コンピュータ評価のための各種のミックス*

高 橋 義 造**

1. はじめに

コンピュータの性能の比較をひとことで行うために従来 Gibson mix が一番都合よく用いられてきた．ひところはわが国で新しいコンピュータが開発されると必ず Gibson mix が発表されるというふうであった．コンピュータの評価は cost/performance ratio によって比較されるべきであるが，コンピュータの cost は一意的に決定するのに対して performance(性能) の方はそうはいかない．同じコンピュータでも記憶容量や入出力機器の構成によって性能は大いに左右される．また全く同じ構成のコンピュータでも使用するソフトウェアによって性能が異るのが一般である．したがってコンピュータの性能をただ一つの指標によって代表させようということなど不可能なことである．

しかしながら，このような事情を十分承知の上で，なおかつコンピュータの性能の指標としてとり上げられるのが各種の「ミックス」である．ミックスの中では Gibson mix がもっとも著名である．この外事務用コンピュータの性能評価のための commercial mix や，また最近では real-time mix なるミックスが提案されている．これらの instruction mix はコンピュータの応用分野によってそれぞれ異るものが採用されるが，さらに細かい目的別のミックスとして，FORTRAN プログラムのコンパイルの性能を表現する FORTRAN mix や，COBOL プログラムのコンパイルに関する COBOL mix というようなミックスも考えることができる．instruction mix に記憶容量と入出力時間を併せて考慮した指標が K. E. Knight によって提案されている[6]．

同じ mix という言葉に job mix というものがある．これはコンピュータで処理されるすべての job の代表となるいくつかの job の集合である．ベンチマークが個別のプログラムであるのに対し job mix は数個ないし数10個の job の集合であるので，コンピュータの性能評価にはより普遍的な指標を求めるのに利用できるであろう．job mix は instruction mix と異り，CPU の性能だけではなく，I/O システムプログラムの性能もふくめたコンピュータシステム全体としての性能評価に役立つ．job mix が実際の job program の一部からできているのに対し，逆行列の計算プログラムや作表などの基本的なプログラムをあつめたプログラムの集合によってコンピュータの性能をはかろうとする試みもある．この基本的な応用プログラムを kernel とよぶ．

これらの各種のミックスについて以下に解説する．

2. Gibson mix

Gibson mix はわが国では何故かもっともよく知られたミックスである．しかるにその起源は長い間はっきりせず，IBM だとか NASA で作られたとか，また Gibson というカクテルに由来しているなどの風説が信じられていたが，最近石田氏の努力によってはじめてその起源が明かにされた[1]．それによると Gibson mix についての論文はやっと1970年になって発案者の J. C. Gibson によって IBM の社内報告として発表されたばかりであった．Gibson mix は Computer の機械命令 (instruction) のうち13群のカテゴリに属する命令の実行時間に，各カテゴリごとにある荷重をかけて平均をとった平均命令実行時間である．J. C. Gibson の原典[2]によるこの荷重を表1に示す．なお奇妙なことにわが国で使われている Gibson mix の荷重は Gibson 氏のものとは多少ことなる．これを表2に示す．

Gibson mix としては上にのべたように平均命令実行時間 (μs) をいうが，時にはこの逆数の，単位時間内の平均命令実行回数 (instruction/sec) をいうこともある．

* Mixes for Computer Performance Evaluation, by Yoshizo Takahashi (Tokyo-Shibaura Electric Co., Ltd.)

** 東京芝浦電気 (株) 電子計算機事業部

表1 Gibson 氏による Gibson mix の荷重

	Instructions	Weight
1	load and store	31.2%
2	fixed point add and subtract	6.1
3	compare	3.8
4	branch	16.6
5	floating add and subtract	6.9
6	floating multiply	3.8
7	floating divide	1.5
8	fixed point multiply	0.6
9	fixed point divide	0.2
10	shifting	4.4
11	logical, And, Or, etc.	1.6
12	instructions not using registers	5.3
13	indexing	18.0

表2 一般に使用されている Gibson mix

	Instructions	Weight
1	add/sub/load/store	33.0%
2	multiply	0.6
3	divide	0.2
4	branch	6.5
5	compare	4.0
6	transfer	17.5
7	shift	4.6
8	And/Or	1.7
9	index	19.0
10	short floating add/sub	5.8
11	long floating add/sub	1.5
12	short floating multiply	3.2
13	long floating multiply	0.8
14	short floating divide	1.3
15	long floating divide	0.3

3. その他の instruction mix

Gibson 氏もその原典の中でのべているように，Gibson mix は万能ではなく，時代おくれの指標であるかもしれない．Gibson mix の荷重は7ケの科学技術計算用の job のプログラムをトレースして各種命令の出現回数の統計をとって求められたものである．従って他の job をあつめてきて統計をとれば異った荷重がえられることになる．従来われわれは Gibson 氏の意図に反して Gibson mix を過大に評価していたのではないだろうか．たとえば

$$\text{ベンチマークの処理速度} = \frac{\text{ソフトウェアの性能}}{\text{Gibson mix}} \qquad (1)$$

というような考えをおこないがちである．すなわちハードウェアの設計者が「せっかくこんなに Gibson mix の小さい，はやいコンピュータを設計したのにソフトウェアがわるいためにベンチマークで他社のマシンにおくれをとった」などとぼやくのである．ソフトウェア屋からいわせれば「Gibson mix ははやいかもしれないが，こんなにテーブル処理のやりにくいコンピュータはない」ということになる．Gibson mix が真にハードウェアの性能を表わしていれば，(1)式は成立つかもしれないが，実際はそうではない．

そこで異る目的の処理プログラムごとにinstruction mix をもとめ，これが Gibson mix といかに異るかを明かしておくことは有益である．一例として TOSBAC-3400 の FORTRAN でかかれたプログラムで，20元の逆行列の計算プログラムをトレースした結果の比率を表3に示す．表3により計算した TOSBAC-3400 のミックス（平均命令実行時間）は Gibson mix の約1.1倍になる．

表3 TOSBAC-3400 による逆行列計算プムラグロの命令比率

	instructions	weight
1	add/sub/load/store	6.0%
2	multiply	11.4
3	divide	0.0
4	branch	10.7
5	compare	9.6
6	transfer	0.0
7	shift	0.0
8	And/Or	0.0
9	index	34.5
10	short floating add/sub	29.0
11	long floating add/sub	0.0
12	short floating multiply	8.6
13	long floating multiply	0.0
14	short floating divide	0.1
15	long floating divide	0.0

一般にあるコンピュータの命令からm個の命令をとり出し，この命令に W_i $(i=1\sim n)$ の荷重を与える．これらの命令の実行時間をそれぞれ $\tau_1, \tau_2, \cdots, \tau_n$ とすると instruction mix μ は

$$\mu = \sum_{i=1}^{n} W_i \tau_i \qquad (2)$$

となる．n個の代表命令としてどのような命令をえらぶか，またそれらに対する荷重をどのように与えるかによって同じコンピュータに対して異る instruction mix がえられるのである．

つぎに Gibson mix 以外の各種のミックスについて述べよう．

3.1 Commercial mix

これも原典は不明であるが，事務用計算機の性能を比較するのに用いられるミックスとして commercial mix というものがある．前章の Gibson mix を科学用ギブソンミックスというのに対してこれを事務用

Gibson mix ということがあるが，これは Gibson 氏も知らないであろう．commercial mix の計算に用いられる荷重を表4に示す[3]．

表4 コマーシャルミックスの荷重

	Instructions	Weight
1	arithmetic operation	9%
2	compare	24
3	move	25
4	jump	31
5	edit	4
6	I/O initiation	7

確たる証拠はないが，この荷重は大体うなづけるものである．

3.2 リアルタイムミックス

これも同様に原典は不明であるが，オンラインリアルタイム用のコンピュータの性能評価に最近使われているのがリアルタイムミックスである．リアルタイムミックスの計算に使われる荷重を表5に示す[4]．

表5 リアルタイムミックスの荷重

	Instructions	Weight
1	load	22%
2	store	22
3	add/sub	11
4	conditional jump	9
5	unconditional jump	8
6	compare	6
7	logical	6
8	shift	6
9	index increment and test	5
10	loader modify index	5

3.3 Command and Control mix

instruction mix では原典のはっきりしないものが多い．この理由を考えてみると発案者がたまたま一群のプログラムをトレースして統計をとった結果を用いたので普遍性について確信がないために論文の発表をためらわれたのではないだろうか？

さてここにのべる command and control mix にははっきりした原典がある[5]．（この名前は筆者が勝手につけたものであることを断っておく．）この mix の計算に用いられる荷重を表6に示す．

この荷重はコマンドコントロール用のプログラムをトレースした統計にもとづいてもとめられたものである．

3.4 FORTRAN mix

筆者がかつて考えたもので FORTRAN プログラムのコンパイル速度に注目して計算機の性能の指標とした FORTRAN mix についてのべる．当時論文として発表しなかったので，同様に原典のない組に属するものである．FORTRAN mix は FORTRAN プログラムのコンパイル時の平均命令実行時間である．すなわち

表6 Command and Control mix の荷重

Instructions		Weight
load/store		47.5%
add/sub	fixed point	8
	increment memory by 1	3
	floating point	0.5
mult/div	fixed point	0.3
	floating point	0.2
logical	And/Or	7
branch	unconditional	5
	on register value	7.5
	index and increment	3.5
testing	register vs. memory	4
	memory vs. 0	2.5
	miscellaneaous signal	0.5
shift/cycling register		7.5
miscellaneous		3

$$\text{FORTRAN mix} = \frac{\text{コンパイル時間 (CPU 時間)}}{\text{コンパイルに要した総実行命令数}}(\text{sec}) \quad (3)$$

である．この FORTRAN mix の測定はトレースを行なったのではなく，カウンターを用いた直接測定法によった．すなわち FORTRAN プログラムをコンパイルする際の実行時間（slave mode の CPU 時間）と，この間に fetch されて slave mode で実行される命令の数を図1のようにして測定し，(3)式によって平均命令実行時間を算定したのである．

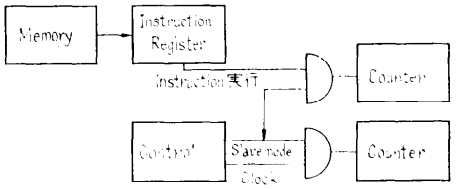

図1 FORTRAN mix の測定法

このようにして測定した FORTRAN mix の値はプログラムによって表7のようにばらつくが，TOSBAC-3400 の場合には Gibson mix に対して10～20％大きな値をとる．同じプログラムを同じ計算機の，別の FORTRAN コンパイルでコンパイルしたときの測定結果を表8に示す．はやいコンパイラはステップ数だけではなく，速い命令をえらんでつくられていることがわかる．

3.5 Instruction mix の直接表示

GE-635 で行なわれ，TOSBAC-5600 でもこれに

表7 FORTRAN mix の測定
(T. ASBAC-3400 FAST FORTRAN)

プログラム番号	ソース・ステートメント数	コンパイル時の		FORTRAN MIX	1ステートメント当りの処理命令数
		実行命令数	実行時間		
	n	N	T sec	$(T/N)\times10^6\mu s$	N/n
1	31	96×10^3	1.08	11.2	3,100
2	46	168×10^3	1.75	10.45	3,650
3	219	$1,020\times10^3$	9.81	10.8	4,650
4	50	168×10^3	1.76	10.5	3,360
5	857	$3,380\times10^3$	34.4	10.2	3,940
6	1,236	$5,500\times10^3$	52.5	9.6	4,450

表8 FORTRAN mix の測定 (TOSBAC-3400 FORTRAN old version)

プログラム番号	ソース・ステートメント数	コンパイル時の		FORTRAN MIX	1ステートメント当りの処理命令数
		実行命令数	実行時間		
	n	N	T sec	$(T/N)\times10^6\mu s$	N/n
3	219	$3,540\times10^3$	39.2	11.1	16,100
4	50	746×10^3	5.3	11.2	9,520
5	857	$9,020\times10^3$	106.3	11.8	10,500

ならって行なわれているのが instruction mix の直接表示である．これは単位時間における命令実行回数が (instructions/sec) コンソールにつけられたメータにつねに表示されるものである．この原理は図2のように命令実行パルスによってトリガされたモノステーブルマルチの出力を直流電流計に流すだけの簡単なものである．

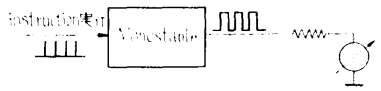

図2 GE-635 における instruction mix の直接表示

処理プログラムがなくなり，モニタで CPU がまわっているときはメータの針が最高の指示に止っており，カードリーダから処理プログラムが投入されると CPU が処理プログラムにかかるのでメータの針が低い指示にかわる．また I/O の終了割込み待ちになるとメータの指示は0になり，命令が実行されず，割込み待ちの状態がよくわかる．少しなれると，このメータのうごきをみていると処理中のプログラムの効率がわかるという．

CDC-3300 では，このメータの代りに命令実行パルスを可聴周波数に低減した上でスピーカに入れて音を出している．処理プログラムがなくなり，モニタでまわったときにはピーという高い音が出る．ジョブを投入すると，これがブーという低い音になり，I/O の割込み待ちが発生するとその瞬間だけ音がとぎれる．この音をきいていると自分のプログラムがコンパイル中か，実行中で，計算を終って出力をディスクに出しているところかなどが実によくわかる．

3.6 Instruction mix の各種測定法

Gibson mix をはじめほとんどの instruction mix は，いくつかの job program をトレースしてえられる全命令をグループに分類し統計をとることによって荷重を求め，この荷重も各グループの命令の平均実行時間より(2)式により計算する方法によってもとめられている．このような統計をとるトレーサをつくることも可能である．

荷重に応じた回数だけその命令をくり返すようなプログラムをつくれば，直接に instruction mix を測定することができる．厳密にこれを行なうためにはアセンブラでプログラムをつくる必要があるので，他のコンピュータに用いるわけにはいかないが，FORTRAN でかいても多少の誤差は出るが，instruction mix の近似値がえられよう．このプログラムを用いると，いろいろなコンピュータの instruction mix が直接測定できるので便利である．

図1の方法によってハードウェアによる instruction の直接測定ができることを示した．この方法を拡張して命令ごとにカウンタをおけば，mix instruction mix の計算の基礎となる荷重の測定もできる．

トレースによらず荷重を測定するのにプログラムのリストにあらわれる命令を分類して，統計をとる方法がある．一般にプログラムには loop やとび越し，サブルーチンなどがあるので，このような方法は正しくないように考えられるが，現実にやってみると実際のデータときわめて近似した結果がえられる．この方法の利点はデータによって結果がかわらないことである．例えば FORTRAN コンパイラをこの方法で解析すると，えられたミックス値はソースプログラムによってかわるおそれがないので mix の目的である一意的な性能の表示には好都合である．

4. Knight の Computing power

K. E. Knight はコンピュータの性能を instruction mix でなく，Computing power P で表現することを提案した[6]．P は CPU の計算速度 t_c，I/O まちの時間 $t_{I/O}$，および記憶容量 M によってきまるとし，次式で定義する．

$$P=\frac{M\times 10^{12}}{t_c+t_{I/O}}. \quad (4)$$

p の詳しい算出法は原論文にゆずるが，CPU の計算速度 t_c について次式で求めることになっていることだけを紹介する．

$$t_c=C_1A_{FI}+C_2A_{FL}+C_3M+C_4D+C_5L. \quad (5)$$

ここに A_{FI} は固定小数点加算時間，A_{FL} は浮動小数点加算時間，M は乗算時間，D は除算時間，L は論理演算時間である．C_1～C_5 はそれぞれの荷重で，科学計算と事務計算でことなり，表9のようにきめられている．

表9 Knight の荷重

	Instructions	Weight	
		sccentific	commercial
1	fixed add		
	a. for computers without index registors or indirect addressing	10%	25%
	b. for computer with index registers or indirect adressing	25	45
2	floating add	10	0
3	multiply	6	1
4	divide	2	0
5	logical operation		
	a. for computers without index registers or indirect adressing	72	74
	b. for computers with index registers or indirect adressing	57	54

5. Calingert の kernel

P. Calingert はコンピュータシステムの評価の一つの指標として kernel を提案している[7]．コンピュータの性能評価に instruction mix を用いるとコンピュータのアドレス方式による差が考慮されない．したがってアドレス方式の異るコンピュータの比較をするときには単に加算時間というよりも A＋B→C のような処理の単位で比較を行なう方がよい．この考えを拡張してベンチマークほど大きくはないが，一つの task の単位となる命令の集合，例えば逆行列の計算といったようなプログラムによって処理速度の比較を行なうのである．このようなプログラムを kernel とよび各種の kernel について実行速度の荷重平均をとるという考えである．

6. Job mix

もっとも現実的な throughput の測定は job mix によるものであろう．job mix はそのコンピュータで処理されるすべての job の傾向を定性的および定量的に代表する多くのプログラムをあつめたもので，job mix の処理時間は数時間におよぶことが普通である．job mix をつくるためには長期間にわたる job 統計をしらべ，実際に処理されたジョブの中から典型的なものをえらび出すことが必要である．これらの job の実行順序も throughput に影響するので注意しなければならない．

7. まとめ

コンピュータの性能評価のための各種のミックスについてのべたが，はじめにものべたようにこれらのミックスでもってコンピュータの性能を一面的に評価することのないように気をつけて頂きたい．またわれわれソフトウェア部門の人間としては，これらのミックス値でソフトウェアの性能を論じようという試みには強く反対せざるをえない．Gibson mix がよいのに throughput がわるいのはソフトウェアのせいにされたり，throughput がよいのはハードウェアのせいにされたり，ソフトウェアはいつになっても正しい評価をうけないという嘆きは筆者だけのものではないであろう．コンピュータシステムの評価法を確立することがハードウェアとソフトウェアのどちらの進歩にも大いに貢献することと信じている．

参考文献

1) 石田晴久：“ギブソンミックスの起源について，”情報処理，Vol. 13, No. 5, 1972, pp. 333～334.
2) Jack C. Gibson: “The Gibson Mix”, IBM Ponghkeepsie Lab. Technical Report, TR 00.2043, June, 1970.
3) 高橋，手塚，三好：“ユーザのためのコンピュータマニュアル，” p. 13, オーム社 (1970).
4) 電子機械工業会：“ミニコンピュータの応用事例調査報告書，” pp. 81～82, 昭和47年3月.
5) E. Raichelson, G. Collins: “A Method for Comparing the Internal Operating Speed of Computer,” CACM, Vol. 7, No. 5, pp. 309～310, May 1964.
6) Kenneth E. Knight: “Changes in Computer Performance,” Datamation, Sept. 1966, pp. 40～54.
7) P. Calingert: “System Performance Evaluation; Survey and Appraisal,” CACM, Vol. 10, No. 1, pp. 12～18, Jan. 1967, Corrigenda on ibid., Vol. 10, No. 4, p. 224, April 1967.

(昭和47年8月28日受付)