア作りに必要とされる

①　オペレーティングシステム

②　各種言語処理系

③　各種ツール

が欲しくなるのは当然である。そして，これらの各種ソフトウェアはただ1つあれば充足されるものではなく，ユーザニーズに応じていくつか用意される必要がある。実際，以降で13種類のシステム記述言語が評価されるが，これらはどれが最も良く，どれが最も悪いというようなリニアな評価は成立しない事情にある。

(3)　ソフトウェア開発の手間

ソフトウェアの開発方式の改良の必要性については，ようやく認識され始め，そして現場ほど痛切な事情にあるといってよい。

しかしながら，どこまでその必要性がソフトウェア開発関係者によって理解され，適切な対処がされているかについては，ソフトウェア開発の若干の実例を見るだけでも，不十分な点が少なくない。

ソフトウェア開発の改良については，過去30年にわたるプログラミング方法論，ソフトウェア工学，現場の知見などによって，次のような方策が提案され，実績もあげられてきた。

① 高水準言語の採用（4～8倍） '60年代

② エディタの活用（2～5倍） '70年代

③ プログラミング環境の改良（2～4倍） '80年代

④ 移植可能・再利用可能ソフトウェアの開発（4～10倍） '80年代

高水準言語は，アセンブリ言語と比較して，5～10倍の機械語を生成することはFORTRAN，COBOL，PL／1の例を見るまでもなく明らかである。また翻訳時機能（"include"など）やマクロ機能によって，その比率を増大させる方式も，'60年以来現場で実用化されてきたところであった。

エディタ，特に，ラインエディタやスクリーンエディタの活用については，汎用機のバッチベースの開発関係者に不可解なところがあり，（実際，使ってみなければその味がわからない）2～5倍の生産性の寄与が可能であるという知見に疑問を呈する向きも多い。しかしながら，テキストエディタも含め，対話型プログラミング環境におけるインタープリティブなプログラミング開発においては，10倍前後の生産性の寄与につながるという実例や意見もある。タイプライタが欧米ほど普及していないわが国において，和文も含めた文書編集にエディタがどれほどの威力を発揮するかは興味深い今後の課題といってよいであろう。

プログラミング環境の改良については，XeroxやMITの実例もあるが，16ビットマイクロコンピュータになじむそれは，Bell研究所のUnixシステムである。各種編集システム，ソースコードコントロールシステム，簡便なコマンドアナライザ，コンパイラコンパイラなどによって，高水準のプログラム開発環境が実現されてきたことは周知の通りである。そのシステム記述言語Cについては，以降で評価される。

移植可能ソフトウェアあるいは再利用可能ソフトウェアについては，サブルーチン，ファンクション，コンパイル単位のライブラリ，ソフトウェアパッケージのような経過をへて，現在UCSD PascalやAdaのような，言語概念にモジュラリティを導入した言語およびその言語処理系が実現されている。まだそのモジュラリティの単位であるunitやpackageのライブラリの共用が本格化していないところから，その実用的評価は下すわけにいかないが，プログラミング生産性向上の歴史の一ページを占める可能性は高い。

これらのソフトウェア開発の手間を省くあるいはソフトウェア開発方法論の改良につながる第一歩として，システム記述言語の役割はそれなりに大きいものがある。特に，4ビット，8ビット時代のアセンブリ言語主導の素朴な開発方式からの脱却の第一歩として，システム記述言語が注目されている。

(4) ソフトウェア教育

マイクロコンピュータシステムやマイクロコンピュータソフトウェアの開発に，開発当事者が現在直面している大きな課題は教育問題である。

既存の汎用中大型機システムを経験した中高年令層の無理解・反感の中で，あるいは学卒数年の経験の浅い若年層を引きつれて，現在の８ビットマイクロコンピュータシステムの開発は曲がりなりにも行われてきた。しかしながら，16ビットマイクロコンピュータソフトウェアにおいては，汎用機のアナロジーや数年のＯＪＴだけで健全な開発が可能である域を，その複雑性において既に超えてしまったとみなしてよい。

ここにおいて，１６ビットマイクロコンピュータソフトウェアの健全な理解のために，当該分野における代表的システム記述言語の理解が必要となる。放置すれば現在反省を強いられている汎用中大型機の複雑巨大な汎用基本ソフトウェア体系の路を，再びたどる危険も感じられる最近のマイクロコンピュータソフトウェアの傾向に対するアンチテーゼとして，16ビットマイクロコンピュータとともにシステム記述言語をよく位置づける必要がある。

5.2 システム記述言語に要求される機能

ここでとりあげる「システム記述言語」は次の通りである。

第１グループには，ハードウェアに依存するプログラミング言語がある。

① Assembler

② Structured Macro Assembler

第２グループには，第１グループよりもハードウェアに依存する度合が比較的少なく，基本ソフトウェアの記述を容易にするために開発されたプログラミング言語がある。

③ ＰＬ／Ｍ

④ ＰＬＺ／ＳＹＳ

⑤ Ｃ

第３グループには，汎用の問題向けのプログラミング言語がある。

⑥ ＰＬ／Ｉ（Ｇ）

⑦ 標 準 Pascal

⑧ ＵＣＳＤ Pascal

⑨ Ada

第４グループには，オペレーティング・システムの記述を中心にし，プロセスの同期機能を持つプログラミング言語がある。

⑩ Concurrent Pascal

⑪ Modula, ModulaⅡ

第５グループには，自己拡張型のプログラミング言語がある。

⑫ ＦＯＲＴＨ

⑬ ＬＩＳＰ

従って，以下のものは対象とされない。

① ハードウェア記述言語（例えばＨＤＬ）

② 事務処理用言語（例えばＣＯＢＯＬ）

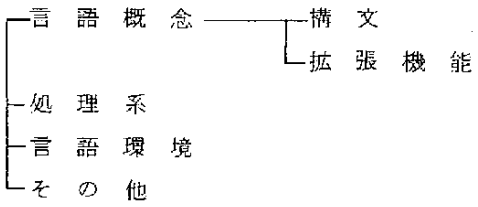
③ 科学技術計算用言語（例えばＦＯＲＴＲＡＮ）

④ 教育用言語（例えば一部のＢＡＳＩＣ）

⑤ その他応用ソフトウェア記述言語など

とりあげた言語のうち，ＦＯＲＴＨとＬＩＳＰは若干系統が異なるが，ＬＩＳＰをＬＩＳＰで記述するという見方からは「システム記述」できるという立場をとった。また，将来性からも除外しにくい意味もあった。

さて，採用した13の言語の評価についてはごく一般的に次のような枠組みを設定した。（表５－１参照）



「構文」として次の項目を設定した。

- データ抽象化
- 型変換
- 分割コンパイル
- 制御構造
- 再帰性
- 併行処理
- 例外処理
- 入出力
- 記述性
- 構文の複雑性
- 習得の容易さ
- その他

「拡張機能」としては次の項目を設定した。

- リスト処理
- マクロ処理

表５－１．言語評価の枠組み

評価項目 / 言語	言語概念（language concepts）		処理系（processor）	言語の環境（environment）	その他
	構文	拡張機能			
①Assembler ②Macro Assembler ③PL／M ④PLZ ⑤C ⑥PL／I（G） ⑦標準Pascal ⑧UCSD Pascal ⑨Ada ⑩Concurrent Pascal ⑪Modula Ⅱ ⑫FORTH ⑬LISP	データ抽象化（abstract data type） 型変換（type conversion） 分割コンパイル（separate compilation） 制御構造（control structure） 再帰性（recursion） 併行処理（concurrency multitasking） 例外処理（exception handling） 入出力（IO） 記述性（descriptiveness） 構文の複雑性（complexity） 習得の容易さ（ease of learning）	リスト処理（list processing） マクロ処理（macro capability） デバッグ機能（debugging feature） プリコンパイル機能（"include" etc）	対話性（interactiveness,batch／interactive） オブジェクトコード効率（object code efficiency） オブジェクトコード規模（object code size） 処理系の規模（compiler size） 診断情報（smart message） 再入可能性（reentrancy） ROM化（direct execution m／c）	移植性（portability） 他言語とのリンケージ（linkage） オペレーティングシステム（operating environment）（例えばunixにおけるC）	実績（popularity） 将来性（promising or not） 当該言語の対象応用分野 ハイフン

- デバッグ機能
- プリコンパイル機能
- その他

また，「処理系」に関する項目は次の通りである。

- 対話性
- オブジェクトコード効率
- オブジェクトコード規模
- 処理系の規模
- 診断情報
- 再入可能性
- ＲＯＭ化
- その他

「言語の環境」に関する項目は，次の通りである。

- 移植性
- 他言語とのリンケージ
- オペレーティングシステム
- その他

なお，「その他」一般的な項目として，次をあげた。

- 実績
- 将来性
- 対象分野
- その他

以上には，基本ソフトウェア体系を記述するにあたって当然とされる言語概念，例えば代入文，goto文の有無あるいは通常の演算子などについての記述は除外されている。評価の前提として，その言語仕様の概要が紹介されている。

以下各項目について，若干の説明と補足を行う。

(1) データ抽象化（ abstract data type ）

データの構成法とそれに対応する基本的操作を合わせて定義し，１つの抽象的な型とみなす技法である。例えば，ストリングやスタックをメモリ上でどう実現されているか意識することなしに，その名前と操作のみで扱うときに有用である。

(2) 型変換（ type corversion ）

データ型については，従来よりも厳密な扱いを要求する傾向がＰａｓｃａｌやＡｄａに見られる。特に，型の名前による型の一致と型の構造による型の一致を区別する（ Ada ）ようになっている。また，配列の大きさについての情報を静的に定めるのではなく，実行時に定める方式，しかもそれを完全にチェックする方式もある。

(3) 分割コンパイル(separate compilation)

大規模プログラムを開発するとき，プログラムを分割して開発する。このとき分割プログラムの翻訳を，翻訳単位間で厳密なチェックを行って，結合編集なしに済ませる方式が分割コンパイルである。トップダウン方式の開発に有用な技法である。

(4) 制御構造(control structure)

特に新規な点はないが，常識的な機能を持つかどうかを調べる。

(5) 再帰性(recursion)

リストやトリーのような再帰的なデータ構造を扱う言語処理系などでは，再帰的な手続き呼び出しが欲しいわけである。

(6) 併行処理(concurrency)

併行処理の中心問題は，タスク間の通信，同期，相互排除の実現方法である。モニタによる方式が一般的であるが，これはタスク間で共有される変数とその操作手続きを１つのモジュールにまとめたものである。モニタには１つのタスクだけが入ることができ，他のタスクは終了まで待たされる。

一方，Adaでは引数を通してタスク間の直接通信を行い，ランデブという方式で同期をとる。各タスクにはエントリが用意され，他のタスクからのエントリ呼び出しとタスクの受付けの実行が一致した時，通信が行われる。

(7) 例外処理(exception handling)

例外処理は，従来ＰＬ／Ｉのon条件のような例外処理部の登録を動的に行う方式が多かった。最近は，例外処理部の記述をプログラムテキストで静的に定めるようになってきている。これによって，プログラムの信頼性の向上や効率の向上あるいはテキストの読みやすさの向上が図られている。

(8) 入出力(input／output)

入出力で特に問題となるのは，ハードウェアよりの低水準記述機能であろう。ここでは，

- ワード，ビットなどの型指定
- 絶対番地指定
- 機械語命令の記述機能

などが問題となる。

(9) 記述性(descriptiveness)

基本ソフトウェア体系の記述に過不足がないかどうかが問題である。

(10) 構文の複雑性(complexity)

これは，次の習得の容易性，そして記述の容易性に影響を与える。

(11) 習得の容易さ(ease of learning)

若年層と中高年令層では，判断規準が異なるであろう。

(12) リスト処理(list processing)

略。

(13) マクロ処理(macro capability)

略。

(14) デバッグ機能(debugging feature)

既述の例外処理機能でカバーできない各種のデバッグに，どのような方式が用意されているかが課題となる。

(15) プリコンパイル機能("include" など)

これは，言語の環境と密接な関係をもっている。

(16) 対話性(interactiveness)

処理系の方式として，コンパイル方式とインタープリタ方式がある。また，エディタ等各種ツールとの併用方式の課題がある。

(17) オブジェクトコード効率(object code efficiency)

実行速度である。

(18) オブジェクトコード規模(object code size)

実行記憶域である。オブジェクトコードの最適化については，システム記述で重要なファクタであるが，今後処理系開発の技法の向上によって大幅な改良が見込まれている。また，ファームウェア化による効率向上も図られよう。

(19) 処理系の規模(compiler size)

処理系の大きさである。

(20) 診断情報(smart message)

診断情報の多寡およびその情報量は，従来の処理系の弱点であったと言ってもよいであろう。今後の言語処理系，特にシステム記述にあたっての診断情報の適切さ(smartness)は，処理系選択の1つの重要なファクタである。

(21) 再入可能性(reentrancy)

再入可能な処理系では，複数のユーザが同一の処理系(の手続き部分)を共用でき，メモリ効率がよい。

(22) ROM化(direct execution)

Pascal Micro Engine，Ada Engineのようなファームウェア化について記述する。

(23) 移植性(portability)

UCSD Pascal やAda においては，ハードウェアやオペレーティングシステムに対する依存性を，コンパイラインタープリタ方式によるユニバーサル・インターミディエト・ランゲージ(中間言語)方式によって乗り越えている。(図5－1参照)

この方式は，基本ソフトウェアのマスマーケットにおける普及に最も有力である。

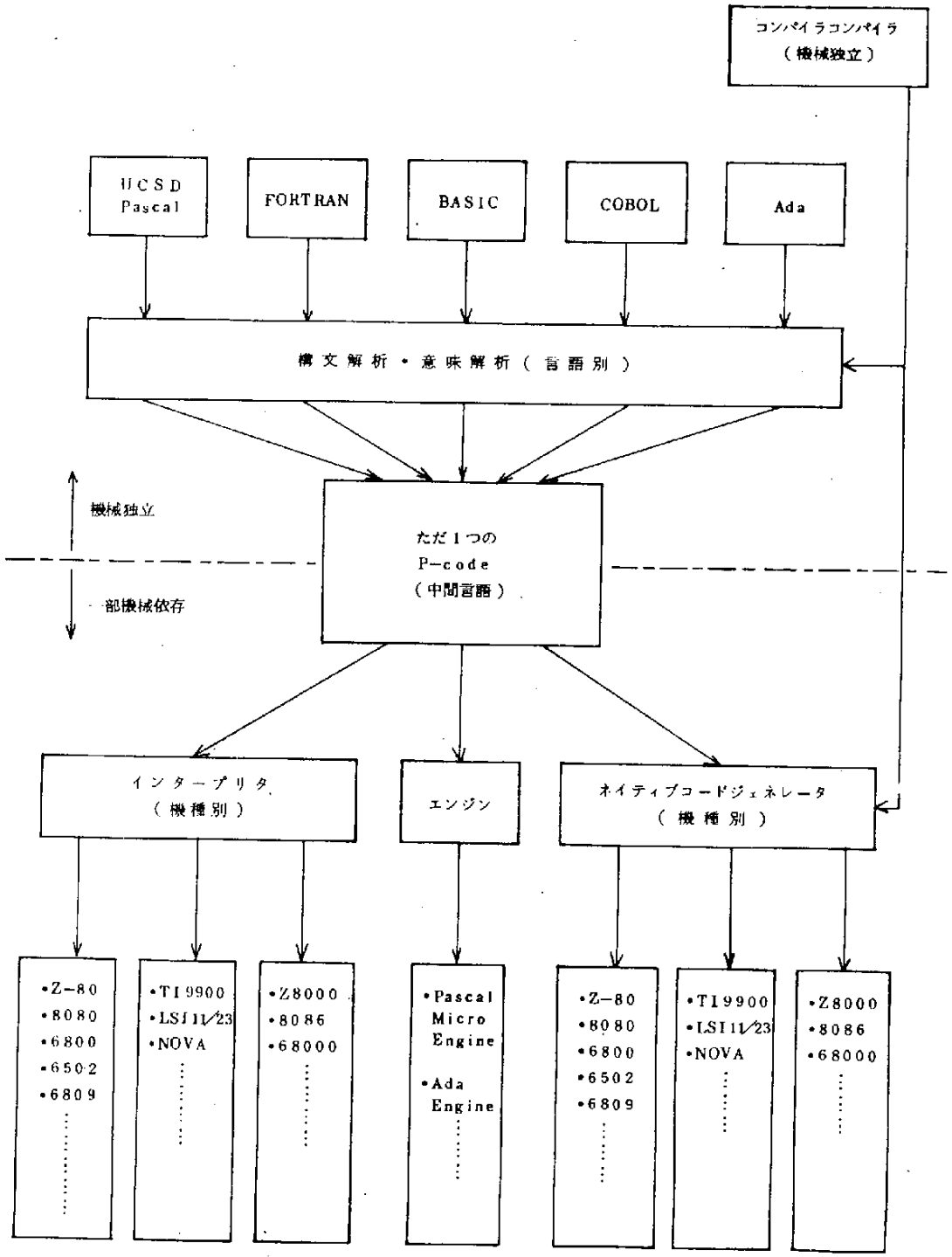

コンパイラコンパイラ
（機械独立）
UCSD
Pascal
FORTRAN
BASIC
COBOL
Ada
構文解析・意味解析（言語別）
機械独立
一部機械依存
ただ1つの
P-code
（中間言語）
インタープリタ
（機種別）
エンジン
ネイティブコードジェネレータ
（機種別）
•Z-80
•8080
•6800
•6502
•6809
•TI9900
•LSI11/23
•NOVA
•Z8000
•8086
•68000
•Pascal
Micro
Engine
•Ada
Engine
•Z-80
•8080
•6800
•6502
•6809
•TI9900
•LSI11/23
•NOVA
•Z8000
•8086
•68000

図５－１　中間言語方式

5.3 代表的システム記述言語の特徴と評価

5.3.1 Assembler （CP／M Assembler）

Assembler 言語は，コンピュータ（以下CPUと略記する）の基本命令（インストラクション）に1対1に対応する命令文によってプログラムを記述する言語である。従ってこの言語を用いるとCPUの基本動作がすべて表現でき，無駄のないきめの細かな処理や実行の速さを要求するプログラム（例えば入出力装置の制御やリアルタイム処理等）を記述するのに適している。また命令の種類は多いが文法は単純なので，プログラムの書きやすさは比較的良いといえる。

しかし，1つ1つのステップが基本的なものであるために，全体としてプログラムのステップ数が多くなり，分岐命令を多用することや名標の参照範囲に制限がないことなども手伝って，極めて読みにくいプログラムを書いてしまう恐れがある。またAssembler言語はCPUの命令に密着しているため，他のCPUとのプログラムの互換性はほとんどなく，種類の多いマイコン全般に利用可能な汎用プログラムを記述することは，プログラムの移植性の面から極めて不利である。

ここでは，最もシンプルなAssemblerの例としてCP／MのAssemblerを取り上げて，言語仕様の概観と評価を行ってみる。なお，CP／Mは各種マイコンに共通に使用できるDigital Research社の汎用オペレーティングシステムであり，CP／M Assemblerはその管理下で稼動する。ちなみにこの汎用のCP／Mの大半が5.3.3で述べるPL／Mで記述されている。

表5－2　CP／MAssembler　言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・ASCII文字セットを使用。 ・名標は，英字で始まる16文字以内の英数字及びドル記号($)から成る文字列。 ・特定の数値を持つ予約語としての10個の名標がある。 ・文はラベル，オペレーション，オペランドより構成される。 ラベルはその文が割り付けられるアドレスに付けられた名標である。 オペレーションは命令の意味を表わすものでCPUに対するものとAssemblerに対するものとがある。 オペランドは命令が作用する対象を表わし，定数や名標，それらを含む式である。 ・コメントは文の間や文字の後ろに任意に挿入できる。
名標と属性	・名標はプログラムのアドレスか数値を表わす。 ・データの型には文字，符号なしの整数（すべて16ビットで表わされ，バイトデータは上位8ビットが0でなければならない）がある。
定　　数	・整定数（2進，8進，10進，16進），文字（列）定数
演算子の種類	・算　術　　単項（＋，－） 2項（＋，－，＊，／，MOD） ・論　理　　単項（NOT） 2項（AND，OR，XOR，SHL，SHR） ・すべてアセンブル時解釈である。
制　御　文	・Jump文（条件，無条件） ・Call文（条件，無条件） ・Return文（条件，無条件） ・Restart文　Halt文　Nop文 ・構造化はされない。
プログラム構造	・プログラムは文の集まりである。 ・プログラムは構造化されず，名標はどこからでも参照できる。 ・ラベルでエントリポイントを，リターン文で戻りを示すことにより，サブルーチンを使うことができる。 サブルーチン間のデータの受け渡しは，ハードウェアのレジスタか名標を用いて行われる。 ・ハードウェア・レジスタに特定値をセットし，5番地をコールすることにより，CP／Mで与えられる外部ルーチンを参照することができる。
入　出　力	IOポート番号を指定してバイト単位で直接行う。

出典：CP／M　ASSEMBLER(ASM)　USER'S　GUIDE(Digital Research)

表5－3　Assemblerの評価（１）

評価項目 ＼ 言語			CP/M　Assembler
言語概念（language concepts）	構文	データ抽象化（abstract data type）	無し。
		形変換（type conversion）	無し。 すべて16ビットの符号なし整数とみなされる。
		分割コンパイル（separate compilation）	無し。
		制御構造（control structure）	構造上の機能は無し。CPUの分岐命令，コール命令等でプログラムの制御を行う。
		再帰性（recursion）	言語としてはなし。ただし，その機能の記述は可能。
		併行処理（concurrency multitasking）	言語としてはなし。ただし，その機能の記述は可能。
		例外処理（exception handling）	言語としてはなし。ただし，その機能の記述は可能。
		入出力（IO）	有り。IOポートを直接参照するバイト単位の入出力。
		記述性（descriptiveness）	writability は比較的よいが，非構造化等の理由で思わぬバグを誘発しやすい。
		構文の複雑性（complexity）	文法の基本的な部分は単純であるが，種類が多く，機能が多岐にわたっているため全体としては複雑である。
		習得の容易さ（ease of learn）	CPUの基本動作に関する基礎知識が必要。
		その他	

表5－3　Assemblerの評価（２）

評価項目＼言語			CP/M　Assembler
言語概念（language concepts）	拡張機能	リスト処理（list processing）	無し。
		マクロ処理（macro capability）	無し。
		デバッグ機能（debugging feature）	別に用意されたＤＤＴ（Dynamic Debugging Tool）によりデバッグ可能。
		プリコンパイル機能（〝include〟etc.）	無し。
		その他	
処理系（processor）		対話性（interactiveness batch/interactive）	バッチ方式のみ。
		オブジェクトコード効率（object code efficiency）	極めて良好。
		オブジェクトコード規模（object code size）	極めて良好。実行時ルーチンは，ＯＳ内の参照によるため，プログラムの規模には影響しない。
		処理系の規模（compiler size）	８ＫＢ。
		診断情報（smart message）	やや不親切。
		再入可能性（reentrancy）	特になし。記述することは可能。
		ROM化（direct execution m/c）	可能。
		その他	アセンブルされた結果は，IntelのHexフォーマットであるから，これを実行するには，別に用意されたローダで機械語に直す必要がある。

表5－3　Assemblerの評価（３）

評価項目＼言語		CP/M Assembler
言語の環境（environment）	移植性（portability）	極めて悪い。
	他言語とのリンケージ（linkage）	リンクするutilityは用意されていない。
	オペレーティングシステム（operating environment）	実行時ルーチンはＣＰ／Ｍ内のものを用いているから，これを使用するプログラムはＣＰ／Ｍの下でなければ実行できない。
	その他	
その他	実績（popularity）	Assembler一般として，ＣＰＵ装置の直接的処理ばかりでなく，それらと直接関係しないシステムの処理やデータ処理などのプログラムの記述に広く用いられている。
	将来性（promising or not）	Assemblerは，現在まで最も多く用いられてきた言語であるが，移植性や読みやすさの点から敬遠ぎみになってきた。しかし，これらの機能を充分持つ高位言語が普及しなければ，問題点を残しながらも，今後も使われていくであろう。
	当該言語の対象応用分野（どの分野に向くか）	リアルタイム処理やプロセス制御などＣＰＵあるいは装置を直接扱う部分を書くのに適している。
	その他	

5.3.2 Macro Assembler （SMAL）

SMALは Structured Macro Assembly Language の略で，その名の示す通り，アセンブラに構造化プログラミングの機能とマクロ機能を取り入れた言語である。1974年にベル研究所のC．Popper によって開発され，IBM 370／168 にインプリメントされた。マイコンに対しては，Chromod Associates により 8080 および 8085 に SMAL／80の名でインプリメントされている。

以後，SMAL／80の主な特徴について述べる。

(1) SMAL／80コンパイラは，マイコン用オペレーティングシステムCP／Mの下で稼動し，マクロプロセッサとアセンブラの2つの機能が独立に働く。

(2) マクロプロセッサの機能を用いると，任意の文字列の変換が可能であるから，プログラムをより抽象化した形で記述することができる。また他のCPUのアセンブラで書いたプログラムを目的のCPU用にアセンブルすることも可能である。

(3) 言語は，構造化プログラミングの制御機構を持ち，アセンブラの欠点である分岐命令のわかりにくさやループ処理のわずらわしさを排除している。

(4) CPUのインストラクションに対応する命令文は，多くのアセンブラが用いているニーモニックによる表現ではなく，レジスタやメモリを変数や配列と見立てた算術式的な記法を採用しており，CPUの命令についてあまり詳しくない人でもわかりやすい。またこの記法によりプログラムが読みやすくなっている。

表5－4　Macro Assembler （SMAL／80）言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・ASCII文字セットを使用。 ・名標は英字で始まる任意の長さ（但し，先頭の8文字のみ有効）の英数字から成る文字列で，文のラベル，数値を表わす。 ・特別の意味に使用される名標として，70余りの予約語がある。 ・文には，互いに，セミコロン(;)で区切られる単文と，それらをまとめて〔〕{}，DO　OD，BEGIN　ENDで囲んだ複文とがある。 ・命令文は代入や数式的記法で表現される。
名標と属性	・名標はプログラムのアドレス（16ビット）か数値（16ビット，8ビット）を表わす。 ・データの型には，文字（1文字が8ビット，2文字が16ビット，3文字以上はない）と符号なしの整数（8ビット，16ビット）がある。
定　　数	整定数（2進，8進，10進，16進），文字（列）定数
演算子の種類	・算術　単項（＋，－） 2項（＋，－，＊，／） ・論理　2項（AND，OR，XOR） すべてコンパイル時解釈。
制　御　文	・構造化が可能 IF（IF　～　THEN　～　ELSE　～，IF　～　GOTO） CALL，RETURN， NEXT，BREAK LOOP文（LOOP　～　PEPEAT　WHILE　～　）
プログラム構造	・プログラムは，単文あるいは複文からなる。 ・1つのプログラム内でのサブルーチン化はできるが，外部ルーチンを参照することは一般にはできない。 ・10種余りのコンパイラに対する疑似命令がある。 ・マクロ処理機能があり，任意の文字列の変換が可能。
入　出　力	・IOポート番号を指定して，バイト単位で直接行う。

出典：SMAL／80　USER'S GUIDE（Chromod Associates）

表５－５　Macro Assemblerの評価（１）

評価項目＼言語			SMAL／80
言語概念（language concepts）	構文	データ抽象化（abstract data type）	無し。
		形変換（type conversion）	無し。
		分割コンパイル（separate compilation）	無し。
		制御構造（control structure）	IF文（IF～THEN～ELSE～，IF～GOTO，CALL，RETURN，NEXT，BREAK）CPUの動作と直結したIF文の種類が多い。LOOP文（LOOP，REPEAT WHILE～）
		再帰性（recursion）	言語としては無し。ただし，その機能の記述は可能。
		併行処理（concurrency multitasking）	言語としては無し。ただし，その機能の記述は可能。
		例外処理（exception handling）	言語としては無し。ただし，その機能の記述は可能。
		入出力（IO）	有り。IOポートを直接参照するバイト単位の入出力。
		記述性（descriptiveness）	レジスタ等のニーモニックはCPU用のものを用いているが，命令文は代入や算術式的な記述ができるから書きやすい。その上，構造化できるからアルゴリズムの正確さやプログラムの読みやすさが増している。
		構文の複雑性（complexity）	命令文一つ一つはCPUの命令であるから単純であるが種類が多い。制御構造は基本的なものである。
		習得の容易さ（ease of learn）	命令文の種類は多いが，代入や算術的な記憶ができておぼえやすい。
		その他	

表５－５　Macro Assemblerの評価（２）

評価項目＼言語			SMAL／80
言語概念（language concepts）	拡張機能	リスト処理（list processing）	言語としてリスト処理機能はない。
		マクロ処理（macro capability）	プリプロセッサにより強力なマクロプロセッサが別に用意されている。これにより，いわゆるプログラム記述言語で書かれたプログラムをSMAL／80のソースコードに変換して，プログラムの正確さ，読みやすさを増すことができる。
		デバッグ機能（debugging feature）	無し。
		プリコンパイル機能（〝include〟etc.）	無し。
		その他	
処理系（processor）		対話性（interactiveness batch/interactive）	バッチ方式のみ。
		オブジェクトコード効率（object code efficiency）	良好。一つの文がほぼＣＰＵの一命令に対応しているので効率はAssemblerとほぼ同じである。
		オブジェクトコード規模（object code size）	実行時ルーチンがないのでソースコードのみがオブジェクトコードとなる。
		処理系の規模（compiler size）	３ＫＢ（Macro Processor） 10ＫＢ（Macro Assembler）
		診断情報（smart message）	コンパイルリストにはエラー番号しか出力されない。ただし種類は15程度であるからそれほど不便はない。
		再入可能性（reentrancy）	特になし。記述することは可能。
		ROM化（direct execution m/c）	可能。
		その他	

表5－5　Macro Assembler の評価（３）

評価項目＼言語		SMAL／80
言語の環境（environment）	移植性（portability）	コンパイラは特定のCPUのニーモニックを含んでいるから移植性は悪い。ただしマクロプロセッサを用いれば，目的のCPUのAssembler用に変換することは可能である。
	他言語とのリンケージ（linkage）	無し。
	オペレーティングシステム（operating environment）	特定のオペレーティング・システムは必要ない。
	その他	
その他	実績（popularity）	SMALはIBM370／168にインプリメントされている。またマクロプロセッサを除いたアセンブラ部はSofTechのMSEFで稼動。SMAL／80は，CP/M上である。
	将来性（promising or not）	構造化Assemblerとしては疑問視される。しかし一般のマクロアセンブラの必要性は残る。
	当該言語の対象応用分野（どの分野に向くか）	リアルタイム処理やプロセス制御については，CPUあるいは装置を直接扱う部分を書くのに適している。また構造化されているのでそれ以外のアルゴリズムも比較的書きやすいであろう。しかし，データ処理分野には不向きであるといえる。
	その他	

5.3.3 ＰＬ／Ｍ

ＰＬ／Ｍは，Intel 社が開発したマイコン用高位言語で，そのコンパイラは通常同社のマイクロコンピュータ開発システムＭＤＳによりＩＳＩＳ－Ⅱオペレーティングシステムの管理下で実行される。

ＰＬ／Ｍには，８ビット系8080 用のＰＬ／Ｍ－80 と16 ビット系8086 用のＰＬ／Ｍ－86とがあり，それぞれのコンパイラが開発されている。この両者は言語仕様の上で若干の相異があるが本質的には同じものであり，後者は前者をより抽象的に，より機能的にしたものと考えられる。ここでは主にＰＬ／Ｍ－86を取り上げ，その言語仕様の概観と評価を行った。以下，言語の特徴を列挙してみる。

(1) プログラムは１つ以上のモジュールから構成され，それぞれ独立にコンパイルできる。またモジュール間の参照は意識的な宣言が必要であるから，プログラムをモジュール化することにより問題の局所化が可能である。

(2) 構造化プログラミングができ，アルゴリズムを明解に記述できる。プログラムの書式が自由であることとを併用すると読み易いプログラムを書くことができる。

(3) 通常の代入文や条件文，ループ制御文に加えて割り込み処理機能や入出力等のＣＰＵに依存する処理のための組込み機能があり，アセンブラと同じような処理の記述が可能である。

(4) 手続きに対してREENTRANT属性を指定することにより，その手続き内のローカル変数がAUTOMATIC変数として割り付けられ，再入可能性や再帰性が保障される。従って，リアルタイムやマルチタスキング，コンパイラ等の構文解析などの処理が記述しやすい。

(5) コンパイルの結果作成されるオブジェクトプログラムはそのままＲＯＭに書き込める形なので，装置制御システム等の開発に有効である。

表5－6　ＰＬ／Ｍ言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・ＡＳＣＩＩおよびＥＢＣＤＩＣ文字セットのサブセットを使用。 ・名標は英字で始まる31文字以内の英数字およびドル記号より成り，変数，手続き，マクロおよびラベルに名前をつけるために使用される。 ・特別の意味を持つ43種の予約語がある。 ・文は宣言文，手続き文及び実行文から成り，実行文はさらに代入文，ＤＯ文ＩＦ文など14種の基本的な文に分けられる。
名標と属性	・変数の属性には，その有効範囲を拡大するＰＵＢＬＩＣ及びＥＸＴＥＲＮＡＬ属性と，指定したロケーションに変数を定義するＡＴ属性がある。 ・固有のロケーションが割り当てられる変数には，ＳＴＡＴＩＣとＡＵＴＯＭＡＴＩＣとがある。 ・データの型には，文字，符号なし整数（８ビット，１６ビット），符号付き整数，実数，ポインタ，基底付の基本型と，それから導出される配列と構造がある。
定　　数	整定数（２進，８進，10進，16進）　文字（列）定数
演算子の種類	・算　術　　単項（＋，－） ２項（＋，－，＊，／，ＭＯＤ，ＰＬＵＳ，ＭＩＮＵＳ） ・論　理　　単項（ＮＯＴ） ２項（ＡＮＤ，ＯＲ，ＸＯＲ） ・関　係　　（＜，＞，＜＝，＞＝，＜＞，＝）
制　御　文	ＩＦ文　（ＩＦ ～ ＴＨＥＮ　～　ＥＬＳＥ　～　） ＤＯ文　（単純ＤＯ，ＤＯ ＷＨＩＬＥ，繰り返しＤＯ，ＤＯ ＣＡＳＥ） ＧＯＴＯ文，ＨＡＬＴ文，ＣＡＬＬ文，ＲＥＴＵＲＮ文
プログラム構造	・プログラムはモジュールの集まりである。モジュールは入れ子にできない。 ・モジュールはモジュール名と，変数や手続きの宣言を含む単純ＤＯブロックから成る。 ・モジュール間のリンケージはＰＵＢＬＩＣ属性，ＥＸＴＥＲＮＡＬ属性により行われる。 ・手続き間のパラメータの受け渡しは，値渡し（ call by　value ），参照渡し（ call by　reference ）　および関数手続きとしての関数値である。 ・手続きはＲＥＥＮＴＲＡＮＴ属性を宣言することにより再入可能性および再帰性を持つ。
入　出　力	言語としては無い。 組み込み機能として，ＩＯポートの直接参照が可能。

出典：ＰＬ／Ｍ－80 Programming Manual（ Intel ），ＰＬ／Ｍ－86 Programming Manual（ Intel ）

表５－７　ＰＬ／Ｍの評価（１）

評価項目＼言語			ＰＬ／Ｍ
言語概念（language concepts）	構文	データ抽象化（abstract data type）	無し。
		形変換（type conversion）	暗黙に行われるものと，組み込み関数により行うものとがある。
		分割コンパイル（separate compilation）	無し。ただし独立コンパイルは可能，プログラムモジュールと呼ばれる単純ＤＯブロックの単位でコンパイルできる。各モジュール間のリンクはPUBLIC，EXTERNAL属性を使って行なわれる。
		制御構造（control structure）	IF文（IF ～ THEN ～ ELSE ～ ） CASE文（DO CASE ～ END ） WHILE文（DO WHILE ～ END ） 繰り返し文（DO範囲 ～ END ）
		再帰性（recursion）	有り。REENTRANT属性が宣言されていれば保障される。
		併行処理（concurrency multitasking）	言語としては無し。
		例外処理（exception handling）	言語としては無し。ただし，例外事象を意識的にチェックする方法はある。
		入出力（ＩＯ）	言語としては無し。組み込み機能の形で用意されている。
		記述性（descriptiveness）	アセンブラで記述できるものはほとんどが記述可能。
		構文の複雑性（complexity）	制御構造の種類やデータの型が少なく簡単である。
		習得の容易き（ease of learn）	比較的容易。構造化言語に慣れていればAssemblerと比べて容易である。
		その他	

表５－７　ＰＬ／Ｍの評価（２）

評価項目 ＼ 言語			ＰＬ／Ｍ
言語概念（language concepts）	拡張機能	リスト処理 （list processing）	言語としては無し。スカラがメモリアドレスとして使えるから，それをポインタとして流用することはできる。
		マクロ処理 （macro capability）	有り。ただしパラメータは使用できない。
		デバッグ機能 （debugging feature）	コンパイラ制御命令＄DEBUGにより，シンボリックデバッグを行える。
		プリコンパイル機能 （〝include〃 etc.）	コンパイラ制御命令＄INCLUDEにより行える。
		その他	数種の組み込み機能がある。
処理系（processor）		対話性 （interactiveness batch/interactive）	バッチ方式のみ。単純なプログラム開発は少ないコマンドで行える。それに加えてコンパイラに対して種々の機能が指定できる。
		オブジェクトコード効率 （object code efficiency）	オブジェクトコードの大きさも，実行時間も共にアセンブラの1.5～2.0倍。基本的な最適化が行われている。
		オブジェクトコード規模 （object code size）	実行時ルーチンがないからソースコードの大きさがそのままオブジェクトコードの大きさに比例する。
		処理系の規模 （compiler size）	100KB程度（オーバレイを含む）
		診断情報 （smart message）	２００種以上のエラーメッセージがある。文章（英文）で出力されるので内容がわかりやすいが，若干あいまいな部分がある。
		再入可能性 （reentrancy）	REENTRANT属性の指定により可能。オブジェクトコードの大きさは，この属性を指定しない場合の1.5倍程度になる。
		ROM化 （direct execution m/c）	可能。コードとデータが分離されている。 リンクプログラムにより処理される。
		その他	Intel MDSシステムにおいては，フロッピディスク使用のため，コンパイル時間が長い。

表5－7　PL／Mの評価（３）

評価項目＼言語		PL／M
言語の環境（environment）	移植性（portability）	比較的良い。
	他言語とのリンケージ（linkage）	Assemblerとリンク可能。 Intel以外で，FORTRANやCOBOLとのリンケージが可能なものもある。
	オペレーティングシステム（operating environment）	実行時ルーチンの必要がないから，特定のオペレーティングシステムは必要ない。しかし，通常ＩＳＩＳ－Ⅱと共に用いられる。
	その他	
その他	実績（popularity）	Intelにおいては，ソフトウェアのほとんどがこれによって書かれている。またDigital ResearchのCP／Mの記述にも使用されている。解説書としては，『PL／Mマイクロコンピュータ・プログラミング（D．Dマクラッケン著石田晴久監訳）』がある。
	将来性（promising or not）	移植性，読みやすさがよく，構文も比較的簡単なので，システム記述用に有望であるといえる。
	当該言語の対象応用分野（どの分野に向くか）	アセンブラ的記述が可能であるから，リアルタイム処理やプロセス制御には向いている。システム記述，特にオペレーティングシステムの記述は，移植性の観点からも有効である。データ処理分野で，科学技術計算や事務計算に対しては，データの型が豊富でないなどにより，不向きであるといえる。
	その他	

5.3.4 ＰＬＺ　（ＰＬＺ／ＳＹＳ）

ＰＬＺはZilog 社がマイコンのシステムプログラム（コンパイラやエディタ等）の記述用に開発した新しい言語ファミリである。言語の概念の多くはPascal 言語に類似しているが，実数や文字列の操作と入出力文は含まれていない。

ＰＬＺファミリのうちＰＬＺ／ＳＹＳは，マイコンのプログラムの中でＣＰＵに依存する箇所を取り除いた部分の記述を行う高位言語である。これに対して，もっぱらＣＰＵ依存部を受け持つのがＰＬＺ／ＡＳＭである。両者は核と呼ばれる基本文法（データの宣言と制御構造）を共有し，その上にそれぞれの高位言語的記述あるいはアセンブラ的記述の機能を備えている。

以下高位言語の観点からＰＬＺ／ＳＹＳに注目してその特徴を概観する。

(1) プログラムを構成するモジュール毎の独立コンパイルができる。

(2) 簡単明瞭な構文により，プログラムのwritability のみならずreadabilityもよい。構造化プログラミングやブロック構造による変数の有効範囲の制限など正確なプログラムを書きやすくするための手段が整っている。制御構造はＩＦ～ＦＩ，ＤＯ～ＯＤのように囲みの形になっているのでその範囲がわかりやすい。

(3) データの型が多く，ユーザ定義も可能である。また型変換のチェック機能も持っている。ポインタを使った高度なリスト処理が可能である。

(4) １つの変数の増減を記述する代入文に対して略記法がある　（＋＝，－＝）。

(5) 言語をシンプルにするために除いてある実数演算や入出力処理は，拡張機能として外部手続きにより与えられることを期待している。

表5－8　ＰＬＺ／ＳＹＳ言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・一般にASCII文字セットを使用。 ・名標は英字で始まる任意の長さの英数字および下線から成る文字列で定数，変数，ループ名および手続きを表わす。 ・名標はそれが参照される前に定義または宣言されていなければならない。 ・特別な意味に使用される名標として40余りの予約語がある。 ・文には，単文として代入文，手書き文，リターン文，ループ制御文があり，構造を持つ文として条件文とループ文とがある。
名標と属性	・記憶域クラスとしては，GLOBAL，EXTERNAL，INTERNAL，LOCALがあり，記憶域割付けと有効範囲が決定付けられる。 ・データの型には，符号なし整数（８ビット，16ビット），符号付き整数（8ビット，16ビット），ポインタ，配列，レコードおよびユーザ定義の型がある。
定　　数	整定数（10進，16進），　文字（列）定数，　アドレス定数
演算子の種類	・算　術　　単項（＋，－，ABS，INC，DEC） 2項（＋，－，＊，／，MOD） ・論　理　　単項（NOT） 2項（AND，OR，XOR） ・関　係　　（＝，<>，＜，＞，<=，>=，ANDIF，ORIF） ・その他　　SIZEOF，＃，↑，型変換演算子
制　御　文	IF文　（IF ～ THEN ～ ELSE ～ FI） SELECT文（IF ～ CASE ～ THEN ～ … ～ ELSE ～ FI） LOOP文　（DO ～ OD，EXIT，RETURN，REPEAT，REPEAT FROM）
プログラム構造	・プログラムはモジュールの集まりである。 ・モジュールは変数の宣言と，手続きの宣言から成る。実行文は手続きの中にのみ書かれる。 ・手続き間の情報交換は，引き数，外部変数及びRETURNS部に示された変数による。 ・引き数およびRETURNS部に示された変数は値渡し（call by value）と参照渡し（call by reference）が可能。 ・手続きは再帰的に呼び出せる。
入　出　力	言語としては無し。

出典：Report on the Programming Language PLZ／SYS（Tod Snook等）

表5－9　PLZの評価（1）

評価項目＼言語			PLZ／SYS
言語概念（language concepts）	構文	データ抽象化（abstract data type）	無し。
		形変換（type conversion）	異種同志の演算，代入は禁止され，自動的な型変換は行われない。特定の演算子を用いて指定しなければならない。
		分割コンパイル（separate compilation）	無し。ただし独立コンパイルは可能。モジュールと宣言されたプログラム単位がコンパイルの単位である。各モジュール間のリンクはGLOBAL，EXTERNAL属性を使って行われる。
		制御構造（control structure）	IF文（IF～THEN～ELSE～FI）IF文はSELECT 文の特別な形のように考えられ，わかりやすい。 SELECT文（IF～CASE～THEN～…～ELSE～FI） LOOP文（DO～OD，EXIT，RETURN，REPEAT，REPEAT FROM）
		再帰性（recursion）	有り。手続き内のLOCAL変数がすべてダイナミックに割り付けられるからどの手続きも，再帰性を持つ。
		併行処理（concurrency multitasking）	無し。
		例外処理（exception handling）	無し。
		入出力（IO）	言語としては無し。標準入出力ライブラリが用意されていて，それを手続きとして使用する。
		記述性（descriptiveness）	通常のアルゴリズムは制御構造で作り出せる。メモリアドレスは，より抽象的な『ポインタ』で書ける。 演算は数学的な式で書け，一部略記もできて便利（＋＝，－＝） LOOP制御が1種類しかないので，WHILE文，REPEAT文などはプログラマが作らねばならない。
		構文の複雑性（complexity）	例外事項がほとんどなく明解である。
		習得の容易さ（ease of learn）	基本的な構文や記述方法が統一されているので習得しやすいが，高度なレコードの処理等は多少熟練を要する。
		その他	

表5－9　ＰＬＺの評価（２）

評価項目 ＼ 言語			PLZ／SYS
言語概念（language concepts）	拡張機能	リスト処理（list processing）	ポインタ変数を用いて容易にしかもより抽象的に行える。
		マクロ処理（macro capability）	言語としてはないが，プリプロセッサとして用意されている。マクロ・ライブラリもある。
		デバッグ機能（debugging feature）	無し。
		プリコンパイル機能（〝include〟etc.）	コンパイラ制御命令＄INCLUDEにより行える。コンパイラにその機能を入れてない場合には，プリプロセッサを使用することもできる。
		その他	
処理系（processor）		対話性（interactiveness batch/interactive）	バッチ方式のみ。コマンドの種類は簡単である。
		オブジェクトコード効率（object code efficiency）	１つの文が平均マシンコードで１５バイト，中間コード（Z—CODE）で7バイトくらい。
		オブジェクトコード規模（object code size）	入出力などを行う場合，実行時ルーチンが結合されるから，小さなプログラムでも比較的大きくなってしまう。
		処理系の規模（compiler size）	
		診断情報（stmart message）	コンパイルリストにエラー番号だけしか出力されないのは不親切。エラー表をその度に参照しなければならない。約240種類別に出力される。
		再入可能性（reentrancy）	手続き内のLOCAL変数がすべてダイナミックに割り付けられるから，どの手続きも再入可能である。
		ROM化（direct execution m/c）	可能。
		その他	

表5－9　ＰＬＺの評価（３）

評価項目＼言語		ＰＬＺ／ＳＹＳ
言語の環境（environment）	移植性（portability）	良好。ただし，ＣＰＵに依存する部分は書けないという制限付きである。
	他言語とのリンケージ（linkage）	同系列のAssembler　PLZ／ASMとリンク可能。 この場合，手続きのパラメータの記述が全く同様にできているので便利。
	オペレーティングシステム（operating environment）	入出力機能等の手続きはプログラム側にリンクされるので実行時に特別なオペレーティングシステムは必要ない。
	その他	
その他	実績（popularity）	Zilog社内で用いられている。 PLZ／SYSを用いたプログラミングの解説書が出版されている。 （Richard Conway著：Introduction to Microprocessor Programming Using PLZ）
	将来性（promising or not）	PLZ／SYSだけではシステム記述の細かい部分は書きにくいが，同系列のPLZ／ASMと共に使用すれば，システム記述言語として期待できる。
	当該言語の対象応用分野（どの分野に向くか）	システム記述の場合，特にCPUに独立な部分の記述に適している。 CPUに依存する部分はむしろPLZ／ASMを用いた方がよい。 科学技術計算や事務計算には不向きである。
	その他	

5.3.5 C言語

C言語はベル研究所において1972年にリッチ（D. M. Ritchie）等によって，UNIXというタイムシェアリングのオペレーティングシステムを記述するために開発された。

言語の概念の多くはBCPL言語に由来しており，初期のUNIXに使用されたB言語を通して間接的に影響を受けているのでBCPL系の言語である。以下に，C言語およびそのコンパイラの主な特徴について述べる。

(1) ストラクチャードプログラミング，モジュール設計，情報隠ぺい技法などの現代的なソフトウェア工学の成果を実現できる機能があり実用性が高い。

(2) 言語は，式の削減，現代的な制御フロー，データ構造，豊富な演算子などの特徴を持ち，明確で簡潔な記述ができる。

(3) コンパイラは，言語仕様が小さいので，単純でコンパクトに作られており，また，データタイプや制御構造はほとんどの計算機で直接サポートされているので，完結したプログラムのインプリメントに必要とされる実行時ルーチンも小さい。

(4) コンパイル時機能として，マクロ機能，ファイルのインクルード機能，条件コンパイル機能があり，また，タイプチェックなどのツールとしてlintと呼ばれるプログラムベリファイヤが用意されており，プログラミング環境がよい。

(5) マシンのアーキテクチャに独立なので，ポータブルなプログラムを記述でき，事実Cコンパイラは，いくつかの機種に移植されている。コンパイラがコンパクトであることを併せればマイコンも含めた多種の計算機に対して高い移植性を有している。

オペレータの優先順位等に僅かな問題点もあるが，アセンブラレベルからコンパイラレベルまで幅広い記述ができ，システム記述用言語として効果的かつ意味ある言語であると言えよう。

表５－10　C言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・文字セットはASCII文字。なお，キーワードは小文字。 ・名標は英字で始まる任意の長さ（但し，先頭の８文字のみ有効）の英数字および下線からなる文字の列で変数名，関数名および文を表わすのに使用。 ・特別の意味に使用される名標として28種のキーワードがあり，すべて予約語である。 ・文には式文，制御文，宣言文および空文などの12種の単一文とそれらを｛｝で囲った複合文がある。
名標と属性	・記憶域クラスとしてautomatic，static，external，registerがあり，記憶域割付けと有効範囲が決定づけられる。 ・データの型には，文字，整数（短，標準（符号あり／なし），長），浮動小数点（単精度，倍精度）の基本型とそれから導出される配列（多次元），関数，ポインタ，構造体，ユニオンおよびユーザ定義の型がある。
定　　数	整定数（８進，10進，16進），浮動小数点定数，文字定数，文字列定数，シンボリック定数
演算子の種類	・間接参照，番地参照，インクリメント，デクリメントなどの12種の単項演算子 ・算術，シフト，関係，ビット論理，論理，代入などの30種の２項演算子 ・条件式を表わす３項演算子
制　御　文	・２分岐（if－else），多分岐（switch） ・トップ判定ループ（while，for），ボトム判定ループ（do－while） ・ループ制御（break，continue） ・goto，　return
プログラム構造	・プログラムは１つ以上の関数の集まりである。 ・関数定義は入れ子にできない。（ＰＬ／Ｉのような内部手続きはない。） ・関数間の情報交換はアーギュメント，外部変数および関数の返す値である。 ・アーギュメントの引渡しは値返し（call by value）のみである。渡されたポインタを受けて値を返すことができる。 ・関数は値を返しても返さなくてもよいが，整数型以外の値を返すときはその型宣言が必要。 ・すべての関数は直接または間接に自分自身を呼出すこと（再帰呼出し）ができる。
入　出　力	・言語としては入出力機能はもたない。 ・標準ライブラリ関数として，一文字入出力，書式化入出力，行入出力他がある。

出典：B. W. Kernighan, D. M. Ritchie「THE C PROGRAMMING LANGUAGE」
The Bell System Technical Journal Vol. 57 No.6

表5－11　Cの評価（1）

評価項目＼言語			C
言語概念（language concepts）	構文	データ抽象化（abstract data type）	ファイル内で局所的なstaticデータとそれを操作する演算としての関数を1つのファイルにまとめることができるので，データ抽象化に対応したモジュール（ファイル）が表現できる。
		形変換（type conversion）	型交換は数少ない変換規則に従って自動的に行われるが，型変換演算子を用いることによって明白なタイプ交換を強制的に行うこともできる。
		分割コンパイル（separate compilation）	ファイル単位（ファイル中には1つ以上の関数が含まれる）にコンパイルすることができる。
		制御構造（control structure）	ストラクチャードプログラミングが可能な制御構造の文（swichなど）が用意されている。
		再帰性（recursion）	直接または間接に自分自身を呼び出すことがすべての関数で可能である。
		併行処理（concurrency multitasking）	言語自身にはない。しかしUNIXオペレーティングシステムでも実証されているように並列処理との整合性はよいと思われる。
		例外処理（exception handling）	言語にはない。OSにデペンドした関数の利用で可。
		入出力（IO）	言語として特に入出力の文はない。標準入出力ライブラリ関数がある。
		記述性（descriptiveness）	簡潔な表現により，1文の機能密度が高く，端末のキーインが他の言語と比較して少ない。また，英小文字を主体としているので英米語使用者には，記述性およびドキュメント性がよい。
		構文の複雑性（complexity）	ポイントを用いた間接データ参照は高度な表現を要する場合に，多少複雑であるが，全般的には他言語（例えばPascal）とそう変らない。
		習得の容易さ（ease of learn）	文字操作や演算子の用法の一部を除けば，他の概念と変らないので，習得は容易である。特に「THE C PROGRAMMING LANGUAGE」（B.W. Kerrngham, D.M. Ritchie）という良い指導書がある。
		その他	BCPL系の言語で，言語の概念はBCPL→B→Cと受けついでいる。但し，BCPL，Bはタイプレス言語である。関数系言語であり，また演算子の種類が豊富である。

表5－11　Cの評価（２）

評価項目＼言語			C
言語概念（language concepts）	拡張機能	リスト処理（list processing）	ポインタデータ，構造体データの操作で容易に可能である。動的記憶域管理はライブラリ関数に依存する。
		マクロ処理（macro capability）	プリコンパイル機能として含まれている。特にアーギュメントつきのマクロ定義はその記述性の向上とポータビリティの向上のために有効である。
		デバッグ機能（debugging feature）	特に言語にはない。
		プリコンパイル機能（〝include〟etc）	前述のマクロ処理，ファイルのインクルード機能および条件コンパイルなどがある。
		その他	新しいデータの型名を定義することができる。
処理系（processor）		対話性（interactiveness batch/interactive）	UNIXでは，標準は会話型であり，バッチでもコンパイルできる。（なお，以後，特に断らない限り処理系はUNIX を想定している）
		オブジェクトコード効率（object code efficiency）	基本的に効率のよいオブジェクトが生成できる言語仕様になっている。また最適化フェーズを持っている。
		オブジェクトコード規模（object code size）	上記と同じで効率の良いコードが生成される。コードとしてシステムライブラリ中の実行時ルーチンの呼出しは不要。
		処理系の規模（compiler size）	C言語自身の記述で約８Ｋステップ（ランタイムルーチンについては不明）。実行時のサイズはPDP－11については不明。なお，C6000（Honeywell 6000）では 57 KW，8080／Z 80 では 48 KB（Whitesmiths 社）。
		診断情報（smart message）	あまり親切ではない。なお実行時に配列添字やアーギュメントタイプのチェックは行っていないが，強力なプログラムベリファイヤであるlintと呼ばれるツールが別に用意されている。
		再入可能性（reentrancy）	OSに依存する。（UNIXでは可能である）
		ROM化（direct execution m/c）	オブジェクトプログラムのROM化処理系によるが，一般に可能である。処理系については，米国BBN社からCマシン（C／70 マルチファクションプロセッサ）が発表されている。
		その他	関数呼出し時のアーギュメントの引渡しは call by value である。コンパイラの出力は機械語の場合とアセンブラ言語の場合がある。

表5－11　Cの評価（3）

評価項目＼言語		C
言語の環境（environment）	移植性（portability）	CコンパイラはDEC　PDP−11上で開発されたがIBMS/370，Honeywell 6000，Interdata 8/32，SEL 86 NOVA，ECLIPSE，VAX−11/780上に移植された実積がある。またこれらの4つのマシン上でのCコンパイラは，ANSI標準版のFORTRANよりずっと互換性があるという事実がある。マイコンとしてはZENIX（i 8086，Z 8000，M 68000）に移植されつつある。
	他言語とのリンケージ（linkage）	FORTRANとリンケージがとれる。
	オペレーティングシステム（operating environment）	UNIX（PDP−11，VAX−11，Interdata 8/32，UNIVAC1107など） RSX−11M（PDP−11：Yourdon社） Idris（i8086，Whitesmiths社） ZENIX（i8086，Z8000，M68000，Microsoft社） GCOS（Honeywell6000）（IBM S/370）
	その他	automatic変数は効率向上のためレジスタにとられる場合があるがあるが，レジスタ宣言は単なるコンパイラに対するヒントであり，特定のハード上のレジスタを示すものではない。
その他	実績（popularity）	UNIXの核の90％およびそのアプリケーションソフトのほとんどがCで記述されている。（UNIXの核は約10Kステップで記述した1Kステップのうち200ステップは効率向上のためで，Cで書けない部分は800ステップである）UNIXは，1980年8月現在で，1800システムが稼動。
	将来性（promising or not）	PascalやAdaと競合するが，以上の理由でCの将来性は高い。Pascalをこのようにシステム記述用として実用的にするには拡張が必要であり，標準化，移植性などに問題が生じる。またAdaはCと比較して，コンパイラの規模はもちろん，実行時ルーチンも大きくなり，サポート可能なコンピュータが制御される危険性がある。
	当該言語の対象応用分野（どの分野に向くか）	UNIXの実積から，OSやそのアプリケーションソフトのほとんどを記述できるのでデータ処理には向いている。ただ，ファイル処理や10進演算がないので事務計算上については充分とは言えないが，元来，他の高級言語に比べて，相対的には低レベル言語であり，アセンブラ言語を少量用いることによって対象分野を拡げることは可能である。
	その他	

5.3.6 PL／I（G）

PL／I（G）は，ANSI PL／Iの適切な汎用サブセットである。

サブセット化に当たっては，次の様な考慮がなされたが，PL／Iらしい特徴は十分引き継いでいる。

- 商業，科学技術およびシステムプログラミングの各分野に適合する。
- 十分小さな計算機システム上でも，処理系の作成が容易に実現できる。
- フルセットに比べて，より学びやすく理解しやすい。
- 使用頻度の少ない機能，代替手段のある機能および誤りやすい機能は制限する。

この結果，サブセットの言語仕様は，フルセットに比べ極めて簡潔になっている。

PL／I（G）言語機能の一般的な特徴を次に述べる。

- ブロック構造（手続き，BEGIN）を持つ。
- ビット，列，整数，実数およびポインタなどの豊富なデータ型。
- 配列や構造体ごとの代入文およびALLOCATE文，FREE文などの記憶域制御。
- シーケンシャル，ダイレクト，キーおよび各種の書式制御を含んだ入出力文。
- 豊富な組込み関数および擬変数。

処理系に関しては，フルセット仕様に近いものは，規模も大きく複雑で今まで大型汎用システム上でしか実現されなかった。また，PL／Iスタイルをした言語は，各種システム上で実現されているが，多くの方言を含みPL／Iとして普及を防げている。

PL／I（G）は，ANSIの完全サブセットとして標準化が進められており，言語規模も小さいので，今後多数の効率のよい処理系が作成される可能性がある。

表5－12　PL／1（G）言語仕様

	仕　様　・　機　能　の　概　略
プログラム構造	・プログラムは手続きの集まりである。 ・外部手続き，内部手続きおよび関数手続きは再帰的に呼出せる。 ・BEGINブロックおよびDOグループがある。 ・名前の有効範囲は手続きおよびBEGINブロックである。 ・データ宣言はDECLARE文により必ず行う必要がある。
名標と属性	・データの型として，BINARY／DECIMAL，FIXED／FLOAT，BIT，CHARACTER，POINTER，LABEL，ENTRY，FILE，PICTUREがある。 ・記憶域クラスとして，STATIC（初期値可），AUTOMATIC，BASED，DEFINED，EXTERNALがある。 データの構造には，配列（上，下限指定），構造体，構造体配列がある。 ・他の属性として，ALIGNED，VARYING，VARIABLE，BUILTINがある。
定　　数	固定小数点，浮動小数点，ビット列，文字列，ラベル，入口，ファイル
演算子の種類	・2項演算子が16種類ある。 （＋，－，＊，／，＊＊，|，¬&，||，＜，＞，＝，＞＝，＜＝，＞＞，＞＜，） ・単項演算子が3種類ある。（＋，－，¬） ・ロケータ修飾子としてー›がある。 ・スカラ，配列および構造体の代入は"＝"で行う。 ・組込関数（58種類）と擬変数（4種類）がある。
制　御　文	・CALL文，RETURN文，関数参照 ・IF文（IF ～ THEN ～ ELSE ～） ・DO文（DO，DO　TO　BY，DO WHILE，DO REPEAT） ・STOP文，GO TO文，空文 ・ON文（ERROR，ENDFILE，ENDPAGE，KEYコンディション） ・SIGNAL文
入　出　力	・OPEN文，CLOSE文 ・レコード入出力文（READ，WRITE，REWRITE，DELETE） ・ストリーム入出力文（GET，PUT） ・FORMAT文 （E，F，P，A，X，R，PAGE，SKIP，COLUMN，TAB，LINE）
そ　の　他	・記憶域制御文として，ALLOCATE文およびFREE文がある。 ・テキスト組込みとして％INCLUDE文

出典：(1)　Draft proposed American National Standard Programming Language PL／I General Purpose Subset，SUBSET／G　Revision 8，1979年

(2)　PL／Iの標準化に関する調査，日本電子工業振興協会　昭和54年3月

表5－13　PL／I（G）の評価（1）

評価項目＼言語			PL／I（G）
言語概念（language concepts）	構文	データ抽象化（abstract data type）	無し。
		形変換（type conversion）	代入文または組込み関数で行う。暗黙のデータ変換は制限している。
		分割コンパイル（separate compilation）	無し。（独立コンパイルは可能）
		制御構造（control structure）	有り。IF～THEN～ELSE，DO WHILE，DO REPEAT，繰り返し形DO，BEGINブロック，PROCEDURE，
		再帰性（recursion）	有り。（オプション）
		併行処理（concurrency multitasking）	無し。
		例外処理（exception handling）	有り。ON文。（ERROR，ENDFILE，ENDPAGE，KEYコンディションのみ）
		入出力（IO）	有り。シーケンシャル，ダイレクト，キー，ストリーム型，レコード型書式指定。
		記述性（descriptiveness）	普通。フリーカラム，キーワードの省略型，省略時解釈，予約語が無し。（保守性の良いプログラムは書ける）。
		構文の複雑性（complexity）	複雑。（特にデータ宣言）
		習得の容易さ（ease of learn）	むずかしい。わかりやすい説明書が少ない。
		その他	

表5－13　ＰＬ／Ｉ（Ｇ）の評価（２）

評価項目＼言語			ＰＬ／Ｉ（Ｇ）
言語概念（language concepts）	拡張機能	リスト処理 （list processing）	ポインタその他で代用可能。
		マクロ処理 （macro capability）	無し。
		デバッグ機能 （debugging feature）	無し。ＳＩＧＮＡＬ文はある。
		プリコンパイル機能 （〝include〟etc）	テキスト単純組込みのみ。（％ＩＮＣＬＵＤ文）
		その他	・ＡＮＳＩ標準ＰＬ／Ｉの完全セット。 ・言語仕様は全体的にまだまだ重い。 ・汎用性は抜群である。
処理系（processor）		対話性 （interactiveness batch/interactive）	無し。完全にバッチ型言語。
		オブジェクトコード効率 （object code efficiency）	良。（ただし高度な最適化が必要）
		オブジェクトコード規模 （object code size）	良。（ただし高度な最適化が必要）実行時ルーチンの構成にも大きく依存する。
		処理系の規模 （compiler size）	大。作成方法のモデルがない。作成者にまかされている部分が多い。
		診断情報 （smart message）	規定されていない。（処理系依存）
		再入可能性 （reentrancy）	有り。（ＲＥＣＵＲＳＩＶＥオプション）
		ＲＯＭ化 （direct execution m/c）	やれる。
		その他	本格的な処理系の作成はこれからである。

表5－13　PL／I（G）の評価（3）

評価項目＼言語		PL／I(G)
言語の環境（environment）	移植性（portability）	これからの課題。作成方法の標準化が必要。
	他言語とのリンケージ（linkage）	とりやすい。各種のデータ型を持つ。
	オペレーティングシステム（operating environment）	汎用ＯＳを想定している。
	その他	
その他	実績（popularity）	・少ない。 ・大型データ処理分野では有り。（フルセット） ・PL／Iライク言語は多数実績有り。
	将来性（promising or not）	順調にのびそう。ただし作成者やユーザ団体の努力が必要。
	当該言語の対象応用分野（どの分野に向くか）	汎用，商業，科学技術，システム記述，その他。
	その他	

5.3.7 標準 Pascal

1971年にスイスのチューリッヒ連邦工科大学のN. Wirth 教授により提案された言語である。Pascal の第1の目的は，プログラミングを教えるのに適した言語を用意することである。プログラミングの体系的な訓練のために，その基礎概念が明解に，かつ自然に反映されている言語として考えられた。第2の目的は，現在使用できる計算機の上で，信頼でき，能率のよい処理系が作られることで，そのための言語としても考えられている。

言語上の特徴は次の通りである。

(1) データ型の概念を強く打出している。プログラマがデータ型を正しくするよう厳しく求めているため，誤りの少ないプログラムが作れるようになっている。

(2) 数え上げとして名前を値として持つ型（スカラ型）を導入し，プログラムのドキュメントとしての性格が強くなるようにしている。

(3) 構造的プログラミングに必要な制御構造を持つ。

(4) データの大きさが動的に決定される要因を極力排することによって，能率のよい目的プログラムを生成できる。

(5) 分割コンパイルはできない。

Pascal は教育用言語として考えられたため，大学，研究機関を中心に普及し，現在かなり多くの計算機で動くようになってきている。表5－14は，Pascal ユーザズ・グループ（米国に本部のある同好者の団体）のメンバが使用している，Pascal コンパイラの動いている計算機名のリストである。個人の提案になる言語で，これだけの普及を見ているものは珍しい。

Pascal は言語仕様が単純なため，コンパイラを小さくできるので，マイクロコンピュータ用言語として適しているといえる。

表５－14　Pascal が稼動している計算機

（出典：Pascal News #19, PUG, Sept. 1980）

ACOS-800
AIM/65
ALGO 2100
Alpha Micro AM-100
Altos ASC-8000
AMC System 29
Amdahl 470
American Microsystems S6800
AMTELCO
Andromeda
Apple II
Astrocom S760
Basin-4
BESM-6
Beta WS-1000
Billings 8080
BTI-4000
BTI-8000
Burroughs B1700/1800
Burroughs B2700
Burroughs B3700/3800-B4700/4800
Burroughs B5500/5700
Burroughs B6700/6800-B7700/7800
CDC 1700/Cyber 18
CDC 3000
CDC 6000,7000/Cyber 70,170
CDC Cyber 200/Star-100
CDC MP-32
CDC MP-60
CDC Omega 480
CII Iris 50
CII Iris 80/10070
Commodore Pet
Computer Automation 21S
Computer Automation LSI-2
Computer Automation LSI-4
Comten (NCR)
COSMAC ELF
CPS-03 (M6800)
Cray Research CRAY-1
Cromemco Z-80
CTL Modular One
Data-100 (Northern Telecom) 78
Data General 600/Nova + microNova
Data General Eclipse
Datapoint
DEC PDP-8
DEC PDP-11
DEC LSI-11 (+114)
DEC PDP-15
DEC VAX 11/780
DECsystem 10
DECsystem 20
Diehl/CTM
Dietz MINCAL 621
Digital Group Z-80
Digital System SD3
Dynabyte DB 8/1
ECD Microming
ES-1022
Exidy Sorcerer Z-80
Ferranti Argus 700
Four-Phase Systems
Foxboro FOX-1
Fujitsu FACOM M190
Fujitsu FACOM 230
Futuredata Z-80
Galaxy 5
General Automation 18/30
General Automation 100
General Automation 220
General Automation 440
GEC 4080
Gimix 6800
GOLEM 8
GRI System 99
Harris 4/6
Harris S135
Harris S200
Harris S500
Heathkit H-8
Heathkit H-11
Hewlett Packard 1000
Hewlett Packard 2000/2100
Hewlett Packard 21MX
Hewlett Packard 3000
HEX-29
Hitachi 8000
Honeywell H316
Honeywell Level 6
Honeywell 6000/Level 66/68
IBM Series 1
IBM System 3
IBM System 32/34
IBM 1130
IBM System 360/370
IBM 3030
IBM 4330
ICL 1900
ICL 2900
ILLIAC IV
IMSAI VDP 40
IMSAI VDP 80
IMSAI 8080/8085
Intel 8080 (+73)
Intel 8085 (+5)
Intel 8086
Itel (National) AS 456
Ithaca Audio
ITT 1652
ITT 2020
Jacquail J-100
KIM-1
LEC-16
Lockheed Sue
Manchester MU-5
Marinchip 9900
MDS-800
MEMBRAIN
Microdata 32/5
Microdata 1630
MITS Altair 680
MITS Altair 8800
MITS Altair Z-80
Mitsubishi MELCOM 7700
3M Linolex
MODCOMP II
MODCOMP IV
Mostek 6502 (+44)
Motorola 6800 (+10)
Motorola 6809
Motorola 68000
Nanodata QM-1
National Semiconductor S-400
National Semiconductor 2900
National Semiconductor PACE
NCR Century
NCR 8000
NEAC-900
NEAC-3200
Norsk Data NORD-10
North Star Horizon (Z-80)
Northwest Micro 85/P
Odell System 85
Ohio Scientific Challenger
Ontel OP-1
PDS-4
Pertec PCC XL40
Pertec PCC 2000
Perkin Elmer Interdata 7/16
Perkin Elmer Interdata 7/32
Perkin Elmer Interdata 8/16
Perkin Elmer Interdata 8/32
Perkin Elmer 3200
Polymorphics 88
Prime P-300
Prime P-400
Prime P-500
Processor Technology SOL-20
Quasar 6800
Quotron 801
Radio Shack TRS-80
RCA 301
RCA 1802
Rockwell 6502
ROLM 1600
RP-16
SBC 80/20
Systems Engineering SEL 32
Systems Engineering SEL 0500
SEMS SOLAR
SEMS T1600
Siemens 4000
Siemens 7000
Singer GP-48
Singer Librascope
Singer System 10
SORD M-222
SPC-16
Sperry SDP-175
SWTP 6800
Sycor (Northern Telecom) 445
Tandem 16
TDL Z-80
TDS-8 (Z-80)
Tektronix 8002
Telefunken 80
Telefunken TR-440
Terak 8510
Three Rivers PERQ
Texas Instruments 980
Texas Instruments 990
Texas Instruments 9900
Texas Instruments ASC
Texas Instruments DX-10
Time Machine TM-600
Univac 418
Univac 90/9000
Univac 1100
Univac V70/77
Univac UYK-7
Vector Graphics MZ
Wang WPS-30
Wang WPS-40
Wang 928
Wang 2200
Western Digital Microengine
Xerox (Honeywell) 560
Xerox (Honeywell) Sigma 3
Xerox (Honeywell) Sigma 5
Xerox (Honeywell) Sigma 6
Xerox (Honeywell) Sigma 7
Xerox (Honeywell) Sigma 8
Xerox (Honeywell) Sigma 9
Xitan Z-80
Zilog Z-80 (+78)
Zilog Z-8000

表5－15　標準Pascal言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	• 文字セットは，ASCII，EBCDIC等特定しない。 • プログラムは6種のシラブル（名前，定数，キーワード，演算記号，区切り記号，注釈）からなる。 • 名前は英字で始まる英数字で，長さの制限はないが，高々8文字で区別する。 • キーワードは35種を用い，すべて予約語である。
名標と属性	• データ属性は型と呼ばれる。6種の単純型（実数，整数，数え上げ，論理，部分範囲，文字）と5種の複合型（集合，配列，レコード，ファイル，ポインタ）がある。 • 演算のオペランド，手続きあるいは関数呼出しの実パラメタは型を正しく守らねばならない。 • 名前は6種の文法要素（関数，手続き，変数，型，定数，フィールド）の命令に用いる。
定　　数	• 実定数，整定数，文字定数，ストリング
演算子の種類	• 実数（加，減，乗，除，符号反転），整数（加，減，乗，除，剰余，符号反転），論理（否定，和，積），集合（和，差，積，要素の存在）， 標準関数17種
制　御　文	• 2分岐（IF ～ THEN ～，IF ～ THEN ～ ELSE） • 多分岐（CASE ～ OF ～，～，… ～ END） • WHILE型繰返し（WHILE ～ DO ～） • UNTIL型繰返し（REPEAT ～ UNTIL ～） • FOR型繰返し（FOR　:= ～ TO ～ DO ～），GOTO
プログラム構造	• プログラムは頭書きと，ブロックよりなる。 • 関数，手続きの宣言は頭書きと，ブロックよりなる。 • ブロックは，ラベル，定数，型，変数，関数，手続きの宣言および実行文からなる。 • すべての宣言は名前を宣言するためのものである。名前はプログラム上参照に先立って宣言されることを原則とする。 • 関数，手続きの宣言が入れ子になり得る。 • 名前の有効範囲はブロックによって区切られ，局所的有効範囲を持つ名前を宣言できる。
入　出　力	• 順編成ファイルの順アクセスのみ行うことができる。 • 文字列のファイル（テキストファイル）については編集入力，編集出力のための標準手続きが用意されている。

出典：K. Jensen，N. Wirth："Pascal User Manual and Report"（1974）

表５－16　標準Pascalの評価（１）

評価項目＼言語			標準Pascal
言語概念（language concepts）	構文	データ抽象化（abstract data type）	不可。
		形変換（type conversion）	文脈により，整数から実数に自動変換される以外は陽に指定する。
		分割コンパイル（separate compilation）	不可。
		制御構造（control structure）	条件分岐（if文，case文），繰り返し（for文，while文，repeat文）
		再帰性（recursion）	可。
		併行処理（concurrency multitasking）	不可。
		例外処理（exception handling）	不可。
		入出力（IO）	逐次ファイルの逐次アクセスのみ。
		記述性（descriptiveness）	秀。
		構文の複雑性（complexity）	例外事項がほとんどなく簡単。
		習得の容易さ（ease of learn）	容易。（プログラミング言語教育に用いられる）
		その他	プログラミングの概念を自然に反映し，かつ現実の計算機に能率的な実現ができるように選ばれた教育的言語である。概念言語として発表されたが，近年多くの処理系が作成されるようになった。

表5－16　標準Pascalの評価（２）

評価項目＼言語			標準Pascal
言語概念（language concepts）	拡張機能	リスト処理 （list processing）	new文，dispose文，ポインタ型データにより可。
		マクロ処理 （macro capability）	不可。
		デバッグ機能 （debugging feature）	言語には含まれていない。
		プリコンパイル機能 （〝include〃 etc）	不可。
		その他	実現されている処理系には，検死報告（post mortem dump）を巧妙かつコンパクトに行っていて便利なものも多い。
処理系（processor）		対話性 （interactiveness batch/interactive）	バッチ処理のみ。
		オブジェクトコード効率 （object code efficiency）	対アセンブラ比２～３倍。
		オブジェクトコード規模 （object code size）	対アセンブラ比２～３倍。
		処理系の規模 （compiler size）	２５０ＫＢ
		診断情報 （stmart message）	やや劣る。
		再入可能性 （reentrancy）	不可。
		ROM化 （direct execution m/c）	不可。
		その他	

表5－16　標準Pascalの評価（３）

評価項目＼言語		標準Pascal
言語の環境（environment）	移植性（portability）	大。
	他言語とのリンケージ（linkage）	不可。
	オペレーティングシステム（operating environment）	特定しない。
	その他	
その他	実績（popularity）	表5－14参照。
	将来性（promising or not）	大。
	当該言語の対象応用分野（どの分野に向くか）	内部処理一般。（入出力が弱い）
	その他	

5.3.8 UCSD Pascal

N. Wirth 等の作成したPascal Pコンパイラ（擬似命令コードを出力する）を母体とし，UCSD（米国加州立大サンディエゴ校）のK. L. Bowles等を中心にして，マイクロコンピュータ用に作られた一連のコンパイラとインタプリタ群である。

言語仕様は標準Pascalを基にし，それに実用的観点から機能の追加を行っている。しかしながら，Pascalの思想から外れたものではない。

擬似命令コードはPコードと呼ばれ，特定の機械のアーキテクチャに存在しない，抽象的機械の命令となっている。

UCSD Pascalのコンパイラは自分自身の言語で書かれている。この言語のコンパイラをある機械で必要とするとき，次の手順を実行することにより，ごく簡単に作成できる。

(1) Pコードのインタープリタをそのコンピュータの機械語で作成する。これはたかだか数キロバイトである。（Pascalで1000行足らず）

(2) 他機種で作成された，コンパイラを自分自身でコンパイルした，Pコードで表現されたコンパイラを用意する。

(3) (2)を(1)で動かして，UCSD Pascalコンパイラをコンパイルする。これをブートストラップといい，原理的には不必要であるが，この結果が(2)と同じであることを確認するために行う。

UCSD Pascalを動かすために，新たに作成すべきプログラムは上記(1)のインタープリタだけであることが，この言語システムの最大の特徴である。

Pコードを機械語とする計算機が考えられる。これによると，インタープリタによって実行せず，直接実行するので実行速度の点でかなり有利となると考えられる。これはすでに実現されている。Western Digital社は，LSI 11のマイクロプログラムのみを変更して実現し，WD／90という商品名で発売している。

表5－17　UCSD　Pascal言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・文字セットはASCII。 ・プログラムは6種のシラブル（名前，定数，キーワード，演算記号，区切り記号，注釈）からなる。 ・名前は英字で始まる英数字で，長さの制限はないが，高々8文字で区別する。 ・キーワードは45種で，すべて予約語である。
名標と属性	・標準Pascalの該当欄に同じ。
定　　数	・実定数，整定数，文字定数，ストリング
演算子の種類	・実数（加，減，乗，除，ベキ，符号反転） ・整数（加，減，乗，除，剰余，符号反転） ・論理（否定，和，積）　・集合（和，差，積，要素の存在） ・標準関数28種
制　御　文	・2分岐（IF ～ THEN ～，IF ～ THEN ～ ELSE ～） ・多分岐（CASE ～ OF ～；～；…；～ END） ・WHILE型繰返し（WHILE ～ DO ～） ・UNTIL型繰返し（REPEAT ～ UNTIL ～） ・FOR型繰返し（FOR　:= ～ TO ～ DO ～） ・GOTO　　・関数，手続きの非定常出口（EXIT）
プログラム構造	・標準Pascalの該当欄に同じ。 ・USES部によって抽象データ型の機能を持たせることが可能。 ・分割コンパイルが可能。
入　出　力	・順編成ファイルについて，順アクセスおよび直接アクセスが可能。 また，会話処理に適するタイミング設定も可能。 ・文字列のファイル（テキストファイル）についても編集入力，編集出力のための標準手続きが用意されている。

出典：SofTech Microsys ems：UCSD Pascal Users Manual 1980

表5－18　UCSDPascalの評価（1）

評価項目 ＼ 言語			UCSD Pascal
言語概念（language concepts）	構文	データ抽象化（abstract data type）	unitにより可。
		形変換（type conversion）	文脈により，整数から実数に自動変換される以外は陽に指定する。
		分割コンパイル（separate compilation）	可。
		制御構造（control structure）	条件分岐（if文, case文），繰り返し（for文, while文, repeat文）
		再帰性（recursion）	可。
		併行処理（concurrency multitasking）	不可。（バージョン4.0では可）
		例外処理（exception handling）	不可。（バージョン4.0では可）
		入出力（IO）	逐次ファイルの逐次アクセスおよびランダムアクセス。
		記述性（descriptiveness）	秀。
		構文の複雑性（complexity）	例外事項がほとんどなく簡単。
		習得の容易さ（ease of learn）	容易。
		その他	標準Pascalに機能強化を行ってある。機械独立な中間言語（Pコードという）へのコンパイルと，各種マイクロコンピュータ対応のインタープリタとに分離して，コンパイラの実現を容易にしている。

表5－18　UCSDPascalの評価（２）

評価項目＼言語			UCSD Pascal
言語概念（language concepts）	拡張機能	リスト処理（list processing）	new, wark, release 文とポインタ型データより可。
		マクロ処理（macro capability）	不可。
		デバッグ機能（debugging feature）	言語には含まれていない。
		プリコンパイル機能（"include" etc）	不可。
		その他	
処理系（processor）		対話性（interactiveness batch/interactive）	コンパイルは基本的にバッチで行われるが，処理系全体としてかなりの対話性を有する。
		オブジェクトコード効率（object code efficiency）	インタープリタにより実行するため低い。対アセンブラ比約5倍。
		オブジェクトコード規模（object code size）	対アセンブラ比1～1.5倍。
		処理系の規模（compiler size）	48KB
		診断情報（stmart message）	やや劣る。
		再入可能性（reentrancy）	不可。
		ROM化（direct execution m/c）	可。Western Digital社，Pascal Microengine あり）
		その他	

表5－18　ＵＣＳＤPascal の評価（３）

評価項目 ＼ 言語		UCSD Pascal
言語の環境（environment）	移植性（portability）	大。
	他言語とのリンケージ（linkage）	バージョン2.0でBASIC，FORTRAN，アセンブラとリンクできる。
	オペレーティングシステム（operating environment）	ＵＣＳＤPascal という名のＯＳないしモニタ。
	その他	
その他	実績（popularity）	表５－１４参照。
	将来性（promising or not）	大。
	当該言語の対象応用分野（どの分野に向くか）	内部処理が主。
	その他	

5.3.9 Ada

1975年に米国防総省は管轄下の計算機システム用の統一的新言語を制定するために，高水準言語作業委員会を設けた。この委員会のまとめた要求仕様書Steelmanに対して提出された予備設計を審査して，1979年に採択されたのがAda言語である。

Ada言語は1980年7月に米国防総省からレファレンスマニュアルが出版されており，これに基いて米国空軍および陸軍が現在コンパイラの作成を発注している状況である。

Ada言語仕様の特徴は以下の通りである。

(1) 基本データ型に有効範囲や，有効値の指定機能があり，データ型のチェックが厳密に行われる。

(2) package機能を用いることにより，抽象データ型を実現する。

(3) プログラムの階層的分割コンパイルにより，トップダウン的プログラム開発が容易になる。

(4) タスクの処理により，並行処理を記述できる。

(5) 例外事項の処理をユーザが自由に規定できる。

(6) ハードウェアに依存した部分を分離して，他の部分と独立に管理できる。

全体的にAda言語はプログラムの共通化を目標として，ハードウェアとの接点を明確化しようとしている。これが抽象データ型や分割コンパイルなどの機能を取り入れる理由となっている。

同時にAda言語は組込みシステムのプログラム作成を考えているので，実時間制御のための並行処理，例外処理機能を含み，メモリまたは速度についてのオプティマイズ機能を持つ。

このような意欲的な試みをAdaは実現しようとしている。これをどの程度のコンパイラで実現できるのか大いに興味がもたれている。

表5－19　Ada言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・文字は英大文字と数字および構文用の特殊文字22種。 英小文字は大文字に変換される。 ・識別子には英数字の他に，下線＿が使える。。 ・コメントは－－と改行で終了する。 ・コンパイラへの要求としてPragmaがある。 ・61種のキーワードがあり，予約語となっている。 ・4種の宣言文と，8種の制御文があり，単文と複文からなる。
識別子と属性	・変数等の記憶域はPackageで指定される。 ・データ型は列挙型，文字型，論理型，整数型，実数型，浮動小数点型，固定小数点型およびこれらのアレイ，レコードからなる。 ・ユーザの規定するデータ型には，値を制限するSubtype，既存の型の性質を引き継ぐDerived Typeとがある。
定　　数	・整定数は10進と2～16までの基底指定とがある。実定数は小数点で区別する。他に文字定数と文字列定数とがある。
演算子の種類	・算術演算子11種，論理演算子3種，比較演算子6種が定められている。 ・Package内で，抽象データ型の記法で，特定のデータ型に対する演算子を定義することができる。
制　御　文	・条件分岐はIf ～ Then ～ Elself ～ Else ～ Eed If で行なうか Case ～ Is When ～ →～ ; When Others →～ End Case; を用いる。 ・繰返しはFor ～ Loop ～ Eed Loop もしくはWhIlf ～ Loop ～ Eed ～ Eed Loopを用い，ExitまたはReturnによって抜け出せる。 ・Goto文がある。
プログラム構造	・プログラムは大きくはライブラリユニットとパッケージとタスクの集合になっている。 ・パッケージやタスクには，副プログラムや関数をトップダウン形式で含めることができる。 ・引数は入力，出力，入出力のいずれかの指定ができる。
入　出　力	・入出力として，ファイル用のINPUT－OUTPUT，Text－IOのパッケージが用意されている。 ・低レベルのI／Oはハードに依存したコードで与える。

出典：U．S．Department of Defense，Reference Manual for the Ada Programming Language（1980年7月）

表5－20　Adaの評価（１）

評価項目＼言語			Ada
言語概念（language concepts）	構文	データ抽象化（abstract data type）	packageによって基本的に実現されている。ただし公理系が与えられているわけではないので，厳密な意味のデータ抽象化に期待されるプログラム証明までは保証できない。
		形変換（type conversion）	数値，アレイ，derived typeでのみ可能。
		分割コンパイル（separate compilation）	階層的に可能
		制御構造（control structure）	Structured Programmingのif～then～else if～else，case～when，loop～exit，gotoの４要求と手続処理かつ制御に使われる。
		再帰性（recursion）	許されている。
		併行処理（concurrency multitasking）	multi－taskingにより実現されている。entry，acceptによる。
		例外処理（exception handling）	例外事項をユーザが宣言できる。システムはConstraia，Numeric，Select Storage，Taskingの各エラーを標準的に例外事項としている。もちろん，handleも書ける。例外処理を起させるraise文がある。
		入出力（ＩＯ）	ファイル用のINPUT－OUTPUT，Text－ＩＯの２種のパッケージが用意されている。低レベルのＩ／Ｏはパッケージが規定されていて，内容はシステム毎にコードで与える。
		記述性（descriptiveness）	良好。
		構文の複雑性（complexity）	かなり複雑。
		習得の容易き（ease of learn）	一般ユーザには困難であろうが，計算機科学等の基本的な事項をマスターした専門家には容易である。
		その他	

表5－20　Adaの評価（２）

評価項目 ＼ 言語			Ada
言語概念（language concepts）	拡張機能	リスト処理（list processing）	access type と allocate 文によって実現される。
		マクロ処理（macro capability）	literal と inline により実現される。
		デバッグ機能（debugging feature）	特にデバッグのための機能は備わっていないが，抽象データ型の機能（ package ）や constraint により容易に実現される。
		プリコンパイル機能（〝include〃 etc）	なし。
		その他	
処理系（processor）		対話性（interactiveness batch/interactive）	現在の所，議論されていない。
		オブジェクトコード効率（object code efficiency）	inline 機能があり，保証する方向でコンパイラが作られる予定。
		オブジェクトコード規模（object code size）	optimize の option として codesize を主眼にする機能がある。かなり考慮が払われる予定。
		処理系の規模（compiler size）	かなり大きくなるのではないかと危ぶまれている。
		診断情報（stmart message）	抽象データ型や分割コンパイルにより，豊富な診断情報を提供するコンパイラを作ることができる。
		再入可能性（reentrancy）	reentrant な task を書くことができる。
		ROM化（direct execution m/c）	共通データを別に定義できるので，コンパイラがROM化コードを生成することは可能である。
		その他	

表5－20　Adaの評価（３）

評価項目＼言語		Ada
言語の環境（environment）	移植性（portability）	Ada開発の主要な理由の一つなので，移植性は保証されるはずである。
	他言語とのリンケージ（linkage）	code部によって（主としてアセンブラ）リンクを取ることができる。
	オペレーティングシステム（operating environment）	現在仕様を作成中。
	その他	
その他	実績（popularity）	なし。
	将来性（promising or not）	米国防総省（DoD）後援により，将来性は極めて高い，と考えられている。
	当該言語の対象応用分野（どの分野に向くか）	制御を主とした工業分野が当面の対象応用分野であるが，実績によっては，より広い分野に適用されるものと考えられている。
	その他	

5.3.10 Concurrent Pascal （CONPAS）

CONPASは，N.Wirth 教授が開発したPascalをもとにこれをシステム記述用の言語としたSequential Pascal システムを開発したPer Brinch Hansen 教授等によって開発されたものである。

Sequential Pascal は，その名が示すように，順次処理のプログラミングを行うために開発されたプログラミングシステムであるが，CONPASは，オペレーティングシステムやリアルタイム処理で見られる併行処理の問題をプログラミングするために必要とされる機能を,process，monitor や class といった型で備えている。

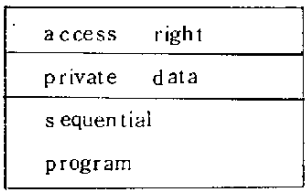
CONPASは，同教授が南カルフォルニア大学に在職中にAl. Hartman 等と共に開発したシステムであるが，このシステムの開発に当っては同教授がかつてＲＣ４０００という計算機のオペレーティングシステム の開発に従事していた時に得た経験がシステムの思想に大きく影響している。

ＣＯＮＰＡＳは，当初ＰＤＰ－１１／４５の上で開発されたが，ＣＯＮＰＡＳシステムそれ自体は，コロラド大学のPascal Distribution Centerの手に渡り，南カルフォルニア大学ばかりでなく全世界で利用できるようになっている。このために，ＣＯＮＰＡＳを利用する大学，研究機関が多くなったばかりでなく，例えばドイツの フィリップス GmbH データシステム社では，オペレーティング システムの開発に当って，ＣＯＮＰＡＳを使用したし，また米国のEnertek社では，8080 マイクロプロセッサのために， Micro ＣＯＮＰＡＳを開発し，これによってリアルタイム処理のプログラムの開発を行っていると報じられている。

Concurrent Pascal システムの思想の１つは，併行処理のプログラムは，いくつかの順次処理から構成されるという考えに立っている。

Concurrent Pascal に於ける processは，順次処理の単位であって，１つの rocessは，access right， private data，sequential program といった部分から構成される。

process の中に定義された private data は，そのprocess の中で処理することができる。



Process の構造

一方，monitor は，access right，shared data sequential operation initialization から構成されている。

ＣＯＮＰＡＳには，queue というモニタだけで利用することができるデータ型が用意されている。

shared data は，例えばバッファであって，ここにあるデータは，各 process からは直接，ここにある内容を取扱うことができなく，monitor を介して処理する。

monitor は，各 rocess を同期をとるために必要な手続きと，初期化を行うための手順を持つ。

access right
synchronization
operation
initialization

Monitor の構成

class は，ＳＩＭＵＬＡというプログラミング言語―実はシミュレーション用言語―で採用されたclass という疑似併行処理の考え方に基づいている。これは，ある process の中で，併行処理，（と云っても置ざり制御）を行う部分のプログラミングを行うのに利用する。

ＣＯＮＰＡＳの第２の特徴は，抽象データ型の採用であろう。

これまでのプログラミング言語では，ＣＯＮＰＡＳが有しているprocess，monitor やclass といった機能は，オペレーション即ち，手続きとして処理するか，あるいは，共通のデータとして処理してきた。しかし，ＣＯＮＰＡＳでは，これらのものを（処理された）データとしている。これまでのプログラミング言語を使用する場合には，どのような構造をしたデータにどのような処理を行ったために作られたどのような構造のデータを使用するといったデータの構造を中心にプログラミングが行われてきた。

ＣＯＮＰＡＳＳでは，抽象データ型の採用によって，どのようなデータを処理するかということが中心となって，データの構造がどのようになっているかということは問題ではなくなった。

このために，ある process で例えばデータバッファの構造を一重バッファから二重バッファに変更したとしてもこの変更が他の process に影響を与えることがないので，信頼性の高いプログラムを作ることができる。

次のプログラム例はＣＯＮＰＡＳによるカード・ツー・プリントのプログラムを説明しやすいように並びかえている。

```
＊  INITALIZE  ＊
      var    inbuffer,  outbuffer : linebuffer;
             reader  :  cardprocess ;
             copier  :  copyprocess ;
             printer :  crintprocess ;
             begin
             init  inbuffer,  outbuffer ;
                   reader ( inbuffer ) ;
                   copier ( inbuffer,  outbuffer ) ;
                   printer ( outbuffer ) ;
             end
```

メインプログラムでは，card process， copy process, print process を起動させている。

＊ CARD PROCESS

```
type card process = process ( buffer : linebuffer );
var param : ioparam ; text, error : line ;
    charno : integer ;
    begin for charno := 1 to 80 do error ( . charno . ) := '?' ;
    param.operation := input;
    with param do
    cycle
    repeat io ( text, param, card device )
    until status <> intervation ;
    if status <> complele then text := error;
    buffer.send ( text );
    end ;
    end ;
```

上のプログラム例は，card process を定義したものである。

これは，process であり引数を１つとることが分る。

処理としては，エラーメッセージのために“？”マークを８０字うめ，カードリーダのオペレーションを行う。

データが読めたら linebuffer の中にある send という手続きを行う。

メイン・プログラムの中では，card process を手続きとしてではなく，抽象的なデータ型として，とらえている。

同じように，このプロセスの中では，line buffer の中にある send という手続きをも抽象的にとらえている。

従って，このプロセスの中では，send という手続きがどのような処理を行っているかということは問題にならない。

一方，linebuffer というモニタは，次のようにプログラミングされている。

＊ Linebuffer ＊

```
type    line buffer = monitor
var     content : line ;   full : boolean ;
        sender ,  receiver : queue ;
        procedure entry send ( text : line );
        begin
        if full then delay ( sender )
        content := text ;  full := true ;
        continue ( receiver );
        end ;
```

```
procedure entry receive ( var : text : line );
begin
if not full then delay ( receiver );
text := content; full := false;
continue ( sender );
end ;
begin full := false end;
```

linebuffer というモニタでは，共有のデータとプロセスの同期を行うために，send receive という手続きと，full という変数を初期化するルーチンを持っている。

プロセスの同期をコントロールするために，sender と receiver という待ち合わせを持っている。card process に於いて，カードリーダよりデータが読まれると，モニタの中の send という手続きによって同期がとられる。

先に送ったものがまだ残っている(full)ときには，要求があったプロセスを receiver というqueue に登録して，空くまで待つ。この間のコントロールは，モニタがコントロールする。空きが出来る(receive という手続きの中のcontinue (sender))と，データ受けとって，full を true にし，receiveという手続きを(continue (receiver)) 再開させる。

copyprocess は，次のように定義されている。

```
type copyprocess = ( inbuffer, outbuffer : linebuffer );
var
consumer : filemarker; text : line;
begin
init consumer ( out buffer );
with buffer, consumer do
cycle receive ( text ); write ( text ) end;
end ;
```

このプロセスでは，データが渡されたら，with consumer do によって，comsumer の中にあるwriteという手続きを実行するように指示しているconsumer では，filemarker というデータ型(実は filemarker というclass)を指定しているので，filemakerの中にあるwriteという手続きを実行することになる。

以上でＣＯＮＰＡＳが持っている併行処理の機能を眺めてきたが，ＣＯＮＰＡＳでは，併行処理をそれぞれの順次処理(cycle ～ end) に置き変えて処理することができるという特徴を持っている。

ＣＯＮＰＡＳコンパイラは，プログラム全体にわたってチェックしなくても，抽象データ型の参照をチェックすることと，class , monitor process の中で，データの参照をチェックすれば良いということになる。

表5－21　Concurrent Pascal 言語仕様

	仕　様　・　機　能　の　概　略
データの型	・文字，整数，実数，論理，テキスト ポインタ，配列，レコード，セット，エニュメレーション ユーザ定義，システム（ process，monitor，class ）
変数の属性	データの有効範囲を指定できる。 値に番号付けを行うことができる。
演算の種類	算術，関係，論理和，論理積，単項＋，－，集合，否定，16種
制　御　文	if，case，for，while，repeat，with，cycle
同　　期	Queue，：empty，delay，continue
起　　動	Start，stop，wait，realtime，attribute
プログラム構造	1　ブロック構造の言語である。 2.　併行処理を順次処理の集まりとしてとらえている。
入　出　力	IO　process のみ
特　　長	・オペレーティング・システムの記述に使用される。 ・seguential Pascal と対で使用される。 ・併行処理を，monitor，process，class という抽象データ型でとらえている。

出典；B.Hansen，A.　The　Architecture　of　Concurrent　Program　Prentice－Hall 1977年
Hansen & A. Har tman，Concurrent Pascal Compiler for mini－computer Lecture
B．otes in Computer Science Springer－Verlag 1977年

表5－22　Concurrent Pascalの評価（1）

評価項目＼言語			Concurrent Pascal
言語概念（language concepts）	構文	データ抽象化（abstract data type）	monitor, Class, process, において，データの抽象化が計られている。
		形変換（type conversion）	無し。
		分割コンパイル（separate compilation）	不可能。
		制御構造（control structure）	if then else , while do , repeat until ,
		再帰性（recursion）	可能。
		併行処理（concurrency multitasking）	可能。
		例外処理（exception handling）	無し。
		入出力（IO）	prefixによってオペレーティングシステムと連動する。 と連
		記述性（descriptiveness）	Pascalと同じように読みやすい。
		構文の複雑性（complexity）	情報の隠ぺいが行われているので，今までの言語になれてきた人にとっては判りにくい点が多いであろう。
		習得の容易き（ease of learn）	Pascalに準ずる。
		その他	開発者の手によってConcurrent Pascalを利用したプログラミング例が発表になっているので，非常に参考になる。

表5－22　Concurrent Pascalの評価（２）

評価項目 ＼ 言語			Concurren Pascal
言語概念（language concepts）	拡張機能	リスト処理（list processing）	無し。
		マクロ処理（macro capability）	無し。
		デバッグ機能（debugging feature）	無し。
		プリコンパイル機能（〝include〃 etc）	無し。
		その他	
処理系（processor）		対話性（interactiveness batch/interactive）	バッチシステムとして作られている。
		オブジェクトコード効率（object code efficiency）	PDP－11の場合にはオブジェクトプログラムがスレッデッドコーディングになっている。
		オブジェクトコード規模（object code size）	インタープリタに相当するカーネルの部分は3Kワードと非常に小さい。
		処理系の規模（compiler size）	PDP－11においては7パス方式を採用している。全体としては55Kワードであるが，各パスとも16Kワードで動くように作られている。
		診断情報（smart message）	余り用意されていない。
		再入可能性（reentrancy）	無し。
		ROM化（direct execution m/c）	無し。
		その他	

表5－22　Concurrent Pascalの評価（３）

評価項目＼言語		Concurrent Pascal
言語の環境（environment）	移植性（portability）	マシンディペンドな部分が少ないので，ソースプログラムは移しやすい。
	他言語とのリンケージ（linkage）	無し。
	オペレーティングシステム（operating environment）	SOLO。 prefixによってオペレーディグ・システムと連動するようになっている。
	その他	Sequential Pascal，Concurrent Pascal，SOLOとも，それぞれの言語を組合せて記述しているので，システムの移植は容易である。
その他	実績（popularity）	米国のEenrtec社では8080用のMICROCONPASでリアルタイム処理のプログラムを開発している。
	将来性（promising or not）	Modula IIが出現するまでは，併行処理にはConcurrent Pascalが使用されてきたが，Modulaの出現によって今後どうなるか不明。
	当該言語の対象応用分野（どの分野に向くか）	リアルタイム処理，オペレーティングシステム　コンパイラの記述。
	その他	

5.3.11 Modula Ⅱ

Modula Ⅱは，Pascalシステムの生みの親であるN. Wirth教授に開発されたミニコン向けのシステム記述言語である。

同教授は，１９６０年代にＩＢＭ３６０のために，システム記述用の言語として，ＰＬ３６０を開発したが，この言語は，余りにもＩＢＭ３６０寄りであった。

このＰＬ３６０と変わって，Modula は，Pascal システムをベースにして，今までアセンブリ言語で行っていたプログラムをこのModulaで記述することができるようにしている。

Modulaは，multi－programming やリアルタイム処理,ひいては入出力装置のコントロールプログラムまで記述することができるようになっているが，特に併行処理については,Concurrent Pascal の影響を受けている。

最初のModula システムは，Pascal で書かれ，ＣＤＣ６６００によってコンパイルし，これによって，ＰＤＰ１１の命令語を作るというクロスコンパイリングの方式がとられた。

Modula は，ＥＴＨで開発したＬＳＩ１１をベースにしたＣＡＩシステムである XS－0システムの開発に使用されている。

Concurrent Pascal と Modula は，両者とも併行処理を記述することができるということから両者とも比較の対象となる。

例えば，Hansen 教授の開発したＳＯＬＯオペレーティングシステムは，Concurrent Pascal(4 %)，Sequential Pascal (92%) とアセンブリ言語(4 %)で書かれている。

これに対して，Modula は，オペレーティングシステムを構成しているあらゆる要素をも，Modula で記述できるようにしようとしている。このため，Modula には，Concurrent Pascalに見られない機能が加わっている。

Modula は，標準Pascal をベースにして出発し，これよりシステムの記述に不必要であると考えられる機能を削除している。

次のような機能が削除された。

① pointer ／ dynamic storage allocation
② Input ／ output
③ file manipulation
④ variant record
⑤ real arithmetic
⑥ go to statement
⑦ for statement
⑧ set , type , subrange Type （Modula Ⅱ で復活 ）

これに代わってmodule , process , system という考え方が導入された。moduleは，Concurrent Pascal のclassに相当するものである

この module は，併行処理を行うことができる単位となるが，またこれは独立にコンパイル

を行う単位となる。

Modula では，Concurrent Pascal と同じように情報の隠ぺいを行うために import，export という機能が用意されている。

Concurent Pascal では，次のような記述は，

```
tipe c = class
var x, y : c ;
procedure p ;
  begin ……x……y……end p;
procedure q ;
  begin ……x……y……end q
end
```

これに対して，Modula では，

```
module M ;
  import t ;
  export p, q, c ;
  tipe c = record x, y : t end ;
  procedure p ( c : c ) ;
  begin ……c . x……c . y……end p ;
  procedure q ( c : c ) ;
  begin ……c . x……c . y……end q
end
```

Modula では，module の組込みを前提としているので，外部より export した場合に識別子の綴りが重なるといったことが起きる。

この重複を避けるために，外部に送るものに対しては，それぞれのモジュールで“export qualifid 変数名”による定義および外部より受け取るものについては“from モジュール名 import 変数名”といった定義がある。

Modula では，通常の module の外に definition module と implementation module の対を使用することができる。

即ち，definition module では，export する object について定義する。

例えば，

```
definition module io ;
export put, get ;
procedure put ( c : ch ) ;
procedure get ( c : ch ) ;
end io ;
```

これに対して，Implementation module では，definition module にある定義について処理手順を記述する。

例えば，

```
implementation module io
var inbuf, outbuf : array [1 … size] of char;
procedure put ( c : char ) ;
begin ……        …… end put ;
procedure get (  );
begin ……        …… end get ;
begin        初期化
end io
```

前述の定義は，メインプログラムでは，次のように利用できる。

```
module main ;
from import put, get,
var ch : char ;
begin
     …… put ( ch ),  …… ch : = get (    )
end main
```

Modula は，例えば使用するオペレーティング・システムなどのシステムに依存する部分をもプログラムすることができるようにするために“system”を導入している。

例えば，

```
MODULE  STORAGE;
FROM  SYSTEM  IMPORT  ADDRESS;
EXPORT  NEW;
VAR  LASTDR : ADDRESS;
PROCEDURE  NEW ( VAR A : ADDRESS; SIZE : CARDINAL );
BEGIN  A : = LASTDR ;
          LASTADR : = LASTADR + SIZE
END  NEW;
```

Modula では，ローレベルのプログラミングを行うために，word，bitset，cardinal といったデータ型の導入や定数を１０進数以外に，１６進数，８進数で記述することができるとか，また，変数を特定のアドレスに割当てるといったことが可能である。

特に，変数を特定のアドレスに割当てることができると，ＰＤＰ－１１やマイコンに見られるメモリマップトＩＯのコントロールプログラムを書く時に非常に便利であろう。

Modula では，Concurrent Programming を行うために，次のような標準的な手続きを用意している。

```
procedure  newprocess (p : proc; a:address ;
                              n : cardinal ; var p1 : process);
```

p ： プロセスとなる手続き名

a ： プロセスのワークスペースのスペースアドレス

n ： ワークスペースのアドレス

```
procedure  transfer (var p1, p2 : process)
```

P1プロセスを止め，P2プロセスを行う。

```
procedure  iotransfer(var p1, p2 : process,
                                    va : cardinal),
```

P1プロセスを止め，Vaで指定した割込み処理を行ったあとP2プロセスを再開する。

表5－23　Modula II 言語仕様

	仕　様　・　機　能　の　概　略
データ型	文字，整数，実数，論理，ビット，カーディナル，ポインタ，配列，レコード セット，エニュメレーション，ユーザ定義
変数の属性	データの有効範囲を指定することができる。 値に番号付けを行うことができる。 アドレスを与えることができる。
演算の種類	算術，関係，論理和，論理積，否定，単項＋，－，集合
制　御　文	if － then － end，　if － then － else － end， if － then － elseif － then － else － end， case － end，　while － do － end，　repeat － until － for － to － by － do － end，　loop － end， with － do － end，　exit，　return
同　　期	wait, send, a waited
プログラム構造	1. ブロック構造を採用している。 2. module という考え方を採用している。 interface module, device module, process とがある。
入　出　力	Modula で記述する。
特　　徴	① アセンブリ言語の代わりに利用することに重点を置いている。 ② 分割コンパイルが可能であるようにするためと，抽象データ型の採用により，変数を他のmodule からでなく，system からimport することができる また他のmodule に対して export することができる。 ③ メモリアドレスを変数として使用することができる。

出典：N.Wirth，Modula： a Language for Modular Programming Software practice experience 1977年7月

N.Wirth，Modula － II Tech, report 39，Institute für Informatik ETH，1980年

表5－24　Modula Ⅱの評価（1）

評価項目 ＼ 言語			Modula Ⅱ
言語概念（language concepts）	構文	データ抽象化（abstract data type）	definition, implementationの外にも，データの抽象化が図られている。
		形変換（type conversion）	無し。
		分割コンパイル（separate compilation）	可能。
		制御構造（control structure）	case, while, do, repeat until, with, do end, for end
		再帰性（recursion）	有り。
		併行処理（concurrency multitasking）	可能。
		例外処理（exception handling）	可能。
		入出力（IO）	無し。
		記述性（descriptiveness）	Pascalと同じように読みやすい。IOコントロールプログラムまで記述することができる。
		構文の複雑性（complexity）	情報の隠ぺいが行われているので，今までの言語になれてきた人にとっては，判りにくい点が多いと思われる。
		習得の容易さ（ease of learn）	プログラミングの方法論がきちんとすると，Concurrent Pascal同様にプログラミングしやすくなる。
		その他	Concurrent Pascalと同様に，開発者自身による応用プログラムが発表になっているのでModulaの使い方で参考になる点が多い。

表5－24　Modula Ⅱの評価（２）

評価項目 ＼ 言語			Modula Ⅱ
言語概念（language concepts）	拡張機能	リスト処理（list processing）	無し。
		マクロ処理（macro capability）	無し。
		デバッグ機能（debugging feature）	無し。
		プリコンパイル機能（〝include〟etc）	無し。
		その他	SYSTEMによってオペレーティングシステムが持っている値を取扱うことができる。
処理系（processor）		対話性（interactiveness batch/interactive）	バッチ用として開発されている。
		オブジェクトコード効率（object code efficiency）	言語自体をアセンブラの代用として開発している。PDP－11の場合にはオブジェクトプログラムの効率は良い。
		オブジェクトコード規模（object code size）	
		処理系の規模（compiler size）	
		診断情報（smart messge）	言語自体が正当性のチェックを行いやすいように作られている。
		再入可能性（reentrancy）	無し。
		ROM化（direct execution m/c）	無し。
		その他	

表5－24　ModulaⅡの評価（3）

評価項目＼言語		Modula Ⅱ
言語の環境（environment）	移植性（portability）	ハードウェアディペンデントな部分はsystemで切り離しているo ＩＯコントロールプログラムをもModulaで記述することができるので，プログラムの移植性は他のシステムで記述したものよりも多い。
	他言語とのリンケージ（linkage）	無し。
	オペレーティングシステム（operating environment 例えばUnixにおけるC）	無し。
	その他	
その他	実績（popularity）	ETH，英国のヨーク大学でPDP－11のModulaを開発している。
	将来性（promising or not）	
	当該言語の対象応用分野（どの分野に向くか）	リアルタイム処理，オペレーティングシステムの記述など。
	その他	

5.3.12 ＦＯＲＴＨ

ＦＯＲＴＨは1960年代にCharles H. Mooreによって考案されたプログラミングシステムであり，今日のＦＯＲＴＨと呼ばれるシステムは，１９７１年にNRAO（Ｎａｔｉｎａｌ Ｒａｄｉｏ Ａｓｔｒｏｎｏｍｙ Ｏｂｓｅｒｖａｔｏｒｙ）でHoneywell Ｈ３１６（16ビット ミニコンピュータ）のために開発された。

このシステムは，電波望遠鏡のデータ収集システムのプログラミングに使用された。

1974年頃になると，Kitt peakにある同天文台において，ＰＤＰ１１を使用した４タスクまで処理できるＦＯＲＴＨが誕生した。

1974年には，ミニコンピュータ用のMiniＦＯＲＴＨが誕生し，これがPoly ＦＯＲＴＨに発展した。

1976年には，マイコン用のMicro ＦＯＲＴＨが設定され，現在ＦＯＲＴＨ80が提案されている。

次に，ＦＯＲＴＨシステムは，スレッデッドコーディング（Threaded Coding）を採用している点にも大きな特長がある。

即ち，ＦＯＲＴＨシステムでは，ディクショナリィを持ち，すべての命令は，このディクショナリィにあるワードに対するポインタとなっている。

ＦＯＲＴＨシステムでは，システムとして持たなければならないワードが決められているが，それ以外のワードは，ユーザが，すでにディクショナリィに登録されているワードを利用して新たに定義することができる。

ＦＯＲＴＨシステムは，基本的には，16ビットの固定小数点データを取扱うが，例えば浮動小数点方式の加算が必要になった時には既にディクショナリィに登録されているワードを使用して処理手順を定義し，これに"ＦＡＤ（名前の長さはシステムによって変わるが４文字～６文字位）という名前を付けてディクショナリィに登録すれば，以後"ＦＡＤ"という名前で浮動小数点の加算を行うことができる。

ＦＯＲＴＨは，対話型のシステムであるので，利用者が，利用時に，自分が必要とするワードを定義して利用することができるが，この定義したワードは１回限りの使用で棄ててもよいし，また，システムの中に組み込んで，ＦＯＲＴＨシステムを拡張することも可能である。

ＦＯＲＴＨシステムは，スタックを中心として処理するシステムであるので，プログラムの記述は逆ポーランド方式を採用する。

即ち，キーボード（あるいはバーチャルメモリ）より入力されたデータは，そのデータが数字であれば，先に定義されている基数に従って，内部表現の数値に変えられ，スタックの上部につまれる。

入力データが文字である場合には，ディクショナリィに登録されているワードを新しく登録されているワードから古くから登録されているワードに向って探していく。

入力された文字と同じ綴りのワードが見つかると，このワードの定義にしたがって処理を行

い結果をスタックの上部にもどす。

また，新しいワードの定義の場合には，定義の内容をコンパイルし，新しい定義とその名前をディクショナリィに登録する。

ＦＯＲＴＨでは，３＋（５×６）は，次のようにプログラミングする。

３　５　６　＊　＋

またある数を２乗するという処理が多くなった場合には，ＳＱＲを次のように定義する。

’ＳＱＲ　：　ＤＵＰ　＊

ある数の２乗を計算するといった場合には，次のようにプログラミングする。

３　ＳＱＲ

これによって３を２乗した値である９がコンソールに出力される。

ＦＯＲＴＨシステムでは，次の１１個のグループの命令を持っている。

① 四 則 演 算
② 入 出 力
③ 基 数 変 換
④ 論 理 演 算
⑤ 比　　較
⑥ ス タ ッ ク の 取 扱 い
⑦ メ モ リ ア ク セ ス
⑧ 定 数 の 設 定
⑨ 領 域 の 確 保
⑩ プログラム制御　BEGIN　END，IF　ELSE　THEN，DO　LOOPなど
⑪ ワ ー ド の 定 義

ＦＯＲＴＨプログラムは，非常に簡略に書けるという利点を持っているが，これをうら返すと，出来上ったプログラムは非常に読みにくいという欠点がある。

しかし，ＦＯＲＴＨシステムの核は，数キロバイトと他のプログラミングシステムと比べて非常に小さくて済むという利点を持っている。

このために，ILLIAC　IVを始めとして８ビットマイクロコンピュータに至る数多くのシステムにおいてＦＯＲＴＨシステムを利用することができるようになっている。

ＦＯＲＴＨシステムは，ユーザスタック，リターンスタック，ハードウェウスタックという３つのレベルのスタックが用意されているので，タスクを代えることは，ＦＯＲＴＨでは，スタックを代えることになるのでこの方式を採用したマルチタスク処理を行うこともできる。

ＦＯＲＴＨシステムは，利用者向けの良いマニュアルがないという欠点があるにも拘らず，年々，利用者が多くなり，採用者の団体として，米国にＦＩＧ（FORTH　INTERESTEP GーROUP），我が国にはＦＩＧ　ＪＡＰＡＮが，そして欧州には，ＥＦＵＧ（European FORTH Users Group）がある。

この外，International Astronical Union がＦＯＲＴＨを採用することに決定したので今後ＦＯＲＴＨの利用者は増加することであろう。

表5－25　ＦＯＲＴＨ言語仕様

	仕　様　・　機　能　の　概　略
データ型	整　数
変数の属性	な　し。
演算の種類	演算，基数変換，論理和，論理積，関係 スタック操作など基本的には２７種
制　御　文	－ IF － ELSE － THEN 文，　DO － LOOP 文， DO － I LOOP 文，　BEGIN － END，　LEAVE
プログラム構造	スタックを利用した対話型。
入　出　力	基本的には，コンソール出力のみ。
特　　徴	FORTHは，自己増殖型の言語であるために，他の手順型の言語と同列で論じることは出来ない。

出典：E. D.Rather，L. Brody ： USING　FORTH，FORTH Inc．1979年４月

表5－26　FORTHの評価（1）

評価項目＼言語			FORTH
言語概念（language concepts）	構文	データ抽象化（abstract data type）	無し。
		形変換（type conversion）	BASE指定によってどのモードに変えることもできる。
		分割コンパイル（separate compilation）	不可能。
		制御構造（control structure）	BEGIN END, IF ELSE THEN, DO LOOP
		再帰性（recursion）	不可能ではないが細工を要する。
		併行処理（concurrency multitasking）	スタックを中心としているので，システムを修正するだけで，Multi－User処理を行うことができる。
		例外処理（exception handling）	無し。
		入出力（IO）	無し。
		記述性（descriptiveness）	読みにくい。
		構文の複雑性（complexity）	
		習得の容易き（ease of learn）	逆ポーランド記述を使用している。スタック中心であることから慣れるまでに時間がかかるが，慣れると容易に使いこなせる。
		その他	

表5－26　FORTHの評価（２）

評価項目 ＼ 言語			FORTH
言語概念（language concepts）	拡張機能	リスト処理（list processing）	無し。
		マクロ処理（macro capability）	通常の意味でのマクロ処理はないが，ワードの定義によって機能を追加することができる。
		デバッグ機能（debugging feature）	無し。
		プリコンパイル機能（〝include〃 etc）	無し。
		その他	
処理系（processor）		対話性（interactiveness batch/interactive）	対話型のシステムとなっている。
		オブジェクトコード効率（object code efficiency）	オブジェクトプログラムはほとんどポインタとなるので比較的効率は良い。
		オブジェクトコード規模（object code size）	ディクショナリィの大きさ＋ユーザプログラムの大きさが，オブジェクトプログラムの大きさとなるので不定。
		処理系の規模（compiler size）	ミニマム　2～3Kバイト，通常のシステムにおいて6K～8Kバイト。
		診断情報（smart message）	無し。
		再入可能性（reentrancy）	無し。
		ROM化（direct execution m/c）	可能なシステムもある。（例えばPoly FORTH）　FORTHマシンがいくつか作られている。
		その他	

表5－26　FORTHの評価（３）

評価項目＼言語		FORTH
言語の環境（environment）	移植性（portability）	FORTHプログラムの移植性は非常に強い。
	他言語とのリンケージ（linkage）	アセンブラが組込まれている。
	オペレーティングシステム（operating environment）	特に指定はない。
	その他	
その他	実績（popularity）	ILLIACⅣから８ビットマイコンまで，数多くのシステムが開発されている。
	将来性（promising or not）	米国にFIG，我が国にFIG JAPAN，欧州にEFUGがある。今後利用者が多くなるであろう。
	当該言語の対象応用分野（どの分野に向くか）	計測，データ収集，データ解析，データベースなど。
	その他	

5.3.13 ＬＩＳＰ

ＬＩＳＰは１９６０年代に当時米国のマサチューセッツ工科大学にいたマッカーシ（Ｊ．McーCarthy）によって開発された。

現在ＬＩＳＰは１９９０年代以降のシステム記述言語として注目を集め始めているが，ＬＩＳＰの対象応用分野は開発当初から一貫して記号処理，人工知能と呼ばれる分野である。

ＬＩＳＰの言語概念は，ラムダ計算法という論理体系に基づいており，数学的な整合性を含んでいる。プログラミング言語の歴史の中でＬＩＳＰはユニークな位置を占める。処理系はインタープリタとコンパイラおよび各種のユーティリティから成り立っている。

特徴としては以下の諸項が挙げられる。

(1) 手続きが関数の集合で規定される。一種の関数型言語である。

(2) データはリストとアトム（数値，ストリング，識別子）から成る。

(3) リスト・データの領域管理はシステムが行う。このためガーベジ・コレクタが付随している。

(4) コンパイラはインタープリタの補助として，出来上ったモジュール（関数集合）を一括して機械語に落すために用いられる。

(5) プログラムはデータ同様リストの形式で表現される。従ってプログラム（関数本体）をデータ同様自由に変更できる。また，データをプログラムとして実行する関数ＥＶＡＬが存在する。

ＬＩＳＰは記憶容量や処理速度の点で問題があるとされてきたが，最近のハードウェア技術の進歩により，余り大きな障害とは考えられなくなった。ＬＩＳＰの持つ「人間的」な側面，すなわち知識をプログラム化し，ユーザにとって使い易いシステムを作る機能がこれからの計算機システムにとって無視しえない要素になると考えられている。

表５－27　ＬＩＳＰ言語仕様

	仕　様　・　機　能　の　概　略
プログラム要素	・文字は任意。（ ASCIIの他カナ，漢字を使うこともある）
	・識別子には特に制限は無い。（特殊記号の前にはエスケープ文字をつける）
	・基本要素は'（'と'）'でつくられるリストである。
	・プログラム単位は（＜関数＞　＜引数＞　…）というリストである。
識別子と属性	・識別子には属性リストが付随している。この属性リストに識別子についての情報がすべて記述されている。
	・変数をどう処理するかは，処理系に任されている。
	・データ型には，リストの他に数値とストリングがある。
	・アレイは任意次数，任意型が許されている。
定　　数	・整定数（底は２～10進任意），浮動小数点数，文字列
演　算　子	・演算子はすべて関数手続き，例（ PLUS　… ），で表わされる。 infix operator が無くて，prefix operatorだけと考えても良い。
制　御　文	・関数の形で制御が与えられる。 PROG ……連続実行（通常のブロック文に相当）， COND …… IF　…　THEN … ELSEIF … ELSEに相当， CATCH … THROW … GLOBAL EXIT MAP … 繰り返し文に相当
プログラム構造	・プログラムは１つ以上の関数の集まりである。
	・関数間の情報交換は引数，外部変数，関数の値による。引数の引渡しは，値，名前の他に，パターンの場合もある。
	・関数に型宣言は不要。
	・すべての関数は再帰呼び出し可能。
入　出　力	・リストに対する標準入出力，READ，PRINTをもつ。
	・その他は，各システムで用意する。

出典：Winston　Horn，"ＬＩＳＰ"，Addison Wesley （ 1981 ）

表5－28　ＬＩＳＰの評価（１）

評価項目＼言語			LISP
言語概念（language concepts）	構文	データ抽象化（abstract data type）	無し。
		形変換（type conversion）	自動的。形変換関数もある。
		分割コンパイル（separate compilation）	無し。
		制御構造（control structure）	関数呼び出し。（単純だが強力）応用面からは，引数渡しの他にパターンによる呼び出しや，データベースとリンクした呼び出しなどがある。
		再帰性（recursion）	本質的。
		併行処理（concurrency multitasking）	元来のLISPには存在しないが，人工知能など応用分野の要請で採用されつつある。LISPではcontext mechanismと呼ばれデータベースの回復や変更を含んだ手続きの並行処理が行なわれる。
		例外処理（exception handling）	元来のLISPには存在しないが，人工知能等応用分野の要請で採用されつつある。LISPではdaemonと呼ぶことが多く，データ操作により起動される。
		入出力（ＩＯ）	共通に規定されているのはリストの入出力（read，print）だけである。
		記述性（descriptiveness）	評価が分れる。記号を主体とするので記述性が高いとする人々と，括弧が多くて記述性が低いと見る人々がいる。計算機プログラム（pretty printer）を用いると記述性が高まる。
		構文の複雑性（complexity）	構文は極めて単純（関数，引数…）の形式だけである。理論的にはλ一計算法に従う。
		習得の容易き（ease of learn）	習得は極めて容易。特に対話的に学習すれば容易である。
		その他	標準規約がなく，処理系によって様相が変わる。ただしLISPプログラムは，一週間程度の労力で他システムに移植することが可能である。

表5－28　ＬＩＳＰの評価（２）

評価項目 ＼ 言語			LISP
言語概念（language concepts）	拡張機能	リスト処理 （list processing）	リスト処理を中心とする。
		マクロ処理 （macro capability）	マクロ展開機能が標準的に装備されつつある。
		デバッグ機能 （debugging feature）	一般的に豊富である。理由はプログラム自体がデータとして扱えること，制御が単純であること，通常対話的に用いることによる。
		プリコンパイル機能 （〝include〞 etc）	標準的に装備されつつある。プログラム ソースの形式と，コンパイルしたオブジェクトの形式にかかわる。
		その他	ＬＩＳＰの拡張は容易である。言語の特徴ともなっている。
処理系（processor）		対話性 （interactiveness batch/interactive）	本質的に対話的に用いられる。
		オブジェクトコード効率 （object code efficiency）	コンパイラに大きく依存する。ＭＩＴで開発した数値計算用コンパイラはメーカ提供のFORTRANコンパイラにまさっていた。
		オブジェクトコード規模 （object code size）	一般に大きなプログラムが多い。１メガバイトでは不足で 20～30 メガバイトが一応の線だと見られている。この量もコンパイラに大きく依存する。Xerox PARCでは，命令を１バイトに圧縮するＬＩＳＰコンパイラが作られている。
		処理系の規模 （compiler size）	一般にＬＩＳＰ自体で書かれているので大きい。
		診断情報 （smart message）	コンパイラに依存するが，通常ＬＩＳＰ自体で書かれているので必要なだけ情報を出すことができる。コンパイル以前にインタープリタにより動作確認をするので不必要だという意見もある。
		再入可能性 （reentrancy）	プログラムは一般に再入可能。純粋に関数形式のみを用いたプログラムは数学的に再入可能性が保証される。
		ROM化 （direct execution m/c）	プログラムは一般にROM化可能である。
		その他	処理系は一般にインタープリタが用いられる。

表5－28　ＬＩＳＰの評価（３）

評価項目＼言語		LISP
言語の環境（environment）	移植性（portability）	優れる。方言が多いので"そのまま"移植はできないが，高々一週間程度の作業で移植可能。特に移植用プログラムを作成できることが強力。
	他言語とのリンケージ（linkage）	通常余り考慮されていない。
	オペレーティングシステム（operating environment）	オペレーティングシステムの機能（主として対話的環境下では）を存分に利用することがLISPの必要条件となっている。
	その他	対話的環境が不可欠。
その他	実績（popularity）	記号処理，人工知能の分野では圧倒的な実績をもつ。
	将来性（promising or not）	極めて有望。従来障害となっていたメモリ・コスト，会話型端末の価格低下が大きく寄与している。"人間的な計算機システムの基本言語として広まる可能性が大きい。
	当該言語の対象応用分野（どの分野に向くか）	記号処理，人工知能，知識工学など。 一般的には人間とのインタフェース部分（自然言語，画像，高度の判断処理，高度の検索機能など）に有望。
	その他	

〔参考〕

各種言語機能一覧表

機能＼言語	PL/M	PL/Z	C	PL/I (G)	UCSD Pascal	Ada	CONPAS	Modula	LISP
○プログラム構造									
ブロック構造	○	○	○	○	○	○	○	○	○
リカーシブ	○	○	○	○	○	○	○	○	○
リエントラント	○	−	−	−	−	○	○	○	−
○変数の絶対番地割付	−	−	−	−	−	−	−	○	−
○変数のレジスタ割付け	−	○	△	−	−	−	−	−	−
○外部変数	○	○	○	○	○	○	−	○	○
○定数									
2進定数	○	−	−	○	−	○	−	○	○
8進定数	○	−	○	−	−	○	○	○	○
10進定数	○	○	○	○	○	○	○	○	○
16進定数	○	○	○	−	−	○	−	○	○
任意の底数	−	−	−	−	−	○	−	−	○
○コンパイル時のマクロ	○	−	○	○	−	○	−	−	○
○ファイルの読込み	○	−	○	○	○	○	○	○	○
○エニューメレーション	−	−	−	−	○	○	○	○	−
○集合	−	−	−	−	○	○	○	○	△
○整数									
8ビットデータ	○	○	○	○	○	○	−	○	○
16ビットデータ	○	○	○	○	○	○	○	○	○
○実数									
32ビット	−	−	○	○	○	○	○	−	○
64ビット	−	−	△	−	○	○	−	−	−
○文字	○	○	○	○	○	○	○	○	○
○文字列	○	○	○	○	○	○	○	○	○
○論理値	○	○	○	○	○	○	○	○	○
○ポインタ	○	○	○	○	○	○	○	○	○
○アドレス	○	○	○	ADDR	−		−	−	△
○スカラ	○	○	○	○	○	○	○	○	−
○配列	○	○	○	○	○	○	○	○	○
○構造体	○	○	○	○	○	○	○	○	−

機能＼言語	PL/M	PL/Z	C	PL/I (G)	UCSD Pascal	Ada	CONPAS	Modula	LISP
○抽象データ型	−	−	△	−	○	○	○	○	−
○エリア管理	STATIC AUTOMATIC BASED ON		○ STATIC	 AUTOMATIC STATIC BASED	NEW DISPOSE		NEW DISPOSE	NEW DISPOSE	自動的
○代入文	○	○	○	○	○	○	○	○	○
○多重代入文	○	−	○	−	−		−	−	○
○演算付代入文(op=)	−	○	○	−	−	−	−	−	−
○演算									
＊，／	○	○	○	○	○	○	○	○	○
↑			−	○		○	−	−	○
MOD	○	○	○	−	○	○	○	○	○
DIV	−			−	○	REM	○	○	○
＋，−	○	○	○	○	○	○	○	○	○
AND,OR,XOR	○	○	○	○	○	○	○	○	○
NOT	○	○	○	○	○	○	○	○	○
ABS	○	○	○	○	○	○	○	○	○
++, −−	−	○	○	−	−	−	−	○	−
シフト	○関数	−	○	−	−	−	−	○	○
○IN	−	−	−	−	○	○	○	○	−
○WITH	−	−	−	−	○	○	○	○	−
○IF	○	○	○	○	○	○	○	○	CONPを用いる
終りのマーク	−	FI	−	−	−	−	−	○	−

機能＼言語	PL/M	PL/Z	C	PL/I (G)	UCSD Pascal	Ada	CONPAS	Modula	LISP
○くり返し制御									△
くり返し	DO	DO REPEAT	FOR CONTINUE	DO REPEAT	FOR	LOOP	FOR	LOOP	
抜け出し	○	EXIT	BREAK	−	−	−	−	○	
終りマーク	○	OD	−	○	−	−	−		
WHILE条件	○	−	○	○	○	○	○	○	−
UNTIL条件	○	−	△	−	○	−	○	○	−
○場合わけ	○	IF CASE THEN	SWITCH	−	CASE	○	CASE	CASE	CONP
○GOTO	○	−	○	○	○	○	−	−	○
○手続き	○	○	○	○	○	○	○	○	○
○関数	○	○	○	○	○	○	○	○	○
○外部参照	○	○	○	○	○	−	−	○	○
○システム参照	−	−	−	−	−	−	−	○	−
○割込み許可，禁止	○	−	−	−	○	−	−	−	−
○例外処理	−	−	−	○	○	○	−	○	△
○並行処理	−	−	−	−	可能になる	○ TASK	PROCESS MONITOR QUEUE	MODULE PROCESS INTERFACE MODULE SIGNAL	−
○独立コンパイル	○	○	○	○	○	○	−	○	○
○アセンブラ	○	PLZ/ASM	AN ANT	○ RMAC	○	INLINE	−	−	△
○処理系の大きさ（8ビットマイクロコンピュータの場合）									
実行時のメモリの大きさ	＊ 100KB	＊ 32KB	60KB	48KB	48KB	−	48KB	−	48KB
システムの大きさ		＊ 40KB	142KB	133KB			130KB		30KB
ライブラリの大きさ		−	48KB	42KB			2KB		

（＊　推定）

5.4 システム記述言語のプログラミング例

それぞれのプログラミング言語の使い易さ，即ちそれぞれの言語の表現力やその言語の持っている機能を知るためには，ある程度の大きさの問題をそれぞれの言語でプログラミングしてみて，これを評価するのが最善であるが，それをレポートの中に収めることができないので，今回は，比較的簡単な問題を採り上げ，これをそれぞれのプログラミング言語によってプログラミングしてみることにした。

第1のグループ

このグループは，使用するマイクロプロセッサに密着する言語を採り上げている。

1つはIntel社が8080のために開発した8080アセンブリ言語に準拠するシステムを使用し，他にはChromod社が8080のために開発したSMAL／80という構造化アセンブリ言語を使用した。

ここではbubble sort方式のプログラムをプログラミングしている。

8080アセンブリ言語を使用した場合を標準的なプログラムと言うとすると，構造化アセンブリ言語を使用した場合には，特にこのプログラミングシステムが表現を判かりやすくするという点に重きを置いているために，出来上がったプログラムが非常に判かりやすくなっている。

マクロ付アセンブリ言語を利用して，SMAL／80に見られるプログラムの構造化を行うことができるが，このシステムによって構造化を行った場合と，SMAL／80を利用した場合とでは，SMAL／80の方が見やすくなっている。

SMAL／80の場合には，prepassとしてマクロプロセッサを利用することができるので，抽象化によるプログラミングよりスタートすることが出来るので，通常のアセンブリ言語を使用する場合に比べて，きちんとしたプログラムを作るということができる。

8080アセンブリ言語によるbubble sortプログラム

```
;       bubble sort
;
size    equ     255
;
        org     100h
bubble:
loop1:
        mvi     d,0     ; clear flag
        lxi     h,n     ;
        mov     b,m     ; set no. of items
                        ; to be sorted
        lxi     h,array ; set start address
loop2:
        dcr     b       ; test for end of loop
        jz      break1
        mov     a,m     ; address(i)
        inx     h
        cmp     m       ; address (i+1)
        jnc     loop2
        mov     c,m     ; exchange
        mov     m,a
        dcr     h
        mov     m,c
        inx     h
        mvi     d,1     ; set exchage flag
        jmp     loop2
break1:
        mov     a,d
        ora     a
        jnz     loop1
        jmp     0       ; return to monitor
n       db      size    ; no of items
array:
        ds      size    ; data area
        end     bubble
```

SMAL／80によるbubble sortプログラム

```
        /* A SMAL/80 BUBBLESORT */

SIZE EQU 255                                    /*set maximum size*/;

BUBBLE:
    LOOP:
        D = 0                                   /*reset flag for this loop*/;
        HL = N; B = M(HL)                       /*get number of elements in array*/;
        HL = ARRAY                              /*point to beginning of array*/;
        LOOP:
            IF /*no more pairs are left*/ --B ZERO
                BREAK;
            A = M(HL); ++HL                     /*get first element of pair*/;
            IF /*first element is less*/ A : M(HL) NEG
                THEN
                    D/*
                    do an interchange*/
                    C = M(HL);
                    M(HL) = A;
                    --HL;
                    M(HL) = C;
                    ++HL;
                    D = 1                       /*set flag*/;
                    }
        REPEAT;
    REPEAT /*until no pairs interchanged in a pass*/
        WHILE DA = D; A = A OR A; NOT ZERO;
RETURN;

/*VARIABLES - */
    N:   BYTE    SIZE                /*size of ARRAY*/;
    ARRAY:       RESERVE SIZE        /*bytes to be sorted*/;

END PROGRAM;
```

第２のグループは，ソフトウェアの記述用に使用されているプログラミング言語を採り上げた。

このグループに属する言語は，ＰＬ／Ｚ，ＰＬ／Ｍ，Ｃである。

それぞれの言語の生い立ちを見ると，このグループの中ではＰＬ／Ｍが一番先に生まれた。この言語は，ＰＬ／Ｉをモデルにしている。

次いで生まれたのがＣであり，最後に生まれたのがＰＬ／Ｚである。

ＰＬ／Ｚは，Pascal とＣが持っている考え方を採用している。

ＰＬ／Ｚでは，bubble sortの方式でプログラミングし，その他の２つの言語は通常のソートの方式でプログラミングしてあるために，これら３つのプログラミング言語でプログラムした例を直接比較することはできない。

これらの３つのプログラミング言語を利用してプログラミングした例は，いずれも同じような表現になっている。

このグループの言語はＯＳレベルのプログラミングに使用されるのでここで大きな問題になってくるのは，システム記述用言語として考えた場合に，ハードウェアが持っている機能をどれだけ能率よく記述あるいは処理できるかということである。

また，表現力と相伴って，処理系がどのように上手く作られているかということも大きな問題になってくる。

しかし，このプログラム側からは，これらの問題は読み取ることは出来ない。

PL/Mによるsortプログラム

```
sort$1:
do:
    declare string (30) byte:
    declare (i,j,temp ) byte:

do i = 0 to last(string) -1:
    do j = i + 1 to last(string):
          if string (i) > string (j) then
          do:
              temp = string (i):
              string (i) = string (j):
              string (j) = temp:
          end:
    end:
end:
end sort$1:
```

PL/Zによるbubble sortプログラム

```
    sort module
    constant
        true :=1
        false :=0
        n := 255
    internal
    t   array [255byte]
    temp byte
    i    byte
    m    byte
    ! sort array t[1:n] using bubble sort !
      sorted :=false
      m := n
      do
        if m < 2 then exit fi
        sorted := true
        i := 2
        do
             if i > m then exit fi
             if t [i-1] > t[i] then
             do
             temp := t[i-1]
             t[i-1] := t[i]
             t[i] := temp
             od
             sorted := false
             fi
             i += 1
        od
        m -= 1
      od
```

言語Cによるsortプログラム

```
A>
A>^S
#define items 100
main ()
     char word [items];
     int  i,j;
     char temp;
{
          for ( i = 1; i = items-1; ++i)
          for ( j = i+1; j = items; ++j )
          if (word [i] > word [j])
          {
          temp = word [i];
          word [i] = word [j];
          word [j] = temp;
          }
}
```

第3グループには汎用プログラミング言語が属する。

ＰＬ／Ｉ，Pascal，Adaといった高水準言語では,表現に若干の差異は見られるもののほとんど同程度であると言うことが出来る。

このレベルの言語になると，抽象型データの取扱いがどのように行えるかということが大きな問題になってくる。

ＰＬ／Ｉによるｓｏｒｔプログラム

```
sort:
procedure option (main);
%replace size 255
dcl
     word (size) character (2);
     temp        character (2);
     (i,j)            fixed;
do   i = 1 to size-1;
  do j = i+1 to size;
          if word (i) > word (j) then
          do;
          temp = word (i);
          word (i) = word (j);
          word (j) = temp;
          end;
  end;
end;
end sort;
```

PASCALによるsortプログラム

```
program sort ;
const n = 100;
var
word : packed array [ 1.. n ] of char;
temp : char;
i, j : int;
begin
    for i:= 1 to n-1 do
      for j:=i+1 to n do
          if word (i) > word (j) then
          begin
             temp := word (i);
             word (i) := word (j);
             word (j) := temp
          end
end.
```

Adaによるsortプログラム

```
sort:
declare
size : constant integer := 255;
word : array [1..size] of character;
temp : character;
i,j :  integer;
begin
     i := 1;
     while i <= size-1 loop
       j := i +1;
       while j <= size loop
           if word[i] > word[j] then
           temp := word[i];
           word[i] := word [j];
           word[j] := temp;
           end if;
       j := j+1;
       end loop;
       i := i +1;
     end loop;
end;
end sort;
```

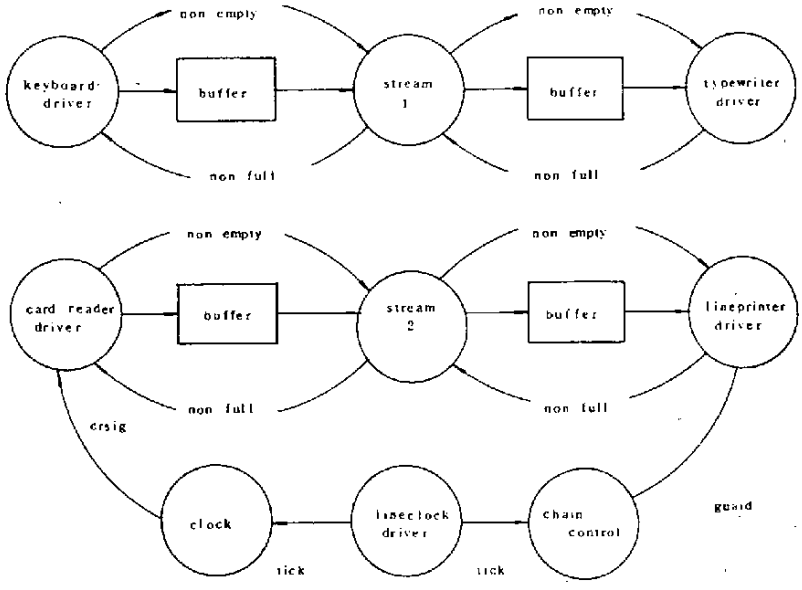
第4グループに属するプログラミング言語にはConcurrent PascalとModulaがある。

Concurrent Pascalは併行処理をスマートに記述しようとしているのに対して，Modulaは，IO driverまでもModulaで記述することができるようにしている。

この点で同じ併行処理を記述することが出来る様にした言語であるのに，両者の考え方に大きな

差異がある。

Modula で取り上げたプログラム例では，次の図で示すような併行処理を行う。



keyboard をcontrol するためのIO control routine は device module で定義することができる。

device module keyboard〔4〕; では，keyboard control するためのpriority が4であることを示している。

nonfull, nonempty : signal; では，process の同期をとるためのsemaphore 変数を定義する。

process driver〔60 B〕; で，IO control programを定義する。

〔60 B〕はkeyboard に割り当てられたinterrupt vector のaddress を定義する。

Kbs〔177560 B〕: bitsでは，keyboard のstatus register に割り当てられているaddress をまたKbb〔177562 B〕: integer; ではkeyboard より入力されたdata をしまうdata regi ー ster のaddress を定義している。

loop よりend までに囲まれた文がkeyboard よりデータを入力するためのcontrol routine である。

kbs〔6〕:= true; doio; kbs〔6〕:= false;という文では，status register の6 bit 目を取扱っている。即ち，6 bit 目をtrueにして割込み許可し，doio で入力待ちし，入力があると6 bit 目をfalseにして，次の入力を禁止している。このように，Modula では，今迄アセンブリ言語でプログラミングしていたようなlower level のプログラミングまでModula でプログラミングすることが出来るようにしている。

このプログラムが利用できる環境ではcard reader より一連のデータが送り出されるときにre—

ady の信号が立たないとか，printer の chain が一定時間経過すると止まってしまうようなことがあるので，これらの control を行うために timer を利用している。

Modula による併行処理プログラム例

(Using of Modula より)

```
module datastreams;
  const lf=12c; ff=14c; cr=15c;
  var crsig:signal;
device module timing[6];
  define tick;
  var tick:signal;
    lcs[177546b] : bits; (* clock status *)
  process driver[100b];
  begin lcs[6]:=true;
    loop doio:send(tick)
  end
  end driver;
begin driver
end timing;

device module keyboard[4];
  define get;
  const n=16; esc=33c;
  var inx,outx,nf:integer;
    nonfull,nonempty:signal;
    buf:array 1:n of char;
  procedure get(var ch:char);
  begin
    if nf=0 then wait(nonempty) end;
    ch:=buf[outx]; outx:=(outx mod n)+1;
    dec(nf); send(nonfull)
  end get;
  procedure driver[60b];
    var kbs[177560b]:bits;
        kbb[177562b]:integer;
    ch:char;
  begin
    loop
      if nf=n then wait(nonfull) end;
      kbs[6]:=true; doio; kbs[6]:=false;
      ch:=char(kbb mod 200b);
      if ch = esc then halt end;
      buf[inx]:=ch; inx:=(inx mod n)+1;
      inc(nf); send(nonempty)
```

```
      end
    end driver;
  begin inx:=1;outx:=1;nf:=0;driver
  end keyboard;
  device module typewriter[4];
    define put;
    const n=64;
    var inx,outx,nf:integer;
      nonfull,nonempty:signal;
      buf:array 1:n of char;

  procedure put(ch:char);
  if nf=n then wait(nonfull) end;
  buf[inx]:=ch;inx:=(inx mod n)+1;
  inc(nf);send(nonempty)
end put;
process driver[64b];
  var tws[177564b]:bits;
      twb[177566b]:char;
begin
  loop
  if nf=0 then wait(nonempty) end;
   twb:=buf[outx];outx:(outx mod n)+1;
   tws[6]:=true;doio;tws[6]:=false;
   dec(nf);send(nonbfull)
  end
end driver;
begin inx:=1;outx:=1;nf:=0;driver
end typewriter;

device module cardreader[6];
  define read;
  use crsie;
  const n=256;
  var inx,outx,ne,nf:integer;
      nonfull,nonempty:signal;
      buf:array 1:n of char;
procedure read(var x:integer);
begin dec(nf);
  if nf<0 then wait(nonempty) end;
  x:=buf[outx];outx:=(outx mod n)+1;
  inc(ne);if ne>=0 then send(nonfull) end;
end read;
```

```
process driver[230b];
  const m=81;
  var crs[177160b]:bits;
      crb[177164b]:bits;
  procedure put(x:integer);
  begin buf[inx]:=x; inx:=(inx mod m)+1;
    inc(nf)
  end put;
begin
  loop dec(ne,m);
    if ne<0 then wait(nonfull) end;
    while not off(crs,[8,9]) do wait(crsie) end;
    crs:=[0,6];
    loop doio;
      when not off(crs,[14,15]) exit
    end;
    put(-1); crs[6]:=false;
    if nf>=0 then send(nonempty) end
  end
end driver;
begin inx:=1; outx:=1; nf:=0; ne:=m; driver
end cardreader;
device module lineprinter[4];
  define write,weol,testchain;
  use lf;
  const n=512;
        dc2=23c; dc4=24c;
        chaindelay=250;
  var inx,outx,ne,nf,del:integer;
      nonfull,nonempty,guard:signal;
      buf:array 1:n of char;
      lps[177514b]:bits;
      lpb[177516b]:char;
procedure write(ch:char);
  if ne<0 then wait(nonfull) end;
  buf[inx]:=ch; inx:=(inx mod n)+1;
end write;
procedure weol;
begin inc(nf); send(nonempty)
end weol;
procedure testchain;
begin
```

```
    if nf>=0 then wait(guard)
    else dec(del);
      if del=0 then
        lps:=dc4;wait(guard)
      end
    end
  end testchain;
  process driver[200b];
    var ch:char;
  begin lpb:=dc3;
    loop dec(nf);
      if nf<0  then
        send(guard);wait(nonempty);
        lpb:=dc3;del:=chaindelay
    end;
    repeat ch:=buff[outx];outx:=(outx mod n)+1;
      inc(ne);lpb:=ch;
       if not lps[7]  then
          lps[6]:=true;doio;lps[6]:=false
       end
    until ch=lf;
    if ne>=0 then send(nonfull) end
    end
  end driver;
  process stream1 (* keyboard to typewriter *)
    use get,put;
    var ch:char;
  begin
    loop get(ch);
      if ch=cr then put(cr);put(cr);put(lf)
               else put(ch)
    end
  end
  end stream1;
  process stream2; (* cardreader to lineprinter *)
    use read,write,weol;
    const eoi=37b;badchar='?';
    var x:integer;ch:char;
        t:array 0:63 of char;
        z:array 0:7 of integer;
    loop read(x);
      if x=eoi then
        repeat read(x) until x<0;
        write(ff);
      else
        while x>=0 do
         wite(ch);read(x)
        end;
         write(lf);weol
```

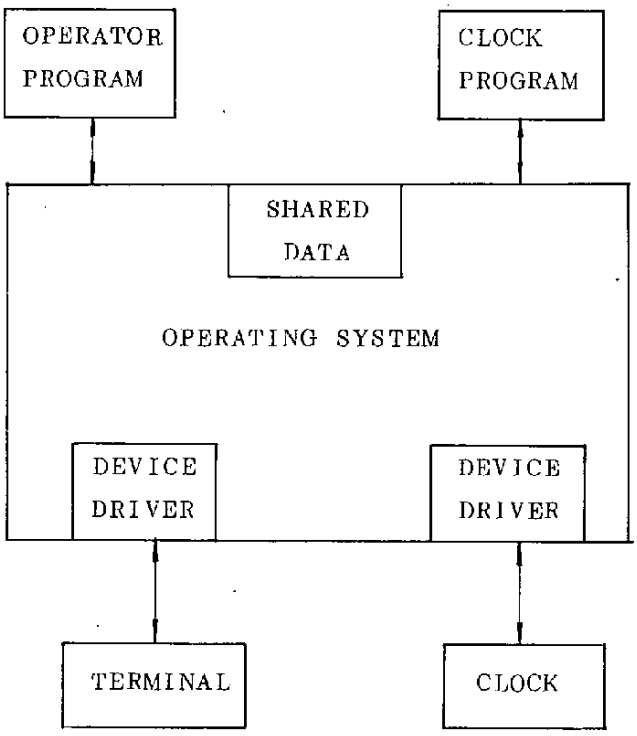
```
    end
  end
end stream2;

process clock;
  use tick,csig;
begin
  loop wait(tick);send(csig)
end clock;
process chaincontrol;
begin
  loop testchain;wait(tick)

  end
end chaincontrol;

begin (* data stream *)
  stream1;stream2;clock;chaincontrol
end datastreams .
```

Concurrent Pascal のプログラム例では，米国のEnertek社が8080マイクロプロセッサのために開発した，Micro CONPAS をベースにしている。

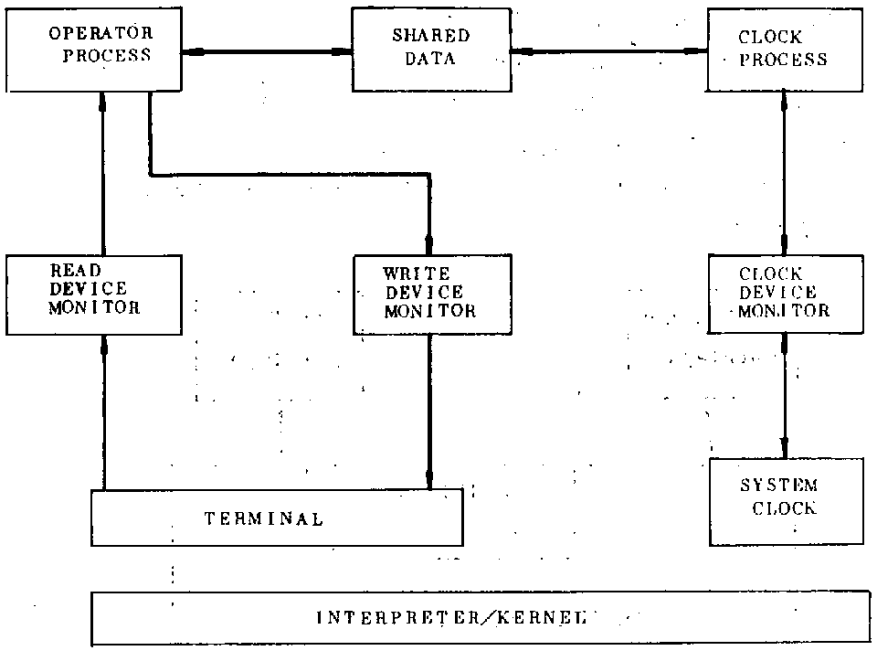
このシステムは，Concurrent Pascal にModula の IO driver の考え方を導入している。

このプログラム例では，オペレータがSというコマンドで時刻の設定を行い，Gというコマンドで時刻の表示を行うというリアルタイム処理を行っている。

通常のプログラミング言語を使用しているプログラミングすると，モニタとこのプログラムモジュールとの関係は図のようになる。



これに対してConcurrent Pascal を使用した場合には次の図のようにすっきりとした型になる。



Concurrent pascal では lower level の program をプログラミングすることが出来ないが，Micro CONPAS では，Concurrent Pascal の記述を生かし，これに lower level のプログラミングを行うことができるようにしている。

INN や OUT が 8080 の入力ポートや出力ポートをコントロールするための命令となっている。

このプログラムでは，init term －out（３）で clock －monitor が priority 3 で起動する。

以下同じように，clock によって，clock monitor，pulse〔6〕によって clock pulse，term－in によって UART － read，tick tock によって clock process が，また operator によって，operator process が起動し，この initial process は消滅する。

clock pulse では clock からの割込みを処理する。

この device monitor が最初に起動された時にはこの System で使用されている 8253 timer を initialize している。

clock process では割込みが起きる度に割込み回路の初期化と，software timer の count up を周期的に行っている。

これに対して operator process は clock と併行して処理が行われる。

この process では，入力をうながす prompt の表示と，keyboard よりの入力及び keyboard より入力されたＳあるいはＧというコマンドの処理を周期的に行っている。

UART－write 及び UART－read という device monitor が，lower level の programming を行うために用意されている。

Micro Concurrent Pascal による併行処理プログラム例

```
(# MICRO CONCURRENT PASCAL PROGRAM #)
CONST
TIMER_CONTROL = ADR(#F3);
TIMER_ZERO    = ADR (#F0);
CTRL_WORD     = ADR (1);
DATA_WORD     = ADR (0);
LINELENGTH    =72;

TYPE
   LINE = ARRAY [1..LINELENGTH] OF CHAR;

CONST
   NULL = '(:0:)';
   BS   = '(:8:)';
   LF   = '(:10:)';
   CR   = '(:13:)';
   NULL2 = '(:0:)(:0:)';

TYPE
  TIME = RECORD
         HOUR, MIN, SEC : INTEGER
         END;
  DISPOSITION = (PROMPT, NEWLINE, STAY);


        ( UART WRITE )
TYPE UART_ERITE   = DEVICE_MON (SELECTOR:INTEGER);
CONST TBE_BIT   = ADR (2);

PROCEDURE ENTRY WRITE (MESSAGE : LINE,
                       DISP    : DISPOSITION);
VAR I : INTEGER;
BEGIN
   REPEAT
        UNTIL (INN(CTRL_WORD) AND TBE_BIT) <> 0 ;
        OUT(#26,CTRL_WORD);
        OUT(#27,CTRL_WORD);
        DOIO;
        I:=1;
        WHILE (MESSAGE[I] <> NUL) AND (I < LINELENGTH) DO
            BEGIN
                OUT (ORD (MESSAGE[I]), DATA_WORD);
                DOIO
                INC (I);
```

```
            END;
         IF (DISP = PROMPT) OR (DISP = NEWLINE) THEN
            BEGIN
                 OUT (ORD (LF), DATA_WORD);
                 DOIO;
                 OUT (ORD (CR), DATA_WORD);
                 DOIO;
            END;
         IF DISP = PROMPT THEN
            BEGIN
                 OUT (ORD ('>'), DATA_WORD);
                 DOIO;
            END;
END;

BEGIN
   { INITIALIZATION CODE FOR UART }
        OUT (0, CTRL_WORD);
        OUT (0, CTRL_WORD);
        OUT (#40, CTRL_WORD);
        OUT (#7A, CRTL_WORD);
        OUT (#27, CRTL_WORD);
END;

TYPE UART_READ = DEVICE_MON (TERM_OUT:UART_WRITE;
                             SELECTOR:INTEGER);

VAR
   PRINT : ARRAY [1..2] OF CHAR;
   THROWAWAY : INTEGER;

PROCEDURE ENTRY READ (VAR MESSAGE : LINE;
                      VAR LENGTH : INTEGER);
VAR LASTCHAR : CHAR;
BEGIN
   LENGTH := 0;
   REPEAT
        IF LENGTH < LINELENGTH THEN
           INC (LENGTH);
        DOIO;
        LASTCHAR := CHR (INN (DATA_WORD));
        IF LASTCHAR = BS THEN
           BEGIN
            IF LENGTH >= 2 THEN
                DEC (LENGTH);
            DEC (LENGTH);
```

```
            END;
          ELSE
            MESSAGE[LENGTH] := LASTCHAR;
          IF LASTCHAR <> CR THEN
            BEGIN
             PRINT[1] := LASTCHAR;
             TERM_OUT.WRITE (PRINT, STAY);
            END;
          UNTIL (LASTCHAR = CR );
          TERM_OUT.WRITE (NULL2,NEWLINE);
     END;

BEGIN
          {INITIALIZATION FOR UART }
     OUT(#7,CTRL_WORD);
     THROWAWAY :=INN (DATA_WORD);
     THROWAWAY := INN (DATA_WORD);
     PRINT[2] := NUL;
END;

{ CLOCK MONITOR }
TYPE CLOCK_MONITOR = MONITOR;
VAR CLOCKTIME : TIME;
PROCEDURE ENTRY TOCK;
   BEGIN
     WITH CLOCKTIME DO
          BEGIN
            INC (SEC);
    IF SEC = 60 THEN
                 BEGIN
                   SEC := 0;
                   INC (MIN);
                   IF MIN = 60 THEN
                     BEGIN
                       MIN := 0;
                       INC (HOUR);
                       IF HOUR = 24 THEN
                          HOUR :=0;
                     END;
          END;
   END;

PROCEDURE ENTRY SETTIME (T : TIME);
   BEGIN
        CLOCKTIME := 0;
```

```
    END;

PROCEDURE ENTRY GETTIME (VAR T : TIME);
    BEGIN
        T := CLOCKTIME;
    END;

BEGIN (INITIALIZATION )
    WITH CLOCKTIME DO
        BEGIN
            HOUR := 0;
            MIN :=  0;
            SEC  := 0;
        END;
END;

( CLOCKPULSE )
TYPE CLOCKPULSE = DEVICE_MON (SELECTOR : INTEGER);
PROCEDURE ENTRY TICK;
BEGIN
        DOIO;
END;

BEGIN
        ( INITIALIZATION FOR 8253 TIMER )
        OUT (#34, TIMER_CONTROL);
        OUT (#96, TIMER_ZERO);
        OUT (0, TIMER_ZERO);
END;

(CLOCK PROCESS )
TYPE CLOCKPROCESS = PROCESS (PULSE : CLOCKPULSE;
                             CLOCK : CLOCK_MONITOR);
BEGIN
        CYCLE
            PULSE.TICK;
            CLOCK.TICK;
        END;
END;
        (OPERATOR PROCESS )
TYPE OPERATORPROCESS = PROCESS (CLOCK : CLOCK_MONITOR;
                                TERM_IN : UART_READ;
                                TERM_OUT : UART_WRITE);
VAR
        BUFFER : LINE;
        LENGTH : INTEGER;
        T      : TIME;
```

```
FUNCTION NUMBER (CHARACTER : CHAR) : INTEGER;
BEGIN
        NUMBER := ORD (CHARACTER) - ORD ('0');
END;

FUNCTION ASCII (NUMBER : INTEGER) : CHAR;
BEGIN
        ASCII := CHR (NUMBER + ORD ('0'));
END;

BEGIN

    CYCLE
        TERM_OUT.WRITE (NULL2, PROMPT);
        TERM_IN.READ   (BUFFER, LENGTH);
        CASE BUFFER[1] OF
'S' :
      BEGIN
        WITH T DO
            BEGIN
                HOUR := 10 * NUMBER (BUFFER[3]) +
                            NUMBER (BUFFER[4]);
                MIN  := 10 * NUMBER (BUFFER[6]) +
                            NUMBER (BUFFER[7]);
                SEC  := 10 * NUMBER (BUFFER[9]) +
                            NUMBER (BUFFER[10]);
            END;
      CLOCK.SETTIME (Y);
      END;

'G' :
      BEGIN
        CLOCK.GETTIME (T);
        WITH T DO
            BEGIN
                BUFFER[1] := ASCII (HOUR DIV 10);

                BUFFER[2] := ASCII (HOUR MOD 10);
                BUFFER[3] := ':';
                BUFFER[4] := ASCII (MIN DIV 10);
                BUFFER[5] := ASCII (MIN MOD 10);
                BUFFER[6] := ':';
                BUFFER[7] := ASCII (SEC DIV 10);
                BUFFER[8] := ASCII (SEC MOD 10);
                BUFFER[9] := NUL;
```

```
                END;
            TERM_OUT (BUFFER, NEWLINE);
            END;
        END;
    END;
END;

        { INITIAL PROCESS }
VAR
    CLOCK : CLOCK_MONITOR;
    TICKTOCK : CLOCKPROCESS;
    PULSE : CLOCKPULSE;
    OPERATOR : OPERATOR PROCESS;
    TERM_IN  : UART_READ;
    TERM_OUT : UART_WRITE;

BEGIN
    INIT TERM_OUT (3);
    INIT CLOCK;
    INIT PULSE (6);
    INIT TERM_IN (TERM_OUT,4);
    INIT TICKTOCK (PULSE, TERM_IN, TERM_OUT);
    INIT OPERATOR (CLOCK, TERM_IN, TERM_OUT);
END.
```

5.5　システム記述言語の総合評価

計算機言語の重要性については，どんなに強調してもしすぎるということはない。実際過去３０年にわたる汎用基本ソフトウェア体系の開発史は，試行錯誤の連続であったわけであり，現在数百万ステップ，あるいは一千万ステップを超えるオペレーティングシステムが過去のCOBOLコードやFORTRANコードを引きずって，その改善の方向を模索している事情にある。

この間，プログラマはアセンブリ言語，COBOL，FORTRAN，ＰＬ／Ｉのような既存の言語でオペレーティングシステムや言語処理系を開発してきたわけであり，同時にそれらの言語を通じてオペレーティングシステム像や言語処理系のイメージを形成してきた。つまり，計算機言語は対象ソフトウェアの構成や性能を，その言語概念によって制約してきたという一面がある。

このような汎用基本ソフトウェア体系の開発に対して，巨大複雑でない適正な中間技術を追求してきたところにマイクロコンピュータソフトウェアの１つの特長があるとみなされる。例えば，TI社における基本ソフトウェア開発用の3000人のプログラマによるUCSD PASCALの採用，16ビットマイクロコンピュータにおけるUnixシステムの普及を想起せよ。

計算機言語の働きは，大ざっぱに言って

① 設計の支援をすること

② プログラマ間の情報伝達に資すること

③ 計算機に対して指令を与えること

であろう。このそれぞれに伝統的な汎用大型基本ソフトウェア体系の場合と，新しい適正技術の場合とで解釈や視点の相異が生じている。この点が，16ビットマイクロコンピュータのシステム記述言語を評価するにあたってまず第一に強調されなければならない。

5.3でとりあげた13種のシステム記述言語については，評価項目として

- 言語概念としての構文
- 言語概念としての拡張機能
- 言語処理系
- 言語の環境
- その他（社会的，経済的側面）

を与えたのであった。このとき，評価の方法としてリニアな基準を与え，点数をつけることは無意味であることはみやすい。

また，16ビットマイクロコンピュータについては従来の汎用機とは桁違いの普及が伴うことが予想され，ユーザの利用も多岐にわたることが見込まれている。従って，システム記述言語の現場における選択も，従来のように１つないし２つ選択すればよいというものではなくて，複数のシステム記述言語を併用するケースが多くなるはずである。

一般的には，プログラミングの生産性（楽しく創造的にプログラミングをする意味も込めて）から判断すると，次のようなプログラミング方法論上の知見が有効であったのは既述の通りであった。

① 高水準言語の採用

② エディタの活用

③ プログラミング環境の改良

④ 移植可能・再利用可能ソフトウェアの開発

システム記述言語を評価するにあたって，単に各種言語を処理系と共に考察するだけでは不十分であって，支援ツール，例えばエディタの性能，また総合的なプログラミング環境の良し悪し，そして大量の流通ソフトウェアが見込まれる時は移植可能性と再利用可能性が重要なファクタとなる。この方向を目指しているものに，UCSD PascalのPシステム，AdaのStoneman，CのUnix ModulaのＸＳ－０などがある。

特に，重要な要素として，移植可能性と再利用可能性がある。移植可能性については，処理系の記述言語と作成方式と関係する。（参照，5.2） 再利用可能性については，特に言語概念としてライブラリ概念を導入したUCSD Pascalの"unit"，Adaの"package"が注目される。しかしながら，これらがわが国でどれ程有効であるかまた普及するかについては，現在のところ判断し難いところである。

実際に，システム記述言語を現場で採用するときには，以上の一般論では律しきれないことは明らかであろう。１つには，現場で採用するに至る組織（企業や研究機関）上の問題がある。デシジョンメーカは通常汎用中大型機の基本ソフトウェア体系になじんだ中高年令層である。あるいは，ハードウェア関係者や門外漢がその任にあたることも多い。これらのデシジョンメーカに，採用のための技術データを与えるプログラマやシステムエンジニアも，過去の汎用機の基本ソフトウェア体系になじんだ人材が多い。計算機言語の視方は，初体験や自身で書いたり読んだりした経験やまた趣味で影響されることは周知の通りである。

このとき，過去10年間以上にわたる中高年令層技術者の経験やプログラムの蓄積の大半がオブソリートになるような事態は，新言語の採用いかんで生じる可能性がある。また，現場の担当者の新言語習得の手間によって，一時的に工期が延びるケースもありうる。

言語の習得にあたっては，特定言語の言語概念で理解するよりも，一般的な計算機言語のパラダイムや言語概念に基づいて学習する方が理解が速いのにもかかわらず，特定言語の不備な手引書やジャーゴンで苦労して解読するようなケースも多い。

別の側面として，当該言語の採用，使用のライフサイクルも考慮されなければならない。その言語をどのようなシステム記述に用いるか，そのシステムの寿命はどれほどか，その言語の後続言語は何になりそうか，一過的な開発でよいかあるいは５年というスコープが必要か，これらが言語のライフサイクルの見込みに関連するファクタである。

また，採用を決定した言語については，サブセッティングやスーパーセッティングをどう行うかも重要なファクタである。ツール群，オペレーティング環境の質も大切である。当然言語は成長するものであるから，ベンダの保守支援体制の質も考慮されなければならない。

以下では，このような実態を捨象して，参考までにわが国で１６ビットマイクロコンピュータのシステム記述言語をごく一般的に評価する。ただし，評価は評価者の意見を反映したものに

なるのは当然のことであり，ここでの評価は本委員会の一致した意見によるものではないことをおことわりしておきたい。評価にあたっては，既述の各種制約条件は考慮せず，プロフェッショナルプログラマがリソースを十二分に利用できるという仮定を置いている。このような評価が有意義であるかどうか，誤解の恐れがないかについてはここでは議論しない。

評価の枠組みは，

(a) 設 計 支 援
(b) 情 報 伝 達
(c) 計 算 機 指 令
(d) 総 合 評 価

という既述のそれを用い，3段階（○，△，×）の評価法を試みる。総合評価は，(a) ～ (c) の平均に相当する。

採り上げる言語は，次に限定する。

(i) Macro Assembler
(ii) PL／M
(iii) C
(iv) UCSD Pascal
(v) Ada
(vi) LISP

評価結果は次の通りである。

言語 ＼ 評価の視点	(a) 設計支援	(b) 情報伝達	(c) 計算機指令	(d) 総合評価
(i) Macro Assembler	×	×	○	×
(ii) PL／M	×	×	△	×
(iii) C	△	△	△	△
(iv) UCSD Pascal	○	○	△	○
(v) Ada	○	○	○	○
(vi) LISP	○	○	△	○

(注) 本表は，委員会のメンバー全員の合意ある評価結果ではない。

このような評価は，ある意味では平均的なプロフェッショナルプログラマの直観以上に，つけ加えるべき情報はないように見える。(i) ～ (vi) とも，オペレーティングシステムなどの記述は何らかの方法で可能であるという意味で等価であるし，LISPを除いて世代的に後から誕生した言語の方が，言語仕様に工夫がこらされているのも当然であろう。

要約すれば，上記評価は主観的なある態度表明というほどの意味を持つに過ぎないだろう。

5.6 システム記述言語の今後の展望

(1) システム記述言語の将来性へのコメント

以下に，本委員会各委員のシステム記述言語に対する将来性についてのコメントを要約する。これらは，5.3で記述されたものと委員会での議論に基づいている。

(a) ASSEMBLER

移植性や読みやすさの点から敬遠ぎみになっているが，アセンブラの記述力に足るだけの高位言語の普及がない限り，問題点を残しながらも，今後も使用されよう。

(b) MACRO ASSEMBLER

一般的に，MACRO ASSEMBLER の必要性は存続する。

(c) PL／(M)

移植性，読みやすさがよく，構文も比較的簡単なので，システム記述用に有望である。

(d) PL／Z

PLZ／SYSだけでは，システム記述の細部は書きにくいが，同系列のPLZ／SYSとともに使用すれば，システム記述言語として期待できる。

(e) C

Pascal や Ada と競合するが，次のような理由で，Cの将来性は高い。

Pascal をCのようにシステム記述用として実用化するには拡張が必要であり，標準化，移植性に問題が生じる。また，Ada は C と比較して，処理系の規模，実行時ルーチンが大きくなり，サポート可能なコンピュータが制限される危険性がある。

(f) PL／I (G)

作成者やユーザ団体の普及の努力によっては，順調に伸びそうである。

(g) 標準Pascal

将来性は大きいとみなされる。

(h) UCSD Pascal

将来性は，大きいとみなされる。

(i) Ada

米国国防総省（DoD）とヨーロッパ共同体（EC）の後援によって，将来性は極めて高い。

(j) Modula Ⅱ

将来性については，予測が現時点では難しい。Pascal を作成したN. Wirth 教授の試みとして，注目される。

(k) Concurrent Pascal

Modula Ⅱ が出現するまでは，併行処理記述に使用されてきたが，Modula の出現により今後どう推移するかは予測し難い。

(l) FORTH

米国にFIG，我が国にFIG Japan，欧州にEFUGができた。今後利用が増大するであろ

う。

(m) LISP

極めて有望である。従来障害となってきたメモリコスト，会話型端末の低価格化が大きく寄与することになろう。"人間的"な計算機システムの基本言語として，普及される可能性が大きい。

これらのコメントについてさらにコメントすると，展望は3～7年のスコープにわたっており，言語によって若干スコープが異なる点に留意する必要がある。委員によっては異なる見解を持つ方々がおられないではないが，平均的な見解とみなして差し支えないと判断される。

これらは，現在入手可能な処理系やオペレーティング環境を基礎に展望されたものであり，今後の市場動向によっては，見通しが変わる可能性が大いにあることも付言しなければならない。

(2) システム記述言語の課題

これらのシステム記述言語が普及・定着するにあたっては，過去のCOBOL，FORTRAN，PL／1と同様に長い年月がかかるとみなされる。そこには，ベンダの普及努力があり，ユーザの消化期間が必要である。新しい芽としては，メーカ主導型でないPascalやAdaのような普及ケースがありうる。

従って，次のような技術課題に対する研究・試作・開発・普及がどのようなペースで進行するかは，十年にわたるようなスコープで考慮する必要があるであろう。

(a) 対象システムのモデル化

オペレーティング環境をどう想定し，各種のモデルを作成してみせるかは，システム記述言語の設計に対する大きなファクタとなる。UNIXやAdaがその意味で注目されるし，N. Wirth 教授のLilith マシン，あるいはXerox PARCにおけるAltoマシンの動向にも注目する必要がある。

(b) 仕様記述言語

プログラミング言語の発展過程の当然の帰結として，設計を支援する機能が，(a)と表裏一体をなして望まれる。注目される動向は，言語設計の技術水準を常にリードし続けてきたALGOL，Pascal，Adaの系譜に属する研究集団によるAbstracto 84である。また，LISPをベースにする方向も，興味深いものがある。

(c) 併行処理記述

併行処理記述は，B. Hansen 教授の言を借りると，21世紀にわたるプログラミング方法論の課題であるような困難な側面を持っている。この課題は，70年代に，Concurrent Pascal，C，Ada，Modula等によって追求されてきた。現在，各種の試みが研究者によって行われているが，データ抽象化や検証可能性などの理論とあわせて，その動向に注目する必要がある。

(d) 検証記述

システム記述言語は，軍事システムや制御システムのような人命に関わるシステムを構築す

る基礎となるものである。この意味で，各種検証理論の進展，SRIなどにおける検証可能オペレーティングシステムの開発，超LSI回路の設計プログラムにおける検証概念の導入などをフォローする必要がある。

(e) 移植性，再利用可能性

UCSD Pascalにおける unit，Adaにおける package のようなモジュラリティの概念が，今後のシステム記述言語に導入される傾向は明らかである。このとき，共用されるモジュールの仕様記述方式，検索の効率向上，モジュールの流通・普及等に技術上の課題が生じることが予想される。これらについての理論上，実施上の進展が注目される。

(f) コンパイラコンパイラ技術

CMUにおける PQCC プロジェクトも含めて，言語構文の記述やテーブル表示により，処理系のパーザ，セマンティックルーチン，さらにはオプティマイザ，コードジェネレータを自動生成する技術は依然として進展しつつあり，これらの動向がフォローされる必要がある。

(g) プログラム開発環境

システム記述言語をそれとして特定し，議論をするやり方に対して，システム記述言語がオペレーティング環境とセットで構築されるケースも多くなった。PWB／UNIX(C)，XS－1(Modula)，UCSD P-SYSTEM(UCSD Pascal)，MASPE(Ada)がその例である。他に，MITやXeroxにおける試みもあり，これらの動向は積極的にフォローされるべきである。

(h) 巨大技術か中間技術か

今後10年のマイクロコンピュータソフトウエアについては，1401，7090，360，370，3033という従来の汎用基本ソフトウエア体系の延長上で議論されるが，UCSD P.

UNIX のような「スモール イズ ビューティフル」という適正技術・中間技術の発想で議論されるかによって，まったく態度や見解が二分されるはずである。汎用基本ソフトウエア体系の延命策には，それなりの市場からやってくる立場と必然性があり，専用基本ソフトウエア体系の普及にはユーザニーズや新興のシステムハウス等の成長からくる立場と必然性がある。

これらの諸点を見過した議論は，ともすると不毛の結論に至るケースが多い。従って，システム記述言語の展望にあたっては，特にどのようなシステムの記述に用いるかという視点が強調されなければならない。

(3) 規範的な展望

公的な機関による規範的な計算機言語に対する施策でもっとも大がかりなものは，Adaに対するDoD（米国国防省）のそれである。

Ada実現のための努力は，1975年に開始され，現在までに次のステップを踏んでいる。

① 既存言語の洗い直し

② 軍産学協同による国際競合入札方式による新言語の設計

③ 言語に伴う支援環境の仕様設定

④ 言語仕様の設定と機種別処理系の委託開発（DoDのみならずECも含む）

⑤ 国際標準化

⑥ 支援環境の標準的機器構成での委託開発

この間，言語設計等の活動には数百人の専門家が動員され，投資規模も既に50億円を超える規範的な施策が続行されている。今後も，特に支援環境の実現とAdaの普及に対して，過去の投資を上まわる施策が見込まれている。

ちなみに，Adaに関する活動の主だったものを欧米に拾うと以下の通りである。

(a) DoD関連

① SofTech社

陸軍の受注で，VAXをホストとし，1602等をターゲットとするAdaの処理系を1982年後半までに開発中。同じく，支援環境を開発中。

空軍からは，IBM 370とCDC 6600をホストとし，Interdata 8－32等をターゲットとするAdaの処理系と支援環境を1983年半ばまでに開発中。

また，検定システムも開発中。以上の受注総額は15億円を超える。

② TRW社

空軍の受注でAdaの形式的定義を処理系実現の立場から行った。

③ New York大学

空軍の受注でAdaの意味的なモデルを作成した。

④ このほかに，空軍から大がかりの発注がソフトウエアハウスに近い将来予定されている。

(b) EC関連

① CII Honeywell Bull／Apsys／Siemens各社

ECからの受注で，Honeywell level 6とSiemens 7760用に，1983年までに9億円を超える規模で開発予定。

② Karlsruhe大学

西独国防省からの受託で，1981年末までにSiemens 7760用にAda処理系を開発予定。

③ Olivetti社

ECからの受注で，デンマークの企業と7億円を超える規模でAdaの処理系を開発中。

④ York大学

英国のScience Research Councilの委託で，Adaの処理系を開発中。

(c) 民間企業関連

① Univac社Defence System Division

"Bda"を検討中。

② DEC社

リサーチ中。

③ Intel 社

VAX をホストとし，iAPX 432 をターゲットとする Ada（マシン）を 1981 年 7 月までに開発中。

④ Intermetrics 社

TCOL Ada を開発中。

⑤ Telesoftware 社

Kennеth Bowles 博士の UCSD Pascal の発展として，1981 年 7 月に Ada のサブセットを出荷予定。

⑥ IBM 社 FSD

PDL から Ada への変換を実施中。

⑦ CSC 社

空軍から Ada 関連のプロジェクトを受注し，検討中。

⑧ CCA 社

Ada のデータベースを 1983 年までに VAX 上で開発中。

⑨ Litton Data Systems

MIL 標準 1862 という 32 ビットの Nebula マシンに対して Ada を開発中。

(d) 大学関連

① UC Berkeley

Pascal と Ada 間の変換系を開発中。

② USC

DEC 10／20 上で Ada の処理系を開発中。

③ Stanford 大学

TI 990 をターゲットとする Ada の処理系を開発中。

④ CMU

Unix 関連の Ada 処理系を開発中。

⑤ Florida 大学

Z 80，CP／M をターゲットとする Ada のサブセットを開発中。

このような実状に対して，わが国における Ada 関連の活動の主だったものは以下の通りである。

(a) 大学・学会関係では，1960 年後半以来の Algol スクール（学派）に属する計算機言語の専門家による Ada の検討があった。一部の成果は，情報処理学会誌や商業誌に紹介されている。

(b) 業界関連では，㈳日本電子工業振興協会による 2 年間にわたる工業用電子計算機言語としての Ada 検討委員会の活動があり，Ada およびその支援環境の紹介が行われた。

(c) ソフトウエア業関連では，協同システム開発㈱により，Ada の検討が産学協同で行われ，

Ada の言語仕様がわが国の計算機言語専門家を中心として，何割か修正された経緯がある。しかしながら，これらに対する投資規模は数百万円の域を出ていない。

民間，特に公社やメーカ等における活動は公表されていないが，1981年現在，処理系なり支援系の開発についてのデータは存在しない。

1つの計算機言語しかもシステム記述という限定された目的の計算機言語の，仕様設定と開発および普及に，欧米においてこれだけの努力が払われた背景は，通常ソフトウェアコストのDoDにおける上昇によって説明されている。一方，DoD主導による新しい1つのシステム記述言語の提唱に対し，産学関係でこれだけの対応がなされたのは，計算機言語の計算機利用における適格な位置づけが，土壌として根底にあった事情が想起されなければならないであろう。

ここには，規範的な施策を打ち出す政策当局のソフトウェアに対する認識とそれを受けて立つソフトウェアの産学関係者の力量があった。

一般的に，16ビットアーキテクチャさらには巨大技術でない適正技術に対するシステム記述言語の設定には次の3つの方途がありうる。

① 既存言語の延長や改良で乗りきろうとする立場

② 有望な言語，例えばAdaやPascalを導入するかあるいはそれらを参考にして若干の改良を加える立場

③ 欧米の事情を参考にはするが，わが国独自の自主技術としてのシステム記述言語を創出する立場

これらの3つの立場は，全く独立のものではなく，政策当局および産業界・学界の力量によって，相互に変りうる相対的なものといってよい。③の立場をとるにせよ，産業界・学界の力量によっては，①，②の立場を当面とらざるをえないかもしれないし，①の立場をとるにせよ，自然に③の立場に移行する可能性もある。ここにおいては，諸技術のなかにおけるコンピュータひいてはソフトウェアという20世紀の技術に対する関係者の認識が判断の大きなファクターとなっている。

この報告では，①～③のどの立場をとるかについて組織だった選択や判断を行うに至っていない。強いて規範的な立場を表明せざるをえないとすれば，5.1～5.5にわたる各委員の分析や主張にその表明を読みとるほかはない。

<参 考 文 献>

Assembler

- CP／M ASSEMBLER(ASM)USERS GUIDE, Digital Research

Structure Macro Assembler

- Chromod : SMAL／80 Users manual
- 〃 : Structured Microprocesson Programming Yourdon Press Qnc.

PLZ／SYS, PLZ／ASM

- Tod Snook, Janet Roberts, Charic Bass , Armen Nahapetain , Mike Fay: Report on the Programming Language PLZ/SYS Springer Verlag 1978

- Z 8000 Data Book, Zilog Inc. CQ出版社

PL／M

- PL／M－80 Programming manual , Intel
- PL／M－86 Reference manual , Intel

PL／I－G

- Draft Proposed American National Standard Programming Language
- Digital Research : PL／I－80 user's manual
- 〃 : PL／I－80 application manual
- 〃 : PL／I－80 LINK manual

Pascal

- K. Jensen, N. E Wirth : PASCAL USER manual and report 2nd edition , Lecture Notes 18 springer Verlag 1976

- P. Materi : PASCAL VS C , a Subjective Comparison in Lecture Notes in Computer Sciences No.79 Springer Verlag 1980.

C.

- Kernighan, Ritchie : The C Programming Language Prentice-Hall
- C notes Yourdon Press Inc.

○ The Bell System Technical Journal July Aug 1978

Ada

○ DoD : Reference manual for the Ada Programming Language July 1980
（bit 別冊 Ada 基準文法書 共立出版）

Concurrent Pascal

○ B.Hansen : The Architecture of Concurrent Program Prentice－Hall 1977

○ 〃 : The Programming Language Concurrent Pascal IEEE Transaction on Software Engineering, June 1975

○ B.Hansen & Hartman : Concurrent Pascal Compiler for minicomputer Lecture Notes Springer Verlag 1977

○ H. A Slidel, G. Grebe : A Kernel for Concurrent PASCAL Operating System in Operating System North － Holland 1979

Modula

○ N.Wirth : The Modula : System Structuring Facility in High Level Programming Language in Language Design and Programming 1980, Lecture Notes 79 Springer Verlag

○ 〃 : A Personal Computer Based on High－Level Language 同 上

○ A. Richardson : A Critique of Modula 同 上

○ N.Wirth : Modula : A Language for Modular programming Software Practice & Experience 7, 1977

○ 〃 : Use of modula 同 上

○ 〃 : Design and Implementation of Modula 同 上

○ 〃 : Modula－2 Tech Report 3 Institute für Informatic ETH 1980

FORTH

○ John S. James : WHAT IS FORTH Byte Aug. 1980

○ Richard Miller & Jill Miller : BREAKFORTH INTO FORTH Byte Aug. 1980

- Kim Harris : FORTH EXTENSIBILITY : OR HOW TO WRITE A COMPILER IN TWENTY－FIVE WORDS OR LESS Byte Aug. 1980
- Charles H.Moore : THE EVOLUTION OF FORTH AN UNUSUAL LANGUAGE Byte Aug. 1980
- E. D. Rather & L. Brody : USING FORTH FORTH Inc. Apr. 1979
- John Cassady : Stacking Strings in FORTH Byte Feb. 1981
- Ulrich Frei : KNIGHT : A Knight's Tour Problem in MMSFORTH Byte Feb. 1981
- W.R.Stevens : FORTH Primer' Kitt Peak National Observatory
- FORTH Inc. : Micro FORTH Technical Manual
- 〃 : Poly FORTH Reference Manual
- 〃 : Micro FORTH Primer
- L. Forseley : TUTORIAL MANUAL Univ. of Rechester
- S. Ewing : CALTECH FORTH MANUAL Calif. Inst. of Technology 1978
- FIG & EFUG : FORTH 77 FORTH 78 STANDARD GROSSARIES FOR THE COMPUTER LANGUAGE FORTH FORTH Standard Team

LISP

- Winston & Horn "LISP" Addison Wesely 1981
- J.R Allen; Anatomy of Lisp Mcgrow Hill 1978

第6章　マイクロコンピュータ・エンドユーザにおける課題と解決策

第6章　マイクロコンピュータ・エンドユーザにおける課題と解決策

6.1　エンドユーザ問題の背景

インテル社より最初の４ビットマイコン４００４が発表されたのは１９７１年である。それより１０年が経過し，マイコンの応用分野も拡大の一途をたどっている。特に家電製品，事務機器，自動車，自動販売機など日常生活の中に深く入りこんでいる製品に使用されるようになって，不特定多数の人と直接接するようになってきた。

マイコンの利用形態には二つの形がある。一つは各種の機器に組込まれて，主として制御用に使用されているものである。この場合その機器の使用者はマイコンを意識することはなく，単にその機器を操作しているとしか思っていない。この形の代表的なものは家電製品や自動車などに見られる。例えば電子レンジのボタンを押して，料理の内容や時間などを入力している主婦は，自分のつくりたい料理に必要な事項を電子レンジに知らせているということは意識していても，それが内蔵されているマイコンにコマンドを与えてプログラムをしているという意識は持っていない。

他の一つは計算機として使用されるものである。これはパーソナルコンピュータで代表されるもので，マイコンの出現により個人用として使用可能な価格帯で一応の機能を持つ計算機が実現した。このような計算機のはしりは電卓（電子式卓上計算機）と考えられるが，現在のマイコンを使用したパーソナルコンピュータは電卓に比べればはるかにレベルがたかく，計算機と呼ぶに十分価するものである。この分野には他にオフィスコンピュータ，ワードプロセッサなども含まれる。いずれにしても使用者は計算機を操作してデータの処理をしていることを意識しており，どういう手順で処理をするかというプログラムにも関心を持っている。しかし従来の計算機の利用形態と異なる点は，使用者が計算機関係の専門技術者ではなく，むしろ計算機の非専門家が使用することである。この点がマイコンのもたらした使用形態上の基本的な変化である。

このようなわけでマイコンは出現以来１０年にして，計算機（この場合はパーソナルコンピュータなどであるが）のユーザ層に質的な変化をもたらした。ミニコンピュータの歴史はマイコンより長いが，ミニコンピュータはマイコンのようなユーザ層の変化をもたらすことはなかった。その意味ではミニコンピュータは従来からある汎用計算機と同じ線上の商品である。従ってミニコンピュータのユーザに生ずる問題は従来の計算機ユーザの問題と同じ性質のものである。しかるにマイコンの場合にはユーザ層が変化し，非専門家の多数層になったため，発生する問題の様相が変化してきている。問題の根本はミニコンピュータの場合と同じであっても，ユーザの相異によりその問題のあらわれ方や解決法は自ずとちがってくるわけである。ここでは前記の二つの形態のうちの後者の場合，すなわち計算機としての応用における問題について検討する。

本報告書は１６ビットマイコンの応用分野と高位言語について述べているものであるが，マイコンの応用分野についての調査は製品分野別に行われるのが一般的である。これはマイコンが工業製品として扱われユーザは各種分野のアセンブリ企業であることを意味している。従ってこのレベルではマイコンメーカとユーザ企業の関係やその間に存在する問題は明らかにされるが，個人レベル

の末端ユーザの実態やその市場などはわからない。本来，末端のユーザの問題はレベルがちがう問題である。マイコンの応用分野の中にパーソナルコンピュータで代表される製品群がある。従来の製品分野別分類ではパーソナルコンピュータという一つの項目が立っだけで，そのパーソナルコンピュータの使用者や用途の実態はわからない。しかしパーソナルコンピュータの用途は小企業の事務計算，学校の教材，家庭のホビーなど多種多様であるのが実態であり，今後はさらに増えると考えられる。このパーソナルコンピュータの分野は個人レベルのユーザが主であり（これがエンドユーザになる）マイコンメーカと末端のユーザが直接につながる唯一の分野である。将来パーソナルコンピュータが普及すると，エンドユーザとメーカの間のパイプも太くしなければならないが，現状ではエンドユーザの実態がわからないので，メーカへフィードバックすべき問題もわからない。例えば本報告のテーマである高位言語についてみても，技術者レベルでの問題はかなり明確に把握されているが，エンドユーザレベルでは問題の本質さえよくわかっていない。

そこで不十分ながら本報告書にエンドユーザに関する一章を設けて，問題点を検討することにした。検討するべき課題としては

(1) 現状における問題点とその解決策

(2) 将来の一般ユーザ（エンドユーザ）に適したマイコンのハードとソフトの技術

の二つがある。(1)は現在のマイコンを前提として，そのかかえている問題と現状に立脚した解決策を考えるものである。具体的な問題の内容については 6.3 節に述べるが，この問題の範囲には単に技術的な問題のみでなく，経済的，社会的な諸問題も含まれる。(2)は視点を変えて，現状の諸問題の根元になっているマイコン技術の考え方を変えて，より一般のユーザに適したマイコン技術を考察するものである。これは現在の問題に対する直接的な解決にはならないが，将来方向としては重要な課題である。本報告ではまず(1)について検討した結果を説明することにし，(2)についての本格的な検討は別の機会にゆずることにする。なお，本検討では資料の不足も多く，不十分なところも多々あるので，これを第一歩と考え，引続き調査検討をすることが必要である。

6.2 エンドユーザの実状

以下の議論でエンドユーザという言葉を使用するので，ここで説明をしておく。エンドユーザとは処理すべき生データをもち，処理されたデータを直接使用する立場にある計算機使用者である。マイコンの場合を考えるとチップメーカからマイコンのチップやボードを購入してパーソナルコンピュータを組み立てる技術者やアセンブリメーカは，マイコンのユーザではあってもエンドユーザではない。したがってメーカとユーザの間の問題があったにしても，エンドユーザの問題とは若干性格が異なる。この場合のエンドユーザは，パーソナルコンピュータを購入して使用する人であって，一般的には計算機の専門家ではないと考えられる。

マイコンの特徴はこのエンドユーザ層が非常に広いことである。通常マイコンのエンドユーザというと学生，サラリーマン，セールスマン，医者，商店主，主婦などに分類されることが多い。しかし，この分類はエンドユーザの購買目的，価格，ユーザ層の分析など市場調査の場合には適して

いるが，技術的問題の場合には意味がない。この場合にはむしろエンドユーザの技術的レベルの方が問題に関係するので，レベル別にユーザを把握できるような分類が望ましい。なぜなら職業別分類では，同じ職業の人の中でも経験度合いや技術レベルが異なるので，これを一括して扱うことには無理があり，問題の焦点がぼやけてくるためである。そこでエンドユーザを計算機システムに対する反応から次の3レベルに分類する。

① 既製プログラムのユーザ

② 専用プログラムのユーザ

③ 専用システムのユーザ

① の既製プログラムのユーザというのは，単にデータのみを入力して処理させるユーザでプログラムはメーカから供給される既製のものを使用する。このような使用法は電卓に見られる使用法で，他に民生家電関係のマイコン組み込み製品も同じ形式である。またオフィスコンピュータやパーソナルコンピュータにおいても，既製のパッケージソフトウェアを使用して，データのみ入力して処理する場合はこの形式になる。いずれにしても定形的な操作で計算機を使用するユーザであるが，現状では多くのエンドユーザがこの段階にあると思われる。② の専用プログラムのユーザとは，簡単なアプリケーションプログラムを自作するユーザで，マイコンのプログラムの基礎を学習した人がこれにあたる。これらのユーザは趣味でやる人もあれば，直接的な仕事上の必要性によってプログラムする人もいる。このクラスはマイコンの普及により今後共に増加する層であるが，また，それだけ問題も多くかかえている層である。③ の専用システムのユーザは，専用プログラムのユーザよりさらに高レベルのユーザで，ハードウェア，ソフトウェアを含めて自分用のシステムを作製することができる人々である。従ってエンドユーザと言っても専門技術者かそれに準ずる教育をうけた人々や非常に熱心で経験をつんだアマチュアなどである。エンドユーザの中でも人数的には少数派である。

このような3レベルを考え，エンドユーザを分類・対応させて，各レベルのユーザの特徴や問題点を検討するためには，エンドユーザの実態をより詳しく知ることが必要である。しかるにエンドユーザの実態を調べた報告は未だ出されていないようである。従来のマイコンのユーザに関する諸調査は，ユーザを一括して扱っているため，エンドユーザも中間的ユーザ（アセンブリメーカの技術者）も一緒にしている。そのためにこの種の調査では，マイコン普及の過程から見て，かなり初期から教育をうけ実用の経験をもっているアセンブリメーカの技術者層（中間的ユーザ）がユーザ層として浮かび上ってくる。従来のマイコンのユーザは，この種のチップメーカから見たユーザが大多数であったから，この調査の結果も十分意味をもつ結果を示していたが，マイコンの普及が進んで本当のエンドユーザが増加してくると，このままの調査結果では必ずしも正しい結果を示さないようになってくる。そこでエンドユーザを抜き出した調査が必要になってきている。このエンドユーザ調査は時期的に見ても，パーソナルコンピュータの普及がはじまったここ1～2年頃に行なわれることが望ましい。そして調査は1回限りでなく，一定年数ごとに数回は行ない，その間の変化状況がわかるようにすることが必要である。

このようにエンドユーザの実態はよくわかっていないのか実状である。そして大多数のエンドユーザの実状は既製プログラムレベルのユーザであり，家電製品的使用状態にあると考えられる。本報告で取り上げようとしている計算機用途の専用プログラムを作成するレベルのユーザは，数が増加してきているとは言っても，学生を中心にしてホビー人口を加えても未だ少数である。その中では数が多いと考えられる学生について予備知識を得るため，今回試みに大学のマイコンクラブに所属する学生の話しをきいてみた。その話しの中から想像される学生ユーザのイメージは次のようなものである。

(1) マイコンの経験年数は個人差があるが，大学に入学してから始めた人が多く，経験年数はあまり長くない。経験年数の割りには知識が豊富であり，特に経験半年程度の１年生でも知識量は驚くべきものがある。この知識は全てマスコミから得たもので，特にマイコン雑誌の威力が大きいことが感じられる。

(2) 学生の中でも理工系の学生はマイコンについて概念程度は知っており，関心もあるが，文科系の学生で興味をもつ者はめずらしい部類に入る。感触では１～２％もいればよい部類らしい。また女子で興味をもつ者は非常に稀で，ほとんどいないのが実状である。この傾向は一般社会人につながるから，この無関心層に興味を持たせることが，将来のエンドユーザ層の拡大につながることになる。

(3) マイコンクラブに所属するような学生は，かなり多数の者がマイコンのオーナーである。彼等の所有するマイコンセットは２０万円クラスのものが一般的であるが，この価格帯ではフロッピディスクがつかないことが問題である。マイコンに投資できる金額は３０万円が一つのリミットであるらしい。

(4) マイコンの用途は８０％程度までゲームである。その他はシミュレーションやプログラムの勉強である。これ以外での積極的な有効用途は特にないらしい。したがってゲームで遊ぶことと，プログラムを作ることを含めてマイコンをいじることを楽しむことが主たる目的である。しかし，一部の学生はメーカのプログラム作成を下請けすることもしており、セミプロ化しつつあることがうかがえる。

このイメージが学生ユーザの全体像を正しく反映しているかどうかについては疑問もあるが，現状におけるマイコンユーザの一つの形態を示していることは確かである。しかしエンドユーザの一部はさらに低年令層化（小中学生）してきているので，別の形態も存在することが考えられる。また社会人ユーザについては推測程度のことしかわからない。そこでエンドユーザ像がわかる程度の範囲で，知識レベル，利用形態，問題点，将来の希望などを調査すると参考になるデータが得られると思われる。

6.3 エンドユーザにおける諸問題

本節ではエンドユーザにかかわる諸問題を説明するが，これらの問題の中には必ずしもマイコンのエンドユーザに個有のものではなく，広く計算機のユーザに共通するものもある。これらの問題

を特にマイコンのエンドユーザの立場に立って述べることにする。なお，問題は性質上技術的問題，経済的問題，社会的問題に分けられるので，この順序にしたがって列挙する。

(1) インタフェースの標準化

パーソナルコンピュータにもキーボード，フロッピディスク，プリンタ，カセットテープなどの標準入出力機器やユーザ固有の専用機器が接続されてシステム化されるようになると，インタフェースの違いが問題になる。A社のシステムにつながる機器がB社のシステムにはつながらないということは日常経験することで，ユーザに多くの不便をもたらしている。特に資金的に予裕の少ない個人ユーザの場合には，システムが大きくなる程ＣＰＵの選択が限られレベルアップも思うようにできないということになる。このような不便はエンドユーザだけでなく，アセンブリメーカなどでも痛感されており，システムバスの標準化が要求されてきた。現在のところ一般的なシステムバスとしては，計測器を中心にしたＩＥＥＥ４８８バス（ＨＰ社の提案をもとにしたのでＨＰＩＢとも言われる）と原子力分野の計測・制御用のＩＥＥＥ４８３バス（ＣＡＭＡＣバス）が標準化されているが，これ以外に標準化のきまっているバスはない。アマチュア用の装置に使用されているＳ１００バスは市場において実質的な標準になっているが，このバスは線数が多く技術的には良い仕様とは言えないので，一般的な工業標準としては問題がある。仕様的には大手のマイコンチップメーカの使用しているバスの方が良い。インテル社のＭＵＬＴＩバスやモトローラ社のExorciserバスなどが有名であるが，今のところこの種のバスが標準になるような動きはない。そう言うわけで技術的にすぐれたシステムバスが直ちに標準化される動きはないが，ユーザとしては出来るだけはやい時期に良いバスが標準化され，同一規格のコネクタで同じ入出力装置をどのシステムのバスにでも接続できるようになることを希望している。

(2) エンドユーザ向き言語

エンドユーザがプログラム開発をする比率は今後増加してゆくと考えられる。その場合最も重要な問題はエンドユーザに適した使いやすいプログラム言語である。この問題には，現在ある言語の中でどの言語が一番適しているかということと，将来どのような言語を開発することが望ましいかという二つのテーマがある。しかしいずれにしても，一つの言語で全てのエンドユーザのあらゆる応用面に最適にするということは考えられないので，応用別に最適な言語を揃えるという傾向になると考えられる。しかし，この場合全ての用途に対して核の部分は同一にして，用途別の部分はその上にのせるような構造が望ましい。そしてユーザのレベルの進歩に対応して核の部分から順次習得してゆくことにより，一つの用途に対してのみならず，他の用途のものに対しても容易に理解し，使用できるようになる。このような言語の仕様は，既存言語の延長では不十分で，ユーザの真の要求をつかみ出すことが必要である。いずれにしても機械（ハードウェア）を売るために動かして見せる道具としての言語ではなく，ユーザの立場で自由に使えて役に立つ言語を考えることが望ましい。なお，本報告書の主テーマの一つが高位言語に関することなので，エンドユーザ向きの言語の問題は6.4節で改めて詳しく論ずることにする。

(3) パッケージソフトウェアの問題

エンドユーザの数が増加し，マイコンの使用目的が多様化する一方，現状では全てのエンドユーザがプログラムを作成するわけでないとすると，既製のプログラムの提供，販売が問題になる。これがパッケージソフトの問題である。マイクロコンピュータやパーソナルコンピュータとパッケージソフトの関係は，丁度レコードプレーヤとレコードの関係と同じであり，ハードウェア（プレーヤ）がある程度普及しないとソフトウェア（レコード）も売れないが，あるレベルをこえると今度は逆にソフトウェアが充実することによってハードウェアの販売量がのび，一層広く普及するようになる。パーソナルコンピュータもそろそろソフトウェアがハードウェアを引張る時期に近ずいてきたと考えられる。米国ではパッケージソフトの開発と流通が盛んであるが，我が国ではまだ未成熟な点が多い。最近のマイコン雑誌にはパッケージソフトの広告も目につくようになったが，弱小メーカが多くて，メーカの技術力，製品の信頼性，保守能力などに疑問が残る。また価格についても特に一般的な評価基準がないので妥当性がよくわからないことが多い。メーカやしかるべきソフトハウスの作成するソフトウェアに比べると，マイコン用として広告されるソフトウェアには価格も桁ちがいに安いものがあるのが実情である。そこでパッケージソフトを普及し正しく流通させるためには，プログラムの内容を保証する何らかの検定システムが必要になる。このシステムではプログラムの内容の正しさる保証すると共に，検定を通過したソフトウェアのリストを公表して，ユーザの普及を推進する。こうするためにはプログラムの仕様をきちんと書くことが要求されることになるので，この過程で仕様の記述方法や記載すべき内容など多くの面で標準化がすすめられることにもなる。エンドユーザの多くが既製プログラムのユーザでマイコンを一種のブラックボックスとして使用するとすれば，各種のアプリケーション用パッケージソフトは今後ますます重要なリソースになるので，流通体制の整備が急務である。

(4) 周辺機器の低価格化

マイコンのエンドユーザが購入するシステムの価格は，特別の場合を除くと２０万円から５０万円程度と考えられる。その中でも２０～３０万円程度が多数を占めるであろう。この価格帯はコンポーネートステレオなどとほぼ同じである。現在市販されているパーソナルコンピュータも，本体価格は多くのものが２０～５０万円程度になっているが，問題は計算機の場合本体だけでは能力が限られるということである。現在の多くのシステムでは本体としてＣＰＵ，メモリ，キーボード，ディスプレイ，電源を持っている。しかし実用上からはこれらの他にプリンタ，外部記憶装置が必要であり，用途によってはその他の装置がつくこともある。このような周辺装置がつくと，システム価格は急激にたかくなり，本体の数倍になることもめずらしくない。このことがユーザに高価格感を与え，周辺装置の低価格化の要求をもたらすもとになる。いま代表的なシステムの価格構成概算を考えてみると，本体部は２０万円，プリンタ７万円，ミニフロッピーディスク装置デュアル３０万円，カセットテープ装置１２万円，カラーディスプレイ装置１０万円である。さらに多数の装置を接続するためのバス拡張ユニット１５万円や各種ケーブルやアダプタ類などを加えると，本体の価格はともかくとして，ディスプレイ，プリンタ，ミニフロッピーディスク装置をつけたシステムは８０万円程度になり，本体の約４倍の価格になる。周辺装置の中で不可欠のものはプリンタと

フロッピーディスク装置である。プリンタは最近マイコン向けの低価格のドットプリンタが販売されるようになり，購入しやすくなった。フロッピーディスク装置はまだその他の装置に比べると高価である。フロッピーディスク装置は使用上の便利さを考えると2セット（2ドライブ）が必要であるから，システムの中では最も高価な装置になる。フロッピーディスク装置の低価格化はユーザの強い要求であり，フロッピーディスク装置を含めて30万円程度になることが希望されている。一方，ミニフロッピーディスクでは大容量化もすすんでいるが，一般のマイコン用としては未だ144Kバイト（フォーマットあり）の記憶容量をもつ片面形が使用されている。両面形などの大容量化は等価的に記憶コストを低下させることになるので，はやく大容量化ミニフッピーディスクが同程度の価格で標準的に装備されるようになることを希望する。

このような標準的周辺装置は量産性があり技術開発も積極的にすすめられているので，低価格化も時間の問題という面があるが，もう少し特殊な周辺装置や回路の低価格も重要な問題である。一例として最近注目を集めている音声合成出力装置や音声入力装置を考えてみると，まだまだ高価でマイコンの入出力装置とするところまではいっていない。しかし音声合成用ＬＳＩは各種開発されてきているし，米国ではマイコン用の音声入力用ボードが15万円程度で市販されているので，価格が下らないわけではない。またもっと基本的な回路としてはＡＤ／ＤＡ変換回路があるが，ＡＤ／ＤＡ変換ＬＳＩが多数開発されていて，安い価格で販売されているのに対して，ＡＤ／ＤＡ変換装置やボードのレベルになると高価格になるところが問題である。要するにこの種の新しい入出力装置では，中心部分がＬＳＩ化されて低価格になってきているのであるから，それに見合って装置価格も低価格化することが必要である。マイコン用としては余分の機能を追加して高価格になるよりも，必要な機能のみで低価格になることが望ましいので，メーカもこの方向で努力されることを希望するものである。

(5) 中古品と下取り

現在一般のエンドユーザに対するパーソナルコンピュータの販売はマイコンショップ，一部の電気店，百貨店，事務機店などで行なわれている。これらの店は新品の販売をしている。マイコンはレベルアップの行われる商品で新製品や上位レベルの製品に買い替えられるが，この場合システム全体がかわるわけではなく，本体のみや，一部の周辺機器のみが買い替えられる場合も多い。この場合古い品の下取りと共にこれを中古品として販売するシステムができていることが望ましい。マイコンの販売量も増加し，販売年数から見ても中古品が出る時期である。このような中古市場は別の形の低価格品供給法である。中古品の販売店はまだ一般的でなく，大阪に委託・中古品コーナを設けたマイコン販売店がある程度にすぎないが，月間20～30台の販売が見込めるということである。パーソナルコンピュータ市場は多数のメーカが新製品を出して競争しており，商品寿命も短かくなっている。ユーザも高機能機種へ買い替える機運が高まっているとともに，低価格品を求める新ユーザ層も増えてきている。ユーザにとっては不要品となるものが買い替えの頭金になり，また新品同様の再生品が保証，サービスつきで手に入るということは非常に望ましいことである。

(6) 入門教育の充実

一般のエンドユーザ層に対しては，適当なマイコンの教育システムが存在しない。企業では社員教育の形でマイコンの教育を行っているし，技術系の学校では技術者の卵として学生に教育をしているが，これら教育はマイコンを扱える技術者の育成を目的としているので，この範囲に含まれない一般のユーザには無関係である。また各種の講習会も主として企業の技術者を対象にしているので事情は同じである。従って学生でも文科系や計算機に無関係の分野の技術系学生にとっては，マイコンを教えてもらう機会はない。また企業に関係ない人々も同じである。このような人達がマイコンを勉強する道は本による独習か，販売店における口伝えしかない。しかし独習はむずかしく一般に誰でも可能というものではない。その理由は大部分の入門書が専門家によって技術者向けに書かれているので，それ以外の人達にとっては，まず使われている言葉からして理解できないことになる。この点ではメーカのカタログや取扱い説明書も同様に技術者指向であり，相当にむずかしいので一般のユーザにはすぐに理解できにくいと思われる。残る方法は販売店の店員にきくことであるが，これも店員のレベルや質の問題があり必ずしも十分とは言えない。実際問題としてあるレベル以上のむずかしさになると店員では対応できなくなる。

教育については小中学生に教育したらよいという考えもあるが，これは１０年，２０年後には役立つであろうが，ここ数年以内の問題の解決にはならない。いま必要なのはユーザの底辺を拡大し，非専門家のエンドユーザのレベルを上げることである。そのためにはまずマイコンに対する恐れやアレルギーをなくすような低レベルの導入教育からはじめて，次第にレベルを上げてゆくような一般向け教育体系をつくり，実施するシステムの確立が必要である。

(7)　コンサルティングの充実

(6)で述べたのはエンドユーザに対する入門教育の問題であったが，ここで述べるのはあるレベルに達したユーザを対象にしたコンサルティングシステムの問題である。これには二つの面がある。一つはエンドユーザの技術上の指導，相談をする体制である。将来一般のエンドユーザが応用プログラムを作成したり，専用システムを開発したりするようなレベルになるとすると，当然それぞれのレベルに応じて技術的な問題にぶつかることが考えられる。この時これらのユーザを指導し，問題解決の相談にあたれる何らかの社会的なシステムが必要である。現状ではメーカも不親切であるし，わずかに力量のある販売店の店員がいるだけであるが，これも指導や相談を本務とするわけではないから，実際にはこのような体制はないということになる。しかしエンドユーザに対するこの種のサポートがないと，エンドユーザがプログラムを作り，実用にするということはむずかしく，エンドユーザは最後まで少数の専門家を別にすれば，単にボタンを押すだけのレベルにとどまることになる。

第二の面は単にエンドユーザの指導，相談にのるだけでなく，より積極的にユーザの作成した応用プログラムやシステムの性能保証をする機関としての役割をすることである。多数のエンドユーザがプログラムやシステムをつくり，これを実用するようになると，当然社会とのかかわり合いが生じてくるので事故防止のことを考えておかなければならない。ここで事故というのは特にプログラムの誤りによる誤動作だけを考える。マイコンが何らかの機器を操作する場合，プログラムの誤

りにより予期せぬ動作をしたり，十分な対応動作ができないと事故につながる。そのためプログラムに事故につながるような欠陥がないことをチェックする何らかの社会的な体制が必要になる。またハードウェアの不良に対しては標準的なテストプログラムを作成してパッケージソフトの一つとして販売するなどの方法が考えられるが，このときのテストプログラムの作成を担当するような機関も必要である。そこで以上のような種々の機能をもつ公的または準公的な機関が全国のエンドユーザの身近に存在するようになれば，エンドユーザのレベルを向上させ，ユーザ層の拡張にも役立つと考えられる。

6.4 エンドユーザ言語の現状と展望

これまで述べたような諸問題の中で本節は特にエンドユーザ言語について現状と将来展望を論じる。

6.4.1 エンドユーザ言語とは

第4章，第5章で述べられているように，今後のマイコン市場でソフトウェアの比重は一層高まると見られている。従ってプログラミング言語の問題が重要なのである。

第5章では各種のマイコン製品のソフトウェアを製造する観点から，システム記述言語を取り上げた。

本節では，エンドユーザが使う言語，すなわち利用者の観点からエンドユーザ言語というものを考える。

エンドユーザ言語としては，「エンドユーザが現在利用している言語」という定義と「エンドユーザ用に設計された言語」という定義とが考えられる。現状ではこの両者は混用されているが，将来は「エンドユーザ用に設計され，エンドユーザに利用されている言語」になるものと考えられている。

エンドユーザ言語の設計においては，システム記述言語の場合とは異なり，利用者側に電子計算機システムに対する知識を仮定できない。従ってシステム記述言語の場合とは異なり，次のような項目を考慮せねばならない。

① 電子計算機の知識の無いユーザでも使用できること。

② エンドユーザ言語の文法や使用法が，容易に学習できること。

③ 作成したプログラムの動き方が容易に理解できること。

④ プログラムのデバッグや改良が容易なこと。

また，エンドユーザにとっては，図形や各種の文字集合やグラフの使用が重要であることから，

⑤ データとして各種の文字集合や図形，グラフが扱えること。

も必要な機能となるであろう。

さらに，言語そのものの機能ではないが，その言語の属性として，

⑥ この言語で書かれたプログラムが多数流通していること。

⑦ この言語の標準仕様が定まっていて，各種のハードウェア（システム）上で使用できるようになっていること。

が実際に必要となるであろう。

また以上の項目を振り返ってみると，いわゆるコンパイラによるソース・プログラムから機械語のモジュールへの変換だけではエンドユーザ言語は成り立たないことがわかる。

すなわち，プログラム作成システムという全体的立場が必要となる。

⑧ プログラミング・システムとして，プログラムの作成，変更，保守を支援するような言語体系になっていること。

以上の８項目がエンドユーザ言語に必要な機能として挙げられる。

6.4.2 エンドユーザ言語の現状

前項で述べたようにエンドユーザ言語の定義には議論の余地はあるものの，必要な機能は大体絞ることができる。

次に，現状について概観する。

(1) エンドユーザ言語の現状概観

エンドユーザ言語の必要性は，この２，３年になってやっと認識され始めた状態である。ハードウェア技術の進歩により，マイクロプロセッサの価格が低下して，個人が計算機を使えるようになったのは，この２，３年のことである。

従ってエンドユーザ言語として取り挙げられたのは，すでに大型機等で開発済みのプログラミング言語で，前項で述べた必要事項を満たしているものということになる。

数年前の実状はもっとお粗末であった。メモリの価格がまだ高価であったことも災いし，エンドユーザはとにかくそのマイコン上で動かしうるプログラミング言語を使うしかなかったのである。

現在でもＢＡＳＩＣがエンドユーザ用言語として認識されている大きな理由の１つは，BASICならどのマイコンでも動くし，またＢＡＳＩＣしか使えないマイコン・システムが多いからでもある。

さて，本項では，ＢＡＳＩＣのように普及してはいないが，その言語上の特徴や設計目的などから，近将来，エンドユーザ言語として広まる可能性を持っている２つの言語についても検討し，報告することにする。

その言語とはＬＩＳＰとSMALLTALKである。BASICを含めて，これらの現在使用可能なエンドユーザ言語の特徴は，

① 会語型であること。

② 文字を割合簡単に扱えること。

③ 文法が比較的簡単であること。

④ プログラムの編集が比較的容易なこと。

が挙げられる。

これらの機能は，必要項目（①，②，③，④）を満たすものである。

(2) ＢＡＳＩＣ

ＢＡＳＩＣは今日マイコン入門者の必須言語としての位置を占めている。

ＢＡＳＩＣの強味は第１に，大抵のマイコンシステムで稼動していること，第２にＢＡＳＩＣで書かれた既存ソフトウェアの蓄積が挙げられる。

元来ＢＡＳＩＣは米国ダートマス大学において，１９６４年にタイムシェアリングの計算機用に開発されたものである。当時はコンパイラで動いていたが，元の意味（Beginners Allpurpose Symbolic Instruction Code）通り初心者用に設計されたプログラミング言語なので習得は容易である。

特に，現在多くのマイコンで使われているようなＢＡＳＩＣインタープリタは，プロセッサが小さく，プログラムの動きが判りやすく出来ているので便利である。

また言語仕様中にエディタが組み込まれているので，プログラムの修正や保守が便利である。

さてＢＡＳＩＣは以上述べた様な利点をもつが，やはり古い言語であり，今日の技術水準でエンドユーザ言語として考えるには以下のような欠点がある。

① 言語の構造が明確ではない。従って大規模なプログラムを作るのが困難である。

② プログラムが逐次制御で動くハードウェアを前提としているので，ユーザが計算機の仕組を理解する必要がある。

③ プログラム変数が極端に制限されており，プログラムが暗号的になり，理解しにくい。

④ 扱えるデータが数値と文字列に限られている。

⑤ プログラミング・システムを構築しようとした場合，プログラム自体の処理が難しい。

(3) ＬＩＳＰ

ＬＩＳＰもＢＡＳＩＣ同様１９６０年代に開発された言語であり，同じく大型計算機のＴＳＳ端末で使用されてきた言語である。

ＬＩＳＰの応用分野はＢＡＳＩＣと異って人工知能と呼ばれる分野を主とする。具体的には，記号とその樹状の構造とを扱う。

ＬＩＳＰの処理系はＬＩＳＰ自体で基本的には６０行位で記述することができる程に簡単なものである。しかし，プログラムをデータ同様に樹状に表現するために記憶領域がかなり必要となる。これが現在のマイコンでＬＩＳＰが余り用いられていない主要原因である。

ＢＡＳＩＣと比べた場合にＬＩＳＰはエンドユーザ言語として次のような項目が魅力的である。

① 言語構造が極めて単純で，数学的である。ＬＩＳＰのプログラムはすべて（＜関数＞＜引数$_1$＞＜引数$_2$＞…）という構造をしている。

② プログラムの実行を，方程式の形で考えたり，計算機への指令で考えたりできる。ただし実際の処理は逐次的となる。

③ プログラム変数には実質的に制限が無い。

④ データとして数値，文字以外に，リストという構造を扱うことができる。

⑤ プログラムとデータが同一のリストという形式で実現されているので，プログラムを変更する処理をプログラムで容易に実現することができる。

⑥ 文字列の扱いが簡単なので，計算機システムと人間との対話を実現するのが容易である。

特にＬＩＳＰのもつプログラムとデータの同一性は，「人間の知識」を計算機に蓄えて改良，利用するのに非常に役立っている。このような「人間性」を言語システムが備えることにより，エンドユーザが安心してプログラムを作れる環境を実現することが可能となる。

ＬＩＳＰはこのように将来のエンドユーザ言語として魅力的な機能を備えているのだが，如何せん１９６０年代開発という時代的制約から以下のような欠点も持っている。

① 図形を扱う機能が無い。

② 並列的な処理を記述する能力が無い。

③ データと手続きとの関係が規定されていない。プログラムが暴走するのを事前にチェックする機能が乏しい。

(4) SMALLTALK

SMALLTALKは１９７０年代に入って「パーソナル・コンピュータ」を念頭に置いて開発された言語である。

歴史が新しいだけに過去のエンドユーザ言語の持っていた欠点を是正する新しい試みが見られる。例えば，

① 図形を容易に扱える。画面上の位置を指定するデバイスが付属しており，その情報をプログラムで扱えるようになっている。文字の形（フォント）もユーザが変更できる。

② プログラムの実行制御がメッセージによる起動方式になっているので，並列処理が容易に記述される。

③ データに対して，これに施せる手続きを規定する。データを主体にしたプログラムを作ることができる。

といった機能を備えている。

SMALLTALKは現在のところ，Xerox 社のＡＬＴＯ等のパーソナル・コンピュータでしか稼動していない。これはディスプレイや入力装置に特殊なものを使用していることにもよる。従ってSMALLTALKは，BASICやＬＩＳＰのようには各種のハードウェア上で稼動していない。

またSMALLTALKのプログラムはテキストの形で蓄えられており，ＬＩＳＰのような構造体として実現されていないので，「知識」をプログラム化し活用する場合に問題があるかもしれない。

6.4.3 将来展望と課題

エンドユーザ言語の本格的な開発はこれからの課題である。前項で紹介したエンドユーザ言語の内，エンドユーザを明確に意識して開発されたのはSMALLTALKだけである。そのSMALL-

TALKに対しても，開発者のKayが自ら不満足であると表明しているように，理想にはまだほど遠い。以下にエンドユーザ言語の将来と課題について述べる。

(1) エンドユーザ言語の将来

エンドユーザ言語の将来は計算機の大衆化にかかっている。ここ数年の動向から予測すれば，数年後にはパーソナル・コンピュータが現在の電卓なみに大衆化することは可能である。

ただし，電卓と違って計算機にはプログラムが必要である。単なる数値演算では電卓に劣り，簡単な情報検索では情報サービスに負けてしまうパーソナル・コンピュータが，大衆化するキー・ポイントはやはりプログラムの多様化と発展性である。

従って計算機の大衆化とエンドユーザ言語の発展は車の両輪である。これはまた，エンドユーザ言語の開発が遅れると，計算機の大衆化が遅れ，結果としてエンドユーザ言語の開発需要が顕在化されないという危険を孕んでいる。

現在進行しつつある情報化社会においては計算機利用技術は過去の「読み書き算盤」に相当する基本技能となることは疑いない。実際今日の日本経済の活力の源の1つは，現場技術者のエレクトロニクス化への基礎技能としての理数科教育の普及である。将来は現在の理数科教育に加えて，計算機を用いる情報処理教育が必要になる。

ゆえにエンドユーザ言語の開発は国家的見地から必要不可欠なものとなるであろう。また大衆化による膨大な市場は各メーカにとっても看過し難い大市場であるから，基本的には開発努力が続けられるであろう。

しかしながら，エンドユーザ言語の開発には種々の難問がある。冒頭に挙げた8つの機能を満足させるのは決して容易ではない。

(2) 課　題

理想的なエンドユーザ言語を開発するためには次のような課題がある。

① 人間の知的な思考過程の解明

エンドユーザ言語ではユーザに計算機の知識を仮定できない。ユーザの知的な思考を自然に表現できる言語でなければならない。そのためには一般人の知的思考過程自体を解明しなければならない。

② 対話機能の研究

現在までのプログラミング言語の多くはユーザが一方的にプログラムを作る形式を仮定している。ところがエンドユーザ言語には対話機能が不可欠である。しかも将来の計算機システムでは現在のキーボート以外に多くの対話手段が考えられる。最適な対話機能の研究を地道に進めていかねばならない。

③ 計算機の理解能力の向上

対話においては手段だけでなく質の向上も問題となる。膨大な数のエンドユーザのプログラム相談は計算機自体の手助けなしには不可能である。エンドユーザ言語にはエンドユーザに対する相談機能，すなわち対象とするプログラム自体やエンドユーザの質問や指令に対す

る理解能力が必要となる。

④ 計算機の低価格下，コンパクト化

いわずもがなのことではあるが，エンドユーザ言語を利用するのは，一般大衆なので計算機システムが高価であったり，かさばるものであってはならない。このハードウェアの開発も決しておろそかには出来ない

6.5 今後の取り組み方

これ迄繰り返し述べてきたように，エンドユーザの問題はマイコンの発生，普及とともに生じた新しい課題である。この問題への取り組み方はまた，２１世紀における国家経済，世界経済に大きく関与するものである。マイコンの大衆化によって，経済的な生産活動ばかりでなく，日常生活，社会生活の形態が革新的に変化することが予想されているからである。政治体制ですらこの急激な変革の波に呑まれないとは限らない。

従って，エンドユーザ問題に対する取り組みは慎重でなければならぬが，同時に技術革新の波に取り残されない様に迅速かつ機敏に対策を講じていかねばならない。

今後の取り組み方に関しては次に述べる３つの大きな目標を同時に達成する方向で考える。

① 実態および動向の正確な把握

② 本質的研究課題に対する国家的取り組み

③ 技術的問題の解明とその解決

6.5.1 実態および動向の正確な把握

大衆としてのエンドユーザは現在その存在が明らかになりつつある状態で，しかもエンドユーザ自身が自分の欲するものを明確には認識していないのが大雑把な現状である。

今後の取り組み方として，まず第一に必要なのは，このエンドユーザの実態を把握し，その問題点を解明し，大衆としてのエンドユーザが進む方向を明確化することである。

実態把握においては現在のマイコン・ユーザに対するアンケートだけでは不充分である。向う１０年ないし２０年間における技術的変革についてのビジョンをもち，現状にとらわれないで将来を見通す指導的な層のもっている意識を把握しておかないと，エンドユーザ問題に対して後手後手にまわる恐れがある。

第１に各界の有識者，特に未来技術に卓見をもつ人や，現状にとらわれない見識家などに，将来のエンドユーザ問題を尋ねてゆく必要がある。

同時に，１０年後および２０年後の理想的エンドユーザ用マイコンに対する技術的な商品イメージを確立する必要がある。

いわば「これが欲しい」というユーザの需要と，「これが可能だ」というメーカの供給とをにらみあわせて将来動向を形成しなければならない。もち論，現在エンドユーザについて日本より進んでいると見られる米国の現状なども大いに参考となろう。

またこの実態，動向の把握は定期的に見直し，軌道修正してゆかねばならない。

6.5.2　本質的な研究課題に対する国家的取り組み

エンドユーザ問題には「人間の思考」に関わる部分が少なくない。例えば理想的なエンドユーザ言語を考える場合，これは「人間はどのように思考するのか」という問いを抜きにしては答えられない。

またエンドユーザに対するシステム・インタフェースを考える場合には，「どのような通信手段が人間にとって最も効率的なのか」を解き明かさないと効果的なものは出来ない。

エンドユーザ問題においては従来とられていた電子計算機を主とし，人間を従とする考え方から，人間を主として電子計算機を従とする考え方に発想を転換しなければならぬ。

このような人間に関わる研究課題については，問題が極めて難解であり，短期的な解決が難かしいので国家的規模で取り組む必要がある。次のような問題が考えられる。

① 人間の思考過程の解明による理想的なエンドユーザ言語の研究

② 人間の思考表現における本質の解明によるエンドユーザ言語の理想的な表記法の研究

③ 人間の認識過程の解明によるシステムの理想的な反応方式の研究

④ エンドユーザの意図を組み取り，適切に要求を表現する人工知能的システムの研究

⑤ 各個人の素質にあった思考法ならびに表現法を発展させる教育システムの研究

以上のような高度の研究課題の他に，国家的見地から検討しなければいけない問題がある。それは，エンドユーザの大衆としての出現による政治的，社会的，経済的な変革に対する国家制度上の検討である。

例えば，現在課題にあがっている「在宅勤務」や「在宅学習」が技術的に可能となった場合に，国家は制度をこれに対してどう変更すべきか，あるいは変更すべきでないか，という事柄である。

従来この種の技術革新に対しては行政上，法制上の措置が常に後を追う形式で整備されてきた。しかしながら６０年代の公害による被害を考えてみても，この種の遅れがもたらす危険性は決して看過しえないものである。

エンドユーザの出現自体は一見無害な技術革新に見えるかも知れないが，その影響の規模は過去の技術革新のどれよりも一層大きいものかも知れない。エンドユーザに対する行政上の保護という問題一つ取り上げてみても本当に何が起るかわかったものではない。

このような政治，社会，経済の変革に対する準備もまた国家的規模で進めなければなるまい。

6.5.3　技術的問題の解明とその解決

技術的問題としてまず取り挙げられるのは

① エンドユーザ用システムの低価格化と高機能化である。前記の国家的研究の遂行によって明らかとなった諸要素を経済的に見合うように開発することである。

ここで問題になると思われるのは，２次記憶装置の小型，軽量，大容量化と，表示装置

の小型，高解像度化である。もち論，システム全体の低価格化は見逃せない技術課題である。

② エンドユーザ用プログラミング言語の開発も技術的課題にあげられる。理想的な言語開発は長期的視点から国家的規模で進めなければいけないが，それ迄手を拱いて待っているわけにもゆかない。特に国内においては，SMALLTALKのような言語の開発が極端に遅れているので，例えばSMALLTALKを試作しその実用性を実験的に確認するといった実践的な取り組み方が必要となる。

またマイコン用のＬＩＳＰ言語の開発も実践的見地から大いに興味あるテーマである。これらの開発―試用―評価―改良といった地道な手順を踏むことによって，研究ベースを支える技術ポテンシャルの増加と，技術的な課題の明確化が可能となり，ひいては理想的なパーソナル・コンピュータ・システムに対するコンセンサスの成立が期待される。

③ 教育問題

上記のようなLISP，SMALLTALK といったエンドユーザ言語を開発し試用する上で問題となるのは教育体制である。

将来エンドユーザ市場が成立する場合にも問題となるのはこの教育体制の整備と，具体的な教育手法とである。

国家的規模でエンドユーザ問題を将来の技術立国の柱と考えるならば，エンドユーザ教育の問題は義務教育課程および初等技術教育課程において施すべき性格のものである。

技術的見地からは，そのような国家的教育体制が整備される以前に，どのようにして膨大な教育需要をさばいてゆくかということになる。ここで鍵となるのはＣＡＩないしパソコン自体を利用した自習システムとなるであろう。

ＣＡＩの基礎となる教材や教育法の開発と安価で効果的ＣＡＩシステムの開発が必要となる。

④ プログラムの流通問題

ソフトウェアのもつ再利用性は具体的にはプログラムの流通となって保証される筈である。エンドユーザがプログラムを書いたり変更したりすれば，その出力であるプログラムの流通が必然的に生じる筈である

また，このような流通市場が形成されて始めて大衆的なエンドユーザが出現するに違いない。この市場がレコードにおけるように少数のメーカが大衆にプログラムを供給する形になるのか，短歌や詩のサークルのように同好者が寄り集まってプログラムを発表，利用する形式になるのかは，現状では判断できない。どちらの要素もある程度取り込む形にはなると思われる。

このプログラム市場をどう育成し，どう発展させるか，また技術要素として何が必要かの解明が必要である。

⑤ ユーザの権利保護機関

大衆としてのエンドユーザが出現すれば，これの権利保護をどうするかが技術的問題となろう。

直接的に問題となるのはソフトウェアの品質保証，ユーザ作成プログラムに対する著作権の問題，欠陥商品によるデータ破壊の保証の問題，欠陥商品の誤動作による損害保償の問題等である。

この権利保護機関は弱者としてのエンドユーザ大衆を保護するという機能以外に，このエンドユーザ大衆の創意工夫を結集して，より良いエンドユーザ・システムの開発を促進するという機能も果すことができる。

以上極く簡単に取り組み方について述べたが結論としては，

A．エンドユーザ・システムの将来動向調査

B．基本的な問題に対する国家的研究テーマの選択と計画策定

C．技術的問題に対する迅速な対応，具体的には，LISP，SMALLTALK といったエンドユーザ言語の試作，評価，低価格システムの開発，教育システムの作成

といった取り組みが現在必要である。

おわりに

おわりに

１６ビット・マイクロコンピュータは，近来その第２世代を迎えたともいわれ，技術面で一定のレベルにまで完成したものとなるとともに，価格面でも十分実用に耐え得るものとなってきた。一方では８ビットまでのマイクロコンピュータでは満たされ難い高度な知識・技術集約型のニーズ，例えばオフィス・コンピュータや高精度，高速の制御機器への利用の要求が高まるすう勢となり，今や１６ビット機種が比重を高めるマイコン第４世代の幕が開かれた。

本報告書では，まずマイクロコンピュータの技術・利用の体系について概観し，１６ビット開発・利用への流れを把えた。次いでそのような１６ビット機の現状をハード，ソフト両面から詳細に調査するとともに，将来を含めて応用の背景と動向を探った。また，大きな課題である高位言語利用について，言語機能への要求の分析と評価を中心に，綿密な検討を加えた。そのために現存する代表的高位言語のすべてを対象として，各言語の特徴を比較的に把握できるような評価方法を案出し，総合評価の試みを通して将来への展望を見出そうとした。さらに重要性がいわれながら意外と実態把握の遅れているエンドユーザの問題にも注目した。エンドユーザとは何であり，その希求するところ，抱える問題は何であるか。それに答えることなしには，長期的な技術の発展は望めないと考えたからである。

それらの調査と考察はかなり思いきった新しい内容を含むものであり，決して容易な仕事ではなかった。以下各章の内容の要点を浮き彫りにしながら，全体のまとめを述べることにしよう。

第２章では，マイクロコンピュータのもつ技術・経済・社会的インパクトの大きさと広さ，将来発展のさらに大きな可能性，そしてその可能性を現実のものとするために解決すべき課題についてまとめた。特に産業経済上も，わが国の国際的立場からも，今後さらに高度な知識集約型先進技術の開発を進める必要性を強調した。そのためにマイコン高位機種の開発と活用，高位言語の活用などによるソフトウェア生産性向上，ハード及びソフトの標準化とフレキシビリティ（流通性，融通性など）改善などを，重要課題として指摘した。

第３章では，現在乃至極めて近い将来市場に登場する代表的１６ビット機種を総覧した。各機種の特徴を比較できるように，(1)ハードウェアの構成と特徴，(2)ソフトウェアの特徴，(3)システムの構成及び(4)開発サポートの各項目を設け，なるべく記述の内容とレベルを統一したものとするよう心掛けた。いわゆるミニコン系１６ビットマイコンについても触れたので，読者の参考に供するところが多いであろう。

第４章では，実際的立場から１６ビットマイコンの導入の背景を述べた後，１６ビット機のどのような機能・性能が，どのような応用分野でそれを選択するモーティベーションとなるかについて分析し，一覧表の形にまとめた（表４－２）。選択・移行の条件としては，演算処理機能，その速度，演算精度，メモリ空間の大きさ，（多重）割込機能の豊富さ，データ幅及び通信機能を取上げた。１６ビット機（及び他の機種）を利用目的に応じて選定する際に，便利な資料となろう。さらに具体的に利用動向をみるために，民生・家電，事務・商業，工業等々の各分野ごとに商品開発のねらいを示し，

多数の具体例を洗い出した。ねらいとしては，情報化，効率化，機能拡大，高精度化，省エネルギー省資源，省力，安全性，利便性，信頼性，福祉，能力開発などが取り上げられた。またこの章では今後の市場動向についても一つの数量的データを示した。

第５章はマイコン用システム記述言語についての評価と展望にあてられた。この章には本報告書の中でも最も多くのページ数があてられているが，１３種の代表的高位言語について詳細な比較評価と今後の開発，活用について展望を示したものである。システムに対する要求がますます高度で複雑・大規模なものとなるにつれ，ソフトウェアの開発の生産性，信頼性，保守性などが大きな問題となるが，その解決のために高位言語の活用は必然となってくる。

ただ現状でも既に多様な言語が登場しており，その全ぼうとそれぞれの特徴をつかみにくいのが難点である。特にユーザ（システム設計者及びエンドユーザ）としてみれば，一つの言語を修得したところにまた別の言語が脈絡もなく現れて，応待にとまどっている。その結果，長い将来にわたってどのような考え方で高位言語を選択し，活用すればよいのか，その基本的指針を見失い，自らのライフサイクルに対する自信をすら喪失する現象を生じている。

この委員会では，この難問に答えるため，まずシステム記述言語のあるべき姿，それに要求される機能について一定の考え方を展開した。その考え方に基づいて多角的な評価の枠組みを定め，評価結果を各言語の仕様とともに理解し易い表形式にまとめた。評価項目は，言語概念，処理系，言語環境，その他に大別され，各項目はさらに詳細な項目に分けられている(表5－1言語評価の枠組み参照)。各言語の性格と分担者により，多少の不揃いはあるにしても，これだけ多数の高位言語を精細に分析し，それを一覧できる成果を得たことは，誠に貴重なものであると信じている。

各言語の評価をさらにまとめて一つの総合評価を試みたのが，第５・５節である。評価の座標を，設計支援，情報伝達，計算機指令各能力に整理し，マクロアセンブラ，ＰＬ／Ｍ，Ｃ，ＵＣＳＤ Pascal，Ada，LISP に的を絞り，一応の結論としてAda に最も高い総合評価を与えている。誤解を避けるために付言しておくが，これはあくまで一つの見方に基づく結論であり，本委員会としては単一の言語にのみ絞ることは危険を伴い，利用目的によって数個の言語を使い分けるべきだとの，基本的態度を確認している。ただ，敢えて一つの見方を示すことによって，大方の活発な議論を誘発し，より広い立場からの意見の集約をはかる出発点にしたいと願ったわけである。委員会としては，このようなことが近い将来実現することを切望する次第である。そのために第５・６節では，各言語に対する手短かなコメントを付け加え，また将来，それらを土台にしてより優れた言語を開発し定着させるための方策について見解を示した。

要するに第５章においては，マイコン用システム記述言語の分析と評価のための資料を，現在可能と考えられる限りにおいて提供し，それに基づいて今後の開発と活用に関する一つの展望を示し得たものと自負している。

最後の第６章は，エンドユーザの実態及びその抱えるニーズと問題点について把握・分析を試み，かつそれらエンドユーザの問題にに対する今後の取り組み方を探ろうとしたものである。マイコン技術は，その登場以来，不特定多数ともいうべき多様な人びとを直接，間接巻き込む形で発展している。

直接型の典型は，パーソナル・コンピュータに代表されるような利用形態である。パーソナル・コンピュータは確かに従来からあるコンピュータと同じ線上にある技術であるが，それらとは比較にならい程多くの人びと，つまり非専門家と直接接することになった。

エンドユーザは利用目的，年令，経験など，誠に多彩であり，一概に論じることはできない。今回は，その一つの層として大学生を選び，その数人とインタビューの機会を設けた。そして，ユーザの主体は理工科系の男子学生であり，彼らは２０～３０万円程度のシステムを所有して，主にマイコン誌などを参考にゲームを楽しんでいる。中にはセミプロ化してメーカ・ソフトの下請けをしているものもあるなどが明らかにされた。また彼らにとっての問題点は，やはりソフトウェア，言語，インタフェース，周辺機器などに関するものであり，その他中古品市場，教育，コンサルティングなどの問題も指摘された。

この章では，また，エンドユーザ言語の現状とあるべき姿について考察を加え，BASIC，LISP及びSMALL TALKの３種について特徴を分析した。特にSMALL TALKは図形情報を容易に扱え，データを主体にしたプログラムを作ることができる会話型言語であり，エンドユーザを明確に意識した唯一のものとして注目に値しよう。課題としては，人間の知的思考過程の解明，対話機能の研究，プログラムやユーザの質問・指令に対するコンピュータの理解能力の向上などの必要性を指摘した。いずれにしても，種々の言語の核（コア）になるような言語を開発し，初期教育の中に定着させることが望まれる。

エンドユーザはマイコン技術の健全な発展にとって最も本質的な存在である。一体人間は何を考え何を望んでいるのかといった基本的・認識論的観点からの解明と，それに対する現実的・技術論的対応策の検討を，産・官・学を含めた広い立場から今後とも重要課題として取り上げるべきであろう。第６・５節ではそれに関するいくつかの具体的な提言を述べた。

本報告はかなり思いきった新しい試みと提言を含むものであるが故に，独断と誤謬もまた少くないかもしれない。大方のご批判とご叱正を得た上で，今後の先進的マイクロコンピュータ応用技術の開発に資することができれば幸いである。

昭和５６年３月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園３－５－８
機械振興会館内
TEL　(434)8211.(代表)

印刷所　株式会社　タ　ケ　ミ　印　刷
東京都千代田区神田司町２－１６
TEL　(254)5840(代表)

55－R 007