FinTech時代、銀行系システムはどうあるべきか(2):

FinTech時代の今、COBOLやPL/I、メインフレームが勘定系システムで必要な理由

本連載では、銀行系システムについて、その要件や歴史を整理しつつ、スマートフォンを使う銀行取引やブロックチェーンなど、新しい技術が及ぼす影響を考察していきます。今回は、メインフレームで COBOL や PL/I が選定された理由やメインフレームの CPU が勘定系システムに使われ続ける理由について、演算や暗号化、圧縮機能の面から解説します。[星野武史, 日本アイ・ビー・エム株式会社]

本連載「FinTech 時代、銀行系システムはどうあるべきか」では、銀行系システムについて、その要件や歴史を整理しつつ、スマートフォンを使う銀行取引やブロックチェーンなど、新しい技術が及ぼす影響を考察していきます。

連載第 1 回の「若手が知らないメインフレームと銀行系システムの歴史&基礎知識」では、銀行オンラインシステムの歴史を振り返りました。第 3 次オンラインシステム構築当時、銀行勘定系システムのトランザクション量を処理できるシステムはメインフレーム以外にはありませんでした。歴史の中でいわゆるオープン系のシステムが勘定系として使われてきていることにも触れました。

では、なぜまだメインフレームが使われているのでしょうか。言語が選定された時代背景を振り返った後、メインフレームのハードウェア機能で特徴的な演算、暗号化、圧縮機能について検証していきます。

●今回の主な内容

- メインフレームで COBOL や PL/I が選定された理由
- CPU と演算の関係～メインフレームの CPU は銀行の預金計算に向いている～
  - 10 進数の表現方法についての基礎知識
  - 2 進数と 10 進数のおさらい～ 2 進数では小数計算は不向き
  - メインフレーム、COBOL や PL/I は 10 進数での計算に有利
- CPU の暗号化機能～メインフレームでは暗号処理も高速～
- CPU での圧縮機能～メインフレームは大量データの処理に向いている～

メインフレームでCOBOLやPL/Iが選定された理由

2016 年現在、メインフレームで稼働する銀行勘定系システムで使用されている言語は COBOL と PL/I があります。PL/I は科学技術計算向けの FORTRAN、事務処理向けの COBOL の長所を取り入れて開発された言語です。

第 3 次オンラインシステムでは、第 2 次オンラインシステムの反省から、開発生産性と保守のしやすさを考慮してプログラミング手法を変えようとしていました。その候補となったのが 1970 年代に提唱された「構造化プログラミング」です。第 3 次オンラインシステムが計画された 1980 年代前半に、標準で構造化プログラミングが可能であった言語は PL/I でした。COBOL は 1985 年の国際規格により構造化プログラミングが可能となり、言語の選択肢は増えました。現在さまざまなシステムで多く使われている C が標準化されたのが 1990 年であり、Java の JDK 1.0 が登場するのは 1996 年になってからです。このような時代背景から、第 3 次オンラインシステムでは COBOL や PL/I が使われるようになったのです。

第 3 次オンラインシステムは前回振り返ったように「半永久のシステムライフ」を持ち、現在も使用されています。そのため、稼働する業務プログラムを開発する言語も積極的なメリットがないと変更されないため、現在でも銀行勘定系システムでは COBOL や PL/I が使われています。

では、なぜメインフレームが使用され続けているのか、まずは演算機能について確認していきます。

CPUと演算の関係～メインフレームのCPUは銀行の預金計算に向いている～

銀行の預金の計算は 10 進数で行われており、預金の入金、出金では加減乗除、利息や利子の計算では乗除算が使われています。金融派生商品の開発評価には 10 進数以外も活躍していると思いますが、消費者の目に見える数値は 10 進数で表記されるものだけです。

メインフレームの前身である「会計機」（「タビュレーティングマシン」。日本では「パンチカードシステム」と呼ばれていました）で扱う数値は 10 進数です。最初の第 1 次オンラインシステムで使われた「IBM 1410」では加減乗除は 10 進数のみ。同時代に科学技術計算専用として利用されていた「IBM 7090」では 2 進数のみを使用して計算をしていました。

1964 年に登場した「System/360」で使用されたオペレーティングシステム「OS/360」になって初めて、10 進数加減乗除計算の命令が 2 進数加減乗除計算と同時に標準命令として採用されました。つまり 10 進数の商用計算と 2 進数の科学技術計算の両方ができるいわゆる汎用コンピュータが登場しました。

10 進数の表現方法についての基礎知識

まずはメインフレームで 10 進数をどのように表現しているかを見てみましょう。

メインフレームでは、文字コードとして、1 バイト（8 ビット）で表現する「EBCDIC」（Extended Binary Coded Decimal Interchange Code、拡張 2 進化 10 進コード）を使用しています。EBCDIC 拡張である漢字を表現する 2 バイト部分はベンダーによって文字コードが異なり互換性がありませんが、ここで述べる数字を含めた 1 バイト部分はほぼ同一です。

10 進数は、「ゾーンフォーマット」または「パックフォーマット」で表現されます。

ゾーンフォーマットを直接操作する命令はなく、もっぱら入出力や数値の編集のために使用されます。例えば、10 進数で「+1234」をゾーンフォーマットで表した場合、図 1 のように 16 進数では「F1F2F3C4」と表現されます。

ゾーンフォーマットは 1 つのバイト（図 1 では「F1」「F2」「F3」「C4」）の右端 4 ビットと左端 4 ビットに分かれ、右端を「数字ビット」（図 1 では 16 進表記で「1」「2」「3」「4」、2 進表記で「0001」「0010」「0011」「0100」）、左端を「ゾーンビット」（図 1 では 16 進表記で「F」「F」「F」「C」、2 進表記で「1111」「1111」「1111」「1100」）と呼びます。最右端の 1 バイト（図 1 では「C4」）の左端 4 ビット（図 1 では 16 進表記で「C」、2 進表記で「1100」）はゾーンビットの替わりに符号（図 1 では「+」つまり「正の数」を表す。詳細は後述）となることもあります。

パックフォーマットは 10 進数演算命令で使用される形式で、最大 16 バイト 31 桁まで利用可能です。「億」＜「兆」＜「京」＜「垓」＜「じょ」＜「穣」＜「溝」と表現される「『1 溝』よりも『1』少ない数」まで表現できますが、もはや天文学的数字で現実感がありません。乗数および除数は 8 バイトまでの長さになります。8 バイトで 15 桁 100 兆まで表現可能なので、整数の乗除であれば日本の国家予算約 100 兆円を乗数または除数にできる長さが確保されています。どれだけ大きな数値が扱えるか実感できたでしょうか。

「+1234」をパックフォーマットで表した場合、図 2 のようになります。データはバイト単位で格納されるため、上位に 0 が付加され、16 進数で「01234C」と表現されます。

1 バイト当たり 2 個の 10 進数数字で構成されていますが（図 2 では 16 進表記で「01」「23」「4C」）、最右端の 4 ビット（図 2 では 16 進表記で「C」）は符号（後述）となります。

符号は正を「C」（1100）、負を「D」（1101）で表わします。ゾーンビットである「F」（1111）、「A」（1010）、「E」（1110）も代替符号の正として扱われ、「B」（1011）は代

図 1 「+1234」をゾーンフォーマットで表した例

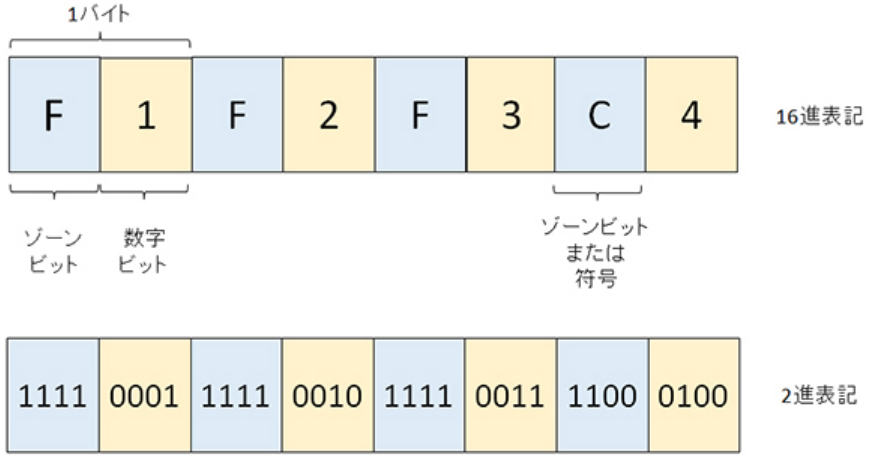
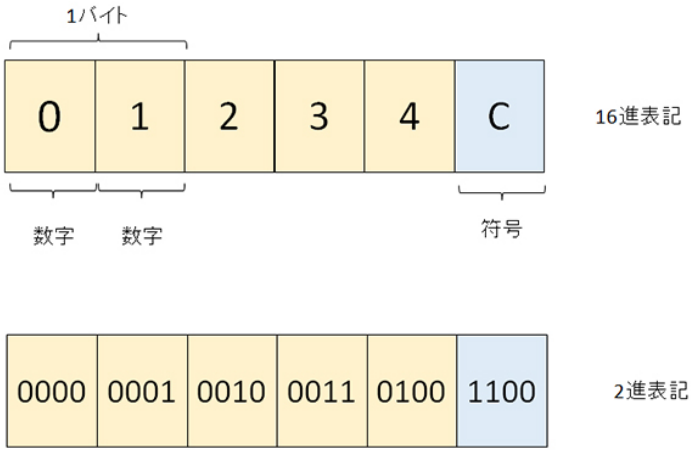


図 2 「+1234」をパックフォーマットで表した例



替符号の負として扱われます。ただし、代替符号は命令の結果としては付かず C（正）や D（負）の符合が付けられます。

なお Intel アーキテクチャでも、パック 10 進数の数値部分は同じ表現ですが、データの格納方法が「リトルエンディアン」となり符号表記も異なります。

2 進数と 10 進数のおさらい～2 進数では小数計算は不向き

0 と 1 のみで表現される 2 進数では、図 3 のように加算も乗算も 0 と 1 の組み合わせの 4 通りしかありません。このため単純な論理装置の組み合わせで回路を構成できるため、高速に計算できるのは 2 進数の演算です。

しかし 2 進数表現では、小数の表現が苦手な場合が多いのです。図 4 のように、10 進数で「0.5」（以下、「0.5(10)」と表記）は 2 進数では「0.1」(以下「0.1(2)」と表記）できますし、「0.25(10)」は「0.01(2)」、「0.125(10)」は「0.001(2)」と表記でき、10 進数から 2 進数に変換し、10 進数に再び変換しても問題ありません。

しかし、例えば「0.1(10)」の場合、「0.0001100110011……(2)」と無限小数となってしまうことは広く知られた問題です。たとえ倍精度の 2 進浮動小数点を使用したとしても「0.1 の近似値」しか表現できません。このため、単純に消費税の計算をしただけで計算結果が異なる例はいくらでもあり、2 進数では小数計算は不向きといえます。

メインフレームでは 10 進数演算の命令がありますが、他のコンピュータではどうしているのでしょうか。

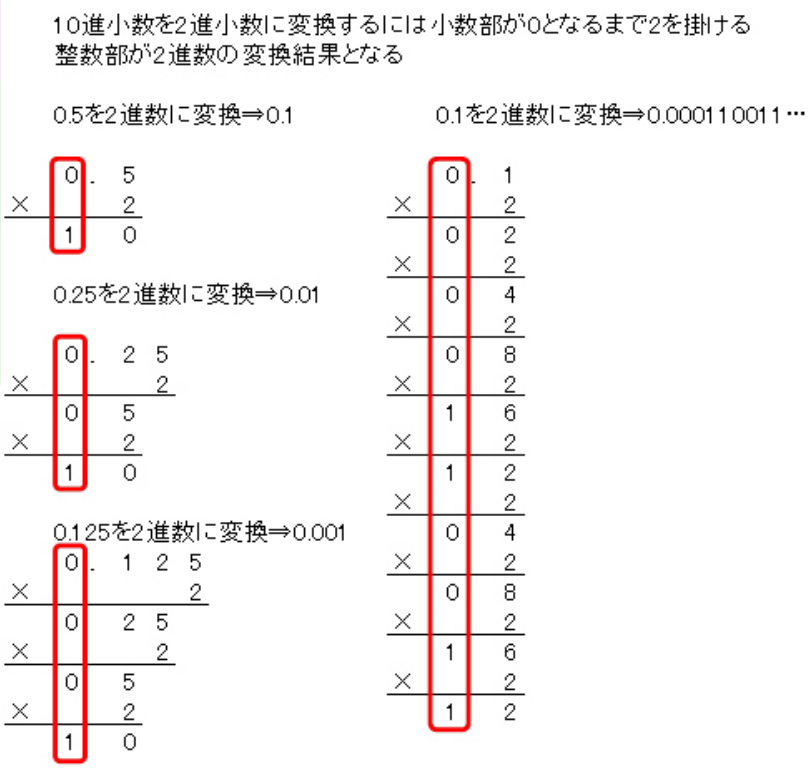
Intel アーキテクチャのマニュアル PDF「Intel 64 and IA-32 Architectures Software Developer's Manual」を見

図 3　2 進数の加算・乗算一覧

+	0	1
0	00	01
1	01	10

×	0	1
0	0	0
1	0	1

図 4　10 進小数を 2 進小数に変換する例



てみると、「10 進演算命令」という項目がありますが、演算後にこれらの命令を実施して 10 進表記の形式に補正（adjust）する命令を指しています。そのため、内部的には 2 進数で演算が行われていることが分かります。さらにメインフレームでは 10 進数関連命令は 64 ビットモードでも使用できますが、Intel アーキテクチャでは 64 ビットモードでは使えないため 32 ビットのプログラムを使わざるを得なくなります。10 進数命令に関しては制約があるといえます。

メインフレーム、COBOL や PL/I は 10 進数での計算に有利

2 進数で計算せずに 10 進数で計算できるように、メインフレームの勘定系システムで使用されている COBOL や PL/I といった言語は、古くから 10 進数浮動小数点のデータ形式を定義し使用できます。それ以外の言語でも、例えば Java では「デシマル型」（java.math.BigDecimal クラス）があって浮動小数点は扱えるようになっています。

これらの演算は浮動小数点演算の標準規格（IEEE754）をサポートしているので、2016 年の現在ではメインフレームとそれ以外のアーキテクチャのコンピュータで機能上の差はなくなりつつあるといえます。しかし、2008 年以降に登場したメインフレームでは 10 進数浮動小数点の演算をハードウェア的にサポートするようになり、演算処理が格段に高速化し Intel 系の CPU に差を付けているようです。

また RISC プロセッサでも、10 進数浮動小数点のハードウェアサポートを装備するようになってきています。ただし、ハードウェアによる浮動小数点演算の高速化のメリットを享受するためには、ハードウェア機能をサポートするコンパイラでプログラムを再コンパイルする必要があります。

CPUの暗号化機能 ～メインフレームでは暗号処理も高速～

銀行の IC キャッシュカードの正当性を示す認証データや、マイナンバーなどの機密情報は暗号化されています。暗号化された状態で ATM や営業店端末から送り出され勘定系システムで処理されます。暗号・復号処理はソフトウェアで行うことができますが、複雑なアルゴリズムで暗号化されているので、CPU 負荷が高くなります。そのため、このようなケースではメインフレームのハードウェア暗号化機能を使用して CPU 負荷を低減するとともに処理時間を短縮しています。

メインフレームで処理する場合、図 5 のような処理形態となり、メインフレームの業務処理の直前までデータは暗号化された状態であるため、セキュリティを確保することができます。

メインフレームでは、たとえ暗号を復号している状態のメモリの状況を DUMP コマンドによって取得した場合であっても、暗号鍵は一切見ることができない仕組みが保たれています。鍵が格納されているデータセットも暗号化されていますし、暗号鍵を使用して復号する装置は耐タンパー性を備えていて、装置を取り外したり、電池を抜き取ったり、装置を分解したりすると暗号鍵はクリアされてしまう構造になっています。メインフレームのシステム一式を無傷で盗まれでもしない限りは暗号文の安全性は保たれているといえます。

図 5 のようなケースでは、勘定系システム以外で認証データを処理する認証サーバ形態を採る場合もあります。このような場合、認証サーバと勘定系システムの間では復号したデータの交換がないことが前提となります。

実はメインフレームでは、暗号化機構の歴史は銀行勘定系オンラインと同じ位の歴史があります。例えば、IBM のメインフレームにおける現在までの主な技術的トピックは以下の通りです。

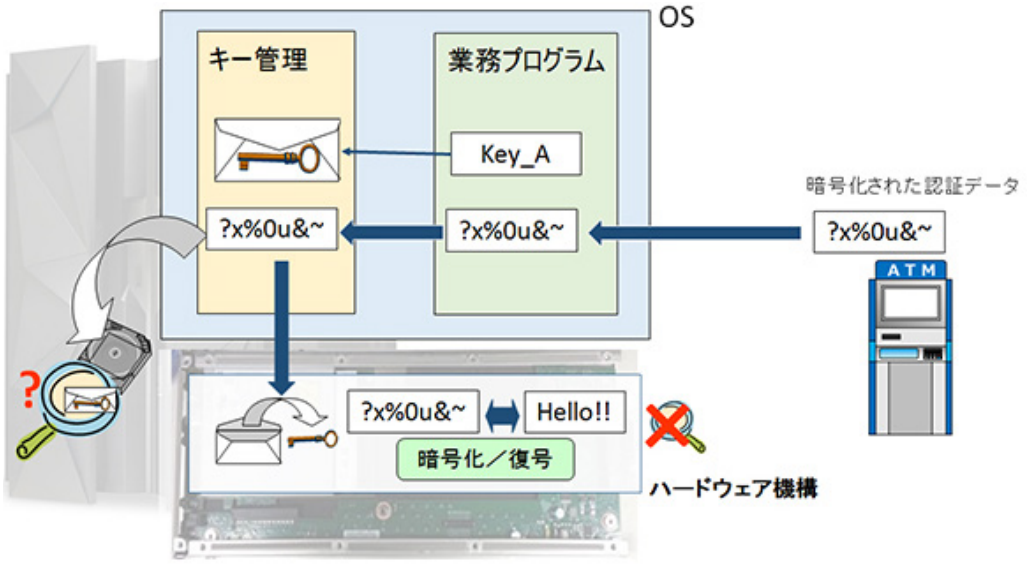
- 1970 年：ハードウェア暗号化処理機構の実装
- 1991 年：OS 機能の一部としてキー管理機能を実装
- 2001 年：電子証明書生成と認証局機能を実装
- 2005 年：メディア暗号化ソリューションが登場
- 2009 年：ディスク装置に自己暗号化機能搭載
- 2012 年：PKCS #11 公開鍵暗号方式標準への対応
- 2016 年：Visa Data Secure Platform（Visa DSP）、ポイントツーポイント暗号化（P2PE）への対応

このように、メインフレームでは暗号化に対応した機能が年々追加され、現在でも進化していることが分かると思います。もちろん RISC プロセッサや Intel プロセッサでも暗号化機能を実装していますが、メインフレームのように幅広くは対応していないようです。

CPUでの圧縮機能 ～メインフレームは大量データの処理に向いている～

銀行オンラインで処理される多くのトランザクションのログやデータベースのバックアップやイメージコピーを取得する場合、テープが利用されます。データ量が少ない場合は特に何も細工せずにデータをテープに取得するのでも問題あ

図 5　メインフレームでの暗号処理概要



りませんが、大量データの場合はいかに高速のテープ装置を使用したとしても時間がいくらあっても足りなくなります。

例えば、銀行オンラインで使用されているデータベースは、最大 2048 に分割でき、分割されたデータベースの最大容量は 4GB ですので、4×2048 ＝ 8192GB つまり 8TB まで拡張できます。このようなデータベースは複数種類あるので、いかに大量のデータを持っているか分かると思います。

これらのデータベースのイメージコピーを取得する場合、分割された単位で取得したとしても時間がかかりそうだということが分かるでしょう。時間を短縮するには高速のテープ装置と高速の I/O インターフェースを使用する以外に、転送するデータ自身を圧縮して転送するデータ量そのものを減らす手段が有効です。

ソフトウェアでデータ圧縮すると CPU パワーが必要となるため、圧縮率と CPU コストのバランスを取ることが必要でした。しかし、メインフレームに装備されるようになったデータ圧縮機能を使用すれば、今までよりはるかに少ない CPU コストしか掛けなくても短時間で圧縮が可能となり、従来に比べてデータの圧縮が簡単に行えるようになっています。

メインフレーム以外でもハードウェアでデータ圧縮機能を備えたサーバも登場していますが、全てのサーバにこの機能は搭載されてはいないようです。

さらに 2016 年に登場したメインフレームでは、OS 自体の機能として圧縮を積極的に活用するようになり、圧縮アルゴリズムの見直しによって効果的で高速な圧縮を提供するとともに、多くのプラットフォームで使われている zlib ベースの圧縮と互換性のある圧縮をサポートするようになって、ハードウェア圧縮による実行時間の短縮と CPU 使用率の低減が、より一層図られています。

現在は、これらの圧縮機能はログのアーカイブやデータベースのイメージコピー、その他のデータのバックアップなどに限定された活用方法でしたが、今後はビッグデータ分析の分野で活用されることが期待されています。

次回は、スマホを活用した FinTechアプリと勘定系システムの連携について

今まで見てきた事柄は、メインフレームの特徴・メリットのほんの一部にしかすぎません。ハードウェアだけではなく、ソフトウェアもバージョンを重ねるごとに進化し、地道な努力の積み重ねでパフォーマンスも飛躍的に向上しています。今後の銀行オンラインシステムの進化に注目したいですね。

次回はいよいよ FinTech の話になります。API を通じたスマホを活用した FinTech アプリと長年の歴史を持つ勘定系システムの連携がどうなっていくのか。専門の筆者にバトンタッチして解説します。お楽しみに。

お問い合わせ先▶ **日本アイ・ビー・エム株式会社**　〒103-0015 東京都中央区日本橋箱崎町19-21　http://ibm.biz/zsystems_jp
IBMアクセスセンター　0120-550-210

※この冊子は、@ IT（http://www.atmarkit.co.jp/）に 2016 年 10 月に掲載されたコンテンツを再構成したものです。　http://www.atmarkit.co.jp/ait/articles/1610/20/news010.html