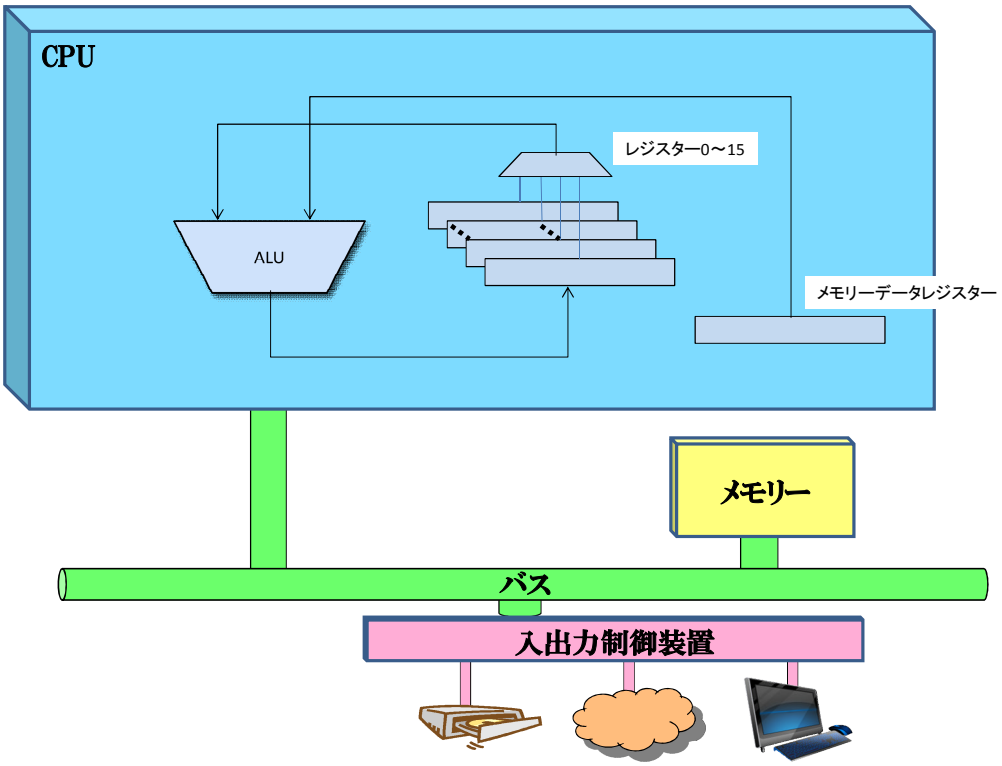
（1） CPU とは

前回の紹介の中で、CPU とメモリーの役割についてご説明しました。

CPU というのは、メモリーに格納されているプログラムを実行するために、プログラムのそれぞれの命令を実行するためのものでした。

そして、CPU の中を少し覗いてみると、下の図のようになっているということを説明しました。



実際の CPU というのはこのように単純ではなく、さまざまな機能を持ったもので構成されています。いきなりすべてを盛り込んだ図をここで示すと、理解の整理ができなくなりますので、少しずつ CPU の機能を説明してみたいと思います。

もう一度、CPU とは何かを一言で表すと、命令を実行するための機能をもったものです。さて、それでは命令を実行するということを、もう少し分解してみましょう。命令を実行する機能をステップバイステップで説明すると以下の通りになります。

① 命令をメモリーから読み出す。

② 何をする命令であるか解読する。

③ 処理するためのデータをメモリーから読み出す（メモリーデータを処理する命令の場合だけ）

④ 処理するためのレジスターを読み出す（レジスターデータを処理する命令の場合だけ）

⑤ データを処理する演算などを実行する

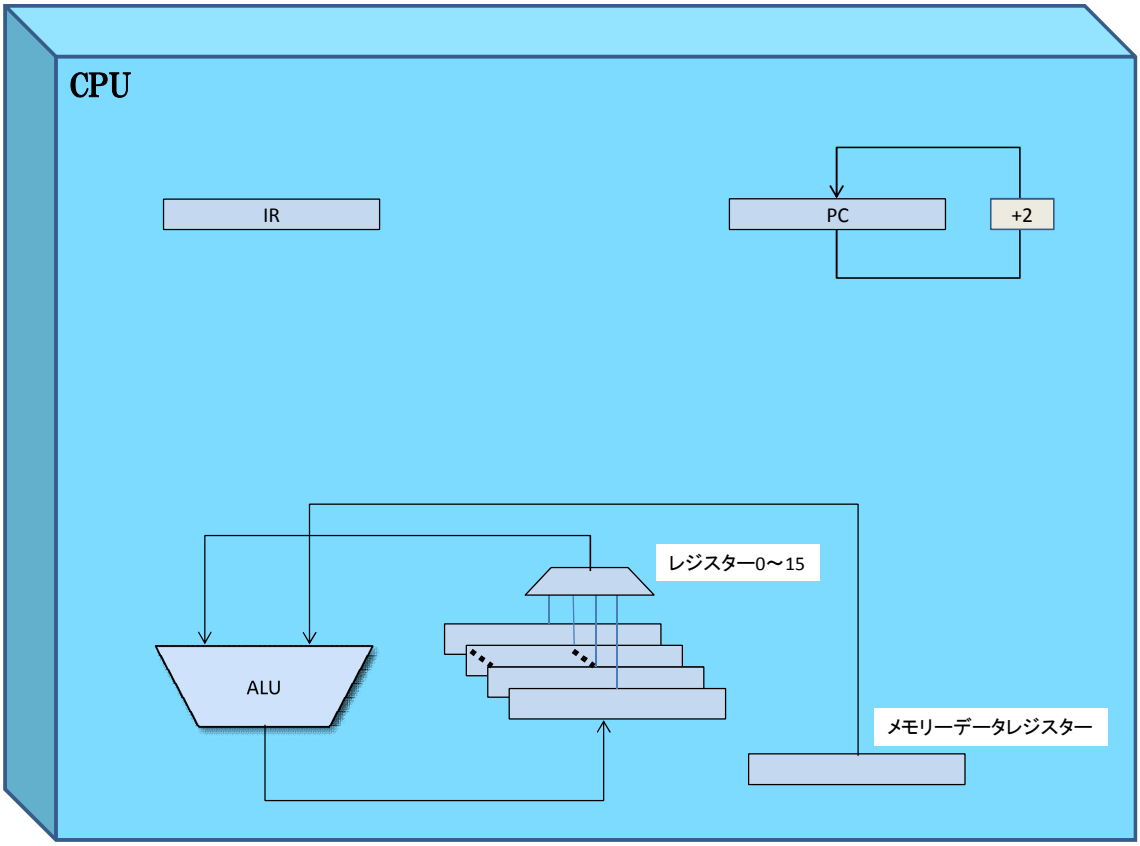
⑥ 演算結果をレジスターに格納する、またはメモリーに格納する

⑦ 次の命令のメモリーアドレスを準備し、次の命令の読み出しに備える。

以上です。③から⑥の機能は命令によっていろいろなケースがありますが、①、②、⑦については、すべての命令に共通の機能です。

さて、下の図は上の図の CPU の部分のみを取り出して、拡大したものですが、①、⑦を実現するものを追加しました。「もの」という言い方も少し格好悪いので部品と呼ぶことにしましょう。PC という部品と IR という部品と「+2」という部品が追加になっています。PC というのは、皆さんおなじみのパーソナルコンピュータの略ではありません。CPU の中にある PC とは「プログラム・カウンター」の P と C の略です。これは、現在実行している命令のメモリーアドレスを保持するためのレジスターです。IR というのは「インストラクション・レジスター」の I と R でインストラクションというのは命令という意味です。つまり、現在実行している命令のコードを保持するためのレジスターです。「+2」という部品はどういう機能を実現するためにあるかというと、前回説明した命令の例で説明しますと、下のように命令が並んでいましたね。0x58100620 というのはロード命令ですが、このロード命令は 4 バイトの長さがあります。つまり、ロード命令を実行しているときの PC は 0x500 ですが、次の命令を読み出す準備として、PC を次の命令のメモリーアドレスにする必要があります。つまり PC の内容に+4 をしなければなりません。命令の種類はこれ以外にもたくさんあり、命令の長さは 2 バイト、4 バイト、6 バイトなどがあります。（CPU によって異なりますが、代表的な CPU ではこの 3 種類の長さの命令となっています）このように 2 バイト、4 バイト、6 バイトの命令の長さを持った CPU のため、「+2」の機能を部品として持ち、ロード命令のように 4 バイトの命令の場合は「+2」機能を 2 回実行、2 バイトの命令であれば 1 回、6 バイト命令では 3 回実行します。

0x500 番地	0x58100620
	0x58200610
	0x5A100600
	0x5B200630
	0x48700508
	0x5B100600



さて、PC と IR と前回の説明の中で紹介したその他の演算実行するため部品たちがそろいましたが、まだ、全体が一体になった感じがありませんね。そこで、②の「何をする命令であるか解読する。」機能の説明をしてみたいと思います。

（2） 命令解読とは

また、部品を追加しました。その中でも特に大きなマイクロプログラムメモリーというものを説明しましょう。また、「プログラム」という言葉が出てきましたが、その前に「マイクロ」という言葉がついています。プログラムは命令を一つずつ実行して機能しますが、一つ一つの命令というのは、実は CPU がマイクロプログラムを実行することによって実現しています。例え方が少し乱暴ですが、高級言語、例えば Java や C 言語の 1 行のファンクションを処理するためには、いくつかの命令からなるプログラムで実現されています。、その Java1 行⇔プログラムと同じような関係が、命令⇔マイクロプログラムという関係です。少し感じがつかめましたか？例えば、ロード命令というのは、下に処理ステップ（①～⑥）を示しましたが、この様なマイクロプログラムを実行することによって実現しています。

① データが格納されているメモリーアドレスをメモリーアドレスレジスター（MAR）に入れ、メモリー読み出しリクエストを出す。

② メモリーから読み出したデータをメモリデータレジスター（MDR）に取り込む。

③ ALU のファンクション（実行する演算）を PASS B(B 入力をそのまま出力)とし、指定したレジスターに格納する。

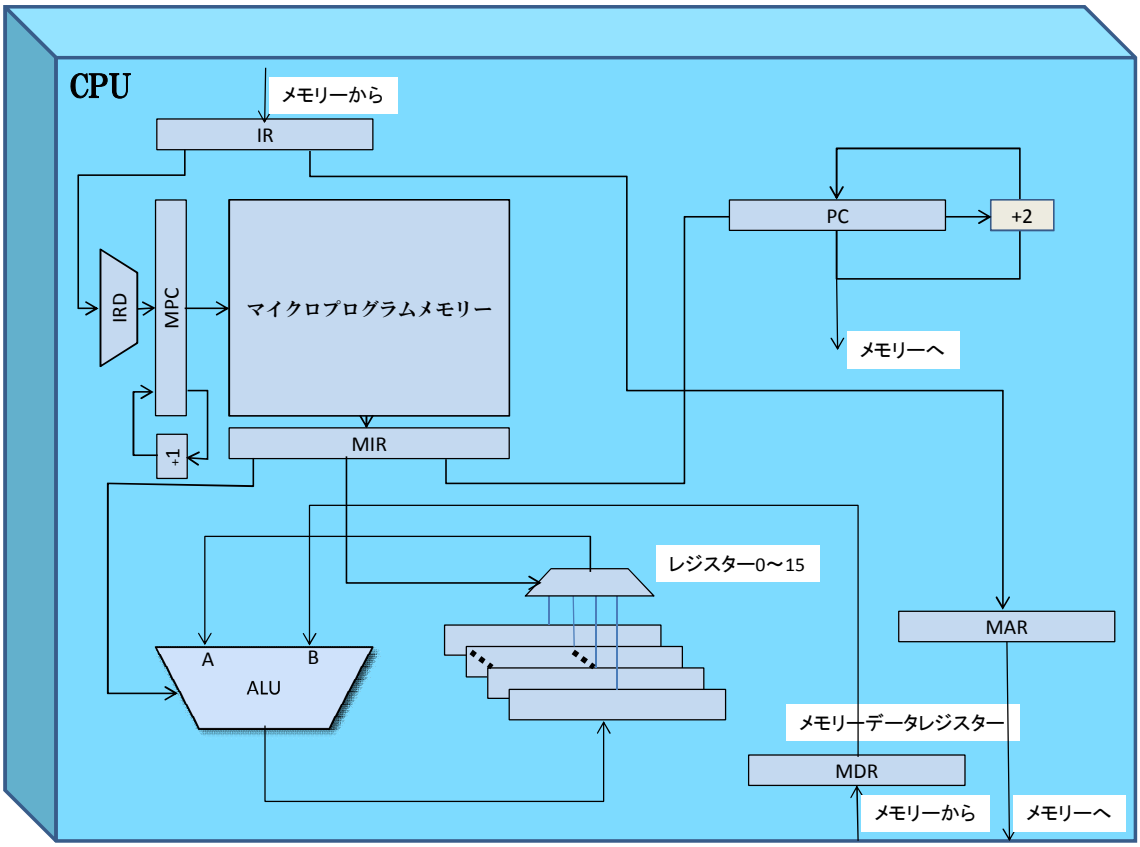
④ PC を+2 する

⑤ PC を+2 する

⑥ 次の命令を読み出すリクエストを出す

いかがでしょうか？まるっきりプログラムそのものです。ですが、実行する内容は CPU の中の部品を直接制御するためのものです。いわゆる、コンピュータとしての命令とは実施する機能のレベルが一段ハードウェア寄りですね。普通の命令やプログラムの前にマイクロとつけることにより、一段下の「細かい処理をする命令」、「細かい処理をするプログラム」という意味合いを持ってそのように「マイクロ」呼ばれています。マイクロプログラムというものがどのようなものかお分かりになりましたか？マイクロプログラムメモリーというのはそのようなマイクロプログラムを格納しておくメモリーです。そして MPC と MIR は大体想像がつくと思いますが、マイクロプログラムカウンターとマイクロ命令レジスター（命令はインストラクションですね）です。

上の図では、IR が一人さみしそうに佇んでいたのですが、その IR が IRD という部品、IRD は MPC という部品につながっています。これは命令の解読をどのように実現しているかという説明になりますが、IR を解読して、IRD（命令レジスターデコーダーの略です。デコーダーというのは、日本語では「復号器」と訳しますが、ここでは、命令コードを解読するという意味としてご理解ください）はその命令を実現するマイクロプログラムの先頭アドレスを生成します。そのマイクロプログラム先頭アドレスを MPC に入れ、そのアドレスで示されるマイクロプログラムメモリーの内容を MIR に読み出します。MIR からは CPU 内のいろいろな部品に信号がつながっており、制御します。最初の MIR が実行されると、MPC が+1 されて、次のマイクロ命令が MIR に読み出されるという具合に処理が進んでいきます。いかがでしょうか？命令の解読のしくみをご理解いただければ、CPU のしくみもかなりご理解いただけたのではないでしょうか？



ここまで、ご理解いただければ、CPU の基本がお分かりになったと言っても過言ではないと思います。あとは枝葉の説明になりますが、枝葉と言っても侮るなかれ、いろいろな技術の工夫がなされていますので、それらをご紹介したいと思います。