数理解析研究所講究録
第155巻 1972年 56-73

56

東大HITAC8800/8700システム と長時間バックグラウンド・ジョブ

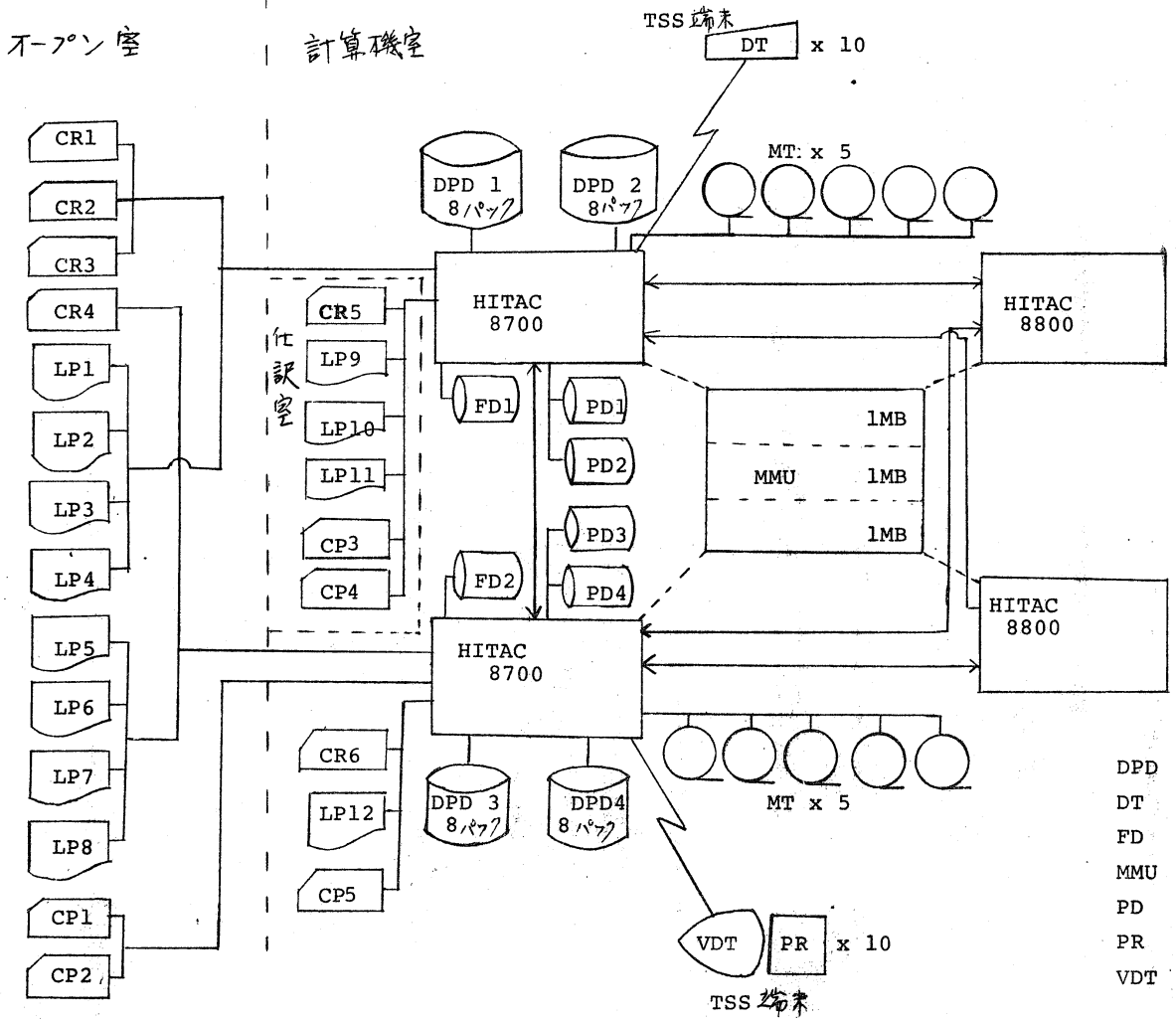
東大　大型計算機センター　　石田　晴久

1. はじめに

東大大型計算機センターでは，1973年1月より図1に示すような超大型電子計算機システムが本格稼動に入ることになっている．このシステムの主な特徴は次の通りである．

(1) 高速である．平均命令実行時間はHITAC8700で160 nsec程度，HITAC8800で700 nsec程度と予想される．したがってH8800はH5020Eより約10倍速いと期待される．しかし各命令の実行時間は，パイプライン制御・バッファーメモリ・アドレス変換（連想レジスタ使用の有無）・コア位置・マルチプロセッシング効果などに大幅に左右されるので，従来の計算機に比して演算速度の推定が困難である．

(2) 主メモリが大きい．コアメモリの大きさは3MB（メガバイト），すなわち768kW（1語32ビット）ある．ただしこのうち管理プログラムが700KB＝175kW程度使うの



DPD　集団ディスク駆動装置
DT　データタイプライタ
FD　磁気ドラム記憶装置
MMU　主記憶装置
PD　高速磁気ドラム記憶装置
PR　プリンタ
VDT　ビデオデータ端末装置

図1　東大 HITAC 8800/8700 システム

58

でユーザ・エリアは2.3MB＝593kw程度になる．また将来リモートバッチ端末が増えるとユーザ・エリアはさらにへることになる．この主メモリではH8800に対しては2MB分が8way（転送速度1.25μsec/64B），1MB分が4way（2.5μsec/64B），またH8700ではすべて4wayのインターリーブが行われる．

(3) 仮想メモリ機構がある．このため，自動オーバーレイ・ダイナミックリンク・リエントラントプログラミング・ページ（1kw）あるいはセグメント（64kw）単位のアクセス制御・非連続領域の有効利用・スワッピング容量などの利点が出てくるが，一方これらは効率低下の原因となる要素もたくさん含んでいる．

(4) バッファー・メモリがある．これは各CPU毎にコアとCPUの中間にあって，当面必要な命令やデータを保持しておいてCPUとメモリのアクセス能率を上げるのに使われる．

	各H8800	各8700
容量	32KB＝8kw	16KB＝4kw
データ幅	8B×2＝4w	8B＝2w
サイクル・タイム	100 nsec	210 nsec

(5) 4台のCPUによるマルチプロセッシング．4台のCPUが全部演算命令を実行している状態では，システム全体として1命令を平均66nsec程度で処理することになる．この4台のCPUの他に4台の入出力処理装置がコアを共用する．こ

のためコアやチャネルでのぶつかり合いなど効率低下の要因もあるが，コアその他のリソースの有効利用が期待でき，またオペレータなど運転要員の数もへらせると期待される．

(6) 補助メモリの容量が大きい．

	台数	容量/台	平均アクセス
ページング・ドラム	4	4.3 MB	10.3 msec
ファイル・ドラム	2	4.5 MB	10.3 msec
集団ディスク	4	233 MB	87.5 msec

このディスクは1974年中に容量800 MBのもの2台と入換える予定になっている．

(7) 機械語はIBM 360のスーパーセットになっている．IBM 360からの拡張部分はほとんど管理プログラムで使われる機能であるから，IBM 360用のユーザプログラムはほとんどそのままかかる．

(8) RAS (Reliability, Availability, Serviceability) を支える機能がある．これには命令の再実行（最高15回まで）・2検出・1訂正の誤り訂正コード（情報64ビット＋チェック8ビット）の使用などが含まれる．また自動再構成（故障したCPU・コアメモリ・入出力処理装置などの切離し）や故障個所指摘テスト (FLT = Fault Locating Test)，エラー情報（最大4kB）の自動ログアウトの機能などもある．

60

(9) オペレーティング・システムが汎用になっている．このためバッチ処理・リモートバッチ処理・TSS などのサービス，個人ファイルの使用ができるようになっている．言語には FORTRAN (compiler/converter/syntax checker), PL/I, BASIC, COBOL などがある．

2. バックグランド・ジョブ

東大システムでは4台の高速の CPU が 768 kw のコア・メモリを共用する形になっているため，次のような マルチプログラミングが行われることになっている．

	タスクの種類	多重度		コア容量
システム・タスク	ジョブ・スケジューラ	1	管理プログラム	0.43MB
	マスタ・スケジューラ	4		
	会話イニシェータ	1		
	入力リーダ (CR 6台)	6	リーダ・ライタ	0.20 MB
	出力ライタ (LP 12台, CP 5台)	17		
ユーザ・タスク	バッチ・タスク	14	各 140KB=35 kw	1.96 MB
	TSS タスク	2	各 100KB (ひとつはスワップ中)	0.30 MB
	(多重度設計	45)	(計	2.89 MB)

ユーザのジョブはいちどシステムの中で実行状態になると (タスクになると)，バッチ処理の場合，図2のように，ディ

ク（またはファイル・ドラム）内の入力ジョブキュー，出力ジョブキュー，ディスク内の入力情報・出力情報，スワッピング・ドラムなどを使って実行されることになる．

この場合，CPUの演算速度が4台で66 msec/ステップ程度であるのに比べて，ドラムやディスクの平均アクセス時間が10 msecないし90 msecと長いため，すべてのユーザ・タスクが入出力待ち（ドラムやディスクへのアクセス待ち）になって，CPU（とくに演算担当のH8800）が遊んでしまう状態が起こりうる．そこでCPUの遊休時間を利用して処理させようというのがここでいうバックグラウンド・ジョブである．

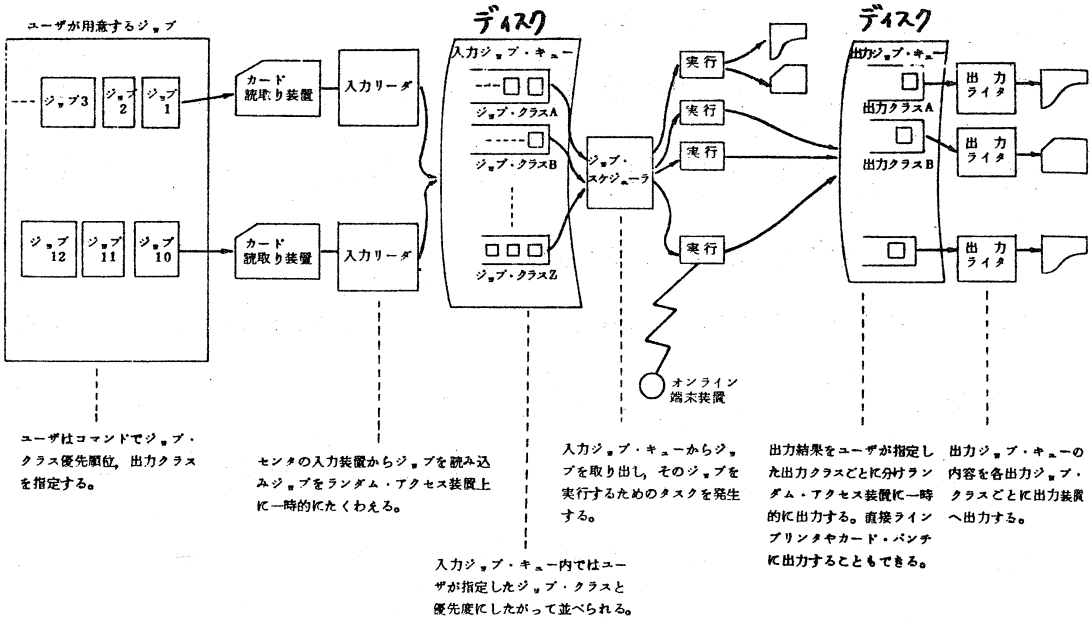


図2　バッチ処理の流れ

62

この目的には，バックグランド・ジョブ用にひとつのジョブクラスを割当て，そのクラスのジョブで発生するタスクの優先順序をうんと低くしておけばよい．バックグランド・タスクの優先順位がずっと低ければ，このタスクは普通はほとんど実行されることがなく，他のクラスのタスクがすべて入出力待ちで実行されなくなって始めて実行されることになる．

このタスクの優先順位を指定するプログラムは，センター・オウン・コーディングによる"タスク優先順位決定モジュール"として，東大センターで管理プログラムに組込む予定である．このモジュールで決定する項目は次の通りである．

(i) 当のジョブを処理するユーザ・タスクのタイム・スライス値（msec 単位）

(ii) 上記タスクの実行優先権（0～255，大きいほど高い）

(iii) 上限優先権（CHAP = CHange Priority マクロで変更できる実行優先権の上限）

バックグランド・ジョブ用のジョブ・クラスについては次のような項目を指定しなければならない．

(a) ラインプリンタ出力のページ数とライン数の上限

(b) カードパンチ出力上限（0にしてよいであろう）

(c) 仮想メモリ上限（1kw のページ単位）＝プログラムの全体的な大きさの上限

(d) 1時に使用できるコアメモリ上限（はみ出した部分はドラムから自動的にオーバレイされる）

(e) CPU使用時間上限（単位 10 msec）

(f) 経過時間（コア時間ともいう）（単位 10 msec）

(g) 取出し比率 R_i（＝0～32767）．X_iをクラスiからすでに取出されたジョブの個数とすると，システムは X_i/R_i を計算し，その小さい順に実行してゆく．たとえばクラスがA, B, C とあって，$R_A : R_B : R_C = 3 : 2 : 1$ ならABCABAの順に取出される．この他に東大センタでは，R_iの低いクラスがたまたまあいているために，そのクラスのジョブが取出されるということを防ぐために，各ジョブの入力時刻も考慮してスケジュリングを行う予定になっている．

バックグランドジョブとしては

(A) プログラムが比較的小さいこと。コア内にあっても余りじゃまにならず，スワップも起きない程度のものがよい．たとえばコアの1割程度の65kWぐらい．

(B) 入出力のデータ量が少いこと．またなるべくまとめて入出力すること．入出力の回数が多いと，ディスク待ちになってバックグランド・ジョブでなくなってしまう．

(C) 計算時間は非常に長いものでよい．たとえば10時間．た

61

だしCPUの空き時間にのみ走るから，ターンアラウンド時間は相当長くなると考えられる．東大の現システムHITAC 5020Eでの実測によるとCPUの稼動率（busy rate）は平均して67%，すなわち平均してCPUは33%の時間は遊んでいたという結果が出ている．次期システムのマルチプログラミングでこの率がどうなるかは非常に興味深い．

(D) 1日の終りにシステムをシャットダウンするときに，ジョブが終了していないときに，強制的にそれを中断させて，翌日それを自動的に再開させることはできない．代りの方法としては，SINF（Set INForm）マクロを発行して，それに応えてオペレータからINFORMコマンドを出してもらって止めることはできる．この場合翌日も続行させるときは，オペレータに起動させてもらう．

(E) 中間結果をみるのが難しい．もっとも簡単なのはプログラムの中からWTO（Write To Operand）コマンドでオペレータ用文字ディスプレイに出すことであるが，それをユーザがみるのは事実上難しい．ラインプリンタへの出力はジョブが終了して出力ファイルが閉じられてからのみである．可能な方法としては，中間結果をある時点で自分の個人用ファイルに書出し，アセンブラ・サブルーチンでそのファイルを閉じておいて，別のジョブでそのファイルを取出す方法がある．

別のジョブにしなければならないのは，異る二つのジョブが同時に同じファイルをアクセス（読出しだけでも）できないように管理プログラムが作られているためである．この中間結果を取出すジョブ用には，もうひとつのジョブ・クラスが必要かもしれない．

中間結果をみるにはTSSモードを使うことも考えられるが，実際には動作中のBATCHジョブにTSSで手を出すことは難しい．（TSS→BATCHは起動の指定だから可能だが，BATCH→TSSはすでに動いているタスクに対するもので，両方の仮想空間が全く異るので難しい．原理的には共通セグメントが設けられれば可能であるが）

(F) バックグランド・ジョブのユーザーは1人のグループ管理者のもとで，ファイル使用量などを管理することもできる．

3．プログラミング上の注意

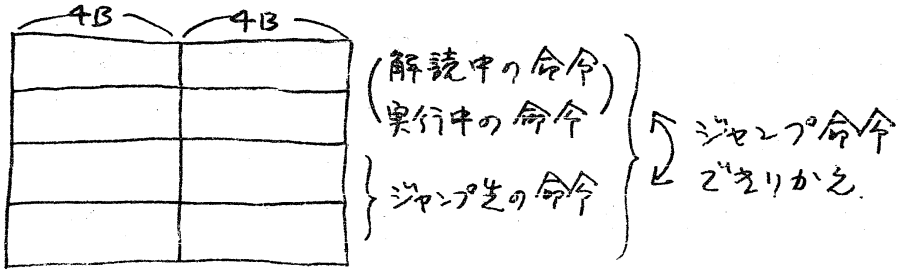
東大の新システムでは，マルチ・プロセッサのぶつかり合い，仮想メモリでのスワッピング，パイプライン制御の乱れ，バッファー・メモリ内の命令の有無など，効率低下の要因がいくつかあるので，それらを避ける注意が必要である．

(a) バッファー・メモリの活用をはかるには，1度に使うデ

66

タとプログラム・ループがなるべく32kB(8kW)に收まるようにする。こうすればプログラムは(データの書込みを除いて)バッファー・メモリの中だけで実行されることになるので,計算速度が向上する。また書込みが行われるときは,バッファーの内容とコアの内容との一致を常にはかるため,コアへも書込まれるが,バッファー・メモリに取込まれている部分へ書込まれるのであれば,連想レジスタにより実アドレスがすぐ分るので,書込みはそれだけ早い。これは図3の太線の流れに相当する。

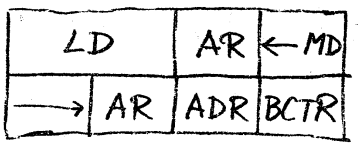
(B) パイプライン制御の効果を生かす。H8800では命令のバッファーが次のように32バイト分ある。そこで行列の積



の計算に出てくる

$$S(I,K) = S(I,J) + A(I,J) \cdot B(J,K)$$

の機械語列の場合のように,



16バイト以内のループなら非常に速い（上の例は15サイクル×50 nsec = 0.75 μsec ループである）ことは頭に入れておくべきである．この場合，先行制御が全く行われていないとすると，LD (8サイクル)，AR (3)，MD (13)，ADR (16)，BCTR (6) で，上のループは41サイクル×50 nsec = 2.05 μsec かかることになる．ただし以上は命令がバッファ・メモリ内にあるとしての話で，もしもバッファ・メモリになけ

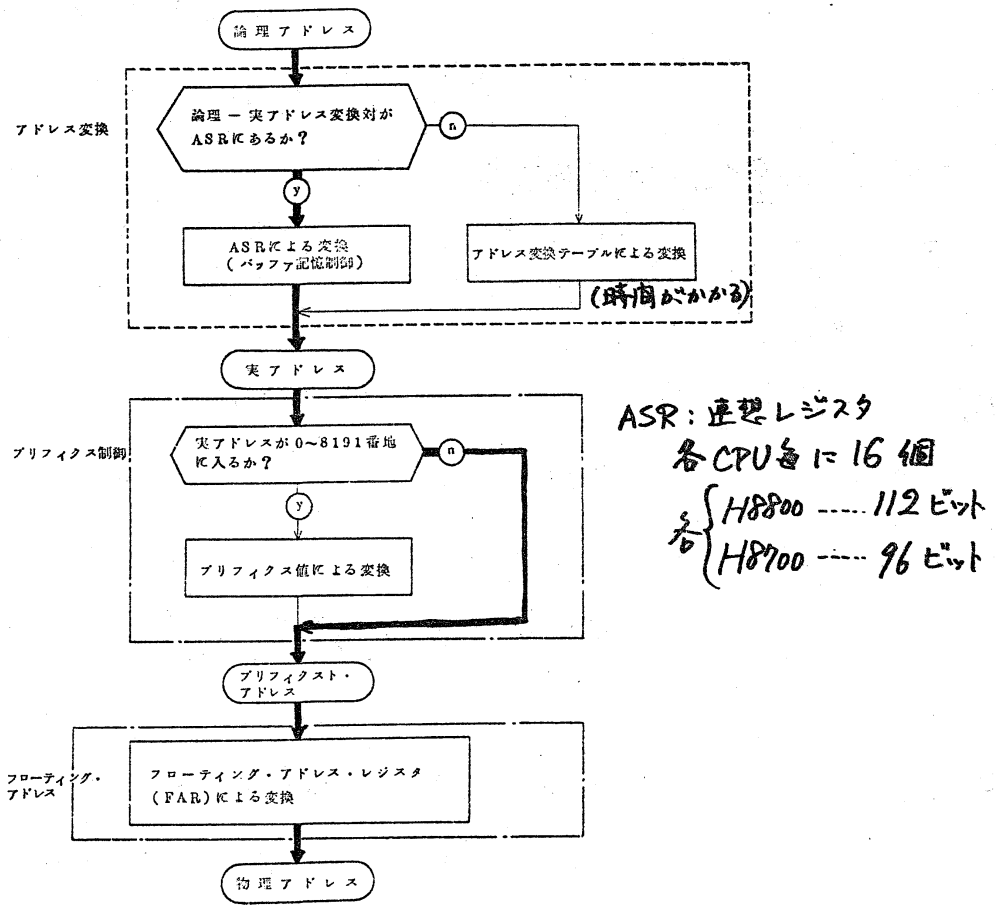


図3　アドレス変換過程

68

ルば，アドレス変換やコアからバッファへの転送などできらに時間がかかる．

(C) H8800/8700 の FORTRAN（およびハードウエア）で扱える数の範囲は次の通りである．

IH)	INTEGER*2	-2^{15}～$2^{15}-1$	−32,768 ～ 32,767	(4桁)
I)	〃 *4	-2^{31}～$2^{31}-1$	−2,147,483,648 ～2,147,483,647	(9桁)
ID)	〃 *8	-2^{63}～$2^{63}-1$	～9,223,372,036,854,775,807	(18桁)
	［負整数は2の補数表示］			
R)	REAL*4	16^{-64}～16^{63}	10^{-78}～10^{75}	約7.2桁
D)	*8	〃	〃	16.8桁
Q)	*16	〃	〃	33.6桁

［負実数は符号＋絶対値表示］

実数の内部表現，例とあげると次のようになる．

	符号	指数部 (excess 64)	仮数部 (24/56/112 ビット)
無限大	0	1111111	.111111 ------ 11111
無限小（非正規化）	0	0000000	.000000 ------ 00001
〃 （正規化）	0	0000000	.000100 ----- 00000
真のゼロ	0	0000000	.000000 ---- 00000

(注) $2^{-24} = 0.596 \times 10^{-7}$

$2^{-56} = 0.138 \times 10^{-16}$

$2^{-112} = 0.192 \times 10^{-33}$

ハードウエアは次のような4倍長演算機能ももっている.

Q ± Q ⟶ Q　normalized

D * D ⟶ Q　〃

Q * Q ⟶ Q　〃

ただしQのわりざんとIDの±*/はソフトで行われる.

(a) 演算に関連したわりこみには次のものがある.

	マスク	実行	結果 FORTRAN(ハード)	FORTRANによる検出	コンディション・コード C.C
整数オーバーフロー(*)	○	○	-----	-----	3
〃 零除数	×	×	メモリ不変	CALL DVCHK(i)	不変
実数・零除数	×	×	メモリ不変	〃	不変
〃 指数オーバーフロー	×	○	∞ (f正規化 e=e-128)	CALL OVFRL(i)	仕セット *1 不変
〃 〃 アンダーフロー	○	○	0 (f正規化 e=e+128)	------	3
〃 零仮数	○	○	-----	------	0

(*) ひきざんは, I−Jの場合

I + 1 + (Jの1の補数) = 答の形で行われる.

したがって

$0-(-\infty) = 0-(100\cdots00) = 0+1+(011\cdots11)$ ⟶ オーバーフロー

$(-\infty)-(-\infty) = 100\cdots00+1+011\cdots11 = 0$ ⟶ オーバーフローなし

こうしたわりこみが起こったときの処置はFORTRANでは,

70

CALL ERSET (IE, IN, ……)

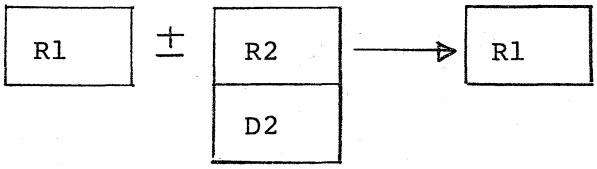
で指定できる．ただしIEはエラー番号，INは打切り回数を表わす．バックグランド・ジョブではエラー処理については万全の配慮が必要である．

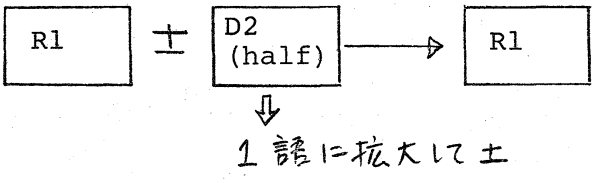
参考文献

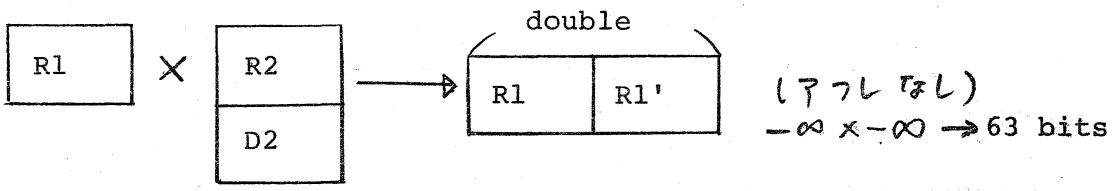
1. 中沢ほか："超高性能電子計算機のシステム設計"ほか4件，情報処理学会大会予稿集，pp. 359－368 (1970)
2. "超大型機特集号"，情報処理学会誌，Vol. 12, No. 8 (1971)
3. P. J. Denning："Virtual Memory", Computing Surveys, Vol. 2, No. 3, pp. 153－189 (September 1970)
4. "HITAC 8700 機能説明書"，日立製作所 (1971)
5. "HITAC 8700 処理装置（拡張モード）"，日立 (1971)
6. "HITAC 8700 管理プログラム概説"，日立 (1971)

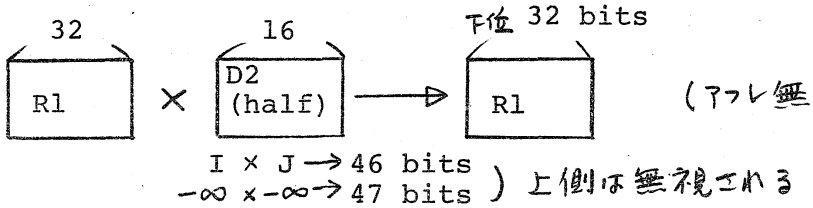
附錄　演算の型　71

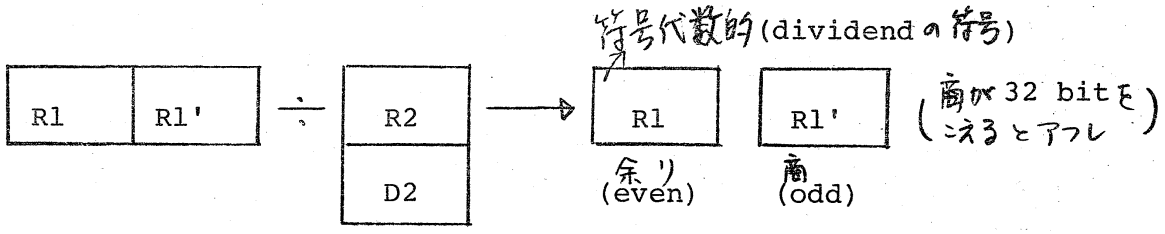
(1) 整数演算





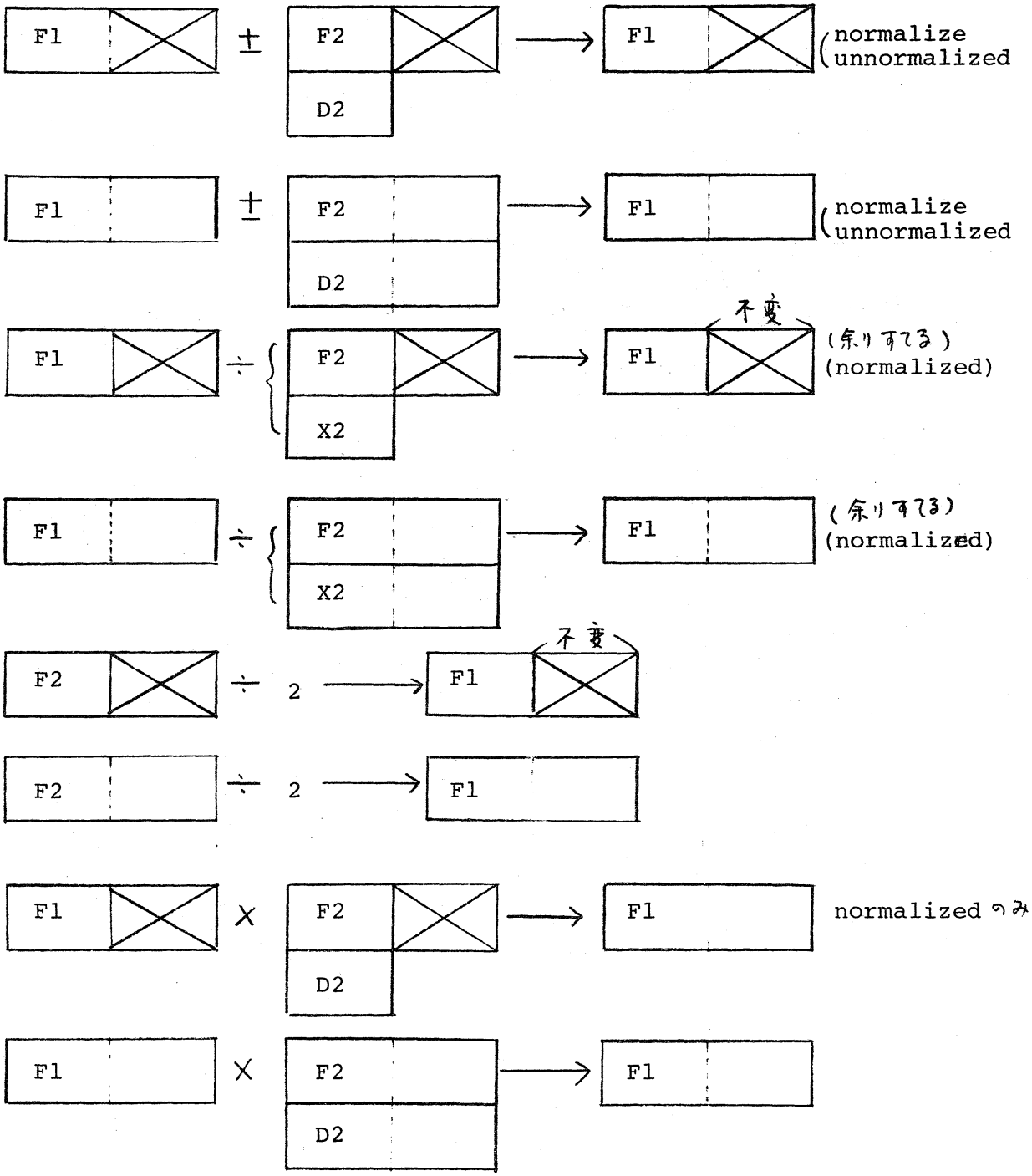






72

(2) 浮動小数点演算


F1
±
F2
D2
F1
(normalize
unnormalized
F1
±
F2
D2
F1
(normalize
unnormalized
不変
F1
÷
F2
X2
F1
(余りすてる)
(normalized)
F1
÷
F2
X2
F1
(余りすてる)
(normalized)
不変
F2
÷
2
F1
F2
÷
2
F1
F1
X
F2
D2
F1
normalized のみ
F1
X
F2
D2
F1

(3) 4倍長演算