

インタプリタ・コプロセッサの設計

9L-7

平井輝久、金本好司、板野肯三
筑波大学

1 はじめに

PL/0 言語を対象にした、ハードウェアによる解析木インタプリタ PATIE-0[1] をより実用的にするに当たり、新しい概念を持つコプロセッサを提案して、このコプロセッサを応用して解析木インタプリタ CPATIE-0(Coprocessor PARse Tree InterpretEr for PL/0) を設計した。CPUとして SPARC を使用して、その特徴を生かしたハードウェア設計を行ない、C 言語でレジスタ転送レベルのシミュレータを作成して性能評価を行った。

2 変換型コプロセッサの概念

コプロセッサとして一般的なものに、図 1(a) に示すような FPU(Floating-Point Unit) が存在する。その FPU は、バスに対して CPU と並列な場所に位置し、アドレスの制御を全て CPU に任せ、フェッチした Data (命令も含む) を CPU と排他的に解釈実行するというのが特徴である。

しかし、ここでは図 1(b) に示すような変換型コプロセッサというものを提案する。変換型コプロセッサは、バスに割り込むように CPU と MEMORY の間に位置し、アドレスの制御を変換型コプロセッサ自身が行ない(このとき、CPU のアドレスラインは意味を持たない)、フェッチした Data を決められた規則に従って別のもの(Data') に変換して CPU に送り、CPU が Data' を解釈実行するという特徴を持つ新しい形のコプロセッサである。この変換型コプロセッサの応用として、解析木インタプリタを設計した。

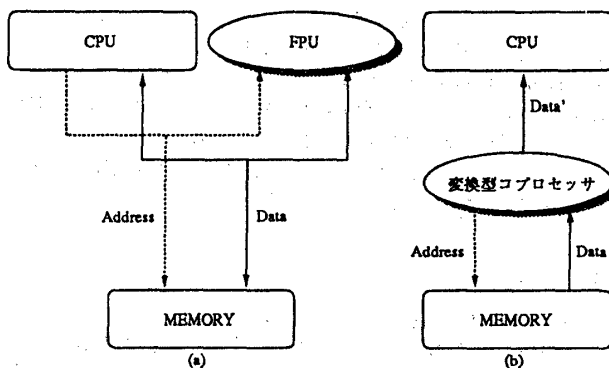


図 1: FPU と変換型コプロセッサ

3 CPATIE-0 の設計

解析木インタプリタの動作には、木をたどってノードをフェッチする処理、フェッチしたノードを認識して演算のシーケンスを生成する処理、この演算を実行する処理があるが、前2者を

変換型コプロセッサで実行し、演算は CPU(SPARC) に送って実行する。ここでは、変換型コプロセッサ CPATIE-0 の入力は解析木のノード、出力は各ノードに対応した SPARC の命令であり、フェッチしたノードを SPARC の命令に変換している。

SPARC を使用するに当たり、SPARC の最大の特徴であるレジスタウィンドウを有効に利用するため、変数や数式評価時の一時変数や関数呼び出し時のパラメータは、できる限りレジスタに割り当て、使用するレジスタの番号や記憶領域のアドレスの情報は、意味解析時にノードに保存した。この意味解析情報を処理するために、レジスタ間のデータ転送命令を SPARC に出力するのか、それともメモリアクセス命令を出力するのかを決定する機構やレジスタ間のデータ転送命令を動的に作り出す機構などが、CPATIE-0 に新たに必要になった。

図 2 に設計した CPATIE-0 のハードウェア構成を示し、以下でそれぞれのユニットについて説明を行なう。

(1) TCU (Traverse Control Unit)

このユニットは、MEMORY からノードをフェッチしたり、分岐処理を行ったりするユニットで、フェッチされたデータは ncode, child1, child2 の 3 つのフィールドを持つ NODE レジスタに格納される。ncode はノードの種類、child1 は子ノード 1 のアドレス、child2 は子ノード 2 のアドレスである。ncode をもとに制御テーブル CTABLE が引ひかれ、そのノードの子ノードのたどり方や意味解析情報の格納場所などの情報が得られる。これらの情報や、分岐処理に必要な条件判断の結果を格納している COND レジスタの値をもとに、TCU_CTR は TCU 全体の制御を行なう。子ノードにアドレスは、そのたどり方によって、そのまま次にフェッチするノードのアドレスとするためにプログラムカウンタ PC に格納したり、後で使うために一旦 PC_STACK にプッシュしたりする。意味解析情報は、ノードの種類によって child1 と child2 のフィールドに格納されていたり、フェッチしたノードの次ワードに格納されていたりするので、両者から OPERAND1 レジスタに格納できるようになっている。レジスタ PC を +4 するのは、フェッチしたノードの次ワードのアドレスを得るためである。ノードの種類と意味解析情報は、それぞれインターフェイスレジスタ NCODE と OPERAND を介して次のユニットに送られる。

(2) RCU (Recursive Control Unit)

このユニットは、各ノードごとに決められた SPARC の命令のシーケンスを作り出すユニットである。TCU から送られた NCODE と OPERAND は、それぞれ MPC レジスタと OPERAND2 レジスタに格納され、ノードの種類を表す NCODE はそのままマイクロプログラムカウンタ MPC になり、制御テーブル ETABLE 中のアドレスを意味することになる。ETABLE には、各ノードごとに、RCU で行なう処理や、次に実行するマイクロプログラムのアドレスや、出力すべき SPARC の命令などが定義されており、これらの情報をもとに、RCU_CTR は RCU 全体の制御を行なう。このマイクロプログラムの実行順序の制御には MPC_STACK が使用される。また、各ノードで必要な意味解析情報を一時的に保存する必要があるために、OPERAND2 レジスタの MPC_STACK へのプッシュ、ポップを許している。OPERAND2 レジスタと RPO レジスタとの演算や代入は、関数のパラメータをレジス

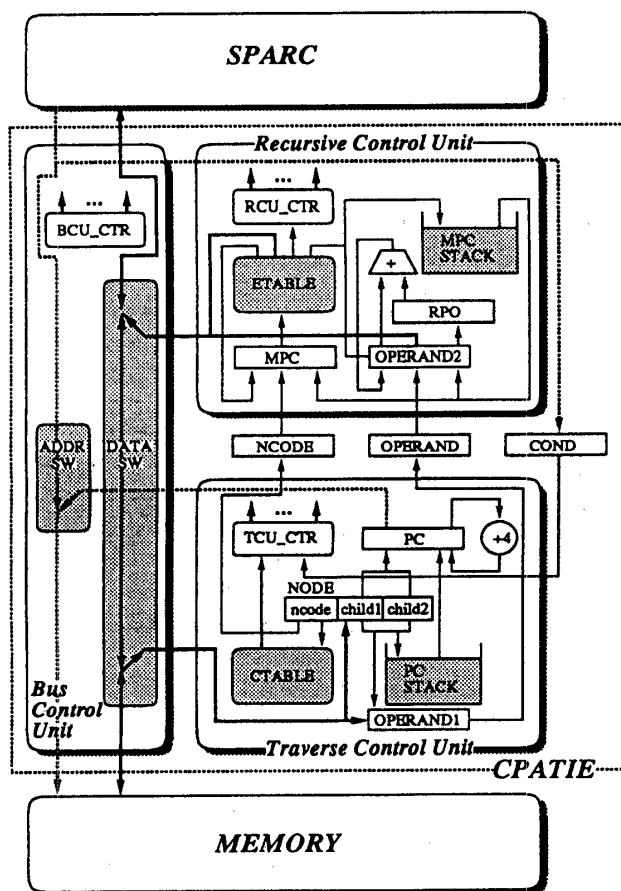


図 2: CPATIE のハードウェア構成

タ間で参照、代入する命令を作り出すために使われる。条件判断の結果は SPARC からアドレスラインを使用して出力され、RCU の制御でインターフェイスレジスタ COND に格納される。この条件判断と、SPARC と CPATIE-0 の一時的な動作停止の処理が RCU に増えたため、RCU の 1 命令の処理にかかる最短クロック数が 2 から 3 に増加した。

(3) BCU (Bus Control Unit)

このユニットは、バスの切替えを行なうユニットである。CPATIE-0 は、変換型コプロセッサとしての動作の他に、変数がメモリ上の記憶領域に割り当てられている場合には、SPARC に直接メモリアクセスを行なわせる必要がある。この問題を解決するために、アドレスバスの切替えを行なう ADDR_SW と、データバスの切替えを行なう DATA_SW を含むユニットを新たに作成した。

4 性能評価

SPARC の ISP 記述 [2] とタイミングチャート [3] をもとに、SPARC と CPATIE-0 のレジスタ転送レベルのシミュレータを作成して、tree11(sort 200)、tree22(matrix 30x30)、tree33(acker(3,4)) の三つの例題を実行した。表 1 に PATIE-0 における DU と CPATIE-0 における SPARC のメモリアクセスの頻度を示す。DU ではデータの演算に内部の STACK も使用しているため、その使用頻度との和を示している。また SPARC では、通常のメモリアクセスの頻度と、トラップによってレジスタウィンドウの退避中に行なわれたメモリアクセスの頻度を

表 1: スタックとメモリのアクセス頻度の比較

		アクセス回数		
		tree11	tree22	tree33
PATIE-0 (DU)	memory	362799	316060	97862
	stack	324796	569112	72030
	SUM	687595	885172	169892
CPATIE-0 (SPARC)	normal	100099	56700	0
	trap	8	8	367634

表 2: CPATIE-0 の各ユニットと SPARC の実行時間

		クロック数 (PATIE-0 に対する比率)		
		tree11	tree22	tree33
TCU		9577970 (1.19)	10837802 (1.25)	2785549 (1.34)
RCU		9577974 (1.19)	10837806 (1.25)	2785560 (1.34)
SPARC (DU)		1534210 (1.07)	1560747 (0.97)	764093 (1.75)

示している。この表から、レジスタ割り当ての効果によりメモリアクセスの頻度がかなり減少していることが分かる。特に tree33 では、トラップによりかなりのメモリアクセスが行なわれているものの、通常の動作では全くメモリアクセスが行なわれていない。

また、CPATIE-0 の各ユニットと SPARC における実行時間と、その実行時間の PATIE-0 に対する比率を表 2 に示す。tree33 ではトラップの処理が頻繁に行なわれるため、実行にかなりの時間がかかっているが、トラップの処理時間を除くと TCU と RCU の実行時間の比率は 1.25、SPARC の実行時間の比率は 0.91 になる。RCU の 1 命令の処理にかかる最短クロック数が 2 から 3 に増加した結果、PATIE-0 に比べて TCU と RCU のクロック数が増加している。しかし、最短クロック数の増加に比例した 1.5 倍よりは少ないため、レジスタ割り当ての効果が現れているものと思われる。また、SPARC と DU の実行時間は、トラップの時間を除くとどの例題も同じぐらいなので、RCU の 1 命令の処理にかかる最短クロック数を少なくするような改良を行なえば、全体のクロック数は PATIE-0 と同レベルにできるものと思われる。

5 おわりに

本論文では、解析木インタプリタを変換型コプロセッサとして設計し、そのシミュレータを作成して性能の評価を行なった。その結果、実行時間ではまだ改善の余地が残されているものの、レジスタ間転送命令の動的な生成やバスの切替えなどの、解析木インタプリタの変換型コプロセッサとしての設計法に関して多くの指針が得られた。

参考文献

- [1] カン ショウウエ, “構文木インタプリタのアーキテクチャの設計”, 筑波大学工学研究科 修士論文 (1989).
- [2] SPARC International, Inc., “The SPARC Architecture Manual Version 8”, (1992).
- [3] Cypress Semiconductor Corporation, “BiCMOS/CMOS DATA BOOK” (1990).