講　　演

Firmware について*

萩　原　　宏**

1. はじめに

最近 firmware が非常に注目されるようになって来て，わが国でもこの方面の研究が活発になり，本大会(昭和48年度第14回大会)においては，題目に firmware という言葉の出ている講演2件，その他これに関連する研究発表が数件ある．

この firmware という言葉は，米国の雑誌 Datamation の1967年1月号に掲載されている Ascher Opler の論文 "Fourth-Generation Software" の中で，つぎのように提案された．すなわち，

To better understand the nature of microprogramming a no-order-set/no-data-structure computer, I believe it worthwhile to introduce a new word into our vocabulary: *firmware*. I use this term to designate microprograms resident in the computer's control memory, which specializes the logical design for a special purpose, e. g., the emulation of another computer. I project a tremendous expansion of firmware—obviously at the expense of hardware but also at the expense of software.

これから明らかなように firmware の基礎をなすのはマイクロプログラミングであるので，まず，マイクロプログラミングの概念とその方式などに関して簡単に述べ，マイクロプログラミングの発展過程をみながら firmware に関係する事項について述べる．続いて firmware の意義と特長，その応用ならびに将来の問題などについて論じよう．

2. マイクロプログラミングについて

マイクロプログラミング方式は1951年M. V. Wilkes によって計算機の制御装置を設計する手法として提案された．すなわち，Fig. 1 に演算装置の一部を示したが，これでレジスタ R1 と R2 の内容の和をレジスタ R3 に入れる操作を行うには，ゲート d および g を開いてレジスタ R1 と R2 の内容を ADDER に入れ，その出力をゲート k を開いて出力ブスに出し，ゲート c を開いてレジスタ R3 に入れればよい．このようなゲートの制御信号を作り出すのは，マイクロプログラミング方式でない方式，すなわち，結線論理(wired logic)方式では論理回路によって行なっているが，マイクロプログラミング方式では，Fig. 2 のように DECODER によって選ばれた水平の線と，必要な制御信号に対応する MATRIX A の垂直の線と結合させることによって作り出そうとするのである．

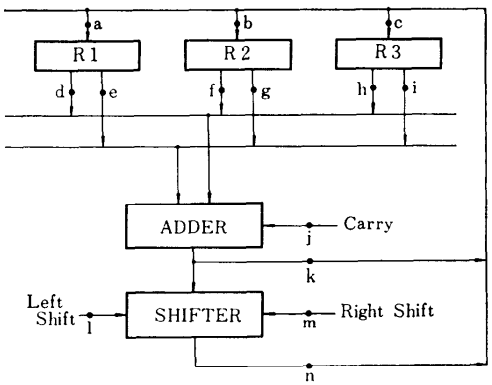

Fig. 1 Arithmetic Unit

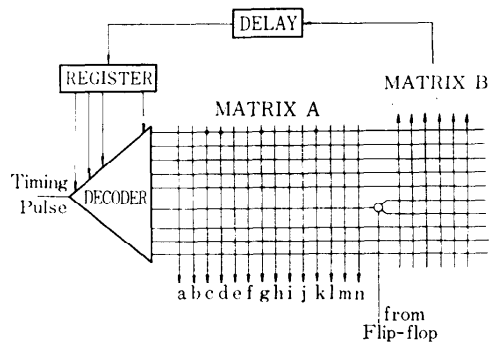

Fig. 2 Microprogram Control Store

* 第14回情報処理学会大会招待講演(昭和48年12月11日)
** 京都大学工学部情報工学教室

時間的に順次このような信号を作り出すためには，図のMATRIX Bに次の制御信号を作り出す水平の線を選択するのに必要な情報を用意しておく．こうすることによって順次所要の水平の線を選択することにより，基本的な操作の組合せとして計算機の命令を実行する制御信号を逐次発生させようとするのである．Wilkes の最初の提案ではこの MATRIX A および B はダイオード・マトリクスで構成し，条件によって分岐する必要があるところでは，図に示したようにFlip-flopからの信号によって MATRIX B で分岐させる方法がとられていた．

この水平の1本の線に含まれている情報を一つの命令とみてマイクロ命令と呼んでいる．このマイクロ命令は必ずしもダイオード・マトリクスに構成する必要はなく，適当な記憶装置（制御記憶と呼ぶ）に格納しておき，これを順次取り出すことにより制御信号を発生させればよい．したがって，Fig. 3 に示すように，

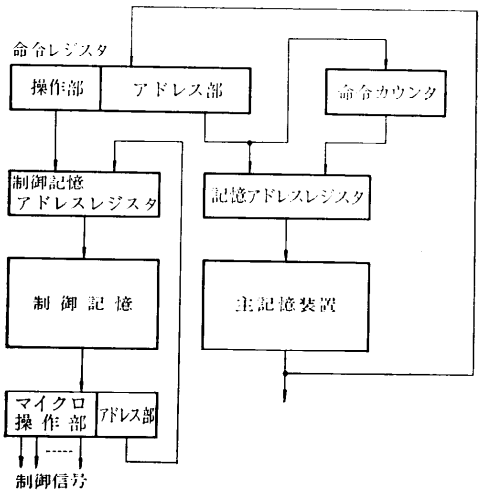

Fig. 3 Microprogram Control Unit

命令レジスタに入れられた機械語命令の操作部を制御記憶アドレス・レジスタ（Fig. 2 の REGISTER に対応する）に転送し，制御記憶の中の所要のマイクロ命令を選択する．以下，マイクロ命令のアドレス部により，次々とマイクロ命令を選択することにより，制御記憶の中に格納されているマイクロ命令の系列，すなわちマイクロプログラムの指定する制御信号の系列を発生させて，命令レジスタに入っている機械語命令の操作を行うのである．

2.1 マイクロ命令の形式

制御記憶に格納されるマイクロ命令の形式には，Fig. 2 に示すようにマイクロ命令の各ビットが制御される点に対応するようなもの(直接制御と呼ばれる)も考えられるが，少し複雑な計算機になると数百ビットの語長が必要になる．そこで，制御信号を互に同時に発生することのないものの組に分けて，各組毎に符号化してマイクロ命令の語長を短縮する方式（符号化制御と呼ぶ）がとられるようになった．

このように方式に対して，更に符号化を進めて，ふつうの機械語の命令に似た形式にする方式がある．これを vertical type（垂直型）と呼んでいる．これに対して前者のものを horizontal type（水平型）と呼ぶ．horizontal type は符号化しても1語のビット数はかなり長いが，同時に制御できる部分が多いのでマイクロ命令のステップ数は少なくなる．これに反し，vertical type では1語のビット数は少ないが，マイクロ命令のステップ数が増えることは避けられない．

2.2 制御記憶

マイクロプログラムを格納する制御記憶としては，高速で読み出すことのできるものが望まれる．そのため，高速の IC メモリが実用になるまでは，主として読み出し専用の固定記憶 (ROM) が用いられた．しかし，高速の IC メモリが実用化されるようになって，書き換え可能な記憶（writable control store, WCS）が用いられるようになり，制御記憶の内容を簡単に書き換えることができるようになるとともに，firmware が現実のものとして注目されるようになったのである．

2.3 2レベル・マイクロプログラミング方式

最近，制御記憶を2段階に分けたマイクロプログラム方式の計算機が現われてきた．すなわち，Fig. 4

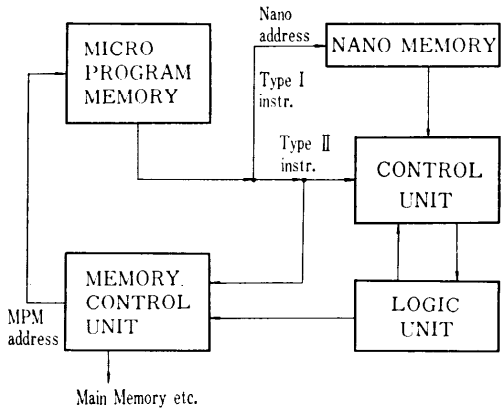

Fig. 4 2 level microprogramming system

に示すように，制御記憶を microprogram memory (MPM) と nano memory とに分け，MPM には vertical type のマイクロプログラムを入れ，nano memory には horizontal type のマイクロ命令を入れておく．第1段の MPM 内の vertical type のマイクロ命令はそのまま制御装置に送られるもの(type II)もあるが，多くは nano memory のアドレスを指定する(type I)ようになっており，nano memory の中の horizontal type のマイクロ命令を指定する．これによって解読器なしで vertical type のマイクロ命令に対する制御信号を生成することができる．第1段の MPM はマイクロプログラムのステップ数だけの語数の容量が必要であるが，vertical type であるので1語のビット数は短かくてすみ，nano memory は horizontal type のマイクロ命令であるので1語のビット数は多いが，マイクロ命令の種類だけの語数があればよい．また，第1段，第2段の制御記憶の読み出しを overlap させることにより，2重構造にしたことによる速度の低下はほとんどない．また，第1段目の vertical type のマイクロ命令に対して第2段目の horizontal type のマイクロ命令の何ステップか（これを nano program という）を対応させることもできる．この方式はマイクロプログラミングの方式として非常に興味あるものである．

3. マイクロプログラミングの発展[1)]

1951年 Wilkes によってマイクロプログラミング方式が提案されてから今日までの発展過程は Table 1 に示すように3期に分けて考えることができる．すなわち，第1期は1951年から1964年までの期間でマイクロプログラミングに関する種々のアイデアが出され，数々の計算機が試作された時代である．第2期は IBM System/360 が発表されて，マイクロプログラミング方式が商用の計算機に広く使われるようになった時期であり，第3期は書き換え可能な制御記憶が実用になり，firmware が活用されるようになった時期と言うことができる．以下，Table 1 の年代順に firmware に関連した事柄について述べよう．

M. T. Wilkes は最初の論文でマイクロプログラムを変更することについてつぎのように論じている．“固定の制御記憶が書き換え可能なものにおきかえられ，必要なときに主記憶から情報が転送されるものとすると，固定した命令コードを全くもたない機械が得られる．プログラマは彼自身の要求に合った命令コードを選択することができるであろうし，また，もし必要ならばプログラムの途中でそれを変更することも可能になるであろう．このような機械は多くの魅惑的な可能性を有している．” すなわち，ダイナミック・マイクロプログラミングの着想が出ている．しかし，その当時の状況ではマイクロプログラムを変更することによる利点よりもむしろ融通性によって生ずる混乱の方が心配された．その後，固定記憶を用いたマイクロプログラミング方式の計算機が種々試作されたが，1961年に至り，stored logic の概念が提唱され，固定した命令コードのない記憶装置に貯えられたマイクロ命令の組合せによって制御する方式の計算機 AN/UYK-1 が発表された．同年，筆者らはマイクロプログラムを取り換えられるような計算機 KT パイロットを設計試作

Table. 1 Historical review of microprogramming

第1期 (1951～64)		
1951	M. V. Wilkes	マイクロプログラミング方式の提案
56	W. Renwick[2)]	EDSAC II
58	G. P. Dinneen	CG 24
59	van der Poel	ZEBRA
60	T. W. Kampe	SD-2
61	H. M. Semarne 他	Stored Logic の概念と AN/UYK-1
〃	萩原他	KT パイロット
〃	H. Isaksson 他	GIER
62	A. Grasselli	pathfinder memory の提案
63	G. B. Gerace	CEP
〃	M. W. Allen 他	CIRRUS
64	W. C. McGee	TRW-133
〃	E. O. Boutwell	PB-440
〃	L. Beck	C-8401
第2期 (1964～69)		
1964	IBM	System/360
65	B. R. S. Buckingham 他	CAS
〃	S. G. Tucker	Emulation
67	A. Opler	Firmware の提案
〃	H. Weber	microprogrammed implementation of EULER
〃	G. A. Rose[3)]	Intergraphic
68	松下[4)]	TOSBAC DN-231
第3期 (1970～)		
1970	N. Bartow 他[5)]	Microdiagnostics
〃	R. W. Cook, M. J. Flynn.	dynamic microprocessor
〃	C. V. Ramamoorthy 他	user-microprogramable computer
〃	S. S. Husson	“Microprogramming: Principles and Practices” の出版
71	A. B. Tucker, M. J. Flynn	dynamic microprogramming
〃	R. H. Eckhouse	MPL.
〃	R. Zacks 他	a firmware APL time-sharing system
72	E. W. Reigel 他	The interpreter
〃	R. F. Rosin 他	QM-1
〃	W. T. Wilner	B-1700
〃	Micro-5[6)]	Firmware monitoring
73	情報処理	マイクロプログラミング特集号
〃	ダイナミック・マイクロプログラミング・シンポジウム[7)]	

した．これでは，制御記憶として光トランジスタとダイオードを直列にした素子を用いたマトリクス（その記憶情報はパンチしたカードで照射する光をマスクすることにより与えられる）やプラグボードなどを用いて，記憶されているマイクロプログラムを簡単に取り換えることができるようにした．これによって，マイクロプログラムを取り換えて種々の実験を行った．

1962年 Grasselli は pathfinder memory を用いて制御記憶のマイクロ命令を指定する方式を提案した．これでは pathfinder memory の内容を変更することによってマイクロプログラムを変更することができる．

1964年に発表された TRW-133, PB-440, C-8401 などは stored logic の概念を具体化したもので，マイクロプログラムを書き換え可能な記憶装置に入れるものであるが，当時の書き換え可能な記憶装置の速度が遅く，これを制御記憶に用いた計算機は速度が著しく低下するため，この考え方は広く一般化しなかった．

1964年 IBM System/360 が発表されて，マイクロプログラミングも第2期に入るが，1965年に発表された CAS はマイクロプログラム作成の手法として注目されるものである．

また，emulation は IBM Systen/360 が旧型機と入れかえられたとき，旧型機のプログラムをそのままの形で System/360 によって能率よく実行するために開発された手法である．それまでは，他機種のプログラムをそのままの形で処理するには，シミュレーション法，すなわち，プログラムによってインタプリティブに実行する方法がとられていたが，これでは処理速度が著しく低下する．そこで，マイクロプログラムを利用して効率よく処理しようとするのが emulation であり，すべての処理をマイクロプログラムで行う場合と，機種の違いによってそれが困難な場合一部プログラムで行う場合とある．後に emulation の概念は拡大されて，実在しない仮想計算機を対象にした emulation も考えられるようになった．

1967年には，最初に述べたように A. Opler によって firmware という言葉が提案された．また，H. Weber は EULER と呼ばれる ALGOL に似た言語をマイクロプログラムを利用して処理する実験システムについて報告している．G. A. Rose は大型計算機とビデオターミナルとのインタフェース制御のためのマイクロプログラム計算機を発表している[3]．1968年の松下氏の発表はマイクロプログラム方式の通信制御装置に関するものである[4]．

1970年代に入り，IC メモリの発達に伴って，書き換え可能な制御記憶が実用になり，マイクロプログラミングも第3期に入った．

N. Bartow はマイクロプログラムによって計算機の故障を発見するマイクロ診断について発表している[5]．それまでのハードウェアによる故障検出方法にくらべて診断のための余分の装置がほとんど必要ないこと，きめ細かい種々の精密なテストが行えることなどの特長がある．

また，制御記憶の中のマイクロプログラムを書き換えて使うダイナミック・マイクロプログラミングに関する論文も種々見られるようになった．

マイクロプログラムを効率よく作成する方法も望まれるようになり，1971年に R. H. Eckhouse によって発表された MPL その他がある．

1972年には先に述べた2レベルの制御記憶をもったシステムである interpreter, QM-1 などが発表され，また，firmware を活用した商用の計算機 B-1700 も発表された．

また，計算機システムの性能評価のため，その動作状況を観測するのに，現在，ハードウェア・モニタ，ソフトウェア・モニタが広く使用されているが，この目的のために firmware を用いることも論じられている[6]．

わが国では1973年の『情報処理』6月号（マイクロプログラミング特集号）においてマイクロプログラミングの現状が紹介され，将来について論じられた．また，1973年7月には本学会プログラミング・シンポジウム委員会により，ダイナミック・マイクロプログラミング・シンポジウムが開催され，ダイナミック・マイクロプログラミングや firmware についての講演ならびに討論が行われた．さらに，本大会では firmware に関する数件の研究発表が行われている．

4. Firmware の意義と特長

計算機システムが hardware と software とで構成されていると考えると，その間のインタフェースとなるものは機械語命令であるということができる．hardware と software の間に新たに firmware を考えると，Fig. 5 に示すように，プログラムを構成する命令は software と firmware との間のインタフェースとなり，この命令を firmware，すなわち，マイクロプログラムが解釈実行することになる．そして，

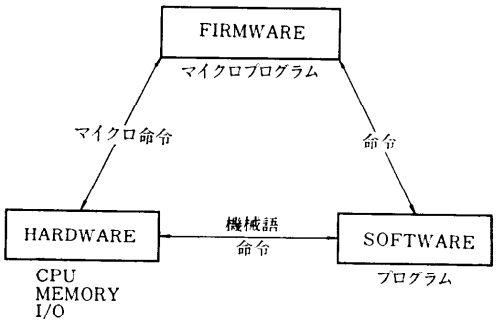

Fig. 5 Relation of Hardware, Firmware and Software

firmware と hardware との間のインタフェースとしてマイクロ命令が存在するのである.

firmware を考えない場合には, 命令コードは機械語命令として hardware によって一義的に決定されてしまい, そのため software は機械語命令, したがって hardware によって大きい制約をうけることになる. マイクロプログラミング方式の計算機ではその命令は firmware によってきまる. したがって, firmware を変更することによって hardware を何ら変更することなく, ある程度自由にその命令を変えることができる. 特に制御記憶が書き換え可能であれば, きわめて容易に命令を変更することができ, 計算機システムの融通性, 柔軟性が著しく増大することになる.

また, firmware を構成するマイクロ命令はマイクロオペレーション, すなわち計算機内の基本的な操作の組合せを指定するものであり, そのため, きめの細かい操作の制御が可能になり, したがって, 問題向きの, あるいは処理に適した制御論理を組立てることができる.

firmware のこのような特長は software の簡単化に役立ち, また, 計算機システムの性能の向上に貢献するものと考えられる. すなわち, software の簡単化により記憶空間が節約され, 問題向きの命令の組を作ることができ, あるいは処理に適した制御論理が組立てられ, さらに余分な操作を減少させることができるため, 処理速度が向上し, 空間, 時間両面での性能の向上が期待される. もちろん, マイクロプログラミングを使わない計算機でその機械語命令が問題によく適合している場合にはマイクロプログラミング方式より処理速度が速いことは確かであるが, 計算機の広汎な利用を考えた場合には, firmware の特長が発揮されるものと思われる.

5. Firmware の応用

firmware の応用について, すでに述べたものをも含めてまとめてみよう.

(1) Emulation すでに述べたようにある計算機で, 他機種のプログラムをそのままの形で処理するのに, firmware を利用して効率よく実行する手法で, emulation を行うとき, システム全体が emulation のために用いられるスタンド・アロン方式と, 実行する計算機のオペレーティング・システムの下で他のジョブと同等の一つのジョブとして扱われるインテグレーテッド方式とがある. 後者ではマルチプログラミングの処理も可能になる.

(2) マイクロ診断 マイクロ命令によって計算機内の論理回路レベルでの制御をすることができる特長を利用して hardware の故障検出をするもので, 計算機システムの RAS の向上に大きく寄与するものであり, 書き換え可能な制御記憶が用いられるようになるとともに, 今後広く活用されるであろう.

(3) 言語処理 高レベル・プログラミング言語の処理に対する応用で, コンパイラ向きの命令を firmware で作り, コンパイラの効率向上をはかる方法, コンパイラを firmware で作る方法, 適当な中間言語を設定し, これを firmware で処理する (この際, 原始プログラムを中間言語に変換するのに機械語による方法と firmware を利用する方法とがある) 方法などが考えられる. 最後の中間言語を原始言語に近づければ, 原始プログラムの直接実行に近づくことになる.

言語処理に対する firmware の応用はダイナミックな処理を必要とする場合, たとえば, 会話型処理や, APL のように言語自体がダイナミックな処理を必要とするようなとき, 特に有効であろう.

(4) 問題向きマクロ命令 いままでの機械語でプログラムすると非常に無駄が多く時間のかかるようなものに有効で, たとえば,

- 高精度算術演算 多倍長の演算を機械語命令で行うと非常に時間がかかることは周知の通りで, firmware の効果が発揮されるであろう.

筆者らは KT パイロット計算機のマイクロプロ

グラムを取替えて当時よく行われていた e あるいは π を多数桁求める実験を行い，極めて有効であることを認めた．その他に，

- 乱数発生
- 関数値の計算[8]

などが考えられるが，これらに対しては firmware に適したアルゴリズムの開発が望まれる．

- ソート，テーブル・サーチ[9] 汎用計算機の場合には大幅な性能改善が期待できよう．

(5) システム・モニタ 計算機システムの動作状況を観測するモニタとしてハードウェア・モニタ，ソフトウェア・モニタが使われているが，firmware を利用することにより，経済的に効率よく種々の情報を集めることができると考えられる．

(6) 特殊目的の処理装置 汎用計算機としてではなく特殊目的のもので firmware が有効と思われる分野をあげると，実時間処理，図形処理，パタン処理，入出力処理，通信制御など，広範囲の応用が考えられる．

6. Firmware の将来

firmware は上に述べたような種々の分野で益々広く利用されるようになるであろうが，そのために解決すべき問題点として，つぎのことが考えられる．

(1) マイクロプログラム作成の手法 マイクロプログラミングは計算機の制御装置のために考えられたものであり，これを利用して効率よい処理を行おうとするのが firmware であるので，その作成に当っては無駄のない効率のよいマイクロプログラムを作り出す必要がある．したがって，現在までのところ主としてマイクロプログラム・アセンブラが使われている．しかし，firmware の応用面が広がるとともに作成量も増大すると思われるので，高レベル言語によるマイクロプログラムの作成が望まれる．このための記述言語とその処理方法の開発は重要な課題の一つであろう．

(2) Firmware 向きのアルゴリズム 現在までいろいろな情報の処理に使われているアルゴリズムは機械語によるプログラミングを対象として考えられていた．しかし，マイクロプログラムでは計算機内の基本操作を制御できるので，この点を考慮したアルゴリズムが開発されれば，firmware の効率は著しく向上することが期待される．

(3) Firmware 向きの hardware の構成 書き換え可能な制御記憶を使ってマイクロプログムラを書き換えて使用する firmware を前提とした計算機の hardware としては，従来の計算機と違った構成をとることが望ましい場合があると考えられる．先に述べた2レベル・マイクロプログラミング方式はその一つの試みであろうが，更に前項に述べた firmware 向きのアルゴリズムが開発されれば，それらをも考慮した構成を考えるべきであろう．

このような問題を解決することにより，firmware を容易に作成し，有効に利用することができるようになれば，待ち望まれている第4世代の計算機に対して，firmware を活用した素晴らしい発展が期待できるであろう．

参考文献

1) 萩原：マイクロプログラミングの発展，情報処理，vol. 14, no. 6, pp. 374～378 (June 1973).

以下には上記に参照していない文献のみを上げる．

2) W. Renwick: EDSAC II, Proc. IEE, vol. 103, pt. B, Suppl. 2, pp. 277～278 (April 1956).
3) G. A. Rose: 'Intergraphic,' a microprogrammed graphical-interface computer, IEEE Trans. on EC, vol. EC-16, no. 6, pp. 773～784 (Dec. 1967).
4) S. Matsushita: A microprogrammed communication control unit, The TOSBAC DN-231, Information Processing 68, pp. 812～817, North-Holland (1969).
5) N. Bartow, R. McGuire: System/360 model 85 microdiagnostics, Proc. SJCC, 1970, pp. 191～197 (1970).
6) Larry Roberts: Features of Firmware monitorings, SIGMICRO Newsletter, vol. 3, no. 4, p. 3 (Jan. 1973).
7) ダイナミック・マイクロプログラミング・シンポジウム報告集，情報処理学会プログラミング・シンポジウム委員会 (1973).
8) 小柳他：マイクロプログラムによる初等関数近似，昭和48年度情報処理学会第14回大会講演予稿集，pp. 171～172 (1973).
9) S. S. Husson: Microprogramming: Principles and Practices, pp. 84～86, Prentice-Hall (1970).