オペレーティング・システムのファームウェア化対象選定法†

長岡満夫†† 中村敏夫†† 森道直††

ソフトウェア機能のファームウェア化によりシステム性能の向上を図る場合，制御記憶が高価格であるなどの理由から，最も少ない制御記憶量で最大の性能向上が達成できるソフトウェア機能の選定法が主要な課題となる．しかし，従来の選定法としては，実行頻度の高い機能をファームウェア化するという定性的な経験則しかなく，選定したソフトウェア機能・性能向上度・制御記憶量との関係については定量化されていない．本論文では，ファームウェア化による性能向上度および所要制御記憶量に関して，ソフトウェア特性をパラメータとする予測式ならびに予測式を利用した選定法を，提案している．さらに，本手法がハードウェア・アーキテクチャの異なる場合にも，有効であることを述べている．

1. はじめに

マイクロプログラム制御可能な計算機において，計算機システムの性能向上の一手段としてソフトウェアのファームウェア化がある．とくにオペレーティングシステム (OS) は計算機システムの処理能力に大きく影響するため，OS の一部のソフトウェア機能をファームウェア化している場合が多い[3)~5)]．

しかし，そのようなソフトウェアのファームウェア化に当たっては，マイクロプログラム格納用の制御記憶が高価格であるなどの理由から，ファームウェア化に最適な（すなわち最も少ない制御記憶量で最大の性能向上が得られる）ソフトウェア機能を選択することが必要となる．

そこで，本論文では OS のファームウェア化手順を4フェーズに分割し，ファームウェア化実験によりソフトウェア/ファームウェアの静的/動的特性の分析を通じて，ハードウェア・アーキテクチャに依存することなくファームウェア化対象を選定する汎用的な手法を提案している．

ソフトウェアのファームウェア化において本手法を適用すれば，ハードウェア・アーキテクチャの異なる計算機上でもマクロにファームウェア化効果を推定することが可能となる．

† A Selecting Method of Operating System Functions to be Microprogrammed by MITSUO NAGAOKA, TOSHIO NAKAMURA and MICHINAO MORI (Information Management Systems Section, Yokosuka Electrical Communication Laboratory, N. T. T.).

†† 日本電信電話公社横須賀電気通信研究所データ蓄積方式研究室

2. OS のファームウェア化と問題点

2.1 問題点

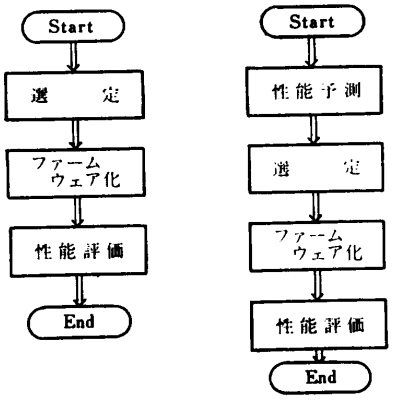
OS のファームウェア化の従来の開発手順，および筆者らが今回採用した開発手順を**図1**に示す．

従来の手順は図1(a)であり，この場合には，ソフトウェア機能の実行頻度，集中度相当の選定基準によりファームウェア化対象を選定し，ファームウェア化を行っている例が多い[3),4),6)]．これらは定性的な経験則でしかなく，ファームウェア化対象のソフトウェア機能とファームウェア化による性能向上度ならびに必要となる制御記憶量との関係については定量化されていなかった．このため，ファームウェア化を行った後，目的とする性能向上度が十分達成されない場合，あるいは制御記憶量の制限内に収まらない場合には，再度ファームウェア化手順を繰り返すことになるという問題点があった．

そこで簡易な手法で性能予測を行い，予測式に基づいたファームウェア化対象の選定法が重要となってくる[2)]．本論文で提示する4フェーズ（図1(b)参照）によるファームウェア開発手順の利点がそこにある．

マイクロプログラム制御方式の計算機ではハードウェア・アーキテクチャの違いにより，ファームウェア化効果が異なる．したがって選定法がハードウェア・アーキテクチャに依存することなく，汎用的な手法として確立するためにはハードウェア・アーキテクチャの特徴を考慮する必要がある．

OS のファームウェア化においてハードウェア・アーキテクチャを考慮した予測法，選定法は現状では見当たらない．



（a）従来の開発手順　　（b）本論文での開発手順

図1 ファームウェア開発手順
Fig. 1 Microprogram development flow.

2.2 ファームウェア化効果算出式の定義

OSのファームウェア化効果として，ソフトウェア機能 r_i をファームウェア化した場合，一つの処理（トランザクション）における計算機システムでの処理時間削減率 (P_i)，および各 r_i ごとの処理時間削減率 (μ_i)，必要とする制御記憶量 (C_i) とする*.

一般にソフトウェア機能 r_i のファームウェア化による性能向上はソフトウェアの動特性集合 ($S_{DSW}{}^i$) とハードウェア（あるいはファームウェア）の動特性集合 ($S_{DHW}{}^i$) に依存するため，以下のように定義する.

(定義1)　$P_i=f_P(S_{DSW}{}^i, S_{DHW}{}^i)$

またファームウェア化に必要となる C_i の増大は，ソフトウェアの静特性集合 ($S_{SSW}{}^i$) とファームウェアの静特性集合 ($S_{SFW}{}^i$) に依存するため，以下のように定義する.

(定義2)　$C_i=f_C(S_{SSW}{}^i, S_{SFW}{}^i)$

ここで，$S_P{}^i$ はファームウェア化による性能向上に大きく影響するソフトウェア動特性，$S_{tr}{}^i$ はトランザクション固有なソフトウェア特性とすると，ファームウェアルーチン r_i の1回走行当りの性能向上度は $f_P{}^1(S_P{}^i)$ で表される．よって P_i は $f_P{}^1(S_P{}^i)$ と $f_P{}^2(S_{tr}{}^i, S_{DHW}{}^i)$ の積で表現される.

$$P_i=f_P(S_{DSW}{}^i, S_{DHW}{}^i)$$
$$=f_P{}^1(S_P{}^i)\cdot f_P{}^2(S_{tr}{}^i, S_{DHW}{}^i)$$

したがって，μ_i は以下のように定義される.

(定義3)　$\mu_i=f_P{}^1(S_P{}^i)$

2.3 選定基準

OSのファームウェア化対象の選定命題が「最も少ないコストで最大の性能向上を得る」であることを考慮して，選定基準を P_i/C_i とすると，定義1, 2より

$$P_i/C_i$$
$$=f_P{}^1(S_P{}^i)\cdot f_P{}^2(S_{tr}{}^i, S_{DHW}{}^i)/f_C(S_{SSW}{}^i, S_{SFW}{}^i)$$

となる.

具体的に本論文でわれわれが採用した選定基準を以下に述べる.

$S_{tr}{}^i$ の要素としては具体的に，トランザクション走行での全動的ステップ数，r_i の1回走行当りの平均動的ステップ数（以後 DS_i と略す），トランザクション走行での r_i の実行回数（以後 m_i と略す），r_i の1回走行当りの動的分岐命令数および動的メモリアクセス回数等がある.

また $S_{DHW}{}^i$ の要素としては，システム・トータルな平均命令実行時間，r_i の平均命令実行時間等である．さらに $S_{SSW}{}^i$ の要素としては，r_i の静的ステップ数（以後 SS_i と略す），r_i の静的分岐命令数および静的メモリアクセス回数等である.

なお，ここで $S_{SFW}{}^i$ の要素は C_i の静的特性であり，$S_{SFW}{}^i$ の各要素の値はファームウェアの仕様に基づくため，一つのファームウェア仕様に閉じた場合の各ソフトウェア機能間で相違はなく一定である.

以上の事項を考慮して得られる選定基準として，以下のリピータビリティを定義する[1].

(定義4)　$R_i=m_i\cdot DS_i/SS_i$

R_i をソフトウェア機能 r_i のリピータビリティと呼ぶ.

3. 実　験

3.1 実験システム

実験システムのハードウェア，ソフトウェアについて以下に述べる.

3.1.1 ハードウェア

実験システムのハードウェアは中規模計算機であり，

（i）マイクロ命令形式

56ビット（うち2ビットはパリティ用）の水平型マイクロ命令

（ii）先行制御方式あり

（iii）ローカルメモリ方式なし

という特徴を有する計算機である.

また，計算機内には，マイクロプログラムの実行状態の情報収集を行うため，1個のマイクロプロセッサが組み込まれている[7]．マイクロプロセッサの起動・終了は測定対象計算機のマイクロ命令により制御さ

* P_i, μ_i を単に性能，C_i を単にコストと呼ぶことがある.

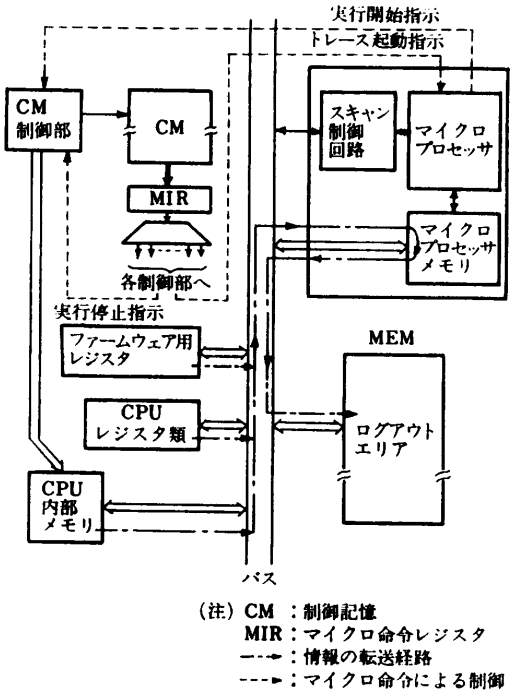


図 2 測定システムの環境
Fig. 2 Block diagram of measurement system.

れ，マイクロ命令の実行アドレス等の情報を測定対象計算機の主記憶へ転送する．さらに主記憶上に転送された情報を磁気テープに転送するツールを用意している．

測定システムの環境を**図 2** に示す．

3.1.2 ソフトウェア

リアルタイムシステム用のオペレーティング・システムであり，サービスとしてはデータベース処理を行っている．

3.2 実験における選定基準

選定基準は 2.3 節で述べたリピータビリティとし，この値の大きいソフトウェア機能から順にファームウェア化することとした．

3.3 ファームウェア化

本実験システムでは，ソフトウェアからファームウェアルーチンへ制御を移す命令は firmware call 命令(以後 FWC 命令と略す) であり，FWC 命令中のサブファンクションコード (SFC) により対応するファームウェアへ分岐する構成となっている．

4. 実験結果の解析

本実験におけるファームウェア化対象機能は，OS の中核部機能であり，

(1) 入出力制御 6 個

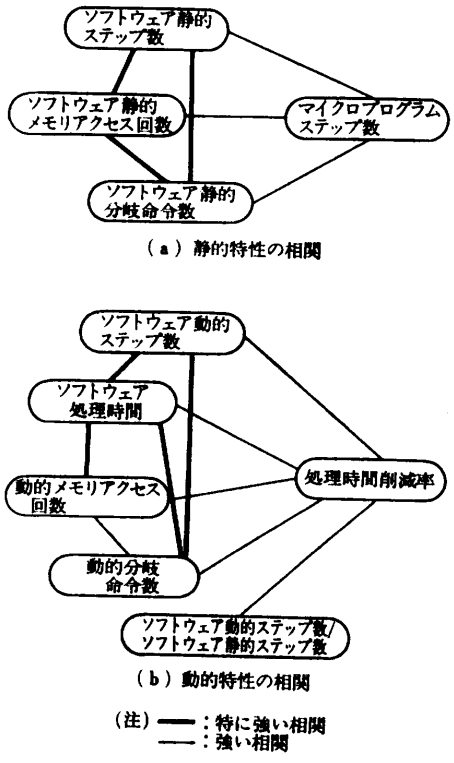


図 3 特性の相関
Fig. 3 Correlations of characteristics. (a) static characteristics, (b) dynamic characteristics.

(2) タスク管理 7 個
(3) 資源管理 5 個
(4) プログラム管理 5 個

の計 23 個であり，これらの機能のファームウェア化効果ならびにソフトウェア特性，ファームウェア特性についてデータ収集をし，以降に示す解析を行った．

4.1 ソフトウェア特性とファームウェア化効果との相関

図 3 にソフトウェア特性，ファームウェア化効果の相関関係を示す．図 3 では，相関の程度を，相関係数を β とすれば，

$0.8 < |\beta| \leq 1$ を “とくに強い相関”
$0.6 < |\beta| \leq 0.8$ を “強い相関”

として示す．

図 3 から以下のことがわかった．

(1) 処理時間削減率 (μ_i) について

μ_i と相関が強いソフトウェア特性としては，DS_i，動的メモリアクセス回数，動的分岐命令数，DS_i/SS_i である．

さらに，**図 4** は上記のソフトウェア特性とマイクロプログラムの特徴との関係を示し，処理時間削減率の

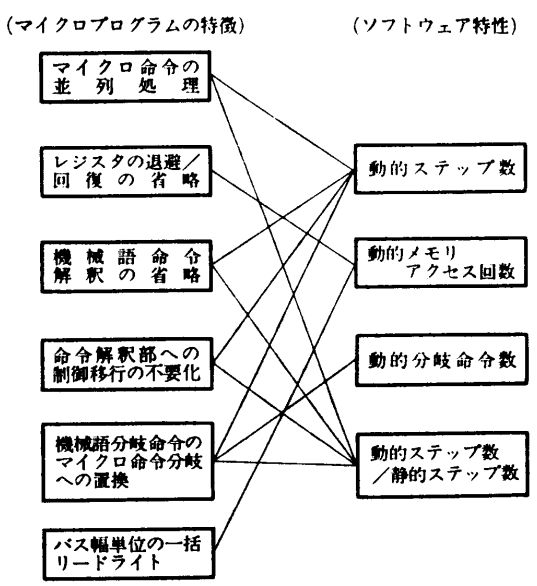


図 4 マイクロプログラムの特徴とソフトウェア特性の関係（性能向上に関して）
Fig. 4 Relation between the feature of microprograms and software characteristics (about performance improvement).

要因を表したものであり，以下のことが言える．

（ⅰ） DS_i, DS_i/SS_i が大きいほど，マイクロ命令の並列処理，機械語命令解釈の省略（とくにオペランドアクセスの削減），機械語の分岐命令のマイクロ命令分岐への置換，命令解釈部への制御移行の不要化に基づくファームウェア化効果が大きくなる．

（ⅱ） 動的メモリアクセス回数が大きいほど，レジスタの退避／回復の省略，バス幅単位のデータの一括リード／ライト処理に基づくファームウェア化効果が大きくなる．

（ⅲ） 動的分岐命令数が多いほど，機械語での分岐命令のマイクロ命令分岐への置換に基づくファームウェア化効果が大きくなる．

（2） 制御記憶量 (C_i) について

C_i (マイクロプログラムステップ数) と相関が強いソフトウェア特性としては，静的メモリアクセス回数，静的分岐命令数，SS_i である．

図 5 は上記のソフトウェア特性とマイクロプログラムの特徴との関係を示し，必要とする制御記憶容量の要因を表したものであり，以下のことが言える．

（ⅰ） ファームウェアの処理構造はソフトウェアの処理構造を反映したものとなり，ほぼ相似の構造となっている．

（ⅱ） メモリアクセス時に数マイクロ命令を要することにより，メモリアクセスに関連するマイクロ命令数がマイクロプログラムに占める割合が大きい．

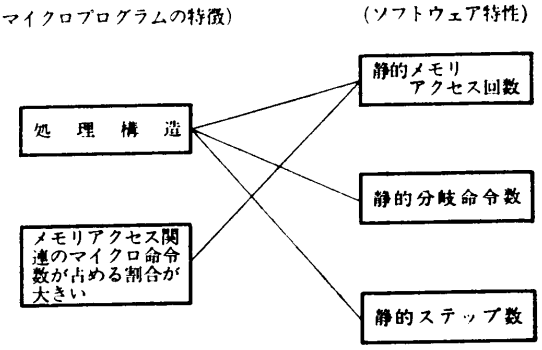


図 5 マイクロプログラムの特徴とソフトウェア特性の関係（制御記憶量に関して）
Fig. 5 Relation between the feature of microprograms and software characteristics (about control memory capacity).

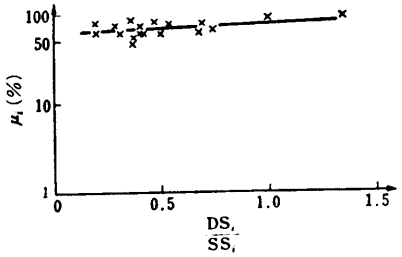


図 6 DS_i/SS_i と μ_i の関係
Fig. 6 Relation between DS_i/SS_i and $\log \mu_i$.

4.2 ファームウェア化効果の予測式の導出

4.1 節で述べた要因分析を通してファームウェア化効果の予測式を導出する．

ここで予測の対象は，2.2 節で述べたファームウェア化効果のうち，処理時間削減率 (μ_i) と静的マイクロプログラム・ステップ数 (C_i) である．

（1） μ_i についての予測

4.1 節で述べたように μ_i に大きく影響するソフトウェア特性はいくつかあるが，回帰直線のあてはまりの良さ，および予測式導出の容易さから最も良効なパラメータとしては DS_i/SS_i が得られ（**図 6** 参照），その結果 μ_i に関する予測式は，

$$\hat{\mu}_i = A \cdot 10^{\alpha \cdot (DS_i/SS_i)}$$

ただし，$\hat{\mu}_i$: μ_i の予測値

となる．

（2） C_i についての予測

4.1 節で述べたように C_i に大きく影響するソフトウェア特性はいくつかあるが，それらの特性（静的メモリアクセス回数，静的分岐命令数，SS_i）は互いに

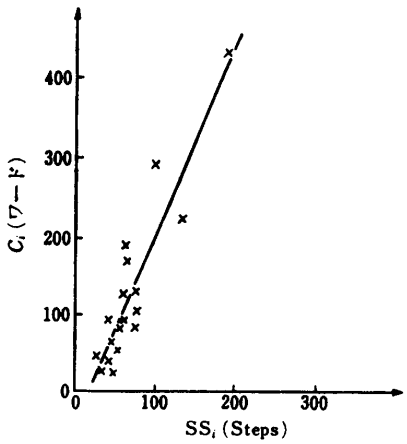


図 7 SS_i と C_i の関係
Fig. 7 Relation between SS_i and C_i.

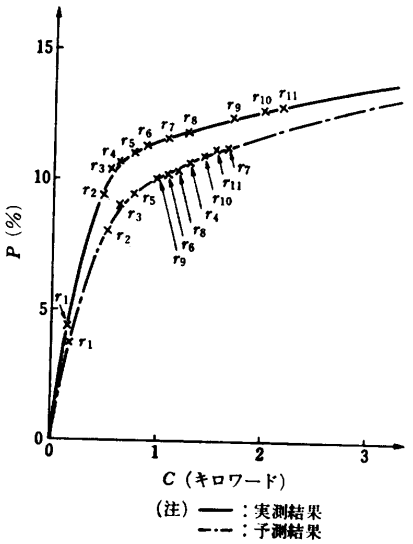


図 8 CとPの関係 (その1)
Fig. 8 Relation between C and P (No. 1).

相関が強いこと，および過去のファームウェア化の経験，予測式導出の容易さ等を考慮して予測パラメータを SS_i で代表させることができる（**図7**参照）．その結果 C_i に関する予測式は，

$$\hat{C}_i=\gamma\cdot SS_i+\varepsilon$$

ただし，$\hat{C}_i$: C_i の予測値

となる．

4.3 ファームウェア化効果予測式の検証

P と C* の関係は**図8**に見られるような曲線（以降，P/C 曲線と呼ぶ）となり，制御記憶が高価格であることから，ファームウェア化に必要とされる制御記憶量は自ずと決ってくる．

したがって，本予測式を用いて $\hat{P}_i$ を求め，$\hat{P}_i/\hat{C}_i$ 比の大きい順にプロットし，その P/C 曲線を利用して性能，コストを満足するようなソフトウェア機能選定を行えばよい．

図8に予測式を用いた P/C 曲線と，実測結果からの P/C 曲線を示す．この図より性能／コスト比に大きく影響するソフトウェア機能（たとえば r_1, r_2, r_3）では予測式による選定結果と評価結果とは同一である（なお，ここで r_1, r_2 は資源管理のうちブロックの開放，確保を，また，r_3 はプログラム管理のうち SVC（スーパーバイザコール）コールスタックの作成を行うものである）．

また，適当な容量の制御記憶量が与えられた場合を考えると，予測式を利用した選定結果と実測結果はほぼ同様であり，予測式によりマクロにファームウェア化対象機能を選定することが可能となった．さらに，予測式ではリピータビリティ等では明確とならない性能向上度，必要とする制御記憶量を定量的に把握できる．

4.4 ハードウェア・アーキテクチャと予測式との関係

本予測式の汎用性を考慮した場合，異なるハードウェア・アーキテクチャ上での本予測式の適用性を検証する必要がある．すなわち，ハードウェア・アーキテクチャの違いと本予測式との関係を明確にする必要がある．

ハードウェア・アーキテクチャとファームウェア化効果との関係については，ハードウェア・アーキテクチャの相違によりファームウェア化効果が異なるという実験結果がいくつか報告されている[8]．たとえば，ローカルメモリ (LM) 方式を採用している計算機では，LM にデータがない場合はブロック転送が生じるが，通常 LM にデータがない確率 (NFP) は小さいため，ファームウェア化によるメモリアクセス回数の削減効果は小さくなり，また命令の読出しおよびオペランドの読出し（命令解釈と称する），命令の実行という一連の動作がパイプライン化される計算機では，ファームウェア化による命令解釈の省略および命令解釈部への制御移行の不要化に基づく効果は小さくなる．

ここではマイクロ命令形式の違い，LM の有無に着目し，水平化レベル**の低いもの（垂直型）から高い

* $P=\Sigma P_i, C=\Sigma C_i$ である．

** マイクロ命令の並列性を表し具体的にはマイクロ命令のフィールド数を表す[9]．

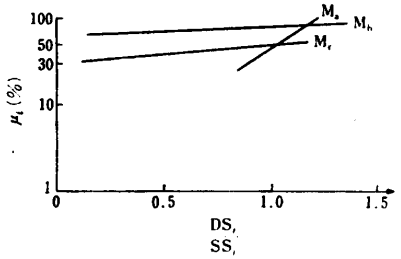


図 9 DS_i/SS_i と μ_i の関係
Fig. 9 Relation between DS_i/SS_i and $\log\mu_i$.

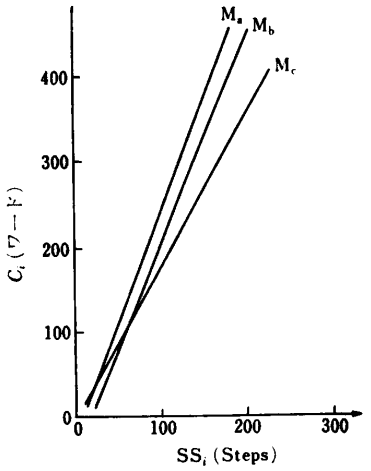


図 10 SS_i と C_i の関係
Fig. 10 Relation between SS_i and C_i.

表 1 ハードウェア・アーキテクチャと予測式の係数
Table 1 Hardware architecture and coefficient of predict expression.

係数 / マシン	A	α	γ	ε	備　考
M_a	0.928	1.692	2.58	−15.86	水平化レベル=2
M_b	62.43	0.104	2.47	−45.76	水平化レベル=8
M_c	31.98	0.136	1.91	−9.42	水平化レベル=9

もの（水平型）という特徴を有する計算機として Ma（水平化レベル=2），Mb（水平化レベル=8，Mc*（水平化レベル=9 かつ LM を有する）の 3 機種を評価対象とし，予測式の係数値とハードウェア・アーキテクチャの特徴を比較する．

なお，ここで，計算機 Mb は本実験システムのことであり，計算機 Ma, Mc については，参考文献 8) に示された実験データの分析結果をもとにしている．

（1） 係数 α に関する分析と考察

計算機 Ma, Mb, Mc の係数をそれぞれ $\alpha_a, \alpha_b, \alpha_c$ と表し，以後他の係数についても同様とする（**図 9, 10** 参照）．

表 1 より $\alpha_a > \alpha_c > \alpha_b$ となっている．この理由としては，以下のことが挙げられる．

① 垂直型マイクロ命令をもつ計算機 Ma では先行制御がなされていないため，ファームウェア化効果の要因として機械語命令解釈の省略による効果が大きいが，一方水平型マイクロ命令をもつ計算機 Mb, Mc では先行制御がなされているために機械語命令解釈の省略による効果は小さくなる．

よって α_a は α_b, α_c よりも大きな値となっている．また，Mb, Mc はほぼ同様のフィールド構成となっているため，α_b, α_c はほぼ同様の値となっている．

② α は DS_i/SS_i に係る係数であり，SS_i がほぼ一定であるような範囲を想定すれば，DS_i が多くなるほど命令解釈の省略による処理時間削減が大きくなる．

（2） 係数 γ に関する分析と考察

表 1 より $\gamma_a > \gamma_b > \gamma_c$ となっている．この理由としては，水平化レベルすなわちマイクロ命令フィールド数が増加するほど，マイクロ操作の並列動作が可能となり，同一処理を少ないステップ数で記述できるからである．

また，Ma, Mc の係数 A について言えば，$A_b > A_c$ となっている．である．この理由は，①Mc が LM 方式であり，メモリアクセス回数削減による効果が少なくなること，②Mb は Mc に比して先行制御のレベルが低いために，命令解釈におけるオペランド読出しが実行時間として現れており，ファームウェア化により，このオペランド（メモリ）アクセス回数削減の効果があることによっている．

5. 結　論

5.1 ファームウェア化効果予測式の適用

ファームウェア化効果予測式を適用する場合，予測式導出においてサンプル・コーディングを行う対象範囲，サンプル数を極力少なくして（マイクロプログラムコーディングの工数を少なくして）高い確度で性能およびコストを予測する必要がある．

サンプルの抽出基準に関しては，4.1 節で述べた要因を用いればよいと考えられ，ここでは一例として

① 静的メモリアクセス回数の大小
② 静的ステップ数の大小
③ 動的ステップ数の大小

を用いた．その結果を表 2 に示し，サンプル数，予測式も合わせて示す．

* Mc のマイクロ命令形式は Mb の場合のフィールド構成とほぼ同様となっている．

表 2 予測式導出のためのサンプルと予測の誤差
Table 2 Sample for predict expression, and prediction error.

サンプル抽出基準	標準誤差			95% の信頼区間を与える誤差			予測式
	p_i	$\log \mu_i$	C_{FWi}	p_i	$\log \mu_i$	C_{FWi}	
静的メモリアクセス回数の大小	0.216	0.077	118.413	0.423	0.151	232.090	$\hat{C}_{FWi}=1.826\cdot SS_i-13.06$ $\hat{\mu}_i=67.99\cdot 10^{0.051\cdot\left(\frac{DS_i}{SS_i}\right)}$
静的ステップ数の大小	0.257	0.081	80.995	0.503	0.158	158.750	$\hat{C}_{FWi}=1.012\cdot SS_i+7.99$ $\hat{\mu}_i=65.74\cdot 10^{0.011\cdot\left(\frac{DS_i}{SS_i}\right)}$
動的ステップ数の大小	0.298	0.072	88.702	0.584	0.142	173.856	$\hat{C}_{FWi}=0.705\cdot SS_i+71.05$ $\hat{\mu}_i=64.36\cdot 10^{0.093\cdot\left(\frac{DS_i}{SS_i}\right)}$

(注) 本データはサンプル数を5とした例を示す.

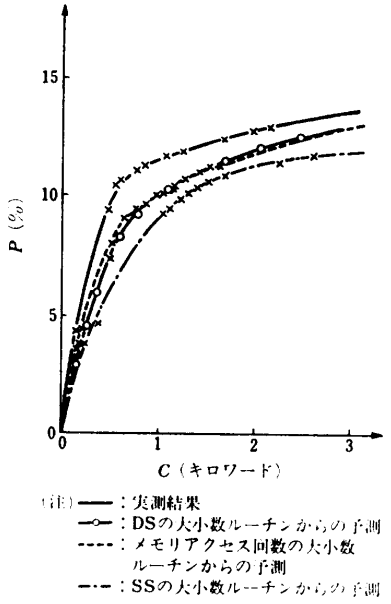


図 11 CとPの関係 (その2)
Fig. 11 Relation between C and P (No. 2).

表2により $p_i(\mu_i)$ についての標準誤差，および95%の信頼区間の誤差について見れば静的ステップ数をサンプル抽出基準とする場合が最もよい結果を得，サンプル数としては4～6ルーチン程度でよいと考えられる．**図11**に各サンプルから導出した予測式を適用した例を示す．

5.2 性能予測・選定フェーズの手順

ファームウェアを導入する計算機システムのソフトウェアトレースデータを前提として，4章で述べた実験結果の解析より性能予測フェーズ，選定フェーズを詳細化する．

(1) 性能予測フェーズ

性能予測フェーズの手順は以下のとおりである．

(i) トレースデータをソフトウェア特性解析ツールにより分析し，ソフトウェア機能単位の m_i, DS_i, SS_i を求める．

(ii) ファームウェア化対象として考えるソフトウェア機能の集合を $\boldsymbol{F}_{sw}$ とする．

(iii) $\boldsymbol{F}_{sw}$ のなかの4～6個のソフトウェア機能を SS_i の大小関係よりランダム抽出し，ファームウェア化のサンプル・コーディングを行う．

(iv) (iii)のサンプル・コーディングよりファームウェア化効果予測式の係数 $A, \alpha, \gamma, \varepsilon$ を求める．

(2) 選定フェーズ

選定フェーズの手順は以下のとおりである．

(i) $\boldsymbol{F}_{sw}$ の各ソフトウェア機能に対して，予測式を適用し $\hat{P}_i/\hat{C}_i$ を求める．

(ii) $\boldsymbol{F}_{sw}$ の各ソフトウェア機能に対して $\hat{P}_i/\hat{C}_i$ を求め，P_i と C_i の二次元空間に $\hat{P}_i/\hat{C}_i$ の大きい順にプロットし，P/C 曲線を作成する．

(iii) P/C 曲線を利用し，以下の条件を満足するソフトウェア機能集合 $\boldsymbol{F}'_{sw}$ を求める．

(a) $\sum_{i=1}^{n} p_i \geq P_{max}$

(b) $\sum_{i=1}^{n} c_i \leq C_{min}$

ここで

P_{max}：ファームウェア化に要求される最大性能

C_{min}：ファームウェア化に要求される最小コスト

n：条件(a)(b)を満たす最小ソフトウェア機能数

6. むすび

本論文ではソフトウェアのファームウェア化を行う場合に，ソフトウェア特性から性能向上度およびフ

ファームウェア化に要する制御記憶量を予測し，ファームウェア化対象機能を選定する方法について示した．

このことは従来選定基準として実行頻度の高い部分をファームウェア化すればよいという経験則[3),4),6)]を実験的に確かめたことになる．

また，本論文で提案しているファームウェア化効果の予測式ではリピータビリティ等では明確にならない性能向上度および制御記憶量が定量的に把握できるのみでなく，ハードウェア・アーキテクチャの相違をも評価できるという利点がある．

今後ファームウェア適用範囲の拡大に伴い，ファームウェア開発の負荷軽減のため開発手順の自動化が望まれるが，本手法はその第一段階として意味があるものと考える．

なお，本論文で述べたハードウェア，ファームウェア化対象ソフトウェアとは異なる環境での本手法の検証は今後の課題である．

謝辞　最後に本研究の遂行にあたりご指導を賜った横須賀電気通信研究所高平データ応用研究室長ならびに村上データ蓄積方式研究室長，および他計算機上のファームウェア化実験データを提供していただいた木村正男，進藤重平，稲垣充広各氏に深謝します．本研究は DIPS プロジェクトの一環として実施したものであり，共同研究にあたられた株式会社日立製作所の関係各位に感謝します．

参考文献

1) 長岡，中村：ソフトウェアのファームウェア化効果の分析と予測に関する考察，情報処理学会第22回全国大会，4J-1 (1981)．
2) 坂村他：ファームウェア化による計算機性能改善度の予測法，電子通信学会論文誌 (D)，J 61-D，9，pp. 711-718 (1978)．
3) Werkheiser, A. H.: Microprogrammed Operating Systems, Proc. 3 rd Annual Workshop on Microprogramming (1970).
4) Sockut, C. H.: Firmware/Hardware Support for Operating Systems: Principle and Selected History, *ACM SIG MICRO Newsletter*, Vol. 16, pp. 17-26 (1975).
5) Brown, G. E. and Eckhouse, R. J., Jr.: Operating System Enhancement Through Firmware, *ACM Newsletter*, Vol. 8, No. 3, pp. 119-128 (1977).
6) Rauscher, T. G. and Agrawala, A. K.: Dynamic Problem-Oriented Redefinition of Computer Architecture via Microprogramming, *IEEE Trans. on Computers*, Vol. C-27, No. 11, pp. 1004-1014 (1978).
7) 外川，星子他：ファームウェアのデバグ効率化に関する一手法，昭和53年度通信学会情報・システム部門全国大会，p. 378 (1978)．
8) 木村，進藤，稲垣：マイクロプログラムの形式とファームウェアの効果について，昭和53年度電子通信学会総合全国大会．p. 1311 (1978)．

(昭和56年8月3日受付)
(昭和56年12月17日採録)