VPP500スカラプロセッサの性能

中島康彦† 大野優人† 竹部好正†

VPP500スカラプロセッサはハードウェア量を抑えつつ複数操作の並列実行を実現するために，3段パイプラインを基本とするLIW方式を採用した．本稿では，VPP500スカラプロセッサの概要および性能測定項目について述べ，次にSPECfp92を用いた測定結果に基づき，VP2600スカラプロセッサと比較しながらVPP500スカラプロセッサについて考察を行う．最後に，浮動小数点演算に要するサイクル数や，キャッシュ容量，ウェイ数を改善することにより，さらに高性能を引き出せることを明らかにする．

Scalar Processor of the VPP500 Parallel Supercomputer

YASUHIKO NAKASHIMA,† YUTO OHNO† and YOSHIMASA TAKEBE†

This paper describes the scalar processor of the VPP500. The 64-bit long instruction word (LIW) architecture allow the issue of up to three operations every clock cycle. The LIW architecture and the three stage pipeline reduce the hardware cost and branch penalty. We summarize the features of the scalar processor, show the way of performance evaluation, show the results of performance measurement using SPECfp92 benchmark program, and consider about the performance characteristics of the scalar of the VPP500 as compared to the scalar of the VP2600. Finally we show that the shorter cycles of the floating-point operation and the larger multiple-way cache efficiently increase the performance.

1. はじめに

VPシリーズベクトル計算機[1]のスカラプロセッサ（以下，スカラと略す）は，Mシリーズ計算機と同じアーキテクチャを採用してきた．しかし，このアーキテクチャは，命令実行順序や割り込み発生時の命令アドレス（Program Counter）の保証を規定しており，命令の並列実行や追い越しなどインプリメントの工夫による高速化を妨げている．VPP500[2)~4)]のスカラでは，このような制約を緩和し，ソフトウェアに内在する命令の並列性をハードウェアの並列処理に効果的に写象することのできる，以下のようなアーキテクチャを採用した．

(1) 最大3操作を同時発行可能とする64ビット固定長の長形式命令語（Long Instruction Word）方式を採用した．スーパスカラ方式に必要な命令スケジューリング機構を不要とし，また超長形式命令語（Very Long Instruction Word）方式の短所である命令サイズ増大を抑えた．

(2) 固定小数点乗算操作，浮動小数点演算操作，主記憶参照操作およびベクトル操作の非同期実行を可能とした．

(3) 条件分岐操作における分岐先アドレス指定をPC相対アドレスのみとすることにより，分岐先アドレス計算および分岐先命令プリフェッチの高速化を可能とした．

2章ではVPP500スカラの概要について，3章では性能測定項目について述べる．割り込み動作などアーキテクチャの詳細については文献5)または6)を参照されたい．4章ではSPECfp92を用いた測定結果を示し，5章では測定結果に基づき，VP2600スカラと比較しながらVPP500スカラの特性について考察を行う．

2. VPP500スカラプロセッサの概要

2.1 構成の概要

VPP500スカラは図1のように，IU，MU，FU，AUと呼ぶ独立に動作するユニットから構成される．

IUは，32本の32ビット汎用レジスタ，2個のALUおよび1個のバレルシフタを有し，最大2個の演算を毎サイクル実行する．

† 富士通株式会社
Fujitsu Limited

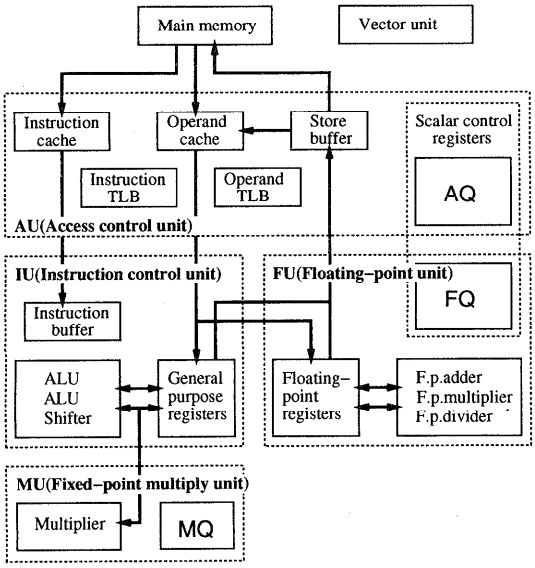


図1 スカラプロセッサの構成
Fig. 1 Scalar processor block diagram.

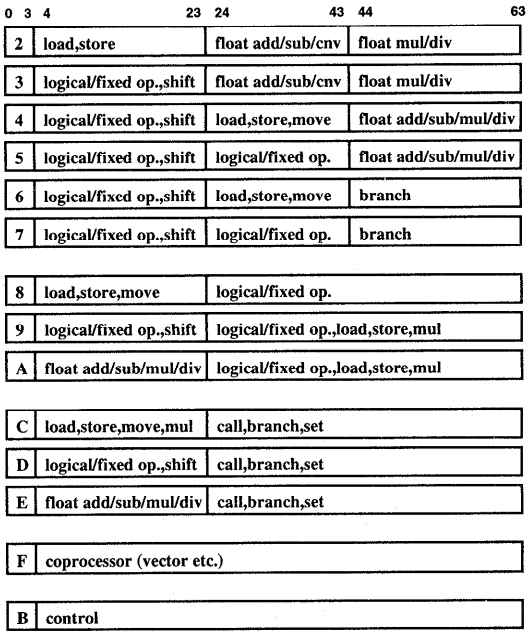


図2 命令の形式
Fig. 2 Instruction formats.

MUは，最大1個の乗算の非同期実行を可能とするキュー（MQ）および乗算器からなる．乗算器はパイプライン化されておらず，実行中の乗算が終了するまで次の乗算が待たされる．

FUは，32本の64ビット浮動小数点レジスタ，加減算操作と乗除算操作の組を最大6組まで非同期実行可能とするキュー（FQ），各1個の加減算器（比較，型変換も行う），乗算器，除算器からなる．加減算と乗算は各1操作を毎サイクル発行可能である．除算はパイプライン化されておらず，実行中の除算が終了するまで次の除算が待たされる．依存関係がない場合，加減乗算は除算を追い越すことができる．

AUは，最大2個の主記憶参照の非同期実行を可能とするキュー（AQ），4エントリの8バイトストアバッファ，命令/オペランドTLB，命令/オペランドキャッシュからなる．オペランドキャッシュから，連続する2本の汎用レジスタまたは2本の浮動小数点レジスタへのロード操作を毎サイクル発行可能である．依存関係がない場合，後続のロード/ストアが先行するロード/ストアを追い越すことができる．各TLBにはダイレクトマップ方式を採用している．エントリ数は各々256，ページサイズは32Kバイトである．65536個のプロセス各々に対して4Gバイトの仮想アドレス空間を提供する動的アドレス変換機構を装備しており，多重仮想アドレス空間を実現している．各キャッシュにはライトスルー方式およびダイレクトマップ方式による物理キャッシュを採用している．容量は各々32Kバイト，ラインサイズは64バイトである．

2.2 命令形式の概要

図2に命令の形式，図3に操作の形式を示す．命令の先頭4ビットが命令の形式を決定する．図3の(*1),(*2)に示すように，ロード，ストア，論理演算，乗算操作に使用できる即値は使用頻度の高い25ビット即値のみとし，20ビット長操作の5ビットオペコードをやりくりしている．(*3)に示すように，条件分岐操作およびcall操作では，PC相対アドレスにより分岐先アドレスを指定する．条件分岐には，2組の固定小数点条件コードレジスタ（各7ビット），浮動小数点条件コードレジスタ（7ビット）の中から1ビットを条件として指定する．

2.3 非同期実行の概要

VPP500スカラは，先行する非同期操作（実行に複数サイクルを要する，固定小数点乗算操作，浮動小数点演算操作，主記憶参照操作およびベクトル操作）の実行終了を待たずに後続命令を毎サイクル発行可能とする非同期実行機構を有する．浮動小数点演算操作は操作をFQにキューイングする同期実行部分と，演算および結果を格納する非同期実行部分に分けて実行される．また，その他の非同期操作は入力オペランドを読み出す同期実行部分と，結果を格納する非同期実行部分に分けて実行される．同期操作（固定小数点演算操作および分岐操作）には非同期実行部分がない．同

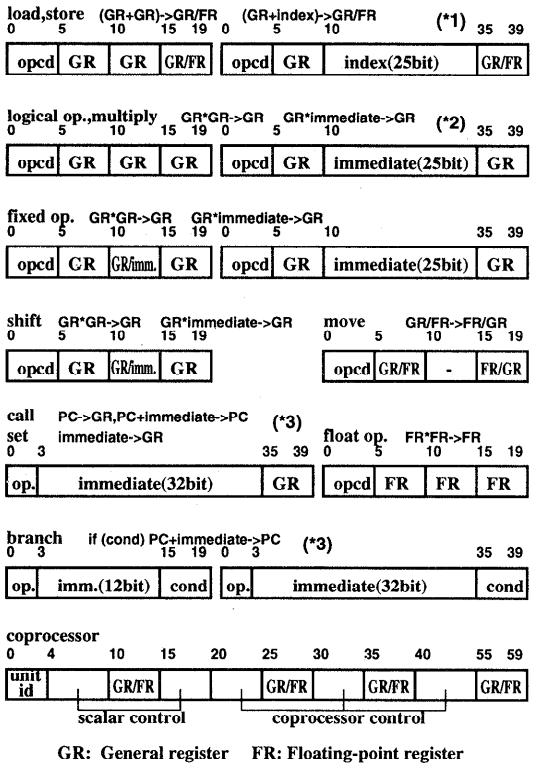


図 3 操作の形式
Fig. 3 Operation formats.

期操作および同期実行部分を IU が，非同期実行部分を他ユニットが各々担当し，固定小数点演算操作，分岐操作，固定小数点乗算操作，浮動小数点演算操作，主記憶参照操作およびベクトル操作を並列実行する．

異なる命令に属する操作の間に依存関係がある場合は，ハードウェアが操作の実行を待ち合わせる．たとえば，先行するロードの結果を後続の浮動小数点演算操作が使う場合は FU が AU を待ち，使わない場合は待たずに浮動小数点演算を開始する．先行操作の非同期実行部分が完了するまで，結果を使う操作を発行しないよう，コンパイラが命令スケジューリングを行うことにより，非同期操作の見かけの実行サイクル数を 1 とすることができ，命令列を毎サイクル発行することが可能となる．

3. 性能測定項目

1) 操作効率およびキャッシュヒット率，2) 操作頻度，3) パイプラインインタロック率を性能測定項目とする．操作効率から LIW 方式の有効性を概観し，キャッシュヒット率，操作頻度およびパイプラインインタロック率から，プログラムごとの VPP500 スカラの挙動を把握する．

3.1 操作効率およびキャッシュヒット率

1 命令を発行するのに要する平均サイクル数を CPI（Cycle Per Instruction），1 命令あたりの平均操作数を OPI（Operation Per Instruction），1 操作を発行するのに要する平均サイクル数を CPO=CPI/OPI（Cycle Per Operation）と定義する．CPI は 1 以上であり，パイプラインインタロックが少ないほど理想値 1 に近づく．1 命令語により最大 3 操作を発行可能であることから，OPI は 1 以上 3 以下であり，1 命令に NOP（No Operation）以外の操作数が多いほど理想値 3 に近づく．CPO は 1/3 以上であり，パイプラインインタロックが少なくかつ命令あたりの操作数が多いほど理想値 1/3 に近づく．

また，毎サイクル 2 個の浮動小数点演算を発行できることに対して，実際に発行した浮動小数点演算の割合を FOR（Floating Operations Ratio）と定義する．平均 5 サイクルに 2 個の浮動小数点演算を発行する場合，FOR は 2 個/(5 サイクル×毎サイクル 2 個)=20%である．オペランドキャッシュのヒット率は OP$，命令キャッシュのヒット率は IF$ と定義する．

3.2 操作頻度

実行した操作の出現頻度を各々，fadd/fsub（浮動小数点加減算），fcmp（同比較），fcnv（同型変換），fmul（同乗算），fdiv（同除算），load（ロード），store（ストア），etc.（固定小数点演算および分岐）と定義する．

3.3 IU に関するインタロック率（D.IF）

命令フェッチパイプラインは，キャッシュ参照権を獲得する P ステージ，TLB およびキャッシュを参照する C ステージからなる．また，命令パイプラインは，命令デコードおよびレジスタ読み出しを行う D ステージ，演算を行う E ステージ，結果をレジスタに格納し，条件コード（Condition Code）および PC を更新する W ステージからなる．各 2 命令分の現命令，次命令および分岐先命令バッファに対し，次命令または分岐先命令を 128 ビット境界から毎サイクル最大 2 命令プリフェッチする．

図 4 に，命令 i1 が CC を生成し，次命令 i2 中の条件分岐操作により，次命令 i3 または分岐先命令 t5 が実行される様子を示す．Stage0 では 2 命令 i1,2 のフェッチ Ci1,2 を行う．Stage1 では命令 i1 のデコード Di1，次 2 命令 i3,4 のプリフェッチ Ci3,4，命令 i2 中に分岐操作を仮定した分岐先アドレス計算 Pt5,6（分岐先アドレスが PC 相対アドレスであることから PC と相対アドレスをつねに加算しておく）を行う．Stage2 では命令 i1 の演算および CC 生成 Ei1，分岐先命令

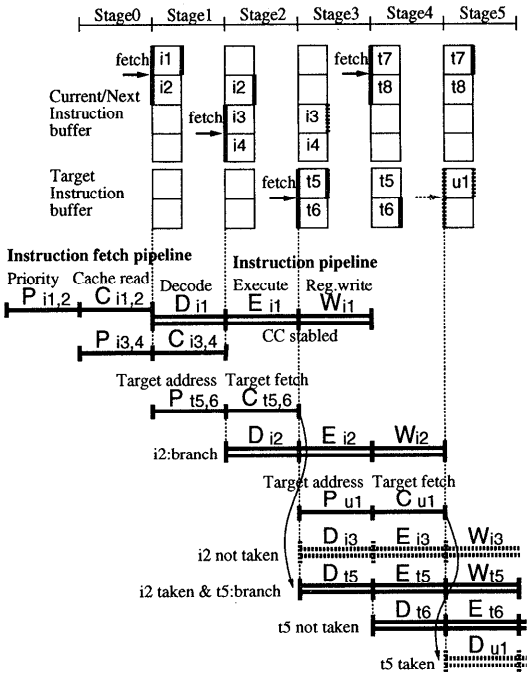


図4 命令パイプライン
Fig. 4 Instruction pipeline.

t5,6 のプリフェッチ Ct5,6，分岐操作を含む命令 i2 のデコード Di2 を行う．この時点において分岐方向が決まり，また，次命令 i3 および分岐先命令 t5 が命令バッファに存在する．Stage3 では分岐条件に応じて即座に次命令 i3 または分岐先命令 t5 のデコードを開始する．

ここで，条件分岐操作における分岐先アドレス指定を PC 相対アドレスのみとした効果について説明する．たとえば，分岐先アドレスがレジスタの内容として指定されると仮定する．この場合には，Ei2 においてレジスタの内容が読み出されて初めて分岐先アドレスが判明する．すなわち，Pt5,6 が Ei2 のステージ（Stage-3）まで遅れ，分岐先命令のデコード Dt5 が Stage-5 となることから，条件分岐に 2 サイクル余分にかかることになる．本 LIW 方式では 2 サイクルに最大 6 操作を発行できるため，この 2 サイクルの短縮は性能向上に大きく寄与する．

次に条件分岐が連続する場合，すなわち，分岐先命令 t5 中の条件分岐操作により，次命令 t6 または分岐先命令 u1 が実行される場合について説明する．Stage3 では次 2 命令 t7,8 のプリフェッチ，分岐先アドレス計算 Pu1 を行う．Ct5,6 と Di2 が同時であったのに対し，分岐先命令 u1 のプリフェッチ Cu1 は Dt5 より遅れるため，この時点では分岐先命令 u1 が命令バッファに存在しない．すなわち，次命令 t6 は Stage4 から実行できるものの，分岐先命令 u1 は 1 ステージ遅れて Stage5 からの実行となる．

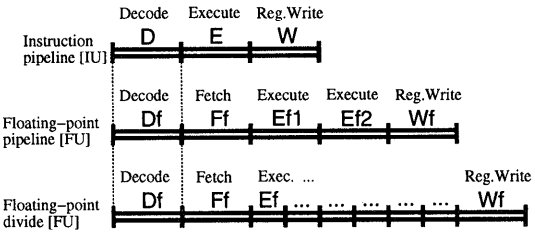


図5 浮動小数点演算パイプライン
Fig. 5 Floating point unit pipeline.

図 4 では，分岐先命令 t5 が 128 ビット境界の前半を占め，t5 と t6 が同時にフェッチされることから t6 の実行が遅れることはない．しかし，t5 が 128 ビット境界の後半を占める場合には，t6 を新たにフェッチすることから，t6 の実行開始が遅れる．

条件分岐の連続，128 ビット境界後半への条件分岐に，命令キャッシュミスを加えて，プログラム走行時間に占める，命令フェッチに関する D ステージの遅れを D.IF と定義する．

3.4 MU に関するインタロック率（E.IP）

乗算には乗数値に依存し 5〜7 サイクルを要する．すなわち，乗算結果を次命令が使う場合は，次命令の E ステージが乗算結果を待つ．プログラム走行時間に占める，汎用レジスタの干渉による E ステージの遅れを E.IP と定義する．E.IP はロード結果を次命令が使う場合の汎用レジスタの干渉（後述）も含む．

3.5 FU に関するインタロック率（D.FQ, D.CC, E.FP）

図 5 に示すように，浮動小数点演算パイプラインは，操作をデコードする Df ステージ，浮動小数点レジスタの干渉検査および読み出しを行う Ff ステージ，演算を行う Ef ステージ，演算結果を浮動小数点レジスタに格納する Wf ステージからなる．除算の Ef ステージは半サイクル長でありステージ数は不定である．

浮動小数点演算操作は，命令パイプラインの D ステージにおいてデコードされると同時に，FQ に空きがあれば FQ にキューイングされる．FQ は 1 命令に記述される加減算操作と乗除算操作の組を 6 組まで格納できる．加減乗算に要するサイクル数は IEEE-754 準拠のための演算処理が増加した分，VP2600 スカラの 2 サイクルに対して 4 サイクルと大きい．すなわち，次命令の入力オペランドとして加減乗算結果を使う場合のインタロックは 3 サイクルである．除算結果

を使う場合は，被除数値に依存し単精度演算では4～6，倍精度演算では4～9サイクルである．

FQ内に存在する依存関係は，浮動小数点演算パイプラインのみのインタロックとなり，演算結果をIUが使うとき（比較が生成するCCの参照，演算結果の主記憶へのストア）に初めて命令パイプラインのインタロックとして観測される．

プログラム走行時間に占める，FQが満杯であることによるDステージの遅れをD.FQ，浮動小数点比較の完了待ち（CCの確定待ち）による条件分岐操作のDステージの遅れをD.CC，浮動小数点演算結果をストアする際のストア操作のEステージの遅れをE.FPと定義する．

3.6 AUに関するインタロック率（E.IP, E.AP, W）

図6に示すように，オペランドパイプラインは，Eステージにおけるアドレス計算結果を元にキャッシュ参照権を獲得するPステージ，TLBおよびキャッシュを参照するCステージ，参照結果を得るRステージ，TLBやキャッシュに書き込みを行うWステージからなる．

TLBおよびキャッシュにヒットするとき，次命令の入力オペランドとして汎用レジスタへのロード結果を使う場合のインタロックは2サイクルである．浮動小数点レジスタへのロード結果を浮動小数点演算が使う場合は，前節に述べたように，浮動小数点演算パイプラインのみのインタロックとなり，命令パイプラインのインタロックとしては観測されない．

主記憶参照操作は，AQにより2個までの非同期動作が可能であり，ロード操作はTLBおよびキャッシュにヒットしている限り，またストア操作はTLBにヒットしている限り毎サイクル発行可能である．ストア操作は，キャッシュがストアスルー方式であることから，キャッシュミスによるインタロックは生じない．すなわち，TLBにヒットしている限り，先行するストア操作を後続のロード操作が追い越すことはない．

一方，非同期動作中の2個のロード操作のうち1個がキャッシュミスを検出した場合には，3個目のロード/ストア操作の発行が待たされる．キャッシュミス時のロード操作のインタロックは17サイクルである．

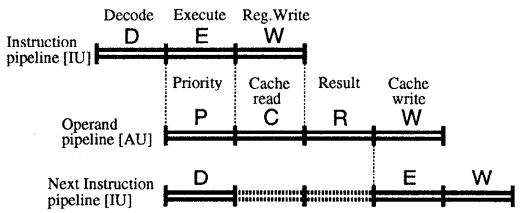


図6 オペランドパイプライン
Fig. 6 Operand pipeline.

乗算結果を次命令が使う場合の汎用レジスタの干渉（前述）に加えて，プログラム走行時間に占める，ロード結果を次命令が使う場合の汎用レジスタの干渉によるEステージの遅れをE.IP（キャッシュミスによる遅れは含まない），ロード結果を次命令が使う場合のキャッシュミスによるEステージの遅れをE.AP，先行ロード操作においてキャッシュミスを検出したことによる後続ロード操作の発行の遅れをWと定義する．

たとえば，ロード結果を直後の命令が使う頻度が高く，キャッシュヒット率が高いプログラムでは，E.IPは高い値，E.APおよびWは低い値を示す．一方，ソフトウェアパイプライニングの適用によりロード結果を2サイクル以上離れた命令が使うプログラムでは，E.IPは低い値を示す．このときキャッシュヒット率が低ければE.APが高くなり，さらにロード操作が毎サイクル発行される場合はWも高い値を示す．

4. SPECfp92を用いた測定結果

VPP500スカラ（サイクルタイム10 ns）上においてSPECfp92の各プログラムを走行した結果をVP2600スカラ（サイクルタイム6.4 ns）に対する相対性能により**表1**の左段に示す．右段は左段に10 ns/6.4 nsを乗じた値であり，同一サイクルタイムを仮定した性能比を表している．幾何平均は1.56である一方，プログラムにより0.73から3.94まで大きなばらつきがある．

性能測定ハードウェア機構によるサンプリング調査

表1 VP2600スカラに対する性能
Table 1 Comparison against VP2600 scalar.

	対VP2600（6.4 ns）比	サイクルタイムにより正規化
013.spice2g6	0.84	1.31
015.doduc	0.71	▲1.11
034.mdljdp2	0.82	1.29
039.wave5	1.07	1.67
047.tomcatv	1.15	1.80
048.ora	0.89	1.40
052.alvinn	2.52	◎3.94
056.ear	1.38	◎2.15
077.mdljsp2	0.77	▲1.20
078.swm256	1.79	◎2.80
089.su2cor	0.96	1.51
090.hydro2d	0.83	1.29
093.nasa7	1.00	1.56
094.fpppp	0.47	▲0.73
幾何平均	1.00	1.56

▲は性能比が特に悪いもの，◎は特に良いものを示す．

表2 操作効率およびキャッシュヒット率
Table 2 Effectiveness of operations and cache hit ratio.

略称	CPI	OPI	CPO	FOR (%)	OP$ (%)	IF$ (%)
spice	2.50	1.14	2.19	1.21	85.33	99.79
doduc	2.28	1.21	1.88	6.74	96.28	97.87
mdljd	2.42	1.17	2.06	12.18	94.95	99.99
wave5	1.80	1.41	1.28	11.88	97.33	99.77
tomca	2.07	1.36	1.53	12.85	91.30	99.98
ora	2.23	1.20	1.85	9.72	100.00	99.99
alvin	1.58	1.66	0.95	19.44	96.40	99.69
ear	2.20	1.35	1.63	4.45	99.51	99.82
mdljs	2.32	1.13	2.04	12.11	98.75	99.99
swm25	1.41	1.73	0.81	33.19	96.27	99.99
su2co	2.45	1.46	1.67	10.52	78.62	99.99
hydro	2.53	1.28	1.97	8.69	89.97	99.60
nasa7	3.33	1.54	2.16	8.24	77.34	99.96
fpppp	2.86	1.25	2.30	7.05	96.74	94.21

下線は際立つ値を示す.

表3 操作頻度 (%)
Table 3 Percentages of operations.

略称	fadd fsub	fcmp	fcnv	fmul	fdiv	load	store	etc.
spice	2.6	0.6	0.1	1.6	0.5	28.8	9.1	56.8
doduc	11.0	3.4	1.3	8.0	1.8	26.3	17.1	31.2
mdljd	22.9	11.3	0.2	14.8	1.0	15.2	9.0	25.7
wave5	12.5	1.7	1.8	13.8	0.6	21.6	14.3	33.6
tomca	22.7	1.2	0.0	14.7	0.6	30.5	11.7	18.5
ora	15.0	2.5	0.0	14.7	3.8	22.3	7.8	33.9
alvin	18.2	0.1	0.7	18.1	0.0	37.0	10.3	15.5
ear	6.9	1.7	0.0	5.9	0.0	26.3	11.4	47.8
mdljs	22.8	11.1	0.2	14.4	1.0	16.4	8.8	25.3
swm25	32.5	0.0	0.0	20.7	0.8	28.3	11.1	6.6
su2co	15.3	0.9	0.7	17.5	0.7	22.6	10.6	31.7
hydro	16.0	7.3	0.0	9.9	1.1	25.4	10.2	30.1
nasa7	17.5	0.4	0.1	17.1	0.5	32.5	13.5	18.5
fpppp	14.9	0.4	0.0	16.9	0.1	32.0	18.4	17.3

下線は際立つ値を示す.

表4 インタロック率 (%)
Table 4 Percentages of interlocks.

略称	W	E.AP	E.IP	E.FP	D.IF	D.FQ	D.CC
spice	0.8	2.5	‡44.0	7.7	4.2	0.5	1.6
doduc	2.7	3.6	3.9	15.1	15.0	4.8	11.7
mdljd	2.7	1.9	2.1	1.8	3.9	12.0	†39.2
wave5	2.5	3.4	3.2	†20.1	4.4	1.8	7.5
tomca	12.9	4.9	8.3	11.4	0.8	8.8	6.5
ora	0.0	0.7	3.9	†21.8	3.8	16.5	6.5
alvin	†23.5	2.2	1.4	2.8	4.4	2.0	0.0
ear	0.5	1.4	3.2	5.0	12.8	0.3	14.8
mdljs	0.3	1.3	2.1	0.9	4.9	10.4	†38.1
swm25	5.3	5.5	0.1	9.2	1.7	8.7	0.1
su2co	†22.3	7.4	1.7	14.3	2.9	7.0	3.6
hydro	7.3	3.1	3.3	11.0	5.9	1.5	†29.5
nasa7	‡40.5	10.7	0.8	10.0	2.2	6.7	0.5
fpppp	1.2	8.7	5.5	†29.0	†25.1	0.3	0.0

下線は10%以上, †は20%以上, ‡は40%以上を示す.

を行って得た各項目の測定結果を**表2**, **表3**, **表4**に示し, 個々のプログラムに関して分析を行う.

4.1 013.spice2g6

LU分解 (ただしループアンローリングおよびソフトウェアパイプライニングの効果がない) が中心である. 浮動小数点演算よりもPIVOTのためのロード, 整数比較, 条件分岐が多くetc.が56.8%と高い. 典型的な `A= 配列(B+I)` および `IF(A.LT.B)GOTO` に対する命令列を示す. 各行先頭の"番号:"は, 命令語の追い番を表し, 命令語間の -2- は2サイクルの命令パイプラインインタロックを表す.

```
1: load     .. 汎用レジスタへの変数Bのロード
   -2-      .. ロード待ちインタロック E.IP
2: add      .. B+Iにより配列の添字を求める
3: shift    .. 4倍して要素アドレスとする
4: load     .. 配列要素からのロード
   -2-      .. ロード待ちインタロック E.IP
5: add      .. ロード結果を使用
-----------------------------------------
1: load     .. 変数Aのロード
   -2-      .. ロード待ちインタロック E.IP
2: compare  .. AとBを比較
3: branch   .. A<Bなら分岐
```

このように1操作のみからなる命令列の頻度が高いためOPIが1.14と極端に悪い. またロード待ちインタロックE.IPが44.0%ときわめて高いため, CPOが2.19と大きい.

4.2 015.doduc

浮動小数点比較に基づく条件分岐が多く, 様々なサブルーチンを頻繁に渡り歩く. ループアンローリングやソフトウェアパイプライニングは困難である. 典型的な `IF(X)100,200,300` に対する命令列を示す.

```
1: f.load     .. 浮動小数点レジスタへのロード
2: f.compare  .. ロード結果Xの判定
   -5-        .. CC確定待ちインタロック D.CC
3: branch     .. 正なら分岐
4: branch     .. 零なら分岐     分岐連続 D.IF
```

CC確定待ちインタロックは, ロード待ちの -2- に比較結果待ちの -3- を加えた -5- となる. このような命令列を反映して, D.CCが11.7%, D.IFが15.0%と各々が高い値を示している. 多くのサブルーチンを渡り歩くことを反映してIF$が97.87%と比較的低いことから, D.IFが高い要因として, 条件分岐の連続, 128ビット境界後半への条件分岐に加えて, 命令キャッシュミスもあげられる.

4.3 034.mdljdp2

fcmpが11.3%であることからも分かるように，浮動小数点演算に基づく条件分岐が非常に多い．部分的にループアンローリングおよびソフトウェアパイプライニングが可能であるが効果は小さい．典型的なIF(A-B.LT.C)THENに対する命令列を示す．

```
1: f.load      .. 浮動小数点レジスタへのロード
2: f.subtract  .. ロード結果Aからを減算
3: f.compare   .. 減算結果の判定
   -8-         .. CC確定待ちインタロック D.CC
4: branch      .. A-B>=Cなら分岐
```

CC確定待ちインタロックは，ロード待ちの -2- に減算待ちの -3- および比較結果待ちの -3- を加えた -8- となる．このような命令列を反映して，D.CCが39.2%と高い値を示している．

4.4 039.wave5

他のプログラムに比べてfcnvが1.8%と型変換が多いのが特徴である．部分的にループアンローリングおよびソフトウェアパイプライニングが可能であるが効果は小さい．例としてAX=BX*X(I)+C, AY=BY*Y(I)+Cに対する命令列を示す．

```
1: f.load      .. X(I) のロード
2: f.load      .. Y(I) のロード
3: f.multiply  .. BX*の乗算
4: f.multiply  .. BY*の乗算
5: f.add       .. +C の加算
6: f.add       .. +C の加算
7: f.convert   .. 整数 AX への型変換
   -8-         .. 結果待ちインタロック E.FP
8: move        .. AX を汎用レジスタへ移動
```

2つのステートメントに対する命令が交互に発行され，依存関係が1命令おきとなっているためインタロックが軽減されている．結果待ちインタロックは，ロード待ちの -1-，乗算結果待ちの -2-，加算結果待ちの -2-，型変換待ちの -3- の合計 -8- である．このようなインタロックが多いことからE.FPは20.1%と高い．

4.5 047.tomcatv

DOループのループアンローリングおよびソフトウェアパイプライニングが可能であり，レジスタ干渉の少ない命令列を生成することが可能である．例としてA(I)=B(I)-C(I)*Dを8重アンローリングしソフトウェアパイプライニングを適用した命令列の一部を示す．

```
1: f.load .. C(I) のロード×8個
2: f.load    および
3: f.load .. B(I) のロード×8個
4: f.load  f.multiply .. *D の乗算×8個
5: f.load  f.multiply
6: f.load  f.multiply
7: f.load  f.multiply
8: f.load  f.multiply
9: f.load  f.multiply
10: f.load  f.multiply  f.subtract .. B(I)-
11: f.load  f.multiply  f.subtract    の減算
12: f.load              f.subtract    ×8個
13: f.load              f.subtract
14: f.load              f.subtract
15: f.load              f.subtract
16: f.load              f.subtract
17: f.store             f.subtract
18: f.store .. ストア×8個
    (以下省略)
```

ただしロードが連続して発行される一方，キャッシュヒット率OP$が91.30%と低いためWが12.9%と高い値を示している．

4.6 048.ora

他のプログラムに比べてfdivが3.8%と除算が多いのが特徴である．ループアンローリングおよびソフトウェアパイプライニングは困難である．039.waveに似て，依存関係のある浮動小数点演算が連続して発行されるため，E.FPが21.8%と高い．倍精度除算は最大9サイクルを要しFQに長時間とどまるため，039.waveと異なり，FQ待ちインタロックのD.FQが16.5%と高い．

4.7 052.alvinn

047.tomcatvと同様，DOループのループアンローリングおよびソフトウェアパイプライニングにより高い性能を引き出せる．Wが23.5%と目立つのは，W以外のインタロック率がきわめて低いためである．OPIが1.66，CPOが0.95，FORが19.44%といずれも良い値を示し，正規化後の対VP2600性能比でも3.94と最も良い値を示している．

4.8 056.ear

052.alvinnと同様，DOループのループアンローリングが可能である．しかし，主なループ中に浮動小数点比較に基づく条件分岐や絶対値演算があるのに対し，VPP500スカラは浮動小数点条件コードを1組しか持たないことから，ソフトウェアパイプライニングによる高速化が難しい．このため，OPIが1.35，CPOが1.63と良い値ではない．また，固定小数点演算により絶対値演算を行っていることから，etc.が47.8%と

高い．

4.9 077.mdljsp2

034.mdljdp2が倍精度演算であるのに対し，本プログラムは単精度演算である．VPP500スカラは単精度/倍精度浮動小数点に関するロード，演算，ストアの素性能が同じであることから，034.mdljdp2のほうがオペランド長が大きくキャッシュヒット率OP$が94.95%と低いことを除けば酷似した測定結果となった．

4.10 078.swm256

DOループのループアンローリングおよびソフトウェアパイプライニングが可能である．各インタロックが小さく，052.alvinnに似た特性を示している．ただし，052.alvinnに対してfaddが32.5%，loadが28.3%と演算が多くロードが少ない．このためOP$が96.27%と同程度であるにもかかわらず，Wが5.3%と低い．その他のインタロックも低く，OPIが1.73，CPOが0.81，FORが33.19%と最も良い値を示し，正規化後の対VP2600性能比でも2.80と良い値を示している．

4.11 089.su2cor

DOループのループアンローリングおよびソフトウェアパイプライニングが可能である．047.tomcatvと同様，ロードが連続して発行される一方，キャッシュヒット率OP$が78.62%ときわめて低いためWが22.3%と高い値を示している．このためCPIが2.45と悪く，正規化後の対VP2600性能比も1.51と良くない．

4.12 090.hydro2d

DOループのループアンローリングが可能である．fcmpが7.3%と浮動小数点比較が多い．VPP500スカラは浮動小数点条件コードを1組しか持たないことから，浮動小数点比較を含めたソフトウェアパイプライニングによる高速化が難しい．このため，D.CCが29.5%と034.mdljdp2および077.mdljsp2に似た特性を示している．

4.13 093.nasa7

DOループのループアンローリングおよびソフトウェアパイプライニングが可能である．047.tomcatvや089.su2corと同様，ロードが連続して発行される一方，キャッシュヒット率OP$が77.34%ときわめて低いためWが40.5%ときわめて高い値を示している．このためCPIが3.33と最も悪く，正規化後の対VP2600性能比も1.56と良くない．

4.14 094.fpppp

ループアンローリングの困難な浮動小数点演算のランダムな集まりである．最内ループ内に条件分岐が少なくかつステートメント量がきわめて多い．最内ループ内の命令サイズは約64Kバイトもあり，IF$が94.21%，D.IFが25.1%であることが示すように命令キャッシュのヒット率が低い．これは015.doducにも見られる特性である．典型的なステートメントおよび対応する命令列を示す．

```
A=A+(B00*C00+B01*C01+B02*C02+B03*C03
    +B04*C04+B05*C05+B06*C06+B07*C07
              (途中省略)
    +B24*C24+B25*C25+B26*C26)*D
-------------------------------------------
 1: f.load B00 ---+
 2: f.load C00 ---+
 3: f.load B01    +
 4: f.load C01    V
 5: f.load B02  f.mult B00*C00 ---+
 6: f.load C02                    +
 7: f.load B03  f.mult B01*C01 ---+
 8: f.load C03                    +
 9: f.load B04  f.mult B02*C02    +
10: f.load C04                    +
11: f.load B05  f.mult B03*C03    V
12: f.load C05  f.add  B00*C00+B01*C01
13: f.load B06  f.mult B04*C04
14: f.load C06
15: f.load B07  f.mult B05*C05
16: f.load C07  f.add B02*C02+B03*C03
              (途中省略)
*1: f.mult *D
*2: f.add  +A
   -E.FP-       .. 演算結果待ちインタロック
*3: f.store A
```

離散的な変数BおよびCをロードするために，レジスタおよび12ビット以上のインデックス値によるアドレス指定が必要である．2.2節に述べたように，このようなロード操作は40ビット長操作しかなく，1命令に最大2個までしか操作を入れることができないため，OPIが1.25と低い．また，最終的にAをストアする際にFQ内の演算終了を待つために，E.FPが29.0%と非常に高い値を示している．

5. 考　　察

測定結果から特に注目すべきインタロックを取り出し，**表5**にグラフとして示す．この結果から，VPP500スカラに関して，以下のことがいえる．

- Wが高い値を示す，047.tomcatv，089.su2cor，

表5 顕著なインタロック
Table 5 Remarkable interlocks.

	W	E.FP	D.IF	D.CC
047.tomcatv	xx	xx		x
089.su2cor	xxxx	xx		
093.nasa7	xxxxxxxx	xx		
039.wave5		xxxx		x
048.ora		xxxx		x
015.doduc		xxx	xxx	xx
094.fpppp		xxxxx	xxxxx	
034.mdljdp2				xxxxxxx
077.mdljsp2				xxxxxxx
056.ear		x	xx	xx
090.hydro2d	x	xx	x	xxxxx
052.alvinn	xxxx			
078.swm256	x	x		

x は5%を表す.

093.nasa7では，オペランドキャッシュが32Kバイトかつダイレクトマップ方式であることから，64Kバイトかつ16ウェイのVP2600スカラに比べるとキャッシュヒット率が低く，せっかくソフトウェアパイプライニングを適用してもこれに見合うだけの十分な性能が出ない．キャッシュ容量およびウェイ数の増加による大幅な性能向上が期待できる．

- E.FPが高い値を示す，039.wave5，048.oraでは，FQがよく機能しており，依存関係のある浮動小数点演算が多数FQにキューイングされている．しかし，演算結果を得てストアする際のインタロックが大きく，加減乗算に要するサイクル数4そのものを削減する努力が必要である．
- D.IFが高い値を示す，015.doduc，094.fppppでは，命令キャッシュのヒット率が低い．特に094.fppppでは，基本ブロックに属する命令が約64Kバイトと巨大であることから，ウェイ数を増やすことよりも容量を増やすことが性能向上のために有効である．
- D.CCが高い値を示す，034.mdljdp2，056.ear，077.mdljsp2，090.hydro2dでは，浮動小数点比較に4サイクルを要すること，または，条件コードレジスタが1組しかないことが性能向上の妨げになっている．比較に要するサイクル数を削減することにより，この4本のプログラムに対して効果がある．また，サイクル数を削減できなくても複数個の条件コードレジスタを設けることにより，比較を含めたループアンローリングおよびソフトウェアパイプライニングの適用が可能となり，056.earおよび090.hydro2dに対して効果がある．
- 052.alvinn，078.swm256のように，キャッシュヒット率が高く，かつソフトウェアパイプライニングの適用によりOPIを高くできる場合には，本LIW方式がきわめて有効に動作し，高い性能を引き出せる（SPECratioは各々，307.6，165.5[6]）．

さて本アーキテクチャは，SPECfp92が登場する以前に，livermoreループなどの比較的小さなベンチマークプログラムを参考にしながら，限られたハードウェア資源の中でいかに高性能を引き出すかの工夫を施し，設計したものである．052.alvinn，078.swm256において高い性能を発揮できたことは，当初の設計どおりである．一方，SPECfp92を走行した結果，キャッシュ容量，ウェイ数，浮動小数点演算レイテンシ，浮動小数点条件コードレジスタ数に関して改善すべき点が明らかになった．

キャッシュ容量，ウェイ数，浮動小数点演算レイテンシについては，ハードウェア物量を鑑み，当時，最大限努力して実現したものであり，反省すべくもない．しかし，浮動小数点条件コードレジスタが足りないためにループアンローリングおよびソフトウェアパイプライニングが適用できず，本LIW方式が有効に機能しなかったことは，設計時の考慮が不足していたためであり，大いに反省しなければならない．この結果を今後の糧としたい．

6. おわりに

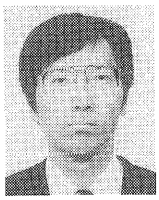
VPP500スカラの開発に際しては，限られたハードウェア資源の中でいかに高性能を引き出すかの工夫を施し，VP2600よりも遅いサイクルタイムであるにもかかわらず，同等の性能を達成した．一方で，浮動小数点条件コードレジスタ数が不足すること，また，直接ハードウェア量の増加につながるけれども，キャッシュ容量，ウェイ数，浮動小数点演算に要するサイクル数について改善することにより，さらに高性能を引き出せることが明らかになった．本稿の結論を今後の課題としたい．

参考文献

1) Uchida, N., Hirai, M., Yoshida, M. and Hotta, K.: Fujitsu VP2000 Series, *Digest of Papers, COMPCON Spring 90*, pp.4–11 (1990).
2) Miura, K., Takamura, M., Sakamoto, Y. and Okada, S.: Overview of the Fujitsu VPP500

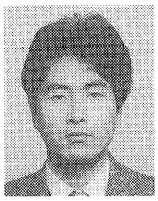
Supercomputer, *Digest of Papers, COMPCON Spring 93* (1993).
3) Utsumi, T., Ikeda, M. and Takamura, M.: Architecture of the VPP500 Parallel Supercomputer, *Proc. Supercomputing '94*, Washington D.C., pp.478–487 (1994).
4) Nakanishi, M., Ina, H. and Miura, K.: A High Performance Linear Equation Solver on the VPP500 Parallel Supercomputer, *Proc. Supercomputing '94*, Washington D.C., pp.803–810 (1994).
5) 中島康彦，北村俊明，田村秀夫，滝内政昭：VPP500 スカラプロセサの特徴，情報処理学会研究報告，ARC-104-17, pp.129–136 (1994).
6) Nakashima, Y., Kitamura, T., Tamura, H., Takiuchi, M. and Miura, K.: Scalar Processor of the VPP500 Parallel Supercomputer, *Proc. 9th ACM Int. Conf. of Supercomputing*, pp.348–356 (1995).

(平成 8 年 8 月 23 日受付)
(平成 9 年 1 月 10 日採録)



中島　康彦（正会員）

1963 年生．1986 年京都大学工学部情報工学科卒業．1988 年同大学院修士課程修了．同年富士通（株）入社．以来，グローバルサーバ事業部にてプロセッサ・アーキテクチャの研究開発に従事．並列処理アーキテクチャに興味を持つ．1987 年本会学術奨励賞．



大野　優人

1963 年生．1982 年県立和歌山工業高校卒業．同年富士通（株）入社．1986 年富士通工業専門学校卒業．1989 年文部省受託研究員として東京大学工学部精密機械工学科に派遣される．入社以来，汎用コンピュータおよびスーパコンピュータの論理設計に従事．

竹部　好正

1964 年生．1987 年早稲田大学理工学部電気工学科卒業．同年より同大学院理工学研究科一般研修生．1988 年富士通（株）入社．以来，スーパーコンピュータ VPP シリーズのプロセッサ開発・設計に従事．