解　説

4. マイクロプログラム技術の応用

4.1 プログラミング言語処理におけるマイクロプログラム技術†

柴　山　　潔††

1. まえがき

本解説においては，前回の特集（1973 年）[1] 以後のマイクロプログラム技術の動向について，特にプログラミング言語処理への応用について概説する．前回の特集号以降には，マイクロプログラム技術の可能性の検討が精力的に行われ，コスト／性能比や技術的な限界などが明らかになっている．本解説では，マイクロプログラム技術のプログラミング言語処理への適用性について，現在における技術的な到達点を示すことを主眼とする．調査の対象とした文献は，主として 1980 年以降のものであり，1970 年代の成果については，歴史的に重要な計算機システム（たとえば，Burroughs の B-1700, Nanodata の QM-1 や Xerox の Alto など）でも割愛した．これらについては，各種の書物[2)-6)] に詳しく述べられているので，それを参考にして欲しい．

調査の対象とした計算機は，各種のマイクロプログラム制御計算機や高級言語計算機である．そして，それらにおけるプログラミング言語処理の方式や支援機構，および各システムのプログラミング言語処理機能とマイクロプログラム技術との関連性などについて調査した．すなわち，マイクロプログラム技術の適用範囲の実例を整理することによって，計算機におけるハードウェア，ファームウェアおよびソフトウェアのトレードオフについて，具体的に明らかにすることを試みた．したがって，計算機アーキテクチャの提案やシミュレーションによる評価のみに留まり，具体的なハードウェア構成が提示されていないので，マイクロプログラム技術がどの程度利用されているかが不明瞭な計算機システムは，本解説の調査範囲外とした．

また，本解説の目標は，プログラミング言語処理におけるマイクロプログラム技術の適用性について述べることにある．したがって，(a)プログラミング言語処理にマイクロプログラム技術を活用していない計算機や，(b)処理の対象が具体的なプログラミング言語ではなく特定の問題（たとえば，データベースや文字列などの記号処理，画像・信号・図形の処理や生成，CAD における回路シミュレーション，数式処理，偏微分方程式や行列演算などの数値計算など）向きの基本操作を専用に行う計算機についても割愛した．

2. プログラミング言語処理におけるハードウェア，ファームウェアおよびソフトウェアのトレードオフ

2.1 プログラミング言語処理におけるトレードオフ

プログラミング言語の処理過程におけるハードウェアとソフトウェアのトレードオフについて，図-1 の

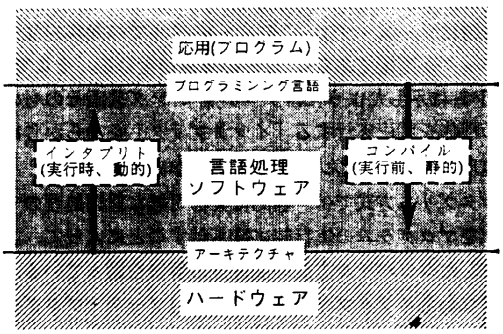

図-1　計算機システムにおけるプログラミング言語とアーキテクチャのセマンティック・ギャップ

† Microprogram Techniques for Programming Language Processings by Kiyoshi SHIBAYAMA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

ような概念図を描くことができる．応用と言語処理ソフトウェアの接点がプログラミング言語であり，ソフトウェアとハードウェアの接点が計算機アーキテクチャである．プログラミング言語と計算機アーキテクチャとの機能差は「セマンティック・ギャップ」と呼ばれる．実際のプログラミング言語処理におけるトレードオフとは，このセマンティック・ギャップを埋めるために行うハードウェア，ファームウェアおよびソフトウェアの機能分担の度合いである．

図-1 に示したように，プログラミング言語とアーキテクチャとのセマンティック・ギャップを埋める仕事は，普通「コンパイラ」と呼ばれるソフトウェアが分担する．すなわち，プログラミング言語で記述されたプログラムを，実行に先立ってコンパイラによりマシン命令列に翻訳するのである．このような過程を主に経て処理されるプログラミング言語は「コンパイル指向言語」と呼ばれる．コンパイル方式では，プログラムの実行前に（静的に）実行可能な最適化処理をできるかぎり行い，実行時によけいな言語処理を持ち越さないようにして，ハードウェアによるマシン命令実行の高速化を図っている．Fortran, Cobol, C, Ada, PL/I, Algol などの手続き型言語の大半は，コンパイル指向型と言える．

しかし，プログラミング言語処理では，プログラムの実行のみではなく，プログラムの開発過程（プログラミング）におけるオーバヘッドを少なくすることも必要とされる．したがって，言語処理をコンパイル方式だけに頼ると，(a)セマンティック・ギャップが大きくなると，コンパイルが複雑になり，それに要する時間が長くなる，(b)普通コンパイル方式によるプログラムの処理過程は一方向に（ソース・プログラムからマシン命令列へ）流れるだけなので，プログラムの開発環境と実行環境が遊離してしまう（たとえば，実行時にソース・プログラム・レベルでの効率的なデバッグが困難である），などの問題が生じる．そこで，図-1 に示したように，プログラミング言語そのものの機能を解釈実行する「インタプリタ」をあらかじめ実現しておく，インタプリト方式が考え出された．インタプリト方式では，静的に実行可能な言語処理の一部をプログラムの実行時に持ち越すことによって，上述したコンパイル方式の問題点の解決を図っている．すなわちインタプリト方式は，(i)ソース・プログラムを高級言語レベルでデバッグできる，(ii)ユーザとシステムが対話しながら処理を進めることができる，(iii)コンパイル過程が複雑になるために利用されなかった言語機能を組み込むことができる，(iv)プログラミング言語処理に必要なハードウェア機能が明確になる，(v)動的なデータ型検査を必要とする言語処理向きである，などの特徴をもつ．APL や Lisp などの会話型言語はプログラムの開発環境にも配慮して設計されており，これらは「インタプリト指向言語」と呼ばれる．

概念的には，セマンティック・ギャップをプログラミング言語側から埋めていく方式がコンパイル方式であり，逆にアーキテクチャ（マシン命令）側から埋めていく方式がインタプリト方式である．もちろん，それぞれのトレードオフを考慮して，セマンティック・ギャップにおける適当な機能レベルを中間言語レベルとして定め，コンパイラとインタプリタとを併用してギャップを埋めようとする方式も多い．また，それぞれの方式の利点を活用するために，両方式を場合に応じて使い分けることが可能なプログラミング言語処理システムもある．このようなシステムの対象言語としては，具体的には最近普及の著しい Lisp や Prolog などの記号処理用言語があげられよう．

図-1 に示したトレードオフの概念図に，ファームウェアが分担する機能部分を付け加えると，図-2 のようになる．応用と言語処理ソフトウェアの接点としてのプログラミング言語の機能レベルの概念は同じであるが，図-1 においてソフトウェアが担当していた言語処理機能を，図-2 ではソフトウェアとファームウェアで分担することになる．図-2 のようなトレードオフの核になる計算機が本解説の主要な対象である「マイクロプログラム制御計算機」である．プログラミング言語処理の観点からみると，ソフトウェアとファームウェアの接点が従来「マシン命令」と呼ばれているアーキテクチャであり，ファームウェアとハードウェアの接点がマイクロ命令レベルのアーキテクチャである．混同を避けるために，マイクロ命令レベルのアーキテクチャを「マイクロアーキテクチャ」，マシ

図-2　ハードウェア，ファームウェアとソフトウェアのトレードオフ

ン命令レベルのアーキテクチャを「マクロアーキテクチャ」と呼ぶこともある．

実際に構築されるプログラミング言語処理システムでは，図-2 の概念図におけるプログラミング言語とマイクロアーキテクチャとの機能差がセマンティック・ギャップとなるわけである．このセマンティック・ギャップを埋める処理を，ソフトウェアとファームウェアが，上述したコンパイル方式とインタプリト方式を併用して分担するわけである．したがって，ソフトウェアとファームウェアのトレードオフと，コンパイル方式とインタプリト方式とのトレードオフとの組み合わせによって，図-3 に示したようにさまざまなプログラミング言語の処理方式が考えられる．

2.2 プログラミング言語処理におけるマイクロプログラム技術の応用

元来，計算機システムにマイクロプログラム技術を導入する目的には次のようなものがある．

(A) ハードウェア機能のソフト化．すなわち，(i)ハードウェア構築時にマシン命令アーキテクチャを変更する余地を残しておきたい，(ii)ハードウェア構成にまだ不確定の要素が残っている試作機の開発である，(iii)制御論理の開発や保守を容易にしたい，(iv)ハードウェアによって構築するにはやや特殊すぎる機能を実現したい，などの「柔軟性」の獲得を目的として，ファームウェアが導入される．

(B) ソフトウェア機能のハード化．すなわち，(i)ソフトウェアによるシミュレーションでは遅いのでエミュレーション機能を導入したい，(ii)専用機能を付加したい，(iii)一部の機能を問題やアルゴリズムに合わせて高速化したい，(iv)ソフトウェアを用いることによって生じるオーバヘッドを小さくしたい，などの「高速性」の獲得を目的として，ファームウェアが導入される．

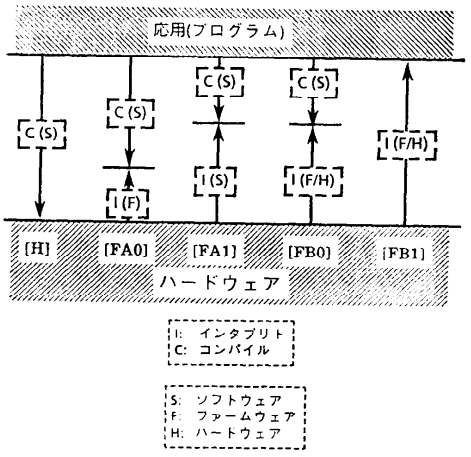

図-3 各種のプログラミング言語処理方式とトレードオフ

これらを基に，具体的にプログラミング言語（特に，高級言語）処理においてマイクロプログラム技術が活用できる点を列挙してみよう．これらはいずれも，言語処理システムの機能を拡張したり，強化する(高速化する）場合に，一般的に言語やソフトウェアがもつ柔軟性や拡張性を失うことなく，ハードウェアを使用するのに近い最適化効果を実現できるという，ファームウェアの特長から派生するものである[7]．

(a) 実行時に多くの処理が持ち越される言語（会話型言語やインタプリト指向言語）で記述されたプログラムの実行を高速化する．

(b) ハードウェア構成とのセマンティック・ギャップが著しい言語(記号処理用言語や非手続き型言語)を高速処理する．

(c) ハードウェア機能のテストや保守における柔軟性を支援する．

(d) 複数の言語や言語ファミリを対象とするシステムを構築する．

(e) 仕様（たとえばデータ型）自体に拡張性を備えた言語を対象言語として，問題向きに拡張可能なシステムを構築する．

(f) 対象とする問題や言語の処理に向いたマシン命令を追加する．(この最適化をダイナミック・マイクロプログラミングによって動的に行う方法が「マシン・チューニング」である．)

(g) プログラムの開発環境と実行環境のトレードオフを，(インタプリタとコンパイラの併用などによって）適正に設定する．

(h) 対象とする言語の機能には本来含まれていないが，ソフトウェア開発において重要な役割を果たすテストやデバッグ機能（たとえば，マシン命令実行によるマシン状態のスナップ・ショット提示機能）などを付加する．

(i) 言語処理方式を研究するために，各種の機能レベルの相異なる仮想計算機アーキテクチャを構築する．

(j) 最適アルゴリズムが確定していない処理機能，たとえば，キャッシュ操作やガーベジ・コレクション (GC) などの記憶管理，またデータを入出力形式に合わせて加工する編集操作，あるいは並行処理のプ

ロセッサ（プロセス）間通信プロトコルなどを試みに実現する．

（k） 専用ハードウェアが装備されていない特殊な，また高機能なマシン命令（たとえば，複雑な構造化データ操作，浮動小数点演算や可変長・多倍長演算，10進数演算，配列・ベクトル演算，ビットや文字列操作など）を実現する．

（l） 高級言語の多様な順序制御論理（ステータスと分岐方式との組み合わせ）の構造化・高速化を実現する．

（m） 例外条件（たとえば，スタックあふれ）に対して，柔軟に対処する．

（n） マシン命令の多様化（たとえば，豊富なアドレッシング・モード）をハードウェア機構と分担して実現する．

（o） 言語処理向きのハードウェア機構(たとえば，タグ・アーキテクチャ，スタック機構）の操作機能(タグによる分岐機能，スタック・ポインタの操作，スタック・フレームへのアクセスなど）を支援する．

特定のプログラミング言語ではなく，一般的な高級言語処理を対象として計算機のファームウェア機能やハードウェア機能を考えると，普通次の2点が中心となる．(1)命令パス系統の強化のためにスタック構造の導入，(2)データ・パス系統の強化にタグ・アーキテクチャの採用，である．(1)は，再帰構造やブロック構造をもつ手続き型言語の制御構造に適合したアーキテクチャの実現を支援するためなどに有効である．(2)のタグ・アーキテクチャの採用による利点には，(i)データ型の検査，保護，変換機能の自動化，(ii)実行時デバッグ機能の支援，(iii)例外条件(配列境界，スタックあふれ，GC 起動，ソフトウェア・トラップなど）の処理支援，(iv)相異なるデータ型に対して包括的な (generic) 命令を適用できることによる命令種類の縮小化，などが考えられる[8]．したがって，高速性以外に柔軟性が要求されるので，ハードウェア機構にマイクロプログラム技術を組み合わせることが多い．

2.3 トレードオフによる計算機の分類

図-3 に示したプログラミング言語処理におけるハードウェア，ファームウェアおよびソフトウェアのトレードオフによって，マイクロプログラム制御計算機を分類してみよう（図-4 参照）．

まず，調査対象とした計算機がファームウェアの分担する機能を有するか否かによって，[H] 布線 (hard-wired) 論理制御方式の計算機と，[F] マイクロプログラム制御計算機，とに分けられる．

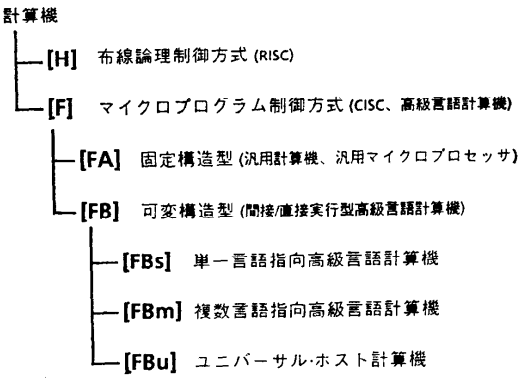

図-4 制御方式と対象言語による計算機の分類

[H] の計算機として最近脚光を浴びている RISC (Reduced Instruction-Set Computer)[9] では，ファームウェアが担当していた言語処理機能をソフトウェアによって実現するのが普通となっている．ソフトウェア・コンパイラの負担は増すが，そのオーバヘッドをハードウェアによる高速実行で帳消しにしようとする試みである．ファームウェアで実現している機能がないので，ソフトウェアとハードウェアの接点がマシン命令アーキテクチャである．VLSI を指向した方式であり，最近のマイクロプロセッサ・チップの開発において採用されている．もっとも，マイクロプログラミング可能な RISC[11] などが存在し，RISC の定義もまちまちなので，必ずしも RISC と呼ばれる計算機すべてが [H] に分類されているわけではない．

[F] に属するマイクロプログラム制御計算機は，さらにマシン命令アーキテクチャの柔軟性の程度により，[FA] 固定構造型計算機と，[FB] 可変構造型計算機とに分けられる．

[FA] マシン命令アーキテクチャが固定されている固定構造型マイクロプログラム制御 (Microprogrammed) 計算機：プログラムは，まずソフトウェアによってプログラミング言語からマシン命令へコンパイルされ，そのマシン命令列がファームウェアによってインタプリトされる．ソフトウェア・コンパイラに言語処理の主要部分を任せる方式である．核となる計算機ハードウェアの設計手法としてマイクロプログラム技術を捉えている．したがって，マイクロプログラム制御計算機であってもマシン命令アーキテクチャが固定されているという意味で，ファームウェアがプログラ

ミング言語処理において本質的に利用されているとは言い難い.

[FA 0]: 汎用計算機や汎用マイクロプロセッサにおいて高級言語を処理する際に普通に採用される方式である. また, 高級言語を指向してマイクロアーキテクチャ (ハードウェア構成) に工夫があるマイクロプロセッサも存在する. また, コンパイル過程の途中に, 何段階かの中間言語機能レベルを設定して, ソフトウェア・コンパイラの拡張性や柔軟性を確保する方式もある.

[FA1]: 会話型言語などのインタプリト指向型言語を対象とする場合には, この方式のように中間言語のソフトウェア・インタプリタを併用することがある.

[FB] マシン命令アーキテクチャが固定されていない可変構造型マイクロプログラム制御 (Microprogrammable) 計算機: ユーザがマイクロプログラムを書き換えることにより簡単に制御論理を変更でき, ファームウェアの特質を活用できる方式である. ソフトウェアとファームウェアとの接点の機能レベルをユーザが自由に設定できるという点で, ファームウェアを本質的に利用していると言えよう. 本解説では, 主として [FB] に属する計算機におけるプログラミング言語処理方式を対象としている.

[FB 0]: [FB] の典型的な例である間接実行型高級言語計算機である. この方式では, 対象とする言語の機能レベルを指向したマシン命令アーキテクチャ (これは [FA] よりも高い機能レベルに設定するのが普通である) を用意して, そのマシン命令のインタプリタをファームウェアで実現する. 対象言語に合わせて中間言語やそのファームウェア・インタプリタを自由に設計することができる. また, 中間言語から直接マイクロ命令列を生成するマイクロプログラム・コンパイル手法を採ることもできる. 対象言語を単一にするか複数にするか, 中間言語レベルをいくつ設けるか, またどの機能レベルに設けるか, コンパイラやインタプリタのそれぞれをハードウェア, ファームウェア, ソフトウェアのいずれで行うのか, などによって各種の方式が考えられる.

[FB1]: ソフトウェアの分担していた言語処理機能の大半をハードウェアないしはファームウェアで代替したものであり直接実行型高級言語計算機と呼ばれる. 対象言語の機能レベルがそのままマシン命令アーキテクチャとなっている. ある程度の柔軟性を確保するためには, インタプリタの機能すべてをハードウェア化するよりも, ファームウェアによって一部を代行する方式が採用されることが多い.

また, [FB] に属する計算機は, 処理対象とするプログラミング言語を特定のものに絞り込むか否かによって, [FBs] 単一言語指向高級言語計算機, [FBm] 複数言語指向高級言語計算機, [FBu] ユニバーサル・ホスト計算機, というように分けることもできる. たとえば, 最近商用化が著しい Lisp マシンや Prolog マシンは [FBs] に, マルチ・パラダイム指向の AI マシンは [FBm] に, それぞれ分類できる. また, [FBm] に分類した計算機は, 複数ではあるが特定のプログラミング言語を対象として明示しているものであり, [FBu] のユニバーサル・ホスト計算機は, 対象として想定した特定のプログラミング言語やマシン命令アーキテクチャがなく, ユーザが応用向きにマイクロプログラムを記述する必要があるもの, という違いがある. 3. では, この分類に従い, 具体的な計算機方式を列挙する.

汎用マイクロプロセッサや高級言語計算機と RISC とのアーキテクチャ上の特徴の比較や, それぞれの一般的なプログラミング言語処理における得失などについては, 本特集号に収録されている別の解説論文や文献[3),10),11)]に詳しく述べられている.

3. プログラミング言語処理におけるマイクロプログラム技術の応用例

3.1 汎用計算機におけるマイクロプログラム技術の応用

まず, マイクロプログラム技術を積極的に利用したアーキテクチャを採用している汎用計算機の具体例を掲げる.

[G1] VAX8600(DEC)[12]: 32ビット CPU は ECL のマイクロセル・アレイをもちいて, 機能別に 6 台のサブ・プロセッサで構成されている. これらを 96 ビット長のマイクロ命令が同時制御する分散マイクロプログラム制御方式を採る. 1マイクロ命令は 80 ns で実行される. マシン命令コードを拡張できる機能を用いて, 問題向き計算機に適応可能である. 実際に, 500 マイクロ命令を消費して, WAM (Prolog 向きの仮想計算機, 3.2 で詳述) 命令と1対1対応するマシン命令セットを作り, マシン命令アーキテクチャを Prolog 向きに拡張した例[13)] が報告されている.

[G2] NCR 9300 (NCR)[14]: CPU は NCR/32-000 という32ビット VLSI チップであり, これは95ビット

長のマイクロ命令で制御される．その制御記憶は，CPU チップに内蔵されている256ワードの ROM である．マシン命令アーキテクチャとして，16ビット長命令をもち，これは64K ワードの命令格納用メモリに格納される．主記憶とは異なる命令格納用メモリを備えており，ユーザがこの命令格納用メモリを書き換えられるので，この計算機はユニバーサル・ホスト計算機の一種とみることもできる．また，WAM 命令をいったんこの命令列に展開する方式で，Prolog マシンのエミュレーションを行っている．

3.2 高級言語計算機

3.2.1 高級言語計算機におけるプログラミング言語の処理方式

単一言語向き，あるいは複数言語向きのいずれの高級言語計算機の処理方式においても，マイクロプログラム技術の特質を活用するならば，対象言語おのおのの構文や意味に対応した適当な中間言語命令のインタプリトをファームウェアで行う方式が普通であろう．すなわち，コンパイラができること（たとえば，種々の最適化）はできるかぎり実行前に行い，実行時にしかできないこと（たとえば，動的な変数束縛）をインタプリタが処理する，というトレードオフが適当である．ファームウェア・インタプリタを用いる間接実行型の場合には，ファームウェアとハードウェアの接点がマイクロ命令アーキテクチャであり，中間言語レベルがそのままマシン命令アーキテクチャとなる．この場合，中間言語レベルの機能を低く（マイクロ命令アーキテクチャに近く）設定し過ぎると，マシン命令のフェッチとデコードがオーバヘッドになる．

マイクロプログラム技術の発展・成熟とともに，各種の高級言語計算機が開発されてきたが，それとともに RISC アプローチを取り入れてトレードオフの改良を図る試みなども行われている[11),15)]．たとえば，対象言語としてシステム記述言語を採用する[82)]，プロセス操作やコンテクスト・スイッチの高速化を図る[65)]，といった OS やコンパイラ指向アーキテクチャなどが具体的に検討され，実験されている．

また，ダイナミック・マイクロプログラミングの応用として有望とされていたにもかかわらず，環境の切り換えやチューニングに要するオーバヘッドのために実用化されなかったマシン・チューニングの概念が，最近では，対象言語の処理方式を設計する過程で活用されている．すなわち，Lisp や Prolog などのインタプリト指向言語においては，プログラムの開発環境がシステム性能を決める重要な要因となり，処理方式も柔軟にして拡張できるように構築する必要がある．プログラミング言語が本質的に備えている拡張性を，できるかぎりオーバヘッドなく実現する技術としてマイクロプログラム制御方式が注目されている．

実際には，最近開発・商用化の著しい Lisp マシンや Prolog マシンの大半は，マイクロプログラム技術を積極的に取り入れている．最近では，1チップの Lisp マシン[29),31)]なども試作されており，並列計算機の要素プロセッサや汎用計算機システムへ組み込む専用プロセッサとして有望であろう．また，これらの記号処理用言語は AI 応用向きであることから，これらの言語の種々のパラダイムを同一計算機上で実現しようという AI マシン（チップ）[83),86)]も提案・試作されている．

以下の各節では，これらの高級言語計算機について，個々の対象言語ごとに具体例を示し，その処理方式や処理機構について説明する．

3.2.2 Lisp の処理機能

逐次型 Lisp の処理に向いた機能としては，次のようなものが考えられる[16),17)]．（1）動的な記憶（ヒープ）管理，（2）リスト格納用構造化メモリ，（3）関数呼び出し制御用スタック，（4）コンテクスト・スイッチ，（5）データ型の検査・比較とその結果による順序制御，（6）リスト・ベクトル演算，などである．いずれもできるかぎりハードウェアで実現することが望ましいが，たとえばスタック管理や GC などにおいては，最適なアルゴリズムや方式が未知である場合も多く，動的な高速処理以外に柔軟性が要求されるので，ファームウェアを併用する場合が多い．また，上記の機能を実現するアーキテクチャとしては，スタック機構を装備したタグ・アーキテクチャが一般的である．

マイクロプログラム技術を応用する処理方式としては，（a）Lisp 処理用マシン命令をマイクロプログラムで用意しておき，そのマシン命令列に Lisp プログラムをソフトウェアでコンパイルし，それをマイクロプログラムでインタプリトするコンパイル-インタプリト方式と，（b）Lisp プログラム（関数）をソフトウェアで直接マイクロプログラムへコンパイルしてしまうマイクロプログラム・コンパイル方式とが考えられる．また，両方式を併用できるシステムもある．

3.2.3 Lisp マシンにおけるマイクロプログラム技術の応用例（付表-1 参考）

[L1] M3L(フランス・Paul Sabatier 大学)[18),19)]：

左ポインタ，右ポインタ，ディスクリプタから成る40ビット長の三つ組をマシン命令とするセル・マシンである．Lisp関数をこのマシン命令列に変換し，それをマイクロプログラムでエミュレート（インタプリト）する．マーク & カウント方式のアルゴリズム格納用PROMとマーク用スタックをGC用ハードウェア機構として備えている．RAMに格納したマイクロプログラムとROMに格納したナノプログラムによる2レベル・マイクロプログラム制御方式を採る．Lispの21関数を670マイクロ命令で記述してある．対象言語に向いたマシン命令（三つ組）アーキテクチャのエミュレータを書けば，各種のプログラミング言語を対象言語とする間接実行型高級言語計算機ができる．実際に，Lisp以外にもPascalやCobol用のインタプリタが書かれている．

[L2] EVLISマシン(大阪大学)[20),21)]：核となるリスト処理プロセッサをEVALIIと呼び，Lispのカーネル関数であるapply, eval, evlis, condなどを含む72関数をマイクロプログラムで記述してある．car/cdr操作機構や診断用インタフェース，ディスパッチ表も兼ねるスクラッチパッド・メモリを装備している．マイクロプログラム・コンパイラも開発されており，それによるとファームウェア・インタプリタによる場合よりも7〜15倍高速のコードを生成できることが報告されている．並列Lisp用のインタプリタやコンパイラもある．

[L3] FACOM ALPHA（富士通）[22),23)]：ハードウェア・スタック（64kバイト）を装備している．Lispインタプリタ，基本Lisp関数，コピー方式GC，および汎用スタック・マシンのマシン命令をマイクロプログラムで記述してある．また，マシン命令からマイクロ命令を呼び出すことができる．LAMDA文やMACRO文，および関数引数関数はいったんマシン命令列へコンパイルされる．ファームウェア・インタプリタは，マシン命令にコンパイルされたコードの5倍程度の性能であるので，マイクロプログラム・コンパイラも必要である，という評価がある．

[L4] Symbolics 3600 (Symbolics)[24)]：カスタムLSIでCPUを構成する商用Lispマシンである．マシン命令アーキテクチャはタグ・アーキテクチャを採るが，強くLisp指向としていることはないので，汎用のワークステーションとして利用できる．マシン命令長が17ビットと短いスタック・マシンである．タグにより，cdrコーディングをサポートする．

[L5] FLATS（理化学研究所)[25)]：並列ハッシング，スタック，リバースcons，GCの各機構をハードウェアで装備している数式処理用タグ・アーキテクチャである．基本Lisp関数やハッシング操作，ビット操作用のマシン命令を230種装備している．

[L6] Explorer (TI)[26),27)]：16ビット長のLisp用高機能スタック・マシン命令をもつ商用Lispマシンである．ビット・フィールド処理，タグ操作，リアルタイムGC，スタック・キャッシュなどのハードウェア機構を装備している．Lisp関数，およびプロセス間排他制御機能やFlavor（オブジェクト指向機能）のインスタンス参照機能がファームウェア化されている．

[L7] ELIS (NTT)[28),29)]：タグ・アーキテクチャをもつ商用のLispチップであり，制御記憶はSRAMで外付けにする．1組のレジスタを多目的（MAR, MDR, ソース，デスティネーション）に使用できる機構を備えている．スタック(32kワード/32ビット)も外付けとしている．対象言語は，TAOと呼ぶLispを基にしたマルチ・パラダイム向き言語である．TAOは，論理型言語やオブジェクト指向言語のパラダイムも融合し，広範囲の記号処理応用を指向した機能を備えている．Common-Lispは元来コンパイル指向言語とされているが，ELISではLisp関数を直接マイクロプログラムでインタプリトする．また，TAOプログラムをいったんLAPコード（バイト・コード）に変換した後，そのコードをマイクロプログラムでインタプリトする手法との比較[30)]を行っている．

[L8] Lispチップ (TI)[31)]：ELISと同じくタグ・アーキテクチャを採り，制御記憶は外付けである．ELISとは異なり，Lispプログラムはいったんマシン(マクロ)命令列にコンパイルされ，それらがマイクロプログラムでインタプリトされる．マクロ命令中のフィールドがマイクロプログラムの開始番地を表している．スタック・トップ格納用のバッファ記憶や，多方向分岐用のディスパッチ・メモリを内蔵している．

そのほかに，Scheme-79 (MIT)[32)]は，リスト形式のSコードをマシン命令とする初期のLispチップである．マシン命令のインタプリタとGCがマイクロプログラムで記述されており，制御論理は2レベルPLAで実現している．

3.2.4 Prologの処理機能

Prologの処理を指向した機能としては，(1)単一化のための動的データ型検査，変数束縛，(2)動的な

記憶管理，(3)コンテクスト・スイッチ（変数束縛環境の切り換え）などがあり，Lisp の処理機能と共通する項目も多い[16]．しかし，Prolog 処理においては，単一化やバックトラックの機能などを必要とする点で，Lisp における記号処理機能がより専用化されていると言える．

Lisp 処理と異なる点としては，(i)バックトラックや述語呼び出しに使用するスタック機能がおのおの専用化できるので，機能別に複数のスタック機構を設けることが有効であること，(ii)単一化処理において変数同士の束縛関係が生じるなど，組み合わせるデータ型によって多種多様な操作を高速に実行する必要があり，マイクロプログラム技術がこの要求に最適であること，などがあげられる．

Prolog マシンの開発でも，種々の処理方式（トレードオフ）が試みられている．最近は，このいずれの方式を採る場合にも，中間言語機能として“WAM”[33),34)]と呼ばれる仮想計算機が設定されることが多い．WAM は，(1)述語の引数をレジスタによって授受する，(2)スタック機能を明確にしている，(3)汎用計算機に近いアーキテクチャであるので，最適化処理にそれと類似の手法が適用できる，などの特徴をもつ．

WAM をエミュレーションの対象とする Prolog の処理方式には，次のようなものがある．

(a) WAM を中間言語とするが，最終的には汎用計算機のマシン命令にコンパイルすることを目標とする，従来型のソフトウェア・コンパイル手法である．WAM のレベルにおける最適化手法が種々開発されているので，それを最大限に利用できる．

(b) WAM をマシン命令として，ファームウェア・インタプリタによって解釈・実行する手法である．商用の Prolog マシンやユニバーサル・ホスト計算機上での Prolog マシンのエミュレーションにおいて盛んに用いられている．

(c) WAM レベルの命令列を直接マイクロプログラムにコンパイルする手法である．マイクロプログラム制御計算機の性能を最大限に引き出すことを狙っている．WAM の提案者らにより，動的に 1 WAM 命令が約 9 マイクロ命令で実行されるというシミュレーションによる評価[34)]が示されている．

また，WAM の改良提案も多くある．たとえば，ZIP マシン[35)]はプロセッサ・モードを 3 個設け，算術命令を追加している．RAP-WAM[36)]は，ユーザが表記した情報を用いて，コンパイル時に変数依存性の検査を行い，条件グラフ表現というテンプレートを用いて，制限された AND 並列処理を実現する試みである．また，汎用計算機を対象とするソフトウェア・コンパイラ[37)]でも，段階的最適化における中間レベルの仮想計算機として WAM を採用している．

3.2.5 Prolog マシンにおけるマイクロプログラム技術の応用例（付表-2 参考）

[P1] PLM (UCB)[38),39)]：Aquarius[40)]という共有メモリ型マルチ・プロセッサ・システムの要素プロセッサであり，WAM レベルの並列プロセス処理マシン命令をもつ．

[P2] PEK（神戸大学）[41),42)]：WAM を改良した中間コード（マシン命令）にいったん Prolog プログラムをコンパイルし，それをマイクロプログラムでインタプリトする．WAM の改良点は，モード宣言の追加，CALL 命令の多様化，環境のグローバル・スタックでの管理，構造共有方式の採用，などである．ハードウェア・スタックやフィールド・ビット処理機構，自動トレイル/UNDO 回路などを装備しているタグ・アーキテクチャである．1 LI (Logical Inference) を 13 マイクロ命令で実行する．

[P3] CHI(日本電気，ICOT)[43)]：WAM をマシン命令アーキテクチャとするスタック・マシンである．180 種のマシン命令のうち 100 種が組み込み述語用である．タグ比較器，スタック操作機構などを 10 フィールドのマイクロ命令で制御する低レベル並列処理方式[94)]を採用している．

[P4] PSI（三菱電機，ICOT)[44)]：Prolog プログラムと同じ機能レベルの表形式マシン命令をマイクロプログラムでインタプリトする．ファームウェアによる直接実行型高級言語計算機のエミュレーションとみることができる．PSI 上で WAM 命令（後述する PSI-II のマシン命令）のエミュレーションを行ったところ，その性能は元の PSI の 2～3 倍となったという報告[45)]がある．

[P5] PSI-II (三菱電機，ICOT)[45)]：PSI と同じく，タグ・アーキテクチャのファームウェア・インタプリタであるが，マシン命令の機能レベルとして WAM が採用された．WAM に，組み込み述語用，カット機能用，動的述語呼び出し用，オブジェクト指向 OS 支援用の各マシン命令を追加している．データ型やマシン命令コードに付加してあるタグによって，直接マイクロプログラムの実行順序を制御することができ

る．PSI-Ⅱ は PSI の 3〜10 倍の性能をもつ．また，PSI-Ⅱ では PSI 上の WAM エミュレータの 1.5〜4 倍の性能向上がみられるが，これはオペランド抽出器やマシン命令の先取り用レジスタなどのハードウェア機構，およびマイクロプログラム化されたページ管理機能などによる．

[P6] IPP（日立製作所）[46]：汎用計算機にファームウェアとハードウェアによる Prolog 向き機構を付加した．付加したハードウェアとしては，タグ操作回路，Prolog 専用レジスタ，ALU，メモリ書き込み用バッファ，トレイル検査回路，およびこれらを制御するマイクロ命令格納用制御記憶である．WAM 命令セットを拡張した 70 種のマシン命令をマイクロプログラムでインタプリトする．ファームウェアによって 3.4 倍，ハードウェアによって 2.7 倍，合計 9.1 倍（Prolog プロセッサを付加しない元の汎用計算機と比べて）の高速化が達成されている．マシン命令プログラムの広域的なデータフロー解析をコンパイル時に行ってレジスタ使用の最適化を図る方法により，さらに 1.2〜3.4 倍の高速化が達成されている[47]．

[P7] VLSI Prolog インタプリタ（イタリア・DEPT）[48]：マシン命令形式はダブル・リンクド・リストであり，これが実行時に直接インタプリトされる．タグ・アーキテクチャを採り，単一化用スタックやレジスタを装備している．

[P8] ASCA (NTT)[49]：Prolog プログラムの内部形式をそのままマイクロプログラムでインタプリトする．変数束縛情報と制御情報を大容量連想メモリに格納し，単一化処理を高速化している．

そのほかに，DAWN（日本電気）[50]は，Shapeup と呼ぶ Prolog に Snobol のパターン照合機能を付加した言語をファームウェアでインタプリトするスタック・アーキテクチャの提案である．

また，試作中の論理型言語向き並列計算機である PIE（東京大学）[51]や KPR（京都大学）[52]では，その要素プロセッサがマイクロプログラム制御方式を採用している．特に，ヘテロジニアス・マルチプロセッサ構成のシステムでは，個々のプロセッサ機能に応じて適当なビット長のマイクロ命令がそれを制御する場合が多い．

3.2.6 その他の単一言語向き高級言語計算機におけるマイクロプログラム技術の応用例

Ada は分散型システムを指向しており，複数の並行タスク／プロセスの管理手法がシステムの性能を決める．Ada の機能は非常に高級であるので，その処理では普通中間言語が設定され，その選定も重要である．

その他の Pascal や C などの手続き型高級言語においても，中間言語や処理方式の選定が言語処理機能の性能を左右する重要な要因となる．

Smalltalk-80 処理用仮想計算機としては，歴史的には「バイト・コード」が著名である．バイト・コードは，スタック・マシン・アーキテクチャ上で実行される．また，その他の Smalltalk-80 マシン用の中間言語アーキテクチャの提案や，それに基づく試作や評価もある[53),54]．Smalltalk-80 処理システムにおいては，(i)コンパイルされたメソッドの管理，(ii)動的なメッセージ生成（手続き呼び出し），(iii)オブジェクトやインスタンスの管理，などがその性能を決める要因となる．

一般的にオブジェクト指向言語は抽象化と領域保護の概念に基づき，プログラミング言語のマルチ・パラダイムの包括を指向していると言える．しかし，オブジェクト指向言語の処理機能はまだ漠然としている．特に，ファームウェアやハードウェア機構として確定できるものは少なく，Smalltalk-80 マシン以外のオブジェクト指向アーキテクチャはいずれもまだ構想や提案の段階にある．

以下に，マイクロプログラム技術を利用している単一言語向き高級言語計算機の最近の具体例を列挙する（**付表-3** 参考）．

[S1] Pascal MICROENGINE (Western Digital)[55]：Pascal 処理向き中間コードである P コードをマシン命令とするスタック・アーキテクチャである．

[S2] MPS 10（フィンランド・Nokia Electronics）[56]：300 種のマシン命令をもつ仮想 Ada マシンをマイクロプログラムによってエミュレートする．1 タスクごとにスタックとタスク状態とがセグメントとして割り付けられるスタック・マシンである．ワーキング・セット管理ルーチンもマイクロプログラムで記述されている．領域保護機能用にタグを用いる．

[S3] POMP（アイルランド・Trinity 大）[57]：Pascal の命令やデータ構造と 1 対 1 対応する 1 バイト長マシン命令をマイクロプログラムでインタプリトするスタック・アーキテクチャである．マシン命令コードとプログラム状態ワードとの組み合わせによりオペランドのデータ型を判定する機構をもつ．スタック・トップ・バッファやディスパッチ・メモリも備え

ている．682 マイクロ命令でマシン命令セットを作成してある．

[S4] Sword 32 (東京大学)[58]，**[S5]** Hobbes (沖電気工業)[59]：バイト・コードのファームウェア・インタプリタである．単純なハードウェア構成が高速化の鍵であると考え，ハードウェアへの投資をバイト・コードやマイクロ命令のディスパッチ機構やスタック機構のみにとどめた．

[S6] Swamp (カナダ・Toronto 大学)[54]：バイト・コードと類似したマシン命令セットを採るが，オブジェクト表現や GC 方式は，Smalltalk-80 向き RISC である SOAR[112] と似ている．コンテクスト・キャッシュをレジスタとして備え，タグ検査やメソッド・キャッシュ操作をハードウェア機能として提供している．

[S7] GF 11 (IBM-Watson 研究所)[60]：量子力学の問題プログラムを Pascal の手続きとして書き，それを大型汎用計算機上でクロス・コンパイルし，グラフ表現にする．そして，それを最適化してマイクロプログラムとし，566 台の要素プロセッサで実行する方式の SIMD 型並列計算機である．

[S8] MIRIS (George Mason 大)[111]：ビット・スライス LSI で構成され，1 マシン命令の実行を 1 マイクロ命令が制御するので，制御記憶に格納されているものはマイクロプログラムというよりも「マイクロ命令ワード」と呼ぶのが適切である．制御記憶は PROM で構成され，現在 64 種類のマシン命令が装備されている．また，2048 ワード (32 ビット) のウィンドウ方式のレジスタを装備している．

[S9] UDEC (筑波大学)[61]：手続き型言語向きの汎用直接実行型高級言語計算機であり，対象言語の構文を制御表に格納し，その意味をマイクロプログラムで記述する方式で複数言語向きにしている．字句解析ユニットや，構文意味認識ユニットはハードウェア化され，連想メモリによって構成された制御表によって実行管理される．制御実行とデータ処理は，MEGA3 と呼ぶマイクロプログラム制御計算機〈64 ビット長マイクロ命令，4 K ワード制御記憶，200 ns マイクロ命令サイクル〉で実行される．Pascal の例では，制御実行用に 612 マイクロ命令，データ処理用に 833 マイクロ命令が静的に消費され，1 トークン処理で動的に 60 マイクロ命令が実行される．

[S10] CRISP (ATT-Bell 研究所)[62]：C 言語向き RISC チップであるが，64 ビット長のマシン命令が多段命令キャッシュ内で 192 ビットの内部表現に変換されるので，これをマイクロ命令ワードとみなすこともできる．手続き呼び出し用マシン命令やスタック・キャッシュを装備している．

[S11] ARC (Aerospace)[63]：Pascal と Fortran を融合した言語 FOCAL で記述されたプログラムの内部表現（可変長字句トークン）を直接マイクロプログラムでインタプリトするスタック・マシンである．計算機アーキテクチャの研究に用いている．

[S12] LTR (フランス・Paul Sabatier 大学)[64]：LTR と呼ぶリアルタイム処理用高級言語をいったん中間言語プログラムにコンパイルし，それをマイクロプログラムでインタプリトする．LTR 言語は，並行動作プログラムを記述可能であり，領域保護機能の概念を備えている．ファームウェア・インタプリタは，2 段階（タスク操作の生成と実行）のパイプライン構造となっており，それぞれが異なるマイクロプログラムによって制御される．

[S13] Transputer, T 414 (イギリス・Inmos)[65]：OCCAM という並列処理用言語をアセンブリ言語とする．C, Fortran, Pascal などの高級言語の処理システムは，OCCAM で記述する．並列／並行プロセス制御用マシン命令をもち，そのプロセス・スケジューラをマイクロプログラムでコーディングしてある．また，乗算やメッセージ転送などの複雑な命令機能もマイクロプログラムで記述してある．110 種類のマシン命令のうち 15 命令コードは 1 サイクルで実行可能である．最近，浮動小数点演算ユニットを内蔵した T 800 が発表された．

そのほかに，OPS5 などのプロダクション・システム (PS) では，競合解消アルゴリズムと照合処理の高速化がシステム性能を左右する．したがって，状態記憶を更新したり，プロダクションを検索する制御部には，高速性と柔軟性が要求され，マイクロプログラム技術の役割も大きい．最近の PS マシンとしては，CMU の RISCF[66]，広島大学の ISAC[67] がある．

意味ネットワーク処理を指向した高並列計算機としては，電総研の IXM[68] や USC の SNAP[69] がある．いずれも，要素プロセッサ中に連想メモリやマイクロプログラム制御機構を備えている．

水平型マイクロ命令によって複数の演算ユニットを制御する，Fortran 向き低レベル並列処理計算機には，CHoPP (CHoPP Computer)[70]，VM アーキテクチャ (Purdue 大学)[71] などがある．

関数型言語 FP のプログラムをマイクロプログラムによってインタプリト（リダクション）する計算機には，FP グラフ・リダクション・マシン（東北大学）[72]，μ3L (Utah 大学)[73]，G-machine (Oregon 大学院)[74]，FFP Machine (North Carolina 大学・Chapel Hill 校)[75] がある．

3.2.7 複数言語向き高級言語計算機の処理方式

プログラミング言語処理におけるマイクロプログラム技術の最も一般的な応用は，複数のプログラミング言語を対象として，それぞれに最適な処理機能をマイクロプログラムで記述する方法である．これは，複数言語向き間接実行型高級言語計算機のエミュレーションを行う機構の実現と言える．もっとも，対象言語は複数としても，ある程度その数を絞っておかなければ，マイクロ・アーキテクチャの機能レベルが低くなりすぎて，柔軟性を得た代償としての実行速度のオーバヘッドが大きくなる．

AI マシンとは，まだはっきりとした概念を形成するものではなく，商業政策として，Lisp や Prolog, Smalltalk-80 といった複数の人工知能 (AI) 応用向き記号処理言語を対象とする計算機システムに冠せられることが多い．明確な対象言語が存在し，その処理機能のファームウェアやハードウェアによる支援機構をもつという点で，後述するユニバーサル・ホスト計算機とは若干性格が異なる．AI マシンは，専用計算機と言うより，AI という漠然とした応用分野における汎用計算機を目指しているものと言えよう．AI マシンの CPU には，汎用マイクロプロセッサを用いる場合も多いが，本解説の調査対象とした AI マシンとしては，特に専用プロセッサを用いたものを考える．AI 応用向き計算機として，プログラミング言語指向ではなく，データや知識処理を指向したものは，オブジェクト指向アーキテクチャとか知識ベース・マシンと呼ばれる．

AI 向きの対象言語として，手続き型，関数型，論理型，オブジェクト指向などのパラダイムを融合して一つの言語としたものがある．前述した，ELIS の対象言語 TAO や PSI の対象言語 KL0 のほかに，Schlumberger が開発中の並列記号処理システム FAIM-1 の対象言語 OIL[76]などである．

AI マシン・アーキテクチャとしては，並列性のサポート，分散型処理構造，動的負荷分散，学習や知識獲得を支援できる機構，大規模記憶空間，などが必要とされよう．複数言語を対象とすることだけではなく，自己拡張性の実現や，未知の処理機能が多いことなどを考慮すると，マイクロプログラム技術が活用できる余地が十分あると考えられる．

3.2.8 複数言語向き高級言語計算機におけるマイクロプログラム技術の応用例（付表-3 参考）

[M1] SWARD (IBM)[77]：ソフトウェアの信頼性向上を目指して，領域保護概念に基づくアドレッシング方式を採るマシン命令アーキテクチャを，マイクロプログラムでエミュレートする．マシン命令は使用頻度に応じて符号化され，可変長セル・トークン列として表される．

[M2] Xerox 1121 (富士ゼロックス)[78]：複数の AI 用プログラミング言語を対象として開発されている Xerox の D マシン・シリーズのうちの商用ワークステーションである．D マシンは，Alto に始まり，Dolphin, Dorado[79),80], Dandelion, DandeTiger[78] と続いている．計算機を構成するハードウェア技術は変遷していても，比較的短いビット長の垂直型マイクロ命令によって対象言語ごとに異なるエミュレータを用意しておく制御方式を，一貫して採用し続けている．Mesa と呼ぶシステム記述言語プログラムは，いったんデバイス・ハンドラというマイクロプログラムでエミュレートされる命令列に変換されて実行される．Lisp プログラムは，基本的な Lisp 関数に対応する Lisp バイト・コードと呼ばれる中間言語命令列に変換され，それがマイクロプログラムによってインタプリトされる．Smalltalk-80 プログラムの場合は，若干複雑であり，バイト・コードがそのままマイクロプログラムによってエミュレートされるものと，いったん Mesa に変換してから Mesa エミュレータによって実行されるもの（特に，OS 機能）とに分けられる．D マシンも，最近では，Dragon[81] という共有メモリ型密結合マルチプロセッサ・システムに進展している．Dragon の要素プロセッサは，RISC ふうの Mesa マシン[82] である．

[M3] AI 32 (日立製作所)[83]：ハードウェアをできるかぎり単純にして，その代わりに制御記憶 (EPROM) や記憶管理ユニット，大容量レジスタ・ファイルをチップ上に搭載しようという試みである．制御記憶を外付けの RAM にして，ダイナミック・アーキテクチャとすることも計画している．

[M4] WINE (東芝)[84),85]：機能スライス LSI で構成したタグ・アーキテクチャの CPU (IP 704 と呼ぶ) を核とするバックエンド型 AI ワークステーショ

ンである．IP 704 は，Prolog 向きに WAM レベル命令を45種，Lisp の基本関数を中心に40種，そして汎用120種のマシン命令セットをもつ（スタック・マシンではなく）レジスタ・マシンである．レジスタを2重化する（切り換えをハードウェア化），データ・バスを2重化する，スタック境界検査をハードウェア化するなどの方式で，Lisp の関数呼び出し，Prolog のバックトラック処理を高速に実行できる．同一のマシン命令に対応する相異なるマイクロプログラムをモードによって切り換えられる機構を，たとえば WAM 命令における読み出し・書き込みモードの切り換えにおいて利用している．

[M5] X-1 (Xenologic)[86]：Prolog 用の WAM 命令に Lisp 用命令を付加したマシン命令アーキテクチャである．Prolog や Lisp プログラムは，いったんこのマシン命令列にコンパイルされる．組み込み関数や組み込み述語もマシン命令列へあらかじめコンパイルされ，ライブラリとなる．ポインタたどり，単一化，変数束縛，バックトラックなどに関する基本操作は，直接マイクロプログラムで記述されている．

[M6] ELI-512 (Yale 大学)[87]：500 ビット以上の非常に長い水平型マイクロ命令によって低レベル並列処理（この方式を "VLIW" と呼んでいる）を行う．Lisp, Fortran, C, Pascal のプログラムを「Nアドレス・コード」と呼ぶ RISC レベルの中間言語命令にコンパイルし，それを「トレース・スケジューリング法」という最適化手法を通してマイクロプログラムを得る．Multiflow Computer が商用機[88] としての開発を引き継いでいる．

また，高級言語の構文を BNF 記法で，意味を SDL という意味定義言語で記述し，それらに 17 個の形式変換規則を適用し，実行時間とプログラム容量とのトレードオフを評価して，最終的にはハードウェアあるいはマイクロプログラムで直接インタプリト可能な中間言語プログラムにする，という一種のアーキテクチャ記述言語の提案[89]もある．

3.3 ユニバーサル・ホスト計算機

3.3.1 ユニバーサル・ホスト計算機の定義

本解説において「ユニバーサル・ホスト計算機」としている計算機は，マイクロプログラム制御方式を採り，特定のマシン命令を意識していない可変構造型であり，ユーザがマイクロプログラミング可能（ダイナミック・マイクロプログラミング）であるものである．もちろん，マイクロ命令格納用の制御記憶とデータやマシン命令格納用の主記憶とは異なるファシリティが充てられ，特にユーザが定義するマシン命令のフェッチやデコードを支援する機構や，マシン命令とマイクロプログラムとのリンク機構を備えているのが普通である．

ユニバーサル・ホスト計算機では，問題ごとに基本操作をマイクロプログラム化することも可能であるが，対象プログラミング言語を設定して，その処理プログラム（多くは，インタプリタ）をファームウェアとして装備する方式を採ることも有効である．これは，間接実行型高級言語計算機のエミュレーションとみなすことができる．ただし，種々の要求を抱える大規模システムにおいては，マイクロプログラムの動的な入れ換えがオーバヘッドとなるので，どちらかというと，パーソナル計算機システムに適している．前述した複数言語向き高級言語計算機とは，(a)エミュレーション対象が高級言語計算機だけではなく汎用計算機などをも含んでいる，(b)問題向きマイクロプログラムを直接書き下すことも多い，などの点で区別している．しかし，複数言語向き高級言語計算機とユニバーサル・ホスト計算機のアーキテクチャ上の区別はなかなか難しいのが現状である．

また，過去には商用機も存在したユニバーサル・ホスト計算機が，結局実験機の域を脱しきれない理由としては，(i)複数のプログラミング言語の利用環境を提供するコストが大きすぎる，(ii)機能レベルに差があるプログラミング言語を対象とする場合には，適当な中間言語向き仮想計算機を構築して，その上により高レベルの言語処理機能をソフトウェアで積み上げる方式で十分である，ことなどが考えられよう．

3.3.2 ユニバーサル・ホスト計算機のプログラミング言語処理への応用例（付表-3 参考）

[U1] MUNAP (宇都宮大学)[90]：Prolog 処理[91]では，Prolog プログラムと同じ機能レベルの中間言語命令をマイクロプログラムでインタプリトし，MUNAP の低レベル並列処理機能を利用して，特に単一化における引数間／構造体間並列処理を実現している．オブジェクト指向言語の処理[92]では，メッセージ・パッシングと1対1対応する4種のマシン命令を設定し，それをファームウェア・インタプリタでエミュレートする．オブジェクト・ポインタやメッセージ・セレクタの並列処理もマイクロプログラムによって実現している．動的には，バイト・コードによる場合の約半分のマシン命令ステップでインタプリトが実行さ

れ，動的に実行されたマイクロ命令ステップも 19% 減少している．また，リスト処理用言語 L^6 のインタプリタに，ソフトウェアのデバッグを支援するマイクロプログラムを動的に埋め込む方式で，マイクロプログラム技術をソフトウェア・テストへ応用する試み[93]も行っている．

[U2] QA-2 (京都大学)[94]：Prolog 処理[95]では，まず Prolog プログラムと同じ機能レベルにある1次元配列形式の中間言語のインタプリタ (QPM-Ih) と，WAM レベルのマシン命令プログラムのインタプリタ (QPM-Il) とを作成し，対象言語の機能レベルと QA-2 アーキテクチャとの相性を比較した．QPM-Il の性能は，QPM-Ih の約 1.6 倍でしかなく，QA-2 のような低レベル並列処理方式のユニバーサル・ホスト計算機においては，高い機能レベルのマシン命令アーキテクチャを直接マイクロプログラムでインタプリトする方式もかなり有効であることがわかった．そこで，WAM レベルのプログラムをさらに QA-2 のマイクロプログラムにコンパイルする方式 (QPM-Cn) を試みたところ，QPM-Ih の約4倍の性能が発揮できた．コンパイルされたマイクロプログラムに局所的最適化を適用すると (QPM-Co)，さらに約 20% 実行マイクロ命令ステップが減少する．最終的には，QPM-Co は QPM-Ih の約5倍の性能を発揮するまでにチューニング（問題適応化）されたことになる．そのほかに，Lisp や C のファームウェアによる処理方式について実験・評価[96],[97]している．

歴史的には，QM-1, Alto, B-1700, BBN の MBB[98]などがユニバーサル・ホスト計算機として著名である．そのほかに，プログラミング言語処理への応用については明示されていないが，ユニバーサル・ホスト計算機と考えられるものには，慶応大学の KMP/II[99]，イギリス・RSRE の FLEX[100]，CDC の VLSI マルチ ISA エミュレータ[101]，ブラジル・リオデジャネイロ連邦大学の MLM[102]，早稲田大学の Proteus[103]などがある．

4. おわりに

アーキテクチャを問題やアルゴリズムに柔軟に適応させて種々の応用の高速化を図りたいという要求にマイクロプログラム技術が適合し，さらに，マイクロプログラムによって実現される制御論理の規則性や一様構造性が VLSI 技術の求める要求によく適合することが，マイクロプログラム技術の進展・成熟を促した．現在では，ファームウェアが計算機システムにおける機能の一端を担うことはごく当たり前になっている．

本解説では，マイクロプログラム技術のプログラミング言語処理への応用について，各種計算機アーキテクチャの処理機能を整理した．紙数の都合で触れることのできなかったアーキテクチャ・パラダイムで重要なものとしては，超並列計算機，データフロー計算機，データベース・マシン，スーパコンピュータなどがある．これらについては，マイクロプログラム技術の本質的な役割がまだ不明確であるものも多いが，アーキテクチャの詳細については文献[104]などで補って欲しい．

特に，データフロー計算機では，(i)対象言語は高級言語であり，そのプログラムから生成されたデータフロー・グラフも高級言語とみなすことができる，(ii)命令やデータ・パケットをマイクロ命令とみることができる，(iii)命令機能に階層性があり，その中でも機能の小さい命令をマイクロ命令とみなせる，(iv)実際に試作されたデータフロー計算機において，その機能ユニットがマイクロプログラム制御方式であることを明示しているもの[105]-[108]が存在する，などの諸点を考慮すると本解説に含むべきであったかもしれないが，これらも紙数の制限で割愛せざるをえなかった．

また，最近開発が盛んになっている並列計算機の，粒度（並列処理単位の機能の大きさ）と要素プロセッサ自体のトレードオフとの関係についても研究を深める必要があろう．

計算機の構成方式に対しても VLSI 技術の進展が大きな影響力を及ぼしていること[109],[110]は，すでに本解説の各所で指摘した．ASIC として専用計算機を作る方式が普及すると，ユニバーサル・ホスト計算機の価値がなくなるのではないか，また，チップ設計においてはチップ上に搭載する機能の選定が重要であり，従来のようなアーキテクチャ設計法は不用になるのではないか，といった指摘もある．しかし，ハードウェアが文字どおり柔軟性に欠けるものであり，応用の拡大とともに種々のプログラミング言語が設計されるであろう将来を考えると，ファームウェアを活用できる場がまだ十分に存在すると判断できよう．

謝辞 日ごろご指導いただく，京都大学工学部 萩原宏先生に深甚なる謝意を表します．また，本解説のテーマに関しまして種々ご討論いただきました，九州大学大学院総合理工学研究科 富田眞治先生，宇都

宮大学工学部 馬場敬信先生，日本電気（株）C & C 研究所 小池誠彦氏，同じく中田登志之氏，三菱電機（株）情報電子研究所 中島浩氏に，心より感謝の意を表します．

参 考 文 献

1) 萩原ほか：マイクロプログラミング特集号，情報処理，Vol. 14, No. 6, pp. 374-465 (1973).
2) 萩原 宏：マイクロプログラミング，産業図書 (1977).
3) 富田眞治：処理装置の構成，VLSI コンピュータ I，元岡編，岩波書店，pp. 11-172 (1984).
4) 馬場敬信：マイクロプログラミング，昭晃堂 (1985).
5) 箱崎，山本：高級言語マシンの実際，産報出版 (1981).
6) 飯塚，田中編：ソフトウェア指向アーキテクチャ，オーム社 (1985).
7) 柴山ほか：高級言語の処理システムにおけるソフトウェア／ファームウェア／ハードウェアのトレードオフについて，情報処理学会第 20 回全国大会講演論文集，2B-3, pp. 21-22 (1979).
8) Gehringer, E. F. and Keedy, J. L.: Tagged Architecture: How Compelling Are its Advantages?, 12th Annual Int. Symp. Comput. Architecture, pp. 162-170 (1985).
9) Stallings, W. ed.: Reduced Instruction Set Computers, IEEE (1986).
10) Hopkins, W. C.: HLLDA Defies RISC: Thoughts on RISCs, CISCs, and HLLDAs, 16 th Annual Workshop Microprogramming, pp. 70-74 (1983).
11) Ditzel, D. R. and Patterson, D. A.: Retrospective on High-Level Language Computer Architecture, 7th Annual Int. Symp. Comput. Architecture, pp. 97-104 (1980).
12) DeRosa, J. et al.: A Design and Implementation of the VAX 8600 Pipeline, IEEE COMPUTER, Vol. 18, No. 5, pp. 38-48 (1985).
13) Gee, J. et al.: The Implementation of Prolog via VAX 8600 Microcode, 19th Annual Workshop Microprogramming, pp. 68-74 (1986).
14) Fagin, B. et al.: Compiling Prolog into Microcode: A Case Study Using the NCR/32-000, 18th Annual Workshop Microprogramming, pp. 79-88 (1985).
15) Kavi, K. et al.: HLL Architecture, Pitfalls and Predirections, 9th Annual Int. Symp. Comput. Architecture, pp. 18-23 (1982).
16) 柴山 潔：記号処理マシン，情報処理，Vol. 28, No. 1, pp. 27-46 (1987).
17) Pleszkun, A. R. and Thazhuthaveetil, M. J.: The Architecture of Lisp Machines, IEEE COMPUTER, Vol. 20, No. 3, pp. 35-44 (1987).
18) Sansonnet, J. P. et al.: Direct Execution of LISP on a List-Directed Architecture, ACM Symp. Architectural Support for Programming Languages and OS's, pp. 132-139 (1982).
19) Sansonnet, J. P. et al.: M3L: A List-Directed Architecture, 7th Annual Int. Symp. Comput. Architecture, pp. 105-112 (1980).
20) 前川ほか：高速 LISP マシンとリスト処理プロセッサ EVALII—アーキテクチャとハードウェア構成，情報処理学会論文誌，Vol. 24, No. 5, pp. 683-688 (1983).
21) 斉藤ほか：高速 LISP マシンとリスト処理プロセッサ EVALII—インタプリタによるマシンの評価，情報処理学会論文誌，Vol. 24, No. 5, pp. 689-695 (1983).
22) 服部ほか：ALPHA—高性能 Lisp マシン，情報処理学会研究報告，Vol. SM-23, No. 1, p. 8 (1983).
23) Yuhara, M. et al.: Evaluation of the FACOM ALPHA Lisp Machine, 13th Annual Int. Symp. Comput. Architecture, pp. 184-190 (1986).
24) Moon, D. A.: Architecture of the Symbolics 3600, 12th Annual Int. Symp. Comput. Architecture, pp. 76-83 (1985).
25) 平木，後藤：数式処理計算機 FLATS のアーキテクチャ，情報処理学会論文誌，Vol. 27, No. 1, pp. 81-89 (1986).
26) 大田ほか：Explorer (KS-301), 高機能ワークステーション，前川，清水編，共立出版，pp. 136-143 (1986).
27) Krueger, S. D. and Bosshart, P. W.: High LISP Performance on a High-Level Language Processor, COMPCON Spring 87, pp. 120-123 (1987).
28) 日比野ほか：Lisp マシン ELIS のアーキテクチャーメモリレジスタの汎用化とその効果—，情報処理学会研究会報告，Vol. SM-24, No. 3, p. 8 (1983).
29) Watanabe, K. et al.: A 32b LISP Processor, 1987 IEEE Int. Solid-State Circuits Conf., THPM16. 4, pp. 200-201 (1987).
30) 神尾ほか：ELIS における Lisp コンパイラ，情報処理学会研究会報告，Vol. SM-40, No. 4, p. 8 (1987).
31) Bosshart, P. W. et al.: A 553K-Transister LISP Processor Chip, 1987 IEEE Int. Solid-State Circuits Conf., THPM 16. 5, pp. 202-203 (1987).
32) Sussman, G. J. et al.: Scheme-79—Lisp on a Chip, IEEE COMPUTER, Vol. 14, No. 7, pp. 10-21 (1981).
33) Tick, E.: Sequential Prolog Machine: Image and Host Architectures, 17th Annual Workshop Microprogramming, pp. 204-216 (1984).

34) Tick, E. and Warren, D. H. D.: Towards a Pipelined Prolog Processor, IEEE 1984 Int. Symp. Logic Programming, pp. 29-40 (1984).
35) Clocksin, W. F.: Design and Simulation of a Sequential Prolog Machine, New Generation Computing, Vol. 3, OHMSHA and Springer-Verlag, pp. 101-120 (1985).
36) Hermenegildo, M. V. and Warren R. A.: Designing a High Performance Parallel Logic Programming System, ACM Comput. Architecture News, pp. 43-52 (1987).
37) 浅川ほか：複数のアーキテクチャをターゲットにした高速 Prolog コンパイラ，情報処理学会研究会報告，Vol. SM-37, No. 3, p. 8 (1986).
38) Dobry, T.P. et al.: Design Decisions Influencing the Microarchitecture for a Prolog Machine, 17th Annual Workshop Microprogramming, pp. 217-231 (1984).
39) Dobry, T. P. et al.: Performance Studies of a Prolog Machine Architecture, 12th Annual Int. Symp. Comput. Architecture, pp. 180-190 (1985).
40) Despain, A. et al.: Aquarius, ACM Comput. Architecture News, Vol. 15, No. 1, pp. 22-34 (1987).
41) 田村ほか：シーケンシャル実行型 Prolog マシン PEK—ハードウェア構成—，情報処理学会論文誌，Vol. 26, No. 5, pp. 855-861 (1985).
42) 宮本ほか：Prolog マシン PEK における Prolog 中間コードについて，情報処理学会研究会報告，Vol. SM-39, No. 2, p. 8 (1986).
43) Nakazaki, R. et al.: Design of a High-Speed Prolog Machine (HPM), 12th Annual Int. Symp. Comput. Architecture, pp. 191-197 (1985).
44) Nakajima, K. et al.: Evaluation of PSI Micro-Interpreter, COMPCON Spring 86, pp. 173-177 (1986).
45) Nakashima, H. and Nakajima, K.: Hardware Architecture of the Sequential Inference Machine: PSI-II, IEEE 1987 Int. Symp. Logic Programming, p. 10 (1987).
46) Abe, S. et al.: High Performance Integrated Prolog Processor IPP, 14th Annual Int. Symp. Comput. Architecture, pp. 100-107 (1987).
47) 阿部ほか：内蔵型 Prolog プロセッサ IPP の最適コンパイル方式，情報処理学会研究会報告，Vol. PL-10, No. 3, p. 7 (1987).
48) Civera, P. L. et al.: An Experimental VLSI Prolog Interpreter: Preliminary Measurements and Results, 14th Annual Int. Symp. Comput. Architecture, pp. 117-126 (1987).
49) 長沼ほか：連想メモリを用いた Prolog マシン (ASCA)，電子情報通信学会技術研究報告，Vol. CPSY86, No. 52, pp. 1-12 (1987).
50) Yokota, M. and Umemura, M.: DAWN: An Architecture toward a Personal Inference Machine, Int. Workshop High-Level Language Comput. Architecture, pp. 105-112 (1982).
51) 平田ほか：PIE の構造メモリ試作ハードウェアの設計について，電子通信学会技術研究報告，Vol. EC85, No. 64, pp. 55-65 (1986).
52) 柴山ほか：論理型言語向き並列計算機 KPR のアーキテクチャ，ICOT The Logic Programming Conf. '87, 9.1, pp. 183-192 (1987).
53) Samples, A. D. et al.: SOAR: Smalltalk without Bytecodes, ACM Object-Oriented Programming Systems, Languages and Applications '86 Conf., pp. 107-118 (1986).
54) Lewis, D. M. et al.: Swamp: A Fast Processor for Smalltalk-80, ACM Object-Oriented Programming Systems, Languages and Applications '86 Conf., pp. 131-139 (1986).
55) Western Digital Corp.: Pascal MICROENGINE, p. 3 (1980).
56) Salava, A. et al.: MPS 10—A Computer Architecture Supporting Ada, Int. Workshop High-Level Language Comput. Architecture, p. 8 (1982).
57) Jones, J.: A Pascal Oriented Microprogrammed Processor, Int. Workshop High-Level Language Comput. Architecture, pp. 152-160 (1982).
58) Suzuki, N. et al.: Sword32: A Bytecode Emulating Microprocessor for Object-Oriented Languages, ICOT Int. Conf. Fifth Generation Computer Systems, pp. 389-397 (1984).
59) 飯塚，山田：Smalltalk-80 専用マシンのハードウェア構成，情報処理学会第 32 回全国大会講演論文集，5S-1, pp. 253-254 (1986).
60) Beetem, J. et al.: The GF11 Parallel Computer, Experimental Parallel Computing Architectures, J. J. Dongarra ed., North-Holland, pp. 255-298 (1987).
61) 板野，佐藤：汎用直接実行型計算機 UDEC のアーキテクチャ，情報処理学会論文誌，Vol. 27, No. 8, pp. 747-753 (1986).
62) Ditzel, D. R. and McLellan H. R.: The Hardware Architecture of the CRISP Microprocessor, 14th Annual Int. Symp. Comput. Architecture, pp. 309-319 (1987).
63) Speckhard, A. E. et al.: The Aerospace Research Computer (ARC): A Status Report, Int. Workshop High-Level Comput. Architecture, pp. 2. 29-36 (1984).
64) Lecussan, B.: A High Level Language Computer System [HLLCS], Int. Workshop High-Level Comput. Architecture, pp. 8. 16-29 (1984).

65) Barron, I. M.: The Transputer and Occam, IFIP Congress 86, pp. 259-265 (1986).
66) Lehr, T. F.: The Implementation of a Production System Machine, Proc. 19th Annual Hawaii Int. Conf. System Sciences, pp. 177-186 (1986).
67) 阿江, 相原: リアルタイム・プロダクションシステム ISAC の構想, 電子情報通信学会技術研究報告, Vol. CPSY 87, No. 1, pp. 1-6 (1987).
68) 樋口ほか: 並列連想記憶を用いた意味ネットワークマシン, 電子通信学会技術研究報告, Vol. EC85, No. 55, pp. 9-20 (1986).
69) Moldvan, D. I.: An Associative Array Architecture Intended for Semantic Network Proc. ACM '84 Annual Conf. The Fifth Generation Challenge, pp. 212-221 (1984).
70) Mankovich, T. E. et al.: CHoPP Principles of Operation, 2nd Int. Conf. Supercomputing, pp. 2-10 (1987).
71) Milutinovic, V. et al.: The VM Architecture: A HLL Microprocessor Architecuture for Dedicated Real-Time Applications, Int. Workshop High-Level Comput. Architecture, pp. 7. 20-27 (1984).
72) 高井ほか: 汎用パイプラインを簡約エンジンに用いた FP グラフリダクションマシン, 情報処理学会研究会報告, Vol. CA-63, No. 14, pp. 129-139 (1986).
73) Castan, M. and Organick, E. I.: μ3L: An HLL-RISC Processor for Parallel Execution of FP-Language Programs, 9th Annual Int. Symp. Comput. Architecture, pp. 239-247(1982).
74) Kieburtz, R. B.: The G-machine: A Fast, Graph-Reduction Evaluator, Functional Programming Languages and Computer Architecture, Springer-Verlag, pp. 400-413 (1985).
75) Mago, G. and Middleton, D.: The FFP Machine—A Progress Report, Int. Workshop High-Level Comput. Architecture, pp. 5. 13-25 (1984).
76) Anderson, J. M. et al.: The Architecture of FAIM-1, IEEE COMPUTER, Vol. 20, No. 1, pp. 55-65 (1987).
77) Myers, G. J. and Buckingham, B. R. S.: A Hardware Implementation of Capability-Based Addressing, ACM Comput. Architecture News, Vol. 8, No. 6, pp. 12-24 (1980).
78) 守屋, 安田: Xerox 1121 と JStar, 高機能ワークステーション, 前川, 清水編, 共立出版, pp. 159-165 (1986).
79) Pier, K. A.: A Retrospective on the Dorado, A High-Performance Personal Computer, 10th Annual Int. Symp. Comput. Architecture, pp. 252-269 (1983).
80) Lampson, B. W. et al.: An Instruction Fetch Unit for a High-Performance Personal Computer, IEEE Trans. Comput., Vol. C-33, No. 8, pp. 712-730 (1984).
81) Monier, L. and Sindhu, P.: The Architecture of the Dragon, COMPCON Spring 85, pp. 118-121 (1985).
82) Johnsson, R. K. and Wick, J. D.: An Overview of the Mesa Processor Architecture, ACM Symp. Architectural Support for Programming Languages and OS's, pp. 20-29 (1982).
83) Nojiri, T. et al.: Microprogrammable Processor for Object-Oriented Architecture, 13th Annual Int. Symp. Comput. Architecture, pp. 74-81 (1986).
84) 斉藤ほか: AI ワークステーションの開発思想, 人工知能学会全国大会 (第1回) 論文集, 5-1, pp. 237-240 (1987).
85) 相川ほか: AI プロセサーのアーキテクチャ, 人工知能学会全国大会 (第1回) 論文集, 5-2, pp. 241-244 (1987).
86) Dobry, T.: A Coprocessor for AI; LISP, Prolog and Data Bases, IEEE COMPCON Spring 87, pp. 396-407 (1987).
87) Fisher, J. A.: Very Long Instruction Word Architectures and the ELI-512, 10th Annual Int. Symp. Comput. Architecture, pp. 140-150 (1983).
88) Fisher, J. A.: VLIW Architectures: Supercomputing via Overlapped Execution, 2nd Int. Conf. Supercomputing, pp. 353-361 (1987).
89) Bose, P. and Davidson, E. S.: Design of Instruction Set Architectures for Support of High-Level Language, 11th Annual Int. Symp. Comput. Architecture, pp. 198-206 (1984).
90) 馬場ほか: 2レベルマイクロプログラム制御計算機 MUNAP のアーキテクチャ, 電子通信学会論文誌, Vol. J64-D, No. 6, pp. 518-525 (1981).
91) 稲川ほか: Prolog の単一化における引数間の並列処理, 電子情報通信学会論文誌, Vol. J70-D, No. 2, pp. 298-306 (1987).
92) 土井ほか: 低レベル並列処理計算機 MUNAP によるオブジェクト指向型言語の高速処理, 電子情報通信学会論文誌, Vol. J70-D, No. 2, pp. 307-314 (1987).
93) 馬場ほか: 計算機アーキテクチャの支援によるソフトウェアテストシステム, 電子情報通信学会論文誌, Vol. J69-D, No. 1, pp. 30-41 (1986).
94) 北村ほか: ユニバーサル・ホスト計算機 QA-2 の低レベル並列処理方式, 情報処理学会論文誌, Vol. 27, No. 4, pp. 445-453 (1986).
95) 柴山ほか: ユニバーサル・ホスト計算機 QA-2 による逐次型 Prolog マシンのエミュレーション, 情報処理学会論文誌, Vol. 27, No. 8, pp. 754-767 (1986).
96) Tomita, S. et al.: A Computer with Low-

Level Parallelism QA-2—Its Applications to 3-D Graphics and Prolog/Lisp Machines—, 13th Annual Int. Symp. Comput. Architecture, pp. 280-289 (1986).
97) 斉藤ほか：低レベル並列処理計算機 QA-2 のためのC言語マイクロプログラム・コンパイラ，情報処理学会第 35 回全国大会講演論文集，1C-6, pp. 103-104 (1987).
98) Kraley, M. et al.: Design of a User-Microprogrammable Building Block, 13th Annual Workshop Microprogramming, pp. 106-114 (1980).
99) Tokoro, M. et al.: A High Level Multi-Lingual Multiprocessor KMP/II, 7th Annual Int. Symp. Comput. Architecture, pp. 325-333 (1980).
100) Kershaw, J.: Two Implementations of the 'FLEX' Machine, 14th Annual Workshop Microprogramming, pp. 25-37 (1981).
101) Wilkes, J. L.: Architecture of a VLSI Multiple ISA Emulator, 17th Annual Workshop Microprogrammimg, pp. 31-36 (1984).
102) Franca, F. M. G. et al.: Design and Realization of MLM: A Multi-Lingual Machine, 19th Annual Workshop Microprogramming, pp. 129-137 (1986).
103) 村田ほか：マイクロプログラマブルプロセッサ "Proteus" アーキテクチャとハードウェア構成，情報処理学会第 34 回全国大会講演論文集，7P-7, pp. 213-214 (1987).
104) 田中ほか：特集：並列処理マシン，情報処理，Vol. 28, No. 1, pp. 3-105 (1987).
105) Gurd, J. et al.: The Manchester Dataflow Computing System, Experimental Parallel Computing Architectures, J. J. Dongarra ed., North-Holland, pp. 177-219 (1987).
106) Patt, Y. N. et al.: Run-Time Generation of HPS Microinstructions from a VAX Instruction Stream, 19th Annual Workshop Microprogramming, pp. 75-81 (1986).
107) Srini, V. P.: A Message-Based Processor for a Dataflow System, Int. Workshop High-Level Comput. Architecture, pp. 1. 10-19 (1984).
108) Kishi, M. et al.: DDDP: A Distributed Data Driven Processor, 10th Annual Int. Symp. Comput. Architecture, pp. 236-242 (1983).
109) Tredennick, N.: The Impact of VLSI on Microprogramming, 19th Annual Workshop Microprogramming, pp. 2-5 (1986).
110) 日比野靖：ISSCC'87 報告 32 ビットマイクロプロセッサの動向，電子情報通信学会技術研究報告，Vol. CPSY87, No. 6, pp. 39-44 (1987).
111) DuBose, D. K. et al.: A Microcoded RISC, 9th Annual Workshop Microprogramming, pp. 124-128 (1986).
112) Ungar, D. et al.: Architecture of SOAR: Smalltalk on a RISC, 11th Annual Int. Symp. Comput. Architecture, pp. 188-197 (1984).

(昭和 62 年 10 月 30 日受付)

付表-1 主要なマイクロプログラム制御 Lisp マシン

番号	名称	開発機関	マイクロ命令長(ビット)	制御記憶容量(ワード)	マイクロ命令サイクル(ns)
L1	M3L	Paul Sabatier大	32 100†	4K 256†	500
L2	EVLISマシン	阪大	48	8K	100~
L3	FACOM ALPHA	富士通	48	32K	160
L4	Symbolics 3600	Symbolics	112	16K	180 ~250
L5	FLATS	理研	240	1024	120 ×2
L6	Explorer	TI	56	16K	142
L7	ELIS	NTT	64	64K	180~
L8	(Lispチップ)	TI	64	32K	30

† ナノ命令長、ナノ命令記憶容量

付表-2 主要なマイクロプログラム制御 Prolog マシン

番号	名称	開発機関	マイクロ命令長(ビット)	制御記憶容量(ワード)	マイクロ命令サイクル(ns)
P1	PLM	UCB	144	1K	100
P2	PEK	神戸大	96	16K	120 ~400
P3	CHI	日本電気 ICOT	80	11K	100
P4	PSI	三菱電機 ICOT	64	16K	200
P5	PSI-II	三菱電機 ICOT	53	16K	167
P6	IPP	日立製作所	64	2K	不明
P7	(VLSI Prolog インタプリタ)	DEPT	110	1.5K	200*
P8	ASCA	NTT	144	4K	200

* シミュレーション

付表-8 主要なマイクロプログラム制御高級言語計算機とユニバーサル・ホスト計算機

番号	名称	開発機関	マイクロ命令長(ビット)	制御記憶容量(ワード)	マイクロ命令サイクル(ns)	対象言語
S1	Pascal MICROENGINE	Western Digital	22	512	不明	Pascal
S2	MPS 10	Nokia Electronics	104	4K	100 ~220	Ada
S3	POMP	Trinity大	72	1K	250	Pascal
S4	Sword 32	東大	不明	4K	125	Smalltalk-80
S5	Swamp	Toronto大	不明	不明	136	Smalltalk-80
S6	Hobbes	沖電気	48	16K	125	Smalltalk-80
S7	GF 11	IBM	200	512K	50	Pascal
S8	MIRIS	George Mason大	64	256	不明	C
M1	SWARD	IBM	52	6K	不明	Cobol, PL/I, Fortran
M2	1121	Xerox	48	16K	137	Mesa, Lisp, Smalltalk-80
M3	AI 32	日立製作所	64	4K	125	Lisp, Prolog, Smalltalk-80
M4	WINE	東芝	96	4K	100	Lisp, Prolog, C
M5	X-1	Xenologic	不明	不明	不明	Lisp, Prolog
U1	MUNAP	宇都宮大	28 40'×4	8K 4K'	550	Prolog, L6, Smalltalk-80
U2	QA-2	京大	256	4K	600~	Prolog, Lisp, C

'ナノ命令長、ナノ命令記憶容量