オペレーティング・システム 25－4
（1984. 12. 14）

三階層メモリ方式のシミュレーションによる評価

金澤 正憲
京都大学 大型計算機センター

1. はじめに

計算機システムにおいて、多量のデータを迅速に処理する必要性が生じてきたり，または，端末から要求された多数のジョブを短い応答時間で処理するために，主記憶装置の大容量化が望まれる。記憶素子の超LSI化によって，技術的にも可能な容量が増大している。この結果，128 MBとか256 MBという大容量の主記憶装置も汎用の大型計算機で実現されている。

CPUの演算速度（素子の動作速度）が速くなるとともに，主記憶の素子も高速化されている。しかし，実装上の問題が大容量化に伴って生じるため，CPUの演算速度と，CPUからみた主記憶のアクセス時間との比は拡がる一方である。

キャッシュ・メモリは，十数年前既に問題となっていたCPUと主記憶との動作速度のギャップを埋めるものとして，IBM System 360/85で最初に実用化された(1)。それ以後に制作された高速の計算機には，必ずキャッシュ・メモリが装備されている。キャッシュ・メモリの効果に関する研究も数多く行なわれている。そのうちの多くのものは、ヒット率を向上させるための技法に関するものである。筆者らは，文献(2)において，アドレス・トレーサから得られたトレース・データを使用したトレース・ドリブン型シミュレータによって、キャッシュ・メモリの有効性を検討した。その結果，プログラムに依存することなく，比較的高いヒット率が得られることが判った。

しかし，CPUと主記憶の動作速度のギャップが大きくなるにつれて，平均のアクセス時間が問題となる。例えば，平均アクセス時間をそれぞれのアクセス時間にアクセスされる割合の重みを付けたものと単純に仮定しよう。ヒット率が0.9で，キャッシュ・メモリと主記憶の動作速度の比が1対10の時，ヒット率が1の場合に比べて2倍程度に遅くなるが，1対30になると4倍程度になる。即ち，後者の平均アクセス時間は，前者の2倍程度になる。新しく開発される機種の性能が，旧機種の2倍から3倍程度である現状を見れば、平均アクセス時間の問題は決して少さくないことが判るだろう。

キャッシュ・メモリは、主記憶に比べて非常に小さい容量，例えば数百から千分の一の大きさで，十分なヒット率を得ている。しかし，上述のような問題に対して，一つの解決方法は、キャッシュ・メモリの容量を増大し更にヒット率を高めることである。最近の計算機は64 KBとか128 KBの容量を持っているものがある。この場合，文献(3)の中でSmithが指摘しているように，(1)物理的な大きさとロジックの複雑さによるキャッシュ・メモリ自体のアクセス時間の増大と(2)価格の問題が現われる。そこで，もう一つの解決方法として，2レベルのキャッシュ・メモリが考えられる。

ここでは，プログラムのトレーサから得られたデータを入力としたトレース・ドリブン型のシミュレータにより，2レベルのキャッシュ・メモリの効果を考えた。対象とした計算機は、京都大学大型計算機センターのFACOM M-382で，トレース用のプログラムは，文献(2)，(4)で用いたプログラムと同じものを用いた。

2. トレーサとプログラム

トレーサは，1つ1つの命令をインタプリティブ的にトレースし，命令のアドレス*，オペランドのアドレス，および，その長さを収集する。このトレーサは，処理プログラムとして動作するため，制御プログラムも全てをトレースはできないが，制御プログラムであっても処理プログラムのモードで動作する部分はトレースできる**。

トレースされたプログラムは，FORTRANコンパイラと科学技術計算のプログラム（実行時）で，文献(2)，(4)で使用したものと同じである。その内，ここのシミュレーションで用いたプログラムとその特徴を表1に示す。

命令部とデータ部の大きさとは，主記憶を32バイトのブロックに分割した時に何ブロック実際に参照するかを求め，それをKBで示したものである。データ部は，主記憶への読出しのみを対象とし，書込みは含んでいない。

サンプル・プログラム	プログラムの大きさ	命令部		データ部	
		大きさ	実行命令数	大きさ	参照回数
① FORTRANコンパイラ	—	120 KB	249万	122 KB	112万
② 代数方程式の根(ベアストウ法)	132 KB	14	43	10	17
③ 固有値問題(ヤコビ法)	148	13	20	10	10
④ 連立一次方程式(掃出し法)	204	13	33	14	17
⑤ ソーティング	148	17	26	13	11

表1．サンプル・プログラム

付録1に，M-382の命令一覧とすべてのサンプル・プログラムから求めた命令の出現確率を示す。

3. シミュレーション・モデル

現在の大型計算機の記憶階層をみると，キャッシュ・メモリ，主記憶，ディスク装置の3つの階層から成っている。それぞれの間での転送の単位は，ブロック（例えば，32バイト）とページ（例えば，4KB）という具合に大きく異なっている。

ここで検討する3階層メモリ方式とは，2レベルのキャッシュ・メモリを持つもの，即ち，第1レベルのキャッシュ・メモリ（以後キャッシュ1と呼ぶ）と第2レベルのキャッシュ・メモリ（以後キャッシュ2と呼ぶ）を持つものである。従って，転送の単位は同一であることを原則とする。

3.1 アクセス要求とレベル

一般に複数レベルの記憶階層を考える時，CPUからのアクセス要求がどのように各レベルに作用するかが問題である。どの要求も全レベルに達するか否かである。

LRU方式の置換え規則で，かつ，LRUスタックの個数が下位レベルの方が大きい場合，全レベルに要求が達するのであれば，ヒット率の考察は簡単である。LRUの特徴から，第iレベルのヒット率はレベルに関係なく，その容量（スタック数）によって一意に定まるからである。(5) 全レベルの動作が第1レベルと同じ速度でなければならないので，実際的とは言えない。

---

* M-382は仮想記憶方式の計算機であるため，正確には論理アドレスである。このシミュレーションでは，プログラム毎に行い，物理アドレスを考慮する必要はない。

** OSIV/F4（制御プログラム）のかなりの部分は処理プログラム・モードで動作する。

CPUからのアクセス要求は、ヒットしたレベルで吸収されて、それ以降のレベルには行かないと考えるのが実際的であろう。即ち、上位のレベルでミスした時にのみアクセス要求が来るので、レベル間での包含関係は成立しない。また、上位のレベルで最も頻繁にアクセスされるブロックは、下位のレベルから見れば、最もアクセスされないブロックになる可能性が高いであろう。

従って、キャッシュ2のヒット率は、キャッシュ1でヒットしない場合におけるヒットの割合を考える必要がある。

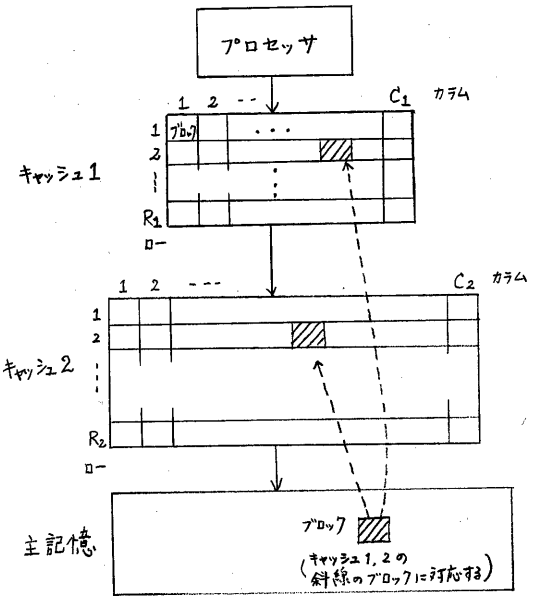

図1. キャッシュ・メモリの構

3.2 キャッシュ・メモリの構造

キャッシュ・メモリの構成は図1に示すようにプロセッサと主記憶の間にキャッシュ1とキャッシュ2がある。両キャッシュは、ブロックに分割されていて、それぞれカラム数が C_1, C_2, ローが R_1, R_2 であるとする。主記憶に関しては図示していないが、同じようにブロックに分割され、キャッシュ1とは C_1 カラム毎に、キャッシュ2とは C_2 カラム毎に区切られてセット・アソシアティブ方式で対応がとられる。

3.3 命令のフェッチとアクセス要求

プロセッサからのアクセス要求には、命令のフェッチ、データのフェッチ、データのストアの3種類がある。命令の長さは可変長であるため(参照付表1)、命令を1つずつ順に取って来るのではなくて、一定の単位で持ってくる。

命令フェッチの単位は対象の計算機の8バイトであるとする。そして、フェッチされた8バイトは命令バッファレジスタに格納される。次の命令がその中にあれば、キャッシュ1に対してアクセス要求を出さないで、命令バッファレジスタから取出される。従って、命令のフェッチ数は、実行される命令の数よりも減少する。

命令の長さが2バイトの場合、最初の1バイトと残りの1バイトが異なるブロックにあることはないが、4又は6バイトの場合はブロックをまたがることがある。このような場合、モデルでは別のブロックへのアクセス要求が生じたとして考慮した。最悪の場合、命令の前半と後半の2回のフェッチが生じ、両方ともミスすることがある。これによって、フェッチ回数は増加するが、命令バッファレジスタの効果が大きいため全体としては、命令フェッチ回数は2.5から3分の1程度に減少した。

以上のことをモデルに取入れたのは、将来、実測データが得られれば、比較し易いと思ったからである。

3.4 データのフェッチとアクセス要求

データの長さは、命令によって異なる。データをフェッチする場合は、各ブロックで高々1回のアクセス要求しか来ないものとした。即ち、32バイトのアクセス要求があり、それが1つのブロックにあれば、1回のアクセスとし、4バイトの要求が2つのブロックにまたがれば2回のアクセスとした。

データのフェッチ回数は、データのフェッチを必要とする命令の数よりも多い。

3.4 置換え規則

キャッシュ・メモリで採用されている置換え規則に，LRU方式とFIFO方式がある。

LRU方式は最も多くの計算機で使用されている方式で，高いヒット率が得られることが多い。しかし，ローの数を多くすると参照毎の更新が大変であるので，ローの数を少なく，例えば4以下にしたり，又は，変則的なLRU方式で実現している。

FIFO方式は，一般にLRUほど高いヒット率が得られないが，その制御が し易い。従って，余り高速でない計算機にみられる。

ここでは，キャッシュ1では，LRUとFIFOの両方式を，キャッシュ2ではLRU方式を検討した。

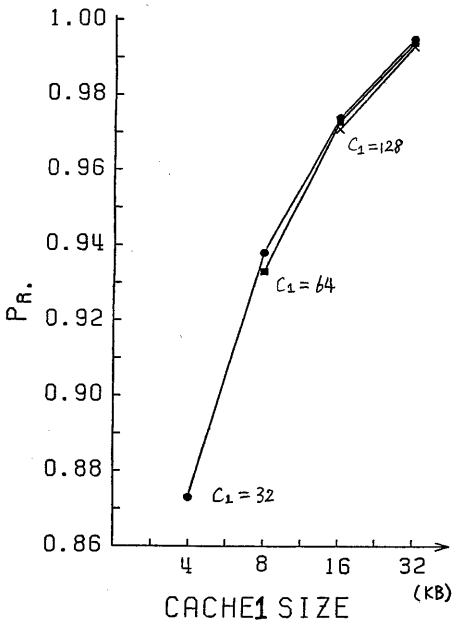

図2. 命令フェッチ・FIFO方式

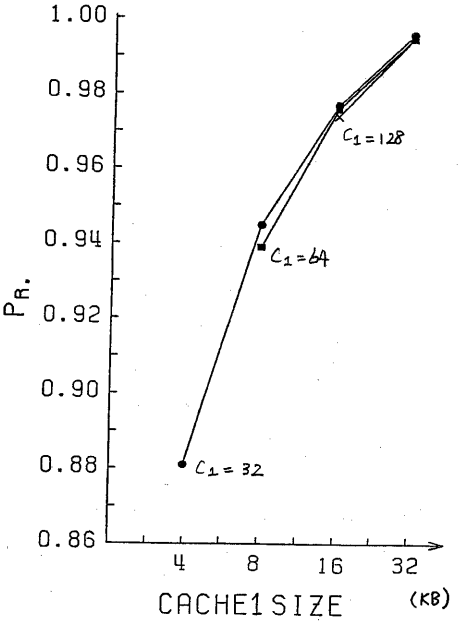

図3. 命令フェッチ・LRU方式

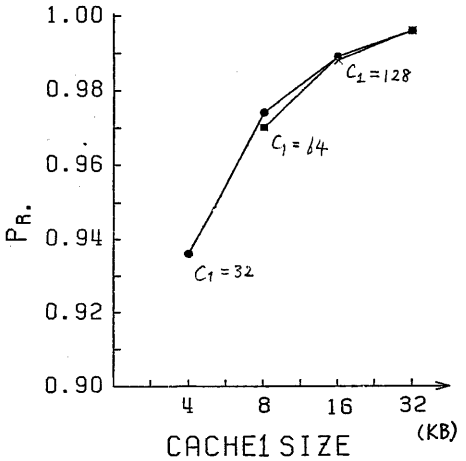

図4. データ・フェッチ・FIFO方式

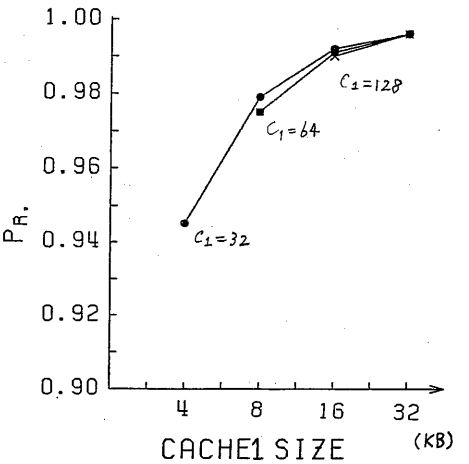

図5. データ・フェッチ・LRU方式

4. シミュレーションの実行と結果

シミュレータの実行に際しては、命令フェッチとデータ・フェッチを区別した。これは両方のフェッチを同等に扱うと、両方のフェッチが同じ様になって　それぞれの特徴が出ないと考えたからである。[6]

まず、トレース・データを読込み、キャッシュ1の動作をシミュレートし、キャッシュ2のためのデータを出力した。次に、その出力データを読込み、キャッシュ2の動作をシミュレートした。なお、先行制御によるアクセス要求はないものとした。

サンプル・プログラム②,③,④では、キャッシュ1が4KBで十分高い(.99以上)得られたため、キャッシュ2でのヒット率は期待できないことが判った。即ち、大きさは、4KBの2.5倍から4倍程度あるにもかかわらず、繰返し使用される大きさは小さいと考えられる。従って、キャッシュ2にあるブロックへのアクセス(キャッシュ1になくて)は殆どないという状況になったと考えられる。

プログラムの大きさが比較的大きいサンプル・プログラム①に対して、その結果を図で示す。図2から図5はキャッシュ1に対するもので、図6から図9はキャッシュ2

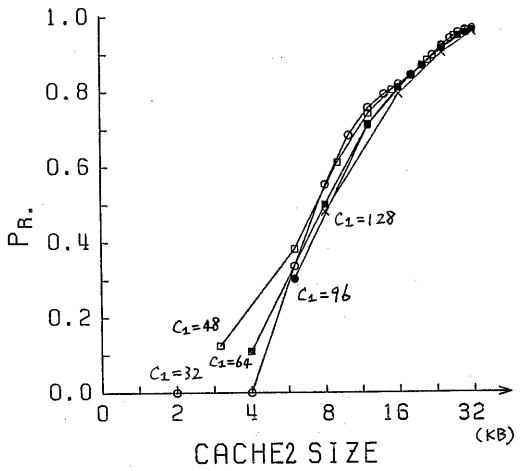

図6. 命令フェッチ(FIFO方式)

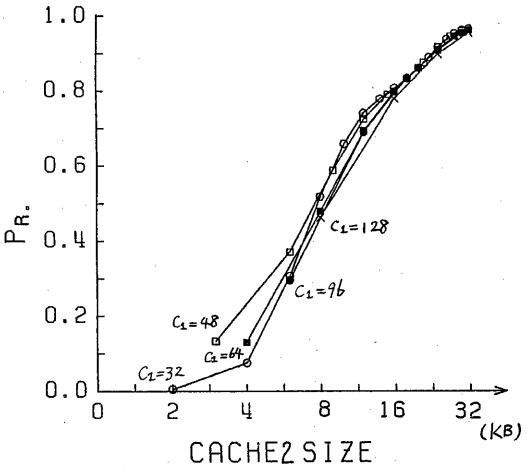

図7. 命令フェッチ(LRU方式)

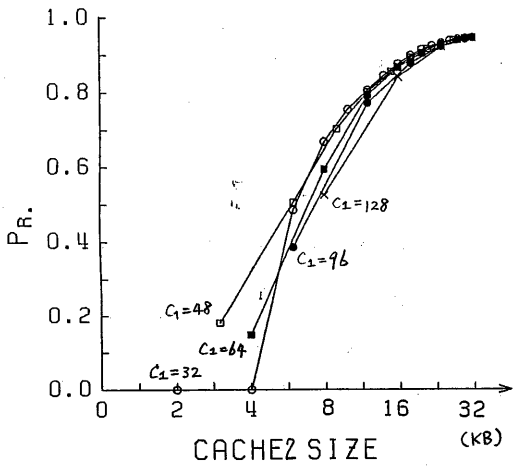

図8. データ・フェッチ(FIFO方式)

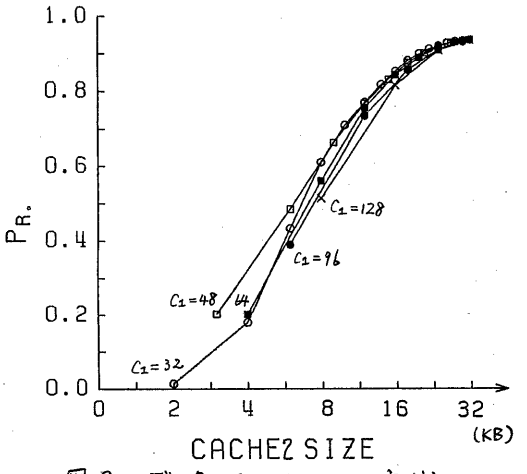

図9. データ・フェッチ(LRU方式)

に対するものである。

図2と3は，命令フェッチに関して，FIFO方式とLRU方式でのヒット率を示している。この結果，LRU方式の方が少し優れているもののその差は比較的少ないことが判った。

また，カラム数を大きくしてもヒット率は余り低下しないことが判った。即ち，キャッシュ1の大きさを増大する時，ローの数を増やすかわりに，カラム数を増やしてもよいことが判った。但し，最低のロー数は4とした。

図4と5は，データ・フェッチに関するヒット率を示している。命令の場合に比べて，ヒット率は高いが，置換え方式やカラム数に関しては命令の場合と同じことが言える。

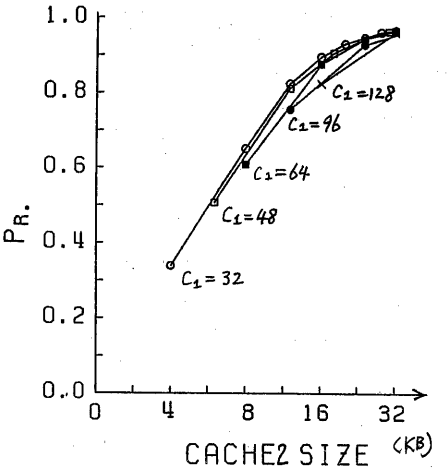

図10. データ・フェッチ（FIFO方式）
64バイト・ブロック

図6から9は，キャッシュ1のシミュレーションにおいて，最低のヒット率の場合（カラム数32，ロー数4，容量4KB）のキャッシュ2の結果である。いずれの場合も置換え規則は前述のようにLRU方式とした。図の表題に記されている方式は，キャッシュ1での置換え規則を示している。

キャッシュ1においては小容量でもかなりのヒット率が得られるのに対し，キャッシュ2においては，キャッシュ1の容量の倍で約0.5，4倍で約0.8になっている。しかし，倍以上の容量になると，命令フェッチとデータ・フェッチ，キャッシュ1でのLRUとFIFO方式，カラム数の違いには関係なく，比較的同じ様なヒット率が得られることが判った。このことは，キャッシュ1でのヒット率の違いにもかかわらずキャッシュ2では同じようなヒット率が得られることが明らかになった。

キャッシュ2の容量がキャッシュ1と同程度か少し大きい場合には，カラム数を変更することに効果が見られた。但し，整数倍になる場合は効果がなくて，1.5倍の48の場合に効果があった。しかし，そのような2のべき乗以外のカラム数にすると制御のための回路が複雑になり，結局は良くないであろう。

キャッシュ2のヒット率を高くし，比較的制御を簡単にするには，その容量をキャッシュ1の4又は8倍程度にし，カラム数を2又は4倍にしても良いことが判った。

ここでのシミュレーションでは，ブロックの大きさは32の固定にしたが，キャッシュ2でのブロックの大きさを64にした時の結果を図10に示す。ヒット率は，殆ど変らなかった。この結果，制御を簡単にするために，キャッシュ2のブロックの大きさを大きくしても良いことが判った。更にブロックの大きさを増大することは，転送の問題も現われてくることが予想されるため，シミュレーションは行なわなかった。

図6と8において，キャッシュ1と2のカラム数が同じ（32）の時，キャッシュ2のヒット率は，キャッシュ1と同じ容量，又は以下の時，必ず0になる。これは，FIFOとLRU方式との兼合いによる本質的な結果であることを注記しておく。

5. おわりに

ここでは，トレーサによって得られた命令とデータ参照のアドレス・データを入力とするトレース・ドリブン型のシミュレータを作成し，２レベルのキャッシュ・メモリを持つ三階層メモリ方式の効果について検討した。

その結果，キャッシュ１では，従来の研究と同じ様に，少ない容量で高いヒット率の得られることが判った．キャッシュ１でのヒット率が比較的悪い場合を取り上げ，キャッシュ２のシミュレーションを行った。キャッシュ２でのヒット率を0.9程度にするには，キャッシュ２の容量をキャッシュ１の４倍から８倍にする必要のあることが判った。また，カラム数を大きくしたり，ブロックの大きさを倍にしてもヒット率は殆ど変化しないことが判った。更に，キャッシュ２のヒット率は，キャッシュ１のヒット率や，置換え規則に関係なく同じような値になることが判った。

シミュレーションの実行に際しては，文献(2)での結果からコンパイラの動作も実行時の動作も余り変らないことが予想されるため，大きなプログラムの実行時の動作を解析しなかったが，確認しておく必要があるだろう。

記憶素子の集積度が上がり，１チップでキャッシュ・メモリを実現するようになってきて，その効果も確かめられてきた[7]。このような方式が大型の高速計算機に実現された時は，２レベルのキャッシュ・メモリも効果があると十分予測できる。

この研究の一部は、京都大学大型計算機センターの開発計画のもとで行った。この研究に対してご助言をいただいた京都大学工学部萩原宏教授と，ご理解をいただいている京都大学大型計算機センター長丹羽義次教授に心から感謝します。筆者が文部省在外研究員としてカリフォルニア大学滞在中に助言をいただいた D. Ferrari 教授と A.J. Smith 教授に感謝します。また トレーサを作成していただいた京都大学大型計算機センター 河野典技官に謝意を表します。

〔参考文献〕

(1) Liptay, J.S., Structual Aspects of the System/360 Model 85 II The Cache, IBM Syst. J. 7, 1 (1968), 15-21.

(2) 中村，北川，金澤，萩原，バッファ・メモリ方式のシミュレーション，情報処理15, 1 (1974), 26-33.

(3) Smith, A.J., Cache Memories, Computing Surveys, 14, 3 (1982) 473-530.

(4) 金澤，北川，萩原，三輪，結合ミックスによるCPUの性能評価，情報処理14, 9 (1973), 715-718.

(5) Chow, C.K., On Optimization of Storage Hierarchies, IBM J. Res. Develop. 18, 3 (1974), 194-203.

(6) 金澤，北川，萩原，バッファ・メモリを有するパイプライン制御計算機の性能評価について，情報処理学会論文誌 22, 6 (1981), 527-534.

(7) Hill, M.D., A.J. Smith, Experimental Evaluation of On-Chip Microprocessor Cache Memories, 11th Annual Symp. on Computer Architecture (SIGARCH Newsletter 12, 3) (1984), 158-166.

(8) 富士通： FACOM Mシリーズ ハードウェア機能説明書 I (命令編)，マニュアル.

(9) 富士通： FACOM M-380/382 ハードウェア解説書，マニュアル.

付録

付表1. 命令の一覧

上4ビット / 下4ビット	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		LPR	LPDR	LPER	STH	ST	STD	STE		STM						SRP*
1		LNR	LNDR	LNER	LA	FLM^{4}				TM^{1}	AI^{1}			MVN*		MVO*
2		LTR	LTDR	LTER	STC			CEI		MVI^{1}	STEXT^{4}			MVC*		PACK*
3		LCR	LCDR	LCER	IC^{1}			CDI		TS^{1}	SVC			MVZ		UNPK*
4		NR	HDR	HER	EX	N^{4}		CIE^{4}		NI^{1}				NC**		
5	BALR	CLR	LRDR	LRER	BAL	CL^{4}		CID^{4}		CLI^{1}				CLC**		
6	BCTR	OR	MXR	AXR	BCT	O^{4}			BXH	OI^{1}				OC**		
7	BCR	XR	MXDR	SXR	BC	X^{4}	MXD^{8}		BXLE	LM*				XC**		
8		LR	LDR	LER	LH^{2}	L^{4}	LD^{8}	LE^{4}	SRL							EAP*
9		CR	CDR	CER	CH^{2}	C^{4}	CD^{8}	CE^{4}	SLL							CP**
10	SVC	AR	ADR	AER	AH^{2}	A^{4}	AD^{8}	AE^{4}	SRA			CS^{4}				AP**
11		SR	SDR	SER	SH^{2}	S^{4}	SD^{8}	SE^{4}	SLA			CDS^{8}				SP**
12		MR	MDR	MER	MH^{2}	M^{4}	MD^{8}	ME^{4}	SRDL					TR**		MP**
13		DR	DDR	DER		D^{4}	DD^{8}	DE^{4}	SLDL			CLM*		TRT**		DP**
14	MVCL*	ALR	AWR	AUR	CVD	AL^{4}	AW^{8}	AU^{4}	SRDA			STCM*		ED**		
15	CLCL**	SLR	SWR	SUR	CVB^{8}	SL^{4}	SW^{8}	SU^{4}	SLDA		MC	ICM*		EDMK**		
命令長	2バイト				4バイト								6バイト			

注. 命令コードの右肩の数字は読出すべきデータの長さを示す。* 印は命令毎に指定される。** 印は命令毎に指定される読出しデータが2つあることを示す。
データのストアに関しては記していない。
命令の内容については文献(8)を参照せよ。

付表2. 命令の出現頻度 (%)

上4ビット / 下4ビット	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.0	0.05	0.01	0.44	0.47	9.05	0.09	3.21	0.0	0.56	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.00	0.0	0.0	3.76	0.0	0.0	0.0	0.0	1.45	0.00	0.0	0.0	0.0	0.0	0.0
2	0.0	2.50	0.00	0.03	1.02	0.0	0.0	0.01	0.0	0.69	0.0	0.0	0.0	0.14	0.0	0.00
3	0.0	0.02	0.0	0.88	2.55	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.04
4	0.0	0.55	0.0	0.10	0.12	0.35	0.0	0.01	0.0	0.10	0.0	0.0	0.0	0.0	0.0	0.0
5	0.50	0.09	0.0	0.06	0.13	0.07	0.0	0.0	0.0	0.44	0.0	0.0	0.0	0.02	0.0	0.0
6	1.02	0.08	0.0	0.0	2.50	0.09	0.0	0.0	0.0	0.13	0.0	0.0	0.0	0.02	0.0	0.0
7	2.36	0.00	0.0	0.0	12.60	0.01	0.0	0.0	0.01	0.00	0.0	0.0	0.0	0.01	0.0	0.0
8	0.0	3.29	0.01	0.41	0.79	18.24	0.11	3.71	0.42	0.78	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	1.62	0.0	0.12	0.13	4.04	0.0	0.31	1.24	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.01	4.17	0.05	0.59	0.10	2.73	0.02	0.98	0.02	0.0	0.0	0.00	0.0	0.0	0.0	0.0
11	0.0	1.77	0.03	0.77	0.02	0.31	0.0	0.07	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.06	0.00	0.42	0.00	0.14	0.04	2.13	0.10	0.0	0.0	0.0	0.0	0.00	0.0	0.0
13	0.0	0.01	0.0	0.07	0.0	0.03	0.0	0.71	0.04	0.0	0.0	0.01	0.0	0.03	0.0	0.0
14	0.00	0.12	0.0	0.0	0.04	0.03	0.04	0.02	0.03	0.0	0.0	0.10	0.0	0.0	0.0	0.0
15	0.0	0.49	0.0	0.02	0.00	0.07	0.0	0.02	0.0	0.0	0.01	0.01	0.0	0.0	0.0	0.0

注. 0.0 は真に0, 0.00 は 0.005未満を示す。