Lorel-2 実行システムの構成とそのファームウェア化の効果†

宮地利雄†† 片山卓也†† 榎本肇††

各種の組合せシステムの問題を記述する事を目的として，筆者らにより開発された高級プログラミング言語 Lorel-2 の処理系である実行システムの構成を述べ，その特徴を明らかにする．本システムでは，内部表現として，セル構造と呼んでいるリスト構造を主として用いているが，そのためのセルプールを仮想記憶上の複数セグメントに分割し，この分割を意識しながら，並列に動作するセル再生タスクを含む記憶管理により局所性を保ちながら管理し，実行効率の向上と大規模な記憶容量を実現している．また，連想検索を高速に実行するための hash 属性については，これを指定した変数にも書きかえが許されるような実現を行い，広い範囲への hash の利用を許している．さらに，セル構造処理機能の主要部がファームウェア化されており，実行速度を中心とする性能改善がはかられている．

1. まえがき

Lorel-2 は，筆者らにより開発されたグラフ，オートマトン，形式言語等の組合せシステムを記述するための高級言語である．その言語仕様は先に報告しているが[1]，豊富なデータ型や Σ 式などの高レベルの記述機能や set や tree データに対する連想検索機能などを特徴としている．処理系は，コンパイラと実行システムを中心として構成され，中型の汎用計算機シリーズ 77 ACOS システム 400 (以降 Acos-400 と呼ぶ) 上で実現している．本論文では，実行システムの構成と特徴を述べるとともに，そのファームウェア化の効果について評価を行う．

本システムは Lorel-2 言語仕様の主要基本機能を完全に提供し，ファームウェア技術の利用や並列処理等の諸技術の採用により高い実行効率を得るとともに，中間言語として OS が提供している既存言語を選びそのプリプロセッサとして Lorel-2 コンパイラを実現することにより他言語で記述された手続きとのリンケージを含む分離コンパイルや仮想記憶など OS が提供する豊富な機能の Lorel-2 からの利用を許していることが特徴である．そして，これらにより Lorel-2 の高度のデータ処理機能を用いて記述された実用規模のプログラムを効率よく実行することを可能にしている．

Lorel-2 実行システムは，概念的にはコンパイラが出力した Lorel-2 基本命令を実行する仮想機械であり，物理的にはファームウェア化部分を含むソフトウェア・システムである．ここで扱うデータの多くは，内部的にはセル構造と呼ばれる一種のリスト構造として記憶されており，これに対する基本的操作の多くは拡張ファームウェアにより実行される．また，セルプールは，OS が提供する仮想記憶空間内の複数セグメントに分割して配置しているが，セル・アクセスの局所性を保つための配慮をしており，さらに，ごみセルを自由セルにもどす再生処理（いわゆるガーベジ・コレクタに相当するが，ごみのセル構造は明白に捨てられるのでマーキングや再配置等の処理は含まない）については，子タスクとして通常の実行と並列に行って効率向上をはかっている．

2. Lorel-2 言語仕様の概略

Lorel-2 は，複雑なデータを扱う論理関係処理を対象とした手続き型の高級言語である．プログラムは手続きの集まりから構成され，各手続きは，文を単位として，条件文や繰返し文等の制御構造を用いて構造的に記述される．使用される変数や手続きは，すべてデータ型とともに明白に宣言することが必要である．

データ型は，単純データである整数，実数，論理値，文字のほか，これをもとに set, tuple, tree, 関数型を利用して複合データ型を再帰的に構成していくことにより，任意に複雑なものを得ることができる．set は同一データ型の要素からなる可変長（空を含む）リストを，tuple は一般には異なるデータ型の要素からなる固定長リストを，tree は指定されたデータ型の節点を持つ n 進木を表す．関数型は引数と返却値のデータ型を指定した関数／手続きを表す．また，特殊な

† Construction of Lorel-2 Executor and its Improvement by Firmware by TOSHIO MIYACHI, TAKUYA KATAYAMA and HAJIME ENOMOTO (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

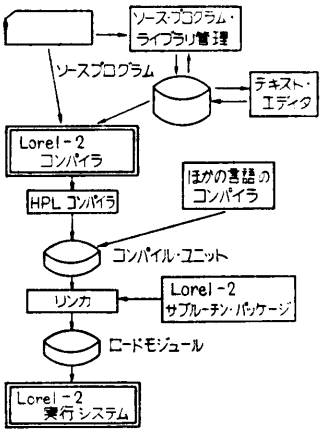


図 1 Lorel-2 処理系の構成
Fig. 1 Configuration of Lorel-2 system.

データ型として特定の set/tree の要素／部分木を参照するための element/subtree がある.

変数は，通常の意味で値の定義や再定義や参照が許され，set, tuple,tree 型の変数ではその値が定義されていれば添字などの方法により部分的に値を選び出し(部分構造変数と呼ぶ) 参照や再定義できる．さらに，set 型の変数では，要素の追加，任意の部分構造変数の削除またはその前後への要素の追加も可能である．これら変数の値の定義または変更の操作は，すべて代入文* の形式で行われる.

また，set や tree 型の変数の宣言時に hash 属性を与えることができ，その変数上で与えられた値を持つ要素／節点が存在するか否かの決定，および与えられた key を持つ要素／節点を捜し出す連想検索などの機能を高速化することができる.

3. 処理系の概要

処理系は，**図1**に示すような構成で，オペレーティング・システムのプログラム管理体系の一部を利用し，その上で実現されている[2),3)].

Lorel-2 コンパイラは，HPL のプリプロセッサとして動作する．すなわち，Lorel-2 のソースプログラムを読み込み，解析とデータ型の検査を行い，Lorel-2 基本命令の列に変換し，これを HPL ソースプログラムの形式で出力する．HPL は，OS が提供している PL/I 系のシステム記述用言語で，機械命令や拡張機械命令を混合使用*することも許されている.

コンパイルは，外部手続きごとに行われ，さらに実行時ルーチンを含めこれらをリンカにより結合し，実行可能なロードモジュールが作られる．これは，ほかの通常の言語と同じ方法で実行されるが，ハードウェア的には拡張命令を含む点だけが異なっている.

このような既成の体系の部分的拡張とプリプロセッサ方式による実現は，①既存の資源や機能を有効に利用することによる実現の容易化，②他言語の手続きとの結合が可能になるため OS の機能をそのまま Lorel-2 利用者に提供できること，などの点で有利であった.

4. 実行システムの構成

4.1 概 要

実行システムは，Lorel-2 実行プログラムの効率的な実行と実行環境の管理を行う．構造的には，**図2**に示されるように Acos-400 の固有ハードウェア／ファームウェア，Lorel-2 用拡張ファームウェア，およびソフトウェアによる実行時ルーチン群からなる.

Lorel-2 実行プログラムは，拡張機械命令を含むことを除き，ほかの言語の実行プログラムと同様の機械語形式のコードだが，概念的には Lorel-2 基本命令の系列であり，実行システムはこれを実行するための仮想機械と考えられる．Lorel-2 基本命令は，概念的な実行の機能単位であり，その実現においては実行時ルーチンの呼出しであったり，拡張機械命令であったり，単一または一群の固有機械命令であったりする.

また，機能的・論理的には**図3**に示すように5つの部分から構成されている．**基本命令実行処理部**では，Lorel-2 実行プログラムの中から基本命令を読出し，これを実行する．このうち，セル構造を取り扱う機能については，その主要部がファームウェア化されていて，拡張機械命令として実現されている．**セルプール**

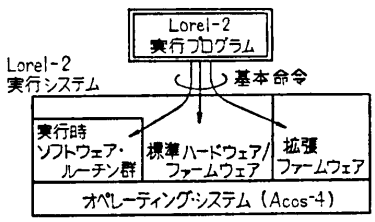


図 2 Lorel-2 実行システムの構成
Fig. 2 Structure of Lorel-2 Executor.

* 代入文には，データの書きかえを行う基本代入文(例. S≔{8.9,1.2,5.6})，set に新しい要素を追加する挿入代入文(例. ↓S{2}≔2.7)，set から要素を削除する要素削除代入文 (例. S{3}≔ε)，element/subtree 型の変数の設定を行う element/subtree 代入文 (例. E≔S{1}) がある.

* 中間言語として HPL を選んでいるのは，この機能を持っている事と，この OS ではコンパイル・ユニットの仕様もアセンブラも公開提供されていないためである.

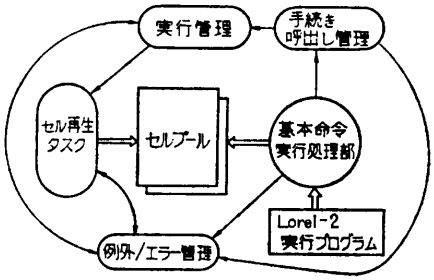
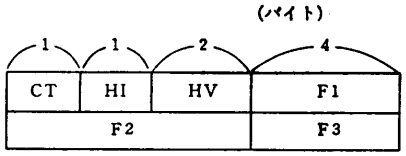


図 3 Lorel-2 実行システムの機能的構造
Fig. 3 Organization of Lorel-2 Executor.



CT: セル・タイプ HI: hash 情報
HV: hash 値計算の中間結果
F1～F3: ポインタ・フィールド
(F2, F3 はデータの格納に使用されることもある.)

図 4 セルのフィールド構成
Fig. 4 Cell field format.

管理では，セルプール上の自由セルとごみセルをリストとして管理し，また，ごみセルから自由セルへの再生処理を行う．なお，この再生処理は子タスクとして主タスクと平行に実行している．**実行管理**では，上述のセル再生処理タスクの起動／停止を含む実行システムの初期化や終了処理などを行う．**手続き呼出し管理**は，Lorel-2 手続きの呼出しや退出に伴う情報の更新を行い，また，外部手続き間における実行時のデータ型検査を行う．なお，再帰的呼出しを含む手続きの呼出し／リターン動作はスタック動作を備えた Acos-400 の標準機能をそのまま利用している．**例外／エラー管理**では，拡張ファームウェアから通知されるエラーやソフトウェア処理を要求する例外，その他の実行システム内から通知されるエラーの検出，分類，処理を行う．

4.2 セル構造

Lorel-2 における通常の変数の値は，set や tree のような動的に大きさが変化するデータを含みうるため，セル構造と呼ばれる一定の形式を備えたリスト構造により記憶される．各変数に対応するセル構造は，変数の各世代ごとにとられるスタック上の変数エントリと呼ばれる記述子に結合されている．セル構造は，セルプール上に用意されたセルを単位としており，セル間は原則として双方向に結合している．双方向結合は，メモリ消費量において不利になることは言うまでもないが，Lorel-2 言語機能中で比較的重要な set の要素の挿入や削除などの操作を効率よく行い，set 要素の逆順のアクセスや tree における子節点から親節点へのアクセスを可能にするためこれを採用した．

表 1 セル・タイプ
Table 1 Cell type variety.

セル・タイプ		機能概要	略号
ヘッド	set 用	set/tuple を代表する節点.	H_s
	tuple 用		H_t
コネクタ	set 用	set/tuple の構成要素を結合する.	C_s
	tuple 用		C_t
単純データセル	integer 用	単純データを格納する.	D_i
	boolean 用		D_b
	real 用		D_r
	string 用		D_s
	procedure用		D_p
参照データセル	element/subtree 用	element/subtree 変数による参照を表わす.	P
	引数用	部分構造を実引数として受け渡す.	SP
	hash 用	データとハッシュ・バケット・リスト間をつなぐ.	HP

セルは，1個が16バイト（4語）の大きさで**図4**に示すようなフィールド構成を持つ．セル構造内のセルは，その機能に従ってCTフィールドで定義されるセル・タイプを持つ．各セル・タイプとその機能の概略を**表1**に示す．HI と HV のフィールドは，hash に関連する情報を格納している．ポインタは32 bit 長で利用可能なセルの個数（88 K セル）に比較して相当に長い．これは各セルの仮想記憶内での論理番地をそのまま利用しているもので，アドレス変換手続きを簡単化してポインタをたどるための時間を短縮した．変数エントリのフィールド構成は，セルの先頭2語の部分と同形式で，CT フィールドは変数エントリである事を示す値がセットされる．**図5**にセル構造の例を示す．

4.3 hash 属性の実現

Lorel-2 では，hash 属性を与えられた変数についても全体または部分的な値の書きかえを許している．我我のシステムでは hash 属性を与えられた変数に対して**図6**のような内部データ構造を用意し，値の書きかえがあるたびに更新していくことにより，これを実現した．ハッシュ表は実行システム内に1個用意されており，各ハッシュ値に対応するバケット・リストへのポインタ配列である．バケット・リストは，セル構造

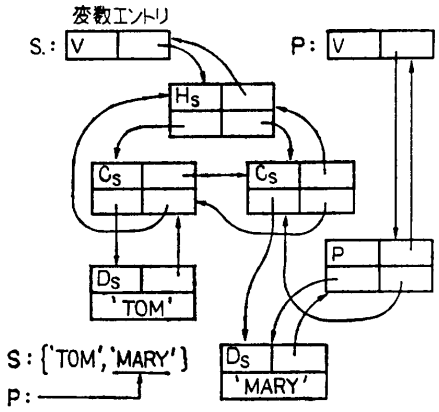


図 5 セル構造の例

Fig. 5 Sample cell structure.

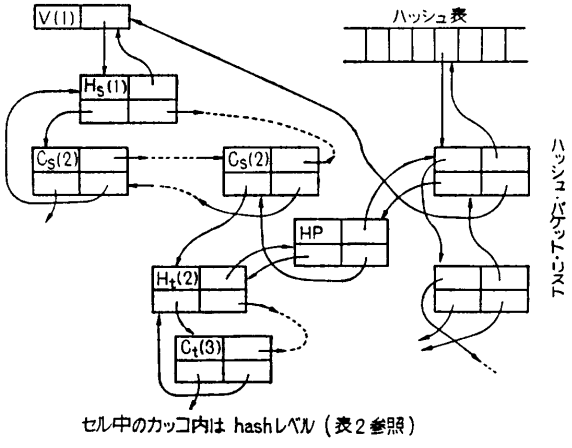


図 6 hash 属性をもつ変数に対する内部表現の例

Fig. 6 Sample data structure for variables with a hash attribute.

と同じセルプール内に作られるが，各セルは4個のポインタのみからなる非標準形式となっている．

hash 属性を与えられた変数の部分的な書きかえに対しても，このデータ構造を矛盾なく効率的に更新するため，セル内の HI フィールドに，**表2**に示す hash タイプと hash レベルの情報を格納しておき，変数またはその一部の値を変更する際にこれを調べ，その結果から必要に応じて hash 値の再計算とバケット・リストの変更，および置換されたセル構造内の HI フィールドのセットを行う．また，hash 値の再計算時の計算量を減らすため，hash 関数をセル構造に対応して帰納的に定義するとともに，セル中の HV フィールドに計算の中間結果を保存しておき，再計算時に利用している．

4.4 セルプールとセルの管理

表 2 HI フィールドの hash 制御情報

Table 2 Hash control information stored in HI field.

hash タイプ	意　味
0	hash とは無関係.
1	tree の所属関係型 hash.
2	set の所属関係型 hash.
3	set の連想検索型 hash.

hash レベル	意　味
1	hash 検索されるデータを部分構造として含む.
2	hash 検索されるデータである.
3	hash 検索されるデータの部分構造である.

（a） セルプール

本システム全体は，Acos-400 の仮想記憶上で動作しており，セグメントと呼ばれる通常 64k バイト以下の大きさの部分空間を単位として記憶管理と保護がなされる．セグメントは，主記憶内常駐度等の属性の指定ができる．この計算機では，いわゆるページングの概念はない．

セルプールは，4K セルずつに分割され，それぞれ64kバイトのセグメントに割り当てられている．セルプールとして22セグメントを用意しており，そのうち1個は主記憶内常駐度が他セグメントよりも高くなるよう指定しており「常駐型」と呼んでいる．ほかの21個を便宜上「非常駐型」と呼ぶ．（常駐型／非常駐型のいずれを利用するかは，原則としてプログラム中で変数ごとに指定するが，一方が不足すればシステムが切換えを行う．）セルプール・セグメントの大きさは，動的に可変で，初期値は0である．常駐型および1個の非常駐型セルプールは，実行システムの初期化時に実行管理部からの要求により4K セルに，ほかの非常駐型セルプールはセルが不足した段階で自由セル管理部からの要求で1K セルずつ4K セルまで拡張して初期化し利用する．

このように，仮想記憶上で複数個に分割されたセルプールを扱う場合，記憶参照の局所性を保つため，①システムによるアクセスの局所化，②セル構造のセル間結合の局所化が重要である．本システムでは，後者については自由セルの管理やセルプール切換えアルゴリズムの工夫で，前者については各セルプールごとに**表3**のように状態を定義しこれをセルプール管理表内に一括して記憶し処理を進めることにより，局所化をはかった．セルプール状態の遷移を**図7**に示す．

（b） 自由セルとごみセルの管理

表 3 セルプールの状態の定義
Table 3 Definition of cell pool state.

状態	自由セル	ごみセル	備考
Original	空	空	初期化される以前である.
Clean	あり	空	
Dirty	?	あり	
Urgent	なし	?	このセルプール上の自由セルを求めて主タスクが待ち状態.
Busy	なし	なし	

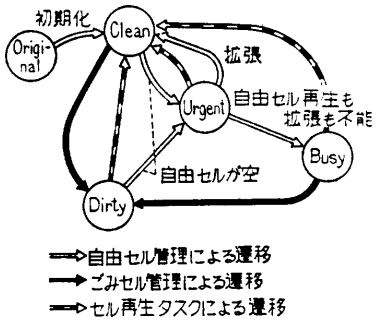


図 7 セルプールの状態遷移図
Fig. 7 State transition diagram of cell pool.

自由セルは，各々が所属するセルプールごとにリストとして線型につながれ，管理される．システムは，現在のセルプールを表示するレジスタを持っており，主タスクは，必要とするセルをこのレジスタが示すセルプールの自由セル・リストから取り出す．このリストが空の場合には，自由セル管理が呼び出され図 8 に示すようにセル再生タスクと同期をとり，これによっても当該セルプールで自由セルが得られない場合には現在のセルプールをほかの可能なものに切り換える．

ごみセルは，不要になったセル構造で，その最上位セルが所属するセルプールのごみセル・リストにつながれる．Lorel-2 では，セル構造が複数の変数に共有されることがないため，セル構造は不要になった時ただちにごみセル・リストへ入れることができる．ごみセル管理は，セル再生タスクと主タスクの同期のため図 9 の動作を行う．

(c) セル再生タスク

ごみセル・リストを構成している不要なセル構造は，セル再生タスクにより分解され，各セルはそれぞれが属するセルプールの自由セル・リストに返される．この処理は，単一 CPU による複数プロセス環境のもとで，主タスクと並列に実行されている．セル再生タスクは，通常の状態では CPU 優先度が主タスク

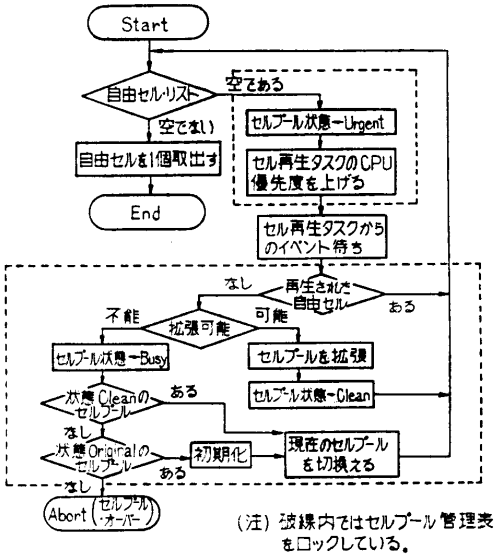


図 8 自由セル管理の動作
Fig. 8 Algorithm of free cell management.

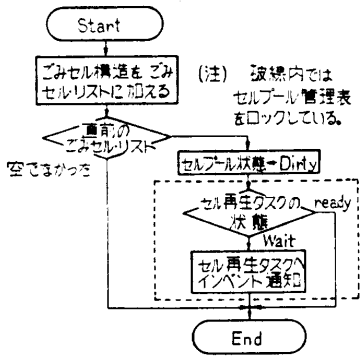


図 9 ごみセル管理の動作
Fig. 9 Algorithm of garbage cell management.

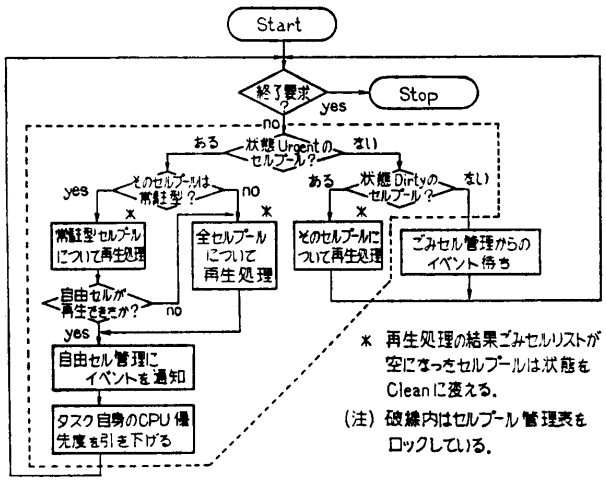


図 10 セル再生タスクの動作
Fig. 10 Algorithm of cell reclamation task.

に対して相対的に1レベルだけ低く指定されていて，主タスクの入出力待ちなどに割り込むように動作している．動作の概略を**図10**に示す．なお，タスク間の同期操作は，Acos-400 の標準機能であるセマフォ命令(機械命令) を利用しており，自由セル・リストとごみセル・リスト上の排他制御については拡張ファームウェアのコーディングの工夫により臨界領域を形成した．

5. ファームウェア化

5.1 概　要

本システムにおいて，セル構造処理のための基本命令は使用頻度が高く，またその多くは複雑な処理を含まず個々の規模も比較的小さい（マイクロプログラムで80～230ステップ）ため，ファームウェア開発上の困難が少なく，しかも効果が大きいものと期待された．拡張命令の呼出しについては，拡張デコール命令を経由する標準的な方法もあるが，OS の改造の必要を避けるとともに拡張ファームウェアに制御が移るまでのオーバヘッドを削減するため，標準命令セットで使用していない命令コードの一部を拡張命令に割り当てることにより新命令を追加する方法を採った．**表4**に使用計算機のマイクロプログラムに関する諸元を掲げる．

5.2 拡張命令の起動と形式

拡張命令を実行するためのマイクロプログラムは，主記憶のファームウェア領域の一部に格納され，ここから実行される．この領域は，計算機を起動した時に外部記憶装置からロードされ，その後は運転終了までソフトウェアによる直接アクセスから保護されている．

拡張命令は，標準命令と同様に主記憶から読出された後，ROS 内の標準ファームウェアによりデコードされる．ここで標準命令セットにない命令コードに対しては，制御を主記憶内に格納された拡張ファームウェアの先頭番地に移すようになっており，デコードからここまでの最小オーバヘッドは4.5μ秒である．その後，拡張命令は拡張ファームウェア内で再度デコードされ，それぞれの実行ルーチンで処理される．以上の制御の流れを**図11**に示す．

表4 Acos-400 のマイクロプログラムの諸元

Table 4 Some data about Acos-400 microprogram.

主記憶サイクル時間：1μ秒/4バイト
ROS 語長：32ビット
　アクセス時間：130n秒
　最大容量：32K語
主記憶の先頭 32K 語の範囲からもマイクロプログラムの実行が可能．
マイクロプログラム・サイクル
　標準：400n秒
　命令により最大 775n 秒まで延びる
マイクロプログラムが主記憶から実行される時には，3～4倍に延ばされる．

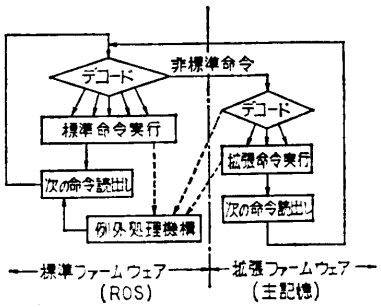
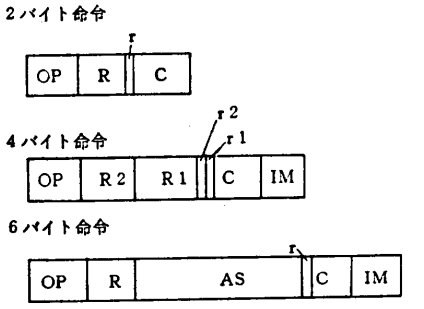


図11 拡張命令の起動

Fig. 11 Initiation of extended instructions and instruction cycle.



OP：命令コード　　C：補助コード
R, R1, R2：レジスタ番号
r, r1, r2：ベースレジスタ／汎用レジスタの指定
AS：オペランド番地の指定
IM：イミーディエイト

図12 拡張命令の形式

Fig. 12 Extended instruction format.

拡張命令の形式は，2, 4, 8バイトの3種類の長さのものがあり，**図12**に示すフィールド構成を持つ．命令内の各フィールド位置は，標準命令とほぼ同じである．これは，デコード用マイクロ命令の機能からの制限と標準ファームウェア内のオペランド番地解析手続きを利用するためである．補助コードCは命令コードの拡張部としてデコードされ，命令コードの不足を補う．レジスタ指定は，8個のベースレジスタ，16個の汎用レジスタの任意のものを指定できる．イミーディエイト IM は，1バイト長の定数を定義する．オペランド番地 AS は，ベース修飾形式で，間接およびインデクス修飾の指定も可能である．

拡張ファームウェアとの情報交換には，上述の命令フィールドのほか，汎用レジスタ中の特定の3個を予

表 5 Lorel-2 基本命令のファームウェア化による速度向上
Table 5 Speed-up of Lorel-2 basic instructions by firmwarelization.

基本命令名	機能概要	実行時間 (μ秒)		比率	マイクロプログラムのステップ数
		ソフトウェア	ファームウェア		
FFC, PGC	自由セル/ごみセル・リストへの付加/からの取出し.	241	54	4.5	合計 120
PFC, FGC		294	81	3.6	
locate (基本)	set や tuple の指定された要素までたどる.	185	52	3.6	85
locate (*)		52	23	2.3	
access	指定されたタイプのセルにアクセス.	190	59	3.2	72
assign	セル構造を置換.	265	66	4.0	175
copy (*)	セル構造を複製.	267	114	2.3	225
compare (*)	セル構造を比較.	263	104	2.5	150
append	セル構造の付加.	369	98	3.8	115
insert	セル構造の挿入.	417	106	3.9	109
delete	set の要素を削除.	470	122	3.9	197
S-iterate	set のセル構造上での繰返し.	434	128	3.4	238
enter, prec, suc		278	50	5.7	64

(注) *を付したものは，1セル当りに換算した時間を示す.

約して特殊目的（現在のセルプールの基底番地の表示等）に使用している．また，拡張ファームウェアからソフトウェアに渡す情報のうち，検出したエラー状態やソフトウェアへの処理要求は（現在のセルプールで自由セルが空になった場合，ハッシュ値の再計算が必要な場合等），標準ファームウェア内の例外処理機構を経由してソフトウェアに通知される．例外処理機構は，リンク時に指定された特定の手続きをファームウェアが与える4語のパラメータを伴って擬似的に呼び出すものである．

5.3 基本命令の処理時間

ファームウェア化した基本命令のうち主要なものについて，命令ごとの実行時間の測定値を，拡張ファームウェアにより拡張命令として実現されたものと，以前に暫定的試験的に使用していたソフトウェア・サブルーチンとの両者を比較して**表5**に示す．ファームウェア化により2～5倍の速度の向上が得られており，さらに現在は主記憶内に格納している拡張ファームウェアを ROS 内に格納することにより，最終的には6～20 倍程度の速度向上率に達すると考えられる．なお，表5に示した多くの命令では，処理されるセル構造をレジスタを介して指定する4バイト長命令と変数エントリを示すオペランド番地を介して指定する6バイト長命令の2種類の形式を持つが，これらについては両者の単純平均値を示した．

ファームウェア化に際しては，本システムが仮想記憶環境下で動作しており，セルプールもその中で複数のセグメントに分割されていることから，アクセスしようとしているセグメントが主記憶内に存在しない場合や，自由セル・リストが空になった場合等におけるマイクロプログラムの実行中断を含む複雑な設計がいくつかの基本命令について必要とされた．locate, copy, compare 等の基本命令の速度向上率が比較的小さいのはそのためで，これらにおいては中断後の再開時にファームウェア内だけで使用する各レジスタについて中断直前の状態を再現して継続できるようにするため，汎用レジスタ上に必要な情報を退避しながら動作している．

なお，ソフトウェア・サブルーチンは，高速性を配慮してコーディングされていたものの，手続き呼出しのオーバヘッド（76.2～96.2μ秒以上）が1回の処理量の少ない命令で重荷になっていたようである．

5.4 Lorel-2 プログラムの速度向上

ファームウェア化による Lorel-2 プログラムの実行時間の短縮は，個々のプログラムで使用する基本命令の割合に大きく影響され，一般的な結論を得にくい．ここでは，2つのプログラム例について，実行速度の測定値と向上率，および拡張ファームウェアが ROS に格納された場合の最終的な速度向上率の推定値を**表6**に示す．これらの例について個々の基本命令の実行時間の割合を算出すると，ソフトウェア版について，Eight Queen では locate 命令が 55% を access 命

表 6 基本命令のファームウェア化による Lorel-2 プログラム例の速度向上

Table 6 Speed-up of sample Lorel-2 programs by firmwarelization of basic instructions.

例題プログラム	実行時間 (秒)		比率	ROS 利用時推定比率
	ソフトウェア	ファームウェア		
Eight Queens	42.8	16.5	2.6	6.7
Bubble Sort	55.4	35.9	1.5	2.0

令が38%，また Bubble Sort では copy 命令が 34% を占めると推定され，これら命令の速度向上が大きな効果を持ったことがわかる．ほかの多くの Lorel-2 プログラムにおいても，ファームウェア化によりこの程度の効率改善がはかられたと見てよいだろう．

5.5 ファームウェア化の評価

前節で測定結果を示した処理速度の向上のほか，ファームウェア化により次の点においても改善された．

(i) 頻繁に呼び出される実行時ルーチンの減少により仮想記憶内での実行システムの占有領域が減少するとともに記憶参照の局所性が高まった．

(ii) セル構造処理機能の呼出しが手続き呼出しから拡張機械命令に置き換えられたことにより，Lorel-2 コンパイラの出力コードが単純化され短縮された．

本システムの主要機能であるセル構造処理部分のファームウェア化により上に述べたように速度的および機能的な性能向上が実現され，これにより本システムの適用問題の範囲が時間量的および手続きの大きさにおいて拡大された．

6. まとめ

論理関係処理用言語 Lorel-2 の処理系の中心に位置する実行システムの構造と特徴の概略を述べ，その記憶構造や hash 属性の実現，記憶管理方式を明らかにした．さらに，そのセル構造処理機能をファームウェア化したことによる速度の改善について測定結果を示し評価を行った．

記憶管理方式については，仮想記憶上で複数セグメントに分割されたセルプールにおいて局所性を保つような管理を行うとともに，セル再生タスクを並列に動作させることにより効率化をはかった．なお，並列ガーベジ・コレクタによる記憶管理は LISP 処理系について実現されたものがいくつかあるが[4),5)]，我々のシステムでは Lorel-2 の言語仕様からごみセル構造を主タスク側で明白に捨てられるため，アルゴリズムを著しく単純化することができた．

セル構造処理機能のファームウェア化による高速化でも，マイクロプログラム・サイクルが比較的遅いなどハードウェア的に不利な条件のほか仮想記憶上に分散したセルプール等による論理的複雑さを考慮すると，ほかのシステムとの比較においても評価[6),7)]され，記憶面等での効果と合わせてシステムの中核部分の性能の向上がはかられた．

現在までにいくつかの研究の中で Lorel-2 を利用してきており，Lorel-2 の記述性と本システムの動作の良好さを確認している．問題点としては，HPL コンパイラの動作の重さから，コンパイル・ユニットを得るまでの処理時間が比較的大きいという多くのプリコンパイラ方式に共通した欠点が挙げられる．現在，デバッグ支援のための機能について改良を検討中である．また，記憶管理について仮想記憶セグメントが主記憶内に存在しているか否かを考慮することによるアルゴリズムの改善も考えられる．

最後に，本研究について大きな助言と協力をいただいたすでに卒業された方々を含む榎本・片山研究室の諸氏に感謝します．

参考文献

1) 榎本 進，宮地，片山，榎本：Lorel-2 言語について，情報処理，Vol. 19, No. 6, pp. 522-530 (1978).
2) 宮地，榎本 進，片山，榎本：Lorel-2 実行システムの構成，信学会計算機研資 EC 77-4 (1977).
3) 宮地，片山，榎本：Lorel-2 とその実行システムの評価，信学会言語とオートマトン研資 AL 79-39 (1979).
4) 日比野靖：並列ガーベッジコレクタを組込んだミニ LISP，情報処理学会第 19 回全国大会論文集 (1978).
5) 薄 隆，田丸喜一郎，所真理雄：マルチプロセッサによる LISP マシンの試作，情報処理学会第 19 回全国大会論文集 (1978).
6) 宮脇富士夫，木下耕二，渡辺勝正，萩原 宏：APL インタプリタのファームウェア化とその効果について，情報処理学会論文誌，Vol. 20, No. 2, pp. 172-178 (1979).
7) 安部憲広，東出正裕，辻 三郎：マイクロプログラムによりミニコンリスプ FLISP の改善，信学会論文誌，J 62-D, 5 pp. 356-357 (1979).

(昭和 54 年 11 月 7 日受付)
(昭和 55 年 11 月 20 日採録)