抽象データ型支援アーキテクチャの一考察

1F-7

神田(コウダ)陽治　田中英彦　元岡達
東京大学　工学部

1° すべてのはじめに

VLSIチップにソフトウェア生産性を高める機能を埋め込むことを考える。研究の背景には、意図する機能を容易に1チップにできる技術発達と、ソフトウェアの生産性向上が求められている事実がある。抽象データ型言語は、同じくソフトウェア生産性向上を目的とした1970年代の言語設計活動の主要な成果であり、†これを支援するアーキテクチャを求めることは上述の最終目標達成の正しい一歩であると考えられる。

2° 抽象概念支援のために考慮すべき項目の提示

抽象データ型*は、問題を表現するデータ構造とそれを直接扱う操作を集め、さらに選ばれた一部の操作を外から呼び出すことを除き全体を封印する。例外となる操作の選択はデータ型定義中に陽に指定したり、データ型使用者の持つ資格によって決められる。この封印により、少ない知識で使えかつ誤った使用を許さないサブシステムを作りだす。

考慮項目を示そう。言語、OS、アーキテクチャの統合設計が不可欠である。

① 抽象化データへの誤った操作をすべて摘発する保護システム
② 問題ごとに作成される種々の抽象データ型すべてに効率よい処理方法
③ 抽象データ型の効率よい実装方法(表現法)、仮想記憶内での取り扱い

3° アーキテクチャ支援、例えばモジュラープログラミングの成功に不可欠

2°の項目を達成したアーキテクチャはモジュールの正しい効率よい実行を可能にする。しかしより本質的なことは、モジュールが扱うデータ型の指定を実行時にまで遅らせることができ、モジュール作成の時点で考えに入れていなかったデータ型も扱いうる柔軟性を生みだすことができる点である。‡

モジュールが記述する仕事は、2種の処理に分けられる。一つが本来の計算処理であり、他がそれを支援する種々の処理である。後者が大きいオーバーヘッドにならぬようにアーキテクチャで支援するわけである。

4° 性能向上の余力を抽象性と生産性向上の目的に利用する提案

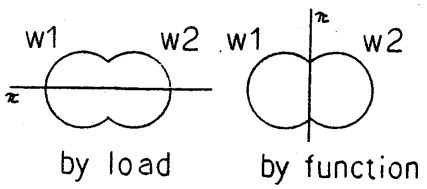
半導体技術の進歩により個人が大きな処理能力を所有できる時代である。その余力をソフトウェア生産性向上に使う。ローカルネットワークは高性能処理ステーションを共有可能とするから、個人用のワークステーションに人間の知的活動を支援する余分の能力を持たせることができる。

† 抽象データ型も完全解ではない。次の世代(1980年～)の研究が進行中
‡ 期待のAdaの多続き名オーバーローディングでは、新規作成が必要である

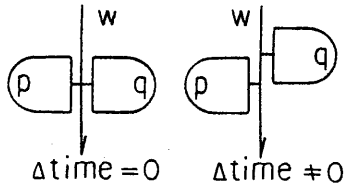
* Mary Shaw, 'The Impact of Abstraction Concerns on Modern Programming Languages' pp1119-1130 the Proceedings of the IEEE special issue on Software Engeneering, September 1980, Vol.68, No. 9

5° 仕事の分割について

3°で述べた本来の処理と支援の処理からなる仕事の分割には、均質に分ける負荷分割，機能単位に区別する機能分割がある。



機能分割の場合，本来の処理と支援の処理を各々機能単位とする分割を考えることができる。 2つの機能単位内の対応するステップ同志を同時に実行しても時間差をもって実行してもよい。それはインプリメンテーションに依る。

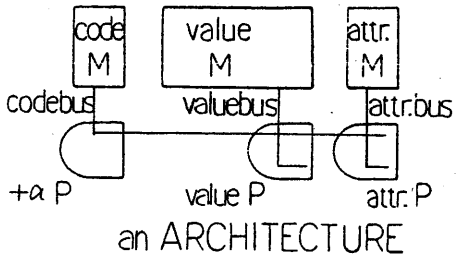


6° ソフトウェア生産性向上を目指す2つのアーキテクチャの例

① リバーシブル実行によるデバッグ支援を目的とするプロセッサーPは、Qに対し常にNステップ先行し，Pがエラーなどで停止すればQもNステップ離れて停止する。 QがQからPまでの各ステップを再現できるなら（Nステップの範囲であるが）精密なリバーシブル実行ができる。

② 値の計算に加え属性の計算もして柔軟性ある動的検査をおこなうプロセッサーPは値の計算，Qは属性の計算をする。 命令とデータを区別し、さらにそのデータを値と属性に分け、それぞれに独立のメモリ，バスを与える。

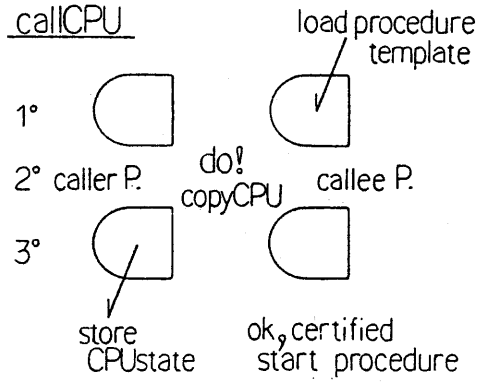
属性メモリはコンパイラーが捨てていた属性情報を再利用する。 高速で幅の広いバスが必要だが小容量でよい。



7° QUICK PROCEDURE CALL の実現

高級言語，特に抽象データ型の言語のプログラムには，手続き呼びだしが多く使われる。 安全かつ速い呼びと戻りの機構が必要である。

6°②の2台で1組と呼ぼう。システムは数組からなっているとする。手続きの呼びだしは，まず遊んでいる組（空組）を選び，それに手続き定義を読み込み，引数間の属性検査をしつつ呼び手組の持つ値情報（CPU状態）を空組へコピーし，最後に空組に起動をかけ手続き本体の実行を開始させる。 呼び手の組は、自ら自分の持つ全情報をメモリへ退避して空組となる。



8° まとめ

抽象化データを扱う基礎アーキテクチャを考察した。 いまだ概念形成の段階であり，生産性向上を目指して一層の考察と具体化が必要である。