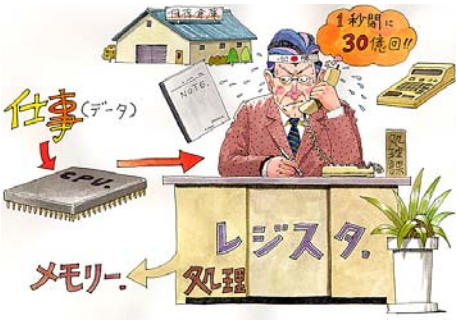
第２回　CPU の計算のしかたは意外に単純、速さで勝負

第二回目は、CPU がデータをどのように処理するかを解説します。
CPU が行っている処理は、意外に単純です。単純だからこそ、高速動作ができるのです。
今日では処理速度を上げるために、いっそう単純化する傾向にあります。CPU の論理的な動作を通じてどのようにそれが実現されるかを説明します。



■CPU の内部構成

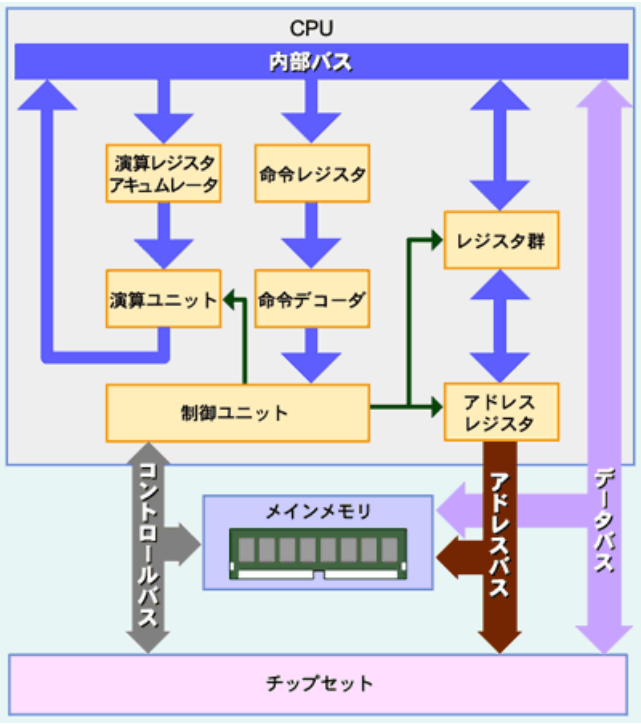
前回は IC（集積回路）の１種として CPU の構造やしくみを紹介しましたが、今回は、CPU のコンピュータとしての構造やしくみを紹介します。

・CPU の機能ブロックの名称とはたらき

数百万から数千万個のトランジスタで構成される論理回路を組み合わせて、CPU の内部にはデータを処理する仕組みが作られています。次の図は、単純化した CPU の内部構成です。各部のはたらきを簡単に説明しておきます。

なお、図中の「バス」と「レジスタ」については、次の項で詳しく説明します。

CPU の内部構成



内部バス

CPU 内の各ブロックと外部とやりとりするデータバスを結んでデータや命令をやりとりします。

データバス

読み込み・書き込み用のデータをやりとりします。

アドレスバス

CPU が読み書きするデータや命令を格納したメモリや I/O のアドレスを送信します。

コントロールバス

CPU からメモリや I/O、メモリから I/O へのデータ転送をコントロールします。

アドレスレジスタ

CPU が命令やデータを読み書きするときに、目的のメモリや I/O ポートのアドレスをアドレスバスに出力するレジスタです。

レジスタ群

CPU が作業するための小容量・高速なメモリ群です。CPU は、このレジスタにデータと命令を読み込んで処理を行います。

命令レジスタ

命令を記録するための専用レジスタ。このレジスタの読み込まれた内容を命令デコーダが翻訳します。

命令デコーダ

命令レジスタに読み込まれた命令を、制御情報に置き換えて制御ユニットに通知します。

制御ユニット

命令に応じた処理を行う CPU 内部の回路を制御します。命令デコーダから送られる制御情報で制御線のスイッチをオン／オフして各機能ブロックやコントロールバスに出力します。

演算レジスタ

演算処理専用のレジスタです。演算結果を置いたり、データを一時的に保存したりします。加減算などの基本的な演算を行う回路が付属しています。アキュムレータと呼びます。

演算ユニット

演算レジスタに読み込まれているデータに所定の演算処理をします。
演算論理装置または ALU(Arithmetic Logic Unit)とも呼ばれます。

・バスは CPU の動脈

バスとは CPU の内部の処理装置どうしや外部のメモリやチップセットとを結ぶ「信号線の束」のことをいいます。データバスには CPU が受け取るデータだけでなく、CPU から送出されるデータも流れます。このように複数の装置がデータの送受信をするために、共有する信号線の束がバスなのです。

CPU 内部には「内部バス」、「データバス」、「アドレスバス」と「コントロールバス」の４つのバスが装備されています。

「内部バス」は CPU 内部の処理装置やデータバスに接続され、内部どうしと内部と外部のデータや命令のやりとりに使用されます。

「データバス」は CPU とメモリや I/O を結び、読み書き用のデータをやりとりするための双方

向バスです。アドレスバスとコントロールバスとセットで動作します。アドレスバスで指定されたメモリや I/O に対して書き込むまたは読み込むデータや命令をやりとりします。読み込みか書き込みかはコントロールバスの信号で判別します。

「アドレスバス」は CPU から外部への一方通行のバスです。CPU のアドレスレジスタから読み書きするデータや命令のアドレスが出力されます。

コントロールバスは、クロックパルスや制御シグナルなどの多種の信号線を含んでいます。動作のタイミングや読み込みか、書き込みか、などを指示するための制御信号をやりとりしています。

データバスとアドレスバスが一度にやりとりできる情報量をビットで表したものがバス幅です。歴代の CPU のビット数（アーキテクチャー）にバスをまとめました。

CPU のビット数とバス幅

	4004	8080	8086	i486	Pentium /2/3
アーキテクチャー	4 ビット	8 ビット	16 ビット	16 ビット	32 ビット
データバス幅	4 ビット	8 ビット	16 ビット	32 ビット	64 ビット
アドレスバス幅	8 ビット	16 ビット	16 ビット	32 ビット	32 ビット

16 ビット CPU 8086 はデータバス幅 16 ビットですから、一度に 16 ビットのデータを読み込むことができます。アドレスバス幅は 16 ビットですから、16 ビットのアドレスを指定することができます。アドレスバスで指定できる範囲のことをアドレス空間と言います。16 ビットの場合、64K ビット（2^{16}=65534 ビット）となりますが、8086 ではセグメントレジスタによるページ切り替えのような機能で 4 ビット分拡張しており、1MB のアドレス空間を持っています。このアドレスバスが 32 ビットになると、アドレス空間は 4GB（ギガバイト=10^9 バイト）まで拡大します。

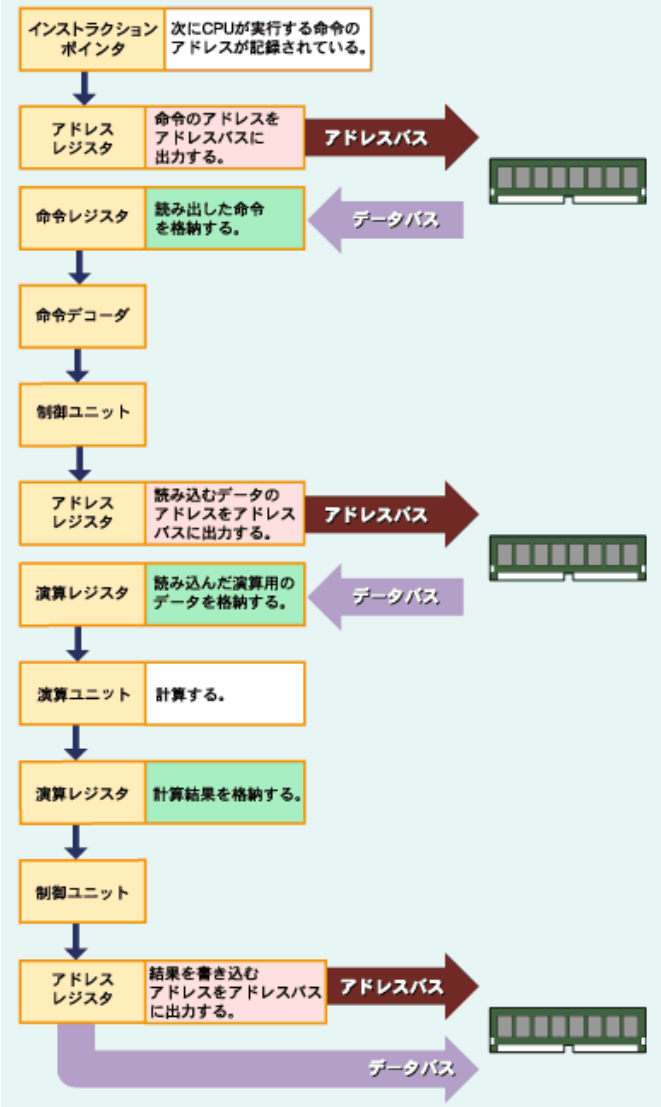
・レジスタは CPU の仕事場所

「レジスタ」は CPU 内部の作業用のメモリで、小容量ですが高速に読み書きできます。

フリップフロップという論理回路の一種で構成されています。フリップフロップ回路は、入力信号があるたびに 1 と 0 を交互に切り替え、次の入力信号が来るまでその状態を保持（記憶）します。動作が単純なので高速に動作しますが、1 ビットに 1 回路が必要で、大きな容量のメモリには向いていません。高速小容量のメモリであるレジスタには最適なのです。フリップフロップは、他にも SRAM（スタティック RAM）や CPU の内部キャッシュに使われています。

CPU は、すべての仕事をこのレジスタを使って行います。

次の図は CPU が演算動作をするときにレジスタを使用する様子を示したものです。



CPU は、「レジスタへの読み込み」、「演算処理」を行って、「レジスタからの出力」するのが仕事の全てです。レジスタは CPU の仕事の道具であり、仕事場所でもあります。そのため、１つひとつに名前が付いており、１つひとつに専用の機能があります。Pentium が内蔵しているレジスタを例に説明します。Pentium のレジスタは 1 本が 32 ビット構成です。これを 16 ビットで使用したり、8 ビット×2 本で使用したりすることができます。

汎用レジスタ

データの記憶や演算に使用できる汎用のレジスタです。演算レジスタとしてもアドレスレジスタとしても使用できます。

ベースポインタ

メインメモリのアドレスのオフセットを格納します。ベースレジスタ、ベースポインタとも呼ばれます。

スタックポインタ

レジスタの値を一時的に保存したり、サブルーチンなどの戻り先を保存したりするためのメモリ領域のアドレスを示します。

インストラクションポインタ
CPU が次に実行する命令のアドレスを格納します。プログラムカウンタともいいます。
フラグレジスタ
桁あふれや演算結果の正負の記号を自動的に読み込み、あらかじめ決められたビットを変化させます。
セグメントレジスタ
メモリのアドレス範囲を選択するのに使います。セグメントセレクタともいいます。

以上のレジスタは、多少名称が異なっている場合もありますが、どの種類の CPU にも装備されています。レジスタには、CPU ごとに機能強化のために独自なものがあります。Pentium にも次のような独自のレジスタが装備されています。

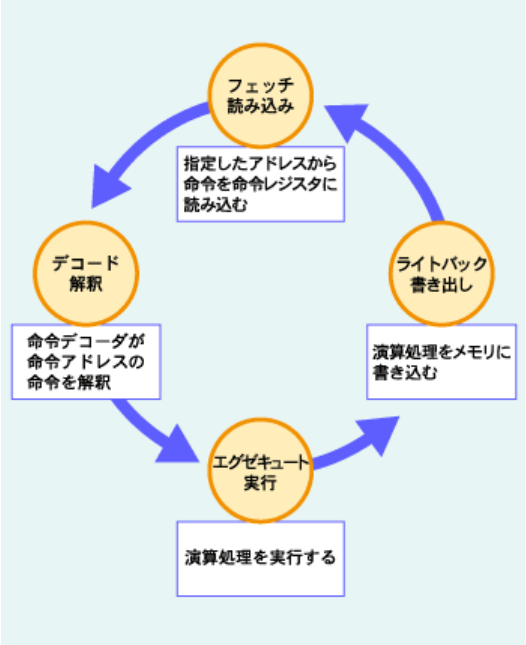
浮動小数点レジスタ
double 型変数など、浮動小数点を使用するプログラムが使用するレジスタ。
MMX レジスタ
一度に複数の整数演算を行うプログラムが使用するレジスタ。64 ビット×8 本
SSE レジスタ
一度に複数の浮動小数点演算を行うプログラムが使用するレジスタ。128 ビット×8 本
システムレジスタ
OS が使用するレジスタ。Pentium のメモリやタスクの管理に使われます。一般のアプリケーションが使うことはできません。
デバッグレジスタ
主にデバッガが利用するレジスタ。

■単純な処理を高速に実行

複雑に見えるアプリケーションソフトの動作時でも、CPU はその動作を 30 億分の 1 秒（クロック 3GHz の CPU）で実行する単純な動作に分解して実行しています。その時の動作の単位を、CPU の動作サイクルといいます。

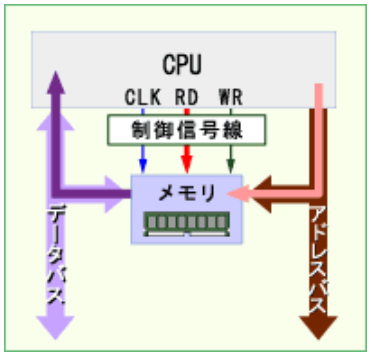
・CPU の内部動作サイクル
次の図は CPU 内部の動作サイクルです。CPU はクロック 1 回ごとに 1 動作ずつ進め、クロック 4 回で 1 サイクルになります。しかし、Pentium など一部の CPU では、内部構造を工夫して（パイプラインやスーパースケーラなど、詳しくは次回説明します）1 クロックで 4 動作進むようにしています。
クロック 3GHz の CPU では、１秒間に 30 億回（3GHz＝3×10^9）の動作が行われることになります。１ずつカウントアップしても 1 秒間に 30 億まで数えられることになります。

CPU の動作サイクル

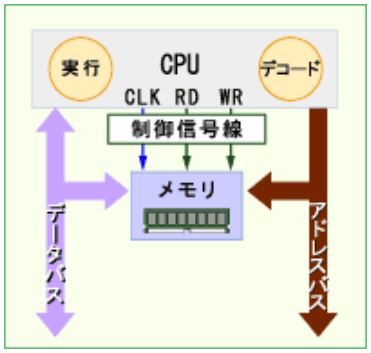


・CPU の外部動作

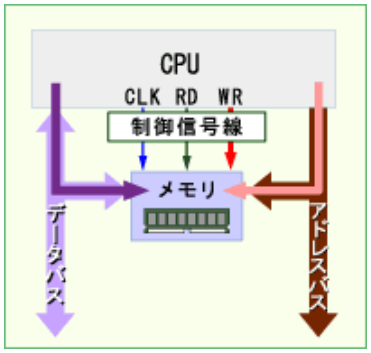
次の図は、今度は CPU が動作中に、外部にはどのように動作しているように見えるかを示しています。



レジスタへの読み込み



命令の解釈と実行



メモリへの書き込み

※CLK はクロック信号：常時出力されています。

CPU はデータバスから命令やデータを読み込んで、計算し、再びデータバスに書き出す動作をしています。CPU がメインメモリや I/O ポートに対して行っているのは、これだけの動作です。この動作を高速に繰り返しています。

PC が複雑な処理や大きな数値の計算、高度な計算を行えるのは、この単純な動作を高速に行うことができるからです。クロック 3GHz の CPU では上記の動作を 30 億分の 1 秒（3×10^9 分の 1 秒）で実行しています。

実際のメモリへの読み書きは CPU にとっては時間がかかる作業ですが、最近では CPU 内部の高速なキャッシュに保存することで高速化が実現しています。その他にも、CPU 内部ではフラグレジスタにプログラムのジャンプやサブルーチンの戻り先を保存したり、アドレスレジスタによりメモリのアドレスをコントロールしたり、プログラムメモリに対してアドレスを自動的に増やしていくインストラクションポインタなど多くの機能が動作しています。

■命令の処理

CPU がどんな命令で動作しているかを紹介しておきましょう。

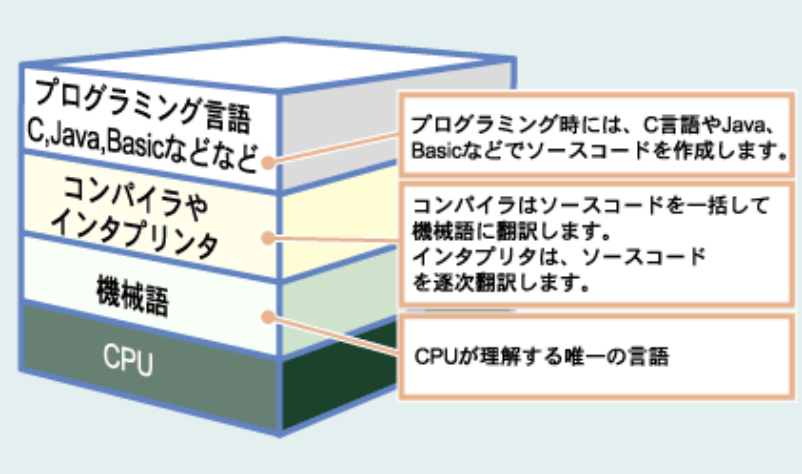
CPU に直接命令を伝えることができるのが機械語です。その機械語を人間が読めるように最低限の英数字で記述できるようにしたのがアセンブラです。

・機械語

機械語は数字の 0 と 1（電気信号のオフとオンに相当）からなる 2 進数か、または 4 ビットずつを 16 進数で表現します。

PC の黎明期の ROM-BASIC 搭載の PC には、機械語モニタとよばれる機械語入力エディタが付属しており、比較的簡単に機械語を見ることができましたが、今日では簡単に見ることはできませんし、また、今日の CPU は OS やアプリケーションが複雑に組み合わさって動作しており、安易に機械語レベルでコードを実行することはお勧めできません。また、現実的に PC のアプリケーションが機械語でプログラミングされることはほとんどありません。

これは機械語が CPU に依存しており、機械語でプログラミングするとなると CPU の種類ごとにプログラムが必要になります。そのため、今日では、「C」や「Basic」などのプログラミング言語でソースコードを作成し、それぞれのコンパイラやインタプリタによって機械語に翻訳する方法が一般的です。こうすればアプリケーションのソースコードは他の CPU と共通で使うことができ、コンパイラやインタプリタだけ CPU 専用のものを用意すればよいことになり、プログラミングが効率的になります。



A レジスタの内容に B レジスタの内容を加えるという処理を機械語では、次のように記述します。

加算する	A レジスタ	B レジスタ	
00010110	110001	110010	（2 進数）
16H	31H	32H	（16 進数）

上記のような、1 と 0 だけまたは 16 進の数字だけのプログラミングは実用的ではありません。

そこで、機械語をもう少し人間が分かるように工夫されたのがアセンブラです。

・アセンブラ

機械語と、アルファベットの文字または文字列を、1 対 1 で対応させたのがアセンブラです。

アルファベットにした部分をニーモニック（mnemonic）と呼びます。ニーモニックは人間が読んだときに、分かりやすいまたは連想しやすい単語になっています。

例えば、mov→ コピーする(move)、cmp　比較する(compare)、jmp →　ジャンプする(jump)などがあります。

この命令の文字列を 2 進数の命令や引数に「翻訳」すれば機械語になります。

前述の機械語の「A レジスタに B レジスタの内容を加える」という処理をアセンブラで記述すると次のようになります。

加算する	A レジスタ	B レジスタ	
ADD	A	B	（アセンブラ）

アセンブラで記述されたプログラムも最終的には機械語に翻訳します。この作業をアセンブルといいます。ベテランの中にはハンドアセンブルという方法をご存じの方もおありでしょう。その通り手作業による翻訳です。また、機械語からアセンブラに翻訳することを逆アセンブルまたはディスアセンブルと呼びます。リバースエンジニアリングの 1 種です。

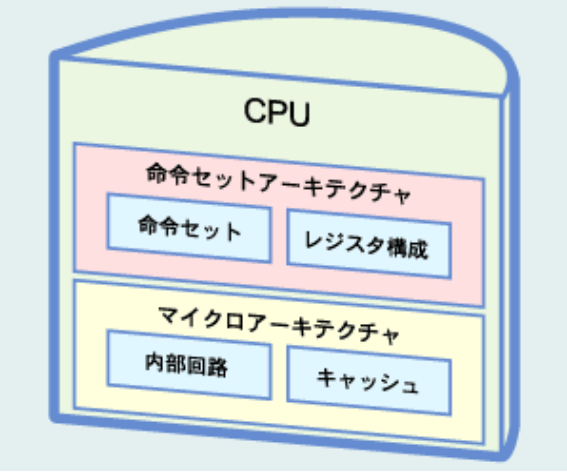
・命令セットとは

CPU で使用できる機械語命令の集まりを命令セットといいます。

x86 系 CPU は、8086 から Pentium 4 に至るまで命令の互換性を保っています。これにより、OS が MS-DOS から Windows3.1、WindowsXP まで一定の互換性を維持しているのです。

しかし、ハードウェアは変化し続けており、旧機種と同じというわけにはいきません。

互換性を保った命令部分を命令セットアーキテクチャー、内部構造やキャッシュなどハードウェア部分をマイクロアーキテクチャーと呼んでいます。



命令セットアーキテクチャーとマイクロアーキテクチャー

MS-DOS や WindowsPC の主流である x86 系、Macintosh に使用されてきた PowerPC など、CPU の種類ごとに命令セットは異なります。同じ CPU でも、OS によっては使用する命令セットは異なります。例えば、Intel 社製の CPU で動作する MacOS は、使用する命令セットが Windows が使用するものとは一部異なると思われます。命令セットには Windows 専用のものがあるからです。

CPU は次々に新製品が登場して、命令セット自体も拡張されていますが、互換のある命令セットだけで組まれたプログラムであれば、基本的には x86 系 CPU ならどれでも動作することになっています。また、ｘ86 は Intel 社製ですが、AMD 社の CPU も同じ命令セットで動作します。これを互換 CPU と呼びます。互換 CPU は同じ機械語の命令セットを持っています。

現在、CPU は 64 ビット化への移行期にあります。ここで問題になるのが、x86 の命令セットです。30 年間にわたり互換性を保ってきたことで、命令体系がすっかり旧式なものになってしまったのです。

アセンブラの例で、「A レジスタ　＋　B レジスタ」を次にように記述しました。

```
add　　A　　B
```

ｘ86 系の命令セットでは、「A レジスタ」と「B レジスタ」の内容を足して、結果を「A レジスタ」に書き込みます。命令の後の引数（オペランドといいます）は２つまでしか使えません。通常は、この後、「A レジスタ」の内容をメモリや I/O に転送しますので、次のような命令が必要です。

```
mov　　A　　C
```

あらたにその命令を用意しなくてはなりません。

ところが、3 オペランドを実装していれば、次のように 1 つの命令で記述することができます。

```
add　　A　　B　　C
```

3 オペランドは、PowerPC などの新しい CPU には実装されています。

互換性を重視する x86 アーキテクチャーでは実現できません。x86 は、1979 年に作られた 8086 を源流とし、8 ビットの 8080 との互換性を保ちながら 16 ビット命令を追加した関係で、扱いやすい命令体系にはなっていないのです。

・x86 の次へ

x86 アーキテクチャーは旧 CPU との互換性を維持することで、ユーザーもメーカーも開発者も大きな恩恵を受けてきました。しかし、技術革新の最も激しい分野であるコンピュータの世界で、しかも CPU が 32 ビットから 64 ビットへの移行期にある今日、ｘ86 アーキテクチャーと、別れ

を考えるときが近いのかもしれません。2007 年 1 月の WindowsVista の登場が大きな転機になりそうです。

今回は命令や実際の処理という面から CPU を解説しましたが、実際にはキャッシュメモリや FPU など、基本機能以外にさまざまな機能が CPU には装備されています。次回は、そういった CPU をフォローする機能とともに、今日の CPU に採用されている注目すべき機能と、今後の CPU の動向を占う技術についてのお話しです。

【参考書籍】

大原雄介『64 ビットがわかる』株式会社技術評論社

馬場敬信『コンピュータのしくみを理解するための 10 章』株式会社技術評論社

蒲地輝尚・水越康博『はじめて読む Pentium マシン語入門編』アスキー