

解説

6. 命令セットアーキテクチャの具体例



6.1 汎用中・大型機 IBM S/370†

中谷登志男†

1. はじめに

S/370 命令セットについて概説する。S/370 命令セットの特徴及び変遷についてはすでによく知られているので最小限にとどめる。中でも、RISC 命令セットとの比較、3090 プロセッサでの実現技術、そして将来の展望を中心に述べていくことにする。

2. S/370 命令セットの特徴

S/370 命令セットで想定されている計算機モデルは、バイト単位でアドレスされるメモリと2の補数表現による32ビット演算器を備えたレジスタ・マシンである。

基本的に2-アドレス命令で構成され、2, 4, 6バイトの三種類の命令長があり、それぞれレジスタ-レジスタ命令、レジスタ-メモリ命令、メモリ-メモリ命令に対応している。

アドレッシング・モードは、基底-変位 (base-displacement) 方式と基底-指標-変位 (base-index-displacement) 方式の二つがあり、変位 (displacement) の大きさは、0~4095 バイトである。汎用レジスタは、16個あり、それぞれ32ビットで、アドレスにもデータにも使用できる。このほか、浮動小数点演算用に4個のレジスタがあり、それぞれ64ビットである。

データ形式としては、16ビットと32ビットの固定小数点数、32ビットと64ビットの浮動小数点数 (基底16)、16バイトまでの (可変長) 十進数、及び256バイトまでの (可変長) 文字列などがある。

3. S/370 命令セットの変遷

1964年にS/360が発表されて以来、現在に至るま

で20数年間にわたり命令セットは基本的に変更されていない。一方、アドレス空間は以下に述べるように大きく拡張された。

S/360は、24ビット単一実アドレス空間を提供していたが、8年後の1972年には、仮想記憶の概念を含むS/370が発表され、複数の24ビット仮想アドレス空間及び単一の24ビット実アドレス空間を提供している。これにより、各ユーザは16メガ・バイトの主記憶域を専有することが可能になり、実質的なアドレス空間の飛躍的な拡大となった。

1980年には、クロス・メモリ機構により、ユーザは、二つの仮想アドレス空間においてデータを移動することが可能になり、また、ページ・アドレス・テーブル・エントリの未使用の2ビットを使うことにより26ビットの実アドレス空間を提供するようになった。

1983年には、S/370 XAが発表になり、31ビットの仮想アドレス空間及び31ビットの実アドレス空間を提供し、仮想記憶域、及び、実記憶域は、2ギガ・バイトにまで拡張された。アドレスの最上位ビットがモード・ビットとして24ビットと31ビットの切り替えのために使用されている。

1988年には、S/370 ESAが発表され、各ユーザは、データ専用の複数の2ギガ・バイト仮想記憶域を専有/共有することが可能になった。

その他の大きな拡張としては、1986年に科学技術計算などの計算中心のアプリケーションを効率良く行うためにベクトル演算命令と16個のベクトル・レジスタが追加された。

4. RISC 命令セットとの比較

S/370の命令セットは、次の二つの観点からRISC命令セットの拡張とみることもできる。

第一はCPI (Cycle Per Instruction) からの観点である。

† An Example of Instruction Set Processor Architecture IBM S/370 by Toshio NAKATANI (IBM Tokyo Research Laboratory).

†† 日本 IBM 東京基礎研究所

RISC では、CPI を 1 に近づけることを設計目標に、ロード・ストア アーキテクチャを採用している。すなわち、RISC では、命令数を限定し、CPI の低減に効果のある重要な（結果的には単純な）命令だけをレジスタ演算命令として実現している。S/370 においても汎用のアプリケーションでよく現れる重要な基本演算はレジスタ演算命令として、1 サイクルで実行されるよう配慮されている。

RISC では、3-アドレス命令を採用していることが多いが、S/370 では、2-アドレス命令が中心である。

2-アドレス命令では、1) レジスタの数を最小限におさえられる、2) 命令長が小さくてすむという二つの長所がある。しかし、ソース・オペランドの一つを消去してしまうので、繰り返し両方のオペランドを使う場合には、次のようにレジスタ間の移動を必要とするという短所がある。たとえば、

```
ADD R1, R2, R3
```

に対して、等価の 2-アドレス命令は（以下、命令の記述法は便宜上のもので実際の S/370 命令表記と必ずしも一致しない）、

```
MOVE R1, R2
```

```
ADD R1, R3
```

となる。ところが、一般には $A \leftarrow A + B$ のように一方のオペランドを消去しても問題のない場合がかなりの頻度でみられる。すなわち、

```
ADD R1, R1, R2
```

となる場合が多く、

```
ADD R1, R2
```

で十分ということになる。これにより、コード・スペースの大幅な節約になり、間接的な CPI の低減にもつながる。

実際、典型的な RISC は、単一 32 ビット命令を採用しているが、S/370 では、半分の 16 ビットでレジスタ演算命令を実現している。このため S/370 では、複数の命令長が存在するが、1) 命令の最上位 2 ビットによる命令長の明示、及び 2) レジスタ指定部分の固定などにより、命令デコードを単純化している。

第二は NIE (Number of Instructions Executed) からの観点である。

S/370 命令セットにおいては、RISC 命令セットの拡張として、NIE の低減に効果のある命令を 1) レジスタ-メモリ命令、あるいは 2) メモリ-メモリ命令として実現している。

たとえば、S-370 のレジスタ-メモリ命令は次のよ

うな場合に NIE の効果的な低減に貢献する。

今、 $A_1, A_2, \dots, A_n$ の総和を求めたいとする。ロード・ストア アーキテクチャでは、

```
LOOP: LOAD R2, A (R1)
```

```
*ADD R0, R2
```

```
ADD R1, R3
```

```
COMP R1, R4
```

```
BNE LOOP
```

となり、本来実行したい加算演算 (*) が 5 サイクルに一度しか実行されないことになる。これを、次のようにレジスタとメモリ間の加算命令 (#) を導入することにより、

```
LOOP: #ADD R0, A (R1)
```

```
ADD R1, R2
```

```
COMP R1, R3
```

```
BNE LOOP
```

と変更することができ、最高 20% の実行時間の節約になりうる。さらには、レジスタの使用数及びコード・サイズの低減という効用もある。

5. S/370 命令セットの実現技術

計算機の性能を向上させる三大要素は、サイクル・タイム、CPI (Cycle Per Instruction)、及び NIE (Number of Instructions Executed) である。このうち、命令セットの設計においては、RISC との比較でみたように、CPI と NIE の二要素が重要である。

これに対し、実際のハードウェアによる実現においては、サイクル・タイムと CPI の二要素を、価格と性能の両方の観点から、どう向上させるかが重要になる。以下、3090 プロセッサを例にとり、S/370 命令セットの実現技術をみていくことにしよう。

まず第一に、サイクル・タイムの短縮のためには、速いテクノロジーを採用すること以外に、高度なパイプライン処理方式によりゲイト数を少なくすることがあげられる。

3090 プロセッサの命令処理部は、1) 命令の先読み、2) 命令のデコード及びオペランド・アドレスの生成、3) オペランドの先読み、4) キュー (queue) での実行待ち、5) 命令実行の 5 つのステージ (pipeline stage) からなる。

一般には、1) 命令バッファに数命令、2) 命令レジスタに 1 命令、3) オペランド・バッファに数命令、4) キューに 3 命令、そして 5) 1 命令が実行されているというように、7 個以上の命令が並行処理さ

れていることになる。

第二に、CPI の低減を目的としては、1) 命令実行数の短縮、2) メモリ・アクセス遅延の短縮、3) 命令及びオペランドの高度な先読み、4) いくつかの命令に関する先実行などがあげられる。

1) 命令実行数の短縮のために、3090 プロセッサの命令実行部は、a) 並列加算器、b) シフタ(shift-er)、c) 順列加算器、d) 汎用レジスタなどから構成され、100 ビットを越える水平型マイクロ・コードで直接制御されている。また、固定小数点及び浮動小数点のための専用掛算器を装備し、マシン・サイクルの半分で動作する高速回路で実現している。

2) メモリ・アクセス時間の短縮のために、3090 プロセッサでは、仮想アドレスでアクセス可能でかつストア・イン(store-in)タイプのキャッシュ・メモリを採用している。

3) 高度な先読みのために、3090 プロセッサでは、三つの命令先読み機構を装備し、現在の命令列以外に二つの異なる命令列を先読みすることを可能にしている。さらに、デコード・ヒストリ・テーブル(decode history table)と呼ばれるハードウェアのテーブルにより、過去の分岐命令の実行結果が記録されており、コンディション・コードを予測することができる。これにより、分岐先の命令を前もってデコードすることが可能である。また、たとえ予測がはずれたときにも、コンディション・コードがセットされると同時に、正しい分岐先の命令をデコードすることが可能である。

4) 先実行の例として、3090 プロセッサでは、ループ終了(loop-closing)のための分岐命令と、アドレス生成のために頻繁に使用されるロード・アドレス(Load Address)命令があげられる。ループ終了のための分岐命令は、コンディション・コードがセットされるのを待たずに先実行され、分岐先の命令を遅延なくデコードすることができる。同様に、ロード・アドレス命令に関しても、先実行し、それに続く命令のアドレス生成に、その結果を利用することが可能である。

6. 将来の展望

これまで述べてきたように、命令セットの設計及びその実現のさまざまな手法により、CPI×NIE の低減が図られてきた。最後に、今後の方向に一つの示唆を与えている一例をみることにしよう。

1986 年に S/370 命令セットに追加されたベクトル演算命令は、次のような場合、CPI×NIE の低減に効果的である。

今、 $A_1, A_2, \dots, A_n$ と $B_1, B_2, \dots, B_n$ の個々の和を $C_1, C_2, \dots, C_n$ として記憶したいとする。既存の命令だけで表すと、

```
LOOP: LOAD  R0, A(R1, R4)
      *ADD   R0, B(R2, R4)
      STORE  R0, C(R3, R4)
      ADD    R4, R5
      COMP   R4, R6
      BNE    LOOP
```

となる。ここで、本来実行したい加算命令(*)は、6 サイクルに 1 回の実行となり、加算器からみると効率よく利用されていないことになる。これを、ベクトル演算命令により書き換えると、

```
VLOAD  VR0, A(R1)
VADD   VR0, B(R2)
VSTORE VR0, C(R3)
```

となる。これらのベクトル演算命令が 3090 プロセッサでデコードされると、新しく追加されたベクトル演算機構により、1) メモリからレジスタへのロード、2) 加算演算、3) レジスタからメモリへのストアが、実質上、連結パイプライン化(chaining)され、毎サイクルごとに加算器から新しい結果が生成される。

これにより、実質上のスループット(throughput)は 6 倍になったことになる。すなわち、CPI は増加したことになるが、NIE は減少したことになり、全体として、CPI×NIE を減少させたことになる。

ただし、実際には、初期遅延、メモリ遅延、ベクトル化の程度などにより性能は左右される。しかし、ベクトル演算の導入は科学技術計算など、同じ演算を繰り返し行う必要のあるアプリケーションには非常に効果的である。

7. おわりに

S/370 命令セットについての詳細は、文献^{5), 6), 7)}にまとめられている。S/360 から S/370 XA そして S/370 ベクトル演算命令に至るそれぞれの設計思想は^{1), 2), 3), 9)}に詳しい。また、S/370 命令セットの変遷は^{4), 8)}が参考になる。実際の各機種での実現方式は⁸⁾、特に 3090 プロセッサに関しては¹⁰⁾に要約されている。

参 考 文 献

- 1) Amdahl, G., Blaauw, G. and Brooks, F.: Architecture of the IBM System/360, IBM Journal of Research and Development 8: 2, pp. 87-101 (1964).
- 2) Buchholz, W.: The IBM System/370 Vector Architecture, IBM Systems Journal 25: 1, pp. 51-62 (1986).
- 3) Case, R. and Padegs, A.: Architecture of the IBM System/370, Communications of the ACM 21: 1, pp. 73-96 (1986).
- 4) Gifford, D. and Spector, A.: Case Study: IBM's System/360-370 Architecture, Communications of the ACM 30: 4, pp. 292-307 (1987).
- 5) IBM: System/370 Principles of Operation, GA-22-7000.
- 6) IBM: IBM System/370 Extended Architecture Principles of Operation, SA-22-7085.
- 7) IBM: IBM System/370 Vector Operations, SA-22-7125.
- 8) Padegs, A.: System/360 and Beyond, IBM Journal of Research and Development 25: 5, pp. 377-390 (1981).
- 9) Padegs, A.: System/370 Extended Architecture: Design Considerations, IBM Journal of Research and Development 27: 3, pp. 198-205 (1983).
- 10) Tucker, S.: The IBM 3090 System: An Overview, IBM Systems Journal 25: 1, pp. 4-19 (1986).

(昭和 63 年 8 月 15 日受付)

訂

本誌第 29 巻 12 号 (1988) pp. 1482~1485 に掲載されました「汎用中・大型機 IBM S/370」の著者中谷登志男氏の申し出により, p. 1484 の右段を以下のとおり削除および訂正します.

(i) 20~25 行まで

「これらのベクトル演算命令が……………新しい結果が生成される」を削除.

(ii) 27 行目

「6 倍」を「2 倍」に訂正.

【訂正理由】

現在のアーキテクチャ (参考文献 7) では, チェイニング (chaining) がサポートされていないため.

正

本誌第 29 巻 11 号 (1988) pp. 1334~1340 に掲載されました伊藤 昇, 熊白侃彦氏の標題「OSI の実現とその課題 (II) メッセージ指向型文書交換システム (MOTIS)」を表紙と目次において, 事務局の校正ミスにより文書交換システムと印刷してしまいました. お詫びと共に訂正いたします.