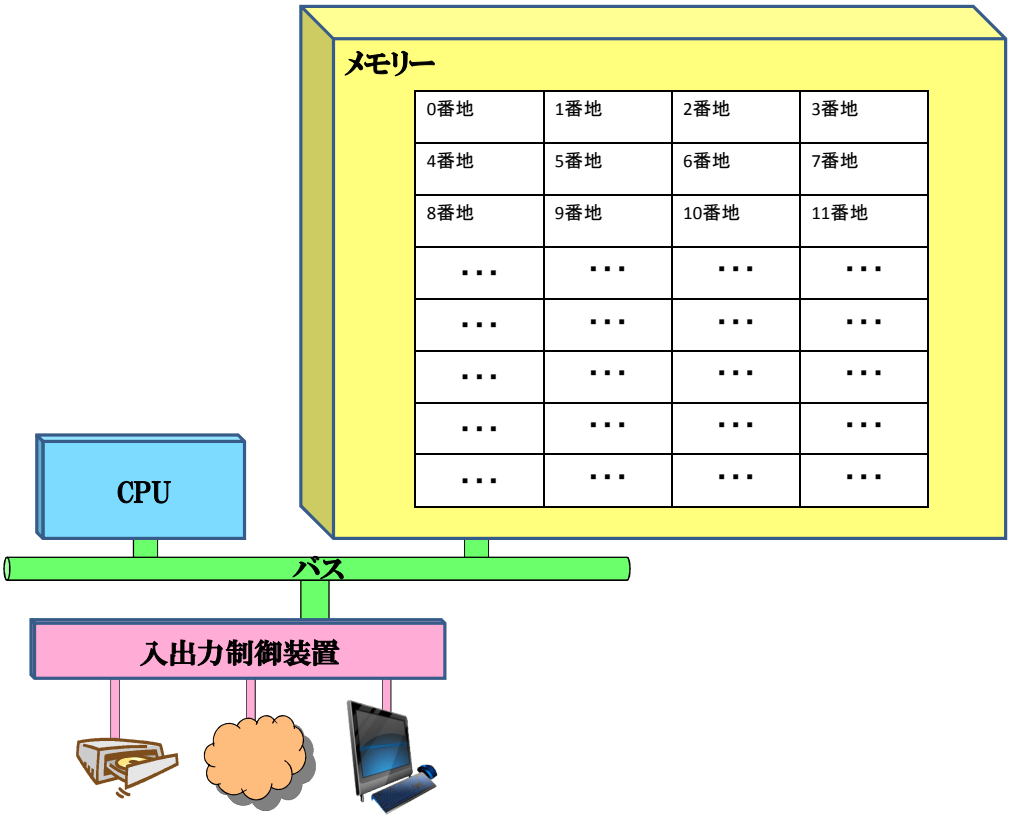
（1） 命令とは

コンピュータはプログラムを実行することによって機能します。ではプログラムというのは何でしょうか？プログラムは命令の組み合わせでできており、データを読んだり、書いたり加工したり、さらには状態によっては分岐（例えば、計算の結果がゼロなら XXX の処理をし、、ゼロでなかったら YYY の処理をする。というように状態によって処理内容を分けるということ）したりします。では、命令というのはどのようなものでしょうか？

命令の説明をする前に、もう少し CPU やメモリーについて説明します。

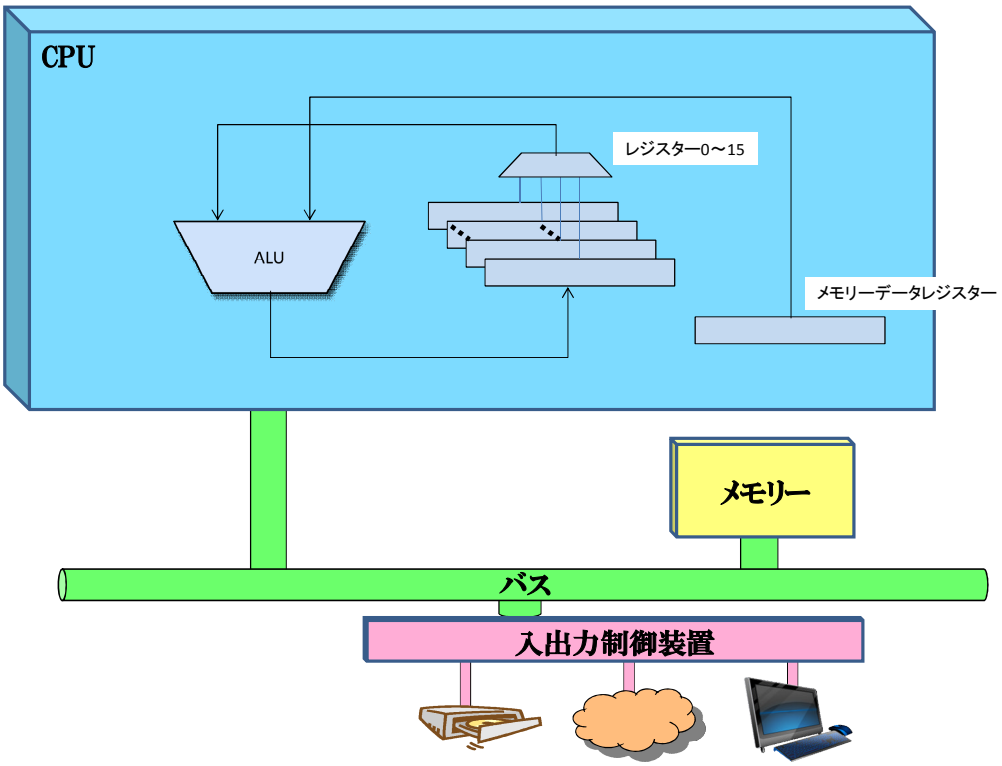
下の図をまずご覧ください。前回のコンピュータのしくみの図のうち、メモリーの部分を拡大しました。メモリーにはご覧のように番地（アドレス）がふられています。一番左の上から 0 番地、1 番地、・・・11 番地、・・・と続きますが、命令がメモリーのデータを読み出したりデータをメモリーに書いたりするときはこの番地を指定します。



次に、下の図をご覧ください。今度は、CPU の部分を拡大してみました。この中で説明しておきたいのは、レジスター0～15 というものです。レジスターというのはデータを保持する入れ物で、記憶装置の一種です。CPU 内にあるため高速にデータの読み書きができます。レジスター0～15 のように 0 から 15 の番号がふられています。命令ではこのレジスターの番号を指定して処理するレジスターを決めます。

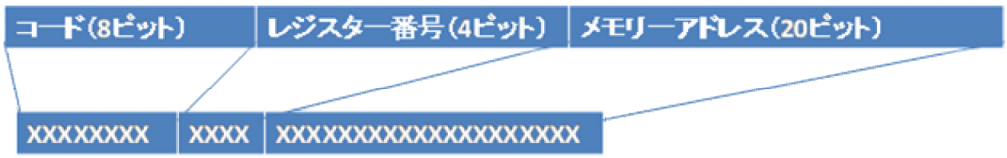
次はメモリーデータレジスターというものです。これもレジスターの一種ですが、メモリーから読み出したデータを保持しておくためのレジスターです。

そして、ALU。これは Arithmetic Logic Unit の略称で、日本語では算術論理演算装置とでも直訳しましょう。単に演算装置ということもありますし、演算論理装置ということもあるようですが、意味としては算術演算、論理演算をする装置です。この入力としてレジスター0～15 が左側、メモリーデータレジスターが右側に入っています。出力はレジスト 0～15 に入っています。これによって、メモリーから読み出したデータとレジスターの内容を演算処理して、レジスターに書き込むことができます。



この説明をふまえて、下の図を見て下さい。これは命令コードの定義です。レジスター番号を指定する部分、メモリーアドレスを指定する部分があります。これは上で説明したように、レジスター番号とメモリーアドレスを指定して、処理するデータを決めるためです。さらに、コードという部分があります。これはどのような処理をするかということを指定するための部分で、ここで加算をするか、減算をするかなどを決めて指示をします。

命令コード



さあ、一気に難しい話になってしまいましたので、少し、おさらいして整理してみたいと思います。

命令とは、レジスターと呼ばれるデータの入れ物と、主記憶装置（メモリー）のデータを処理することができます。加算や減算、読み書きという処理などです。

主記憶装置にはアドレス（番地）があり、それを指定して、その番地のデータを処理します。レジスターにも番号が振られていて、その番号で指定されたレジスターを処理します。

ここまでで、気が付いたことと思いますが、コンピュータの中では、CPU にしろ、メモリーにしろ、命令にしろ、全部、1 か 0 で表現されています。レジスターの番号も番号 0 は‘0000’、番号 1 は’0001’などと表され、命令のコードも例えば、’00000000’から’11111111’の 256 種類の二進数で表されます。すべてが 0 か 1 で表され、定義され、処理されている。これがデジタルの世界です。

さて、命令について、その概要を説明しましたので、具体的な説明をしたいと思います。下表はコンピュータが備えている命令の例を示しています。実際のコンピュータでは、数百もの命令の種類がありますが、ここでは、説明を簡単にするため、この 5 つの命令だけを紹介します。

命令名	機能
L（ロード）命令	指定したメモリーアドレスのデータを指定したレジスターに書き込む。（ロードする）
ST（ストア）命令	指定したメモリーアドレスに指定したレジスターのデータを書き込む。（ストアする）
A（加算）命令	指定したレジスターの内容に指定したメモリーアドレスのデータを加算し、レジスターに書き込む。
S（減算）命令	指定したレジスターの内容から指定したメモリーアドレスのデータを減算し、レジスターに書き込む。
BC（条件分岐）命令	指定した条件が成立した場合は指定したメモリーアドレスに分岐し、条件が成立しない場合は、次の命令に処理が進む。

先ほどの命令コードの定義と、この一覧表がどういう関係になっているのかよくわかりませんね。では、それぞれの命令の命令コード定義（例）を以下にします。

	コード	レジスター番号	メモリーアドレス
L(ロード)命令	01011000	XXXX	XXXXXXXXXXXXXXXXXXXX
ST(ストア)命令	01011001	XXXX	XXXXXXXXXXXXXXXXXXXX
A(加算)命令	01011010	XXXX	XXXXXXXXXXXXXXXXXXXX
S(減算)命令	01011011	XXXX	XXXXXXXXXXXXXXXXXXXX
		条件	
BC(条件分岐)命令	01001000	XXXX	XXXXXXXXXXXXXXXXXXXX

それぞれの命令はコードがあり、レジスター番号を指定でき、メモリーアドレスも指定できます。先ほど、レジスター番号は 0 から 15 あると言いましたが、2 進数 4 ビットの数字では､16 種類の数字が指定できますので､0 から 15 の 16 種類の番号が指定できます。後程 16 進数の説明をしますが、そこでさらに詳しく説明しますので、ここでは 4 ビットで 16 種類指定できるとだけで覚えておいてください。BC(条件分岐)命令では、レジスター番号ではなく、分岐をするかしないかを決める条件を 4 ビットのコードで指定します。例えば、’1111’ならば、無条件で分岐をする。’0100’ならば、この分岐命令の直前の演算結果がゼロの時は分岐し、ゼロでなければ分岐しない。などと使われます。

だんだん、プログラムというものが組めそうな気になってきませんか？この 5 個の命令だけでも、被乗数の加算を乗数回だけ繰り返すという処理で、掛け算を実行するプログラムを組むことができますね。ただ、高速な乗算処理とは言えませんし、掛け算を実行する命令があるのでプログラムで作る必要はないのですが・・・

（2） 16 進数、アセンブリー言語について

さて、命令の定義がわかったところで、例えば「8 番地の内容とレジスター2 の内容を加算して、レジスター2 に入れる。」という命令を作ってみましょう。

01011010001000000000000000001000

となります。最初の 8 ビットは加算命令ですから’01011010’ですね。次の 4 ビットはレジスター番号 2 ですから’0010’次の 20 ビットはメモリーアドレスが 8 番地ですから’00000000000000001000’です。

えーっこんな調子でプログラムなど組めるか！！と、お叱りの声が聞こえてきます。これは何とかしないといけません。コンピュータの中は何度も言って申し訳ありませんが、デジタルの世界で、2 進数で表されます。しかし、上の例のようにだらだらと長い数字の羅列では、参ってしまいます。そこで、少し知恵を使って、何ビットかまとめて一つの数字にしてみましょう。

まず、2 ビットずつまとめてみたらどうでしょうか。

01　01　10　10　00　10　00　00　00　00　00　00　00　00　10　00

2 ビットで表される 2 進数は 00、01、10、11 の 4 種類しかありませんので、それぞれに数字を対応させると、00 を 0 として、

2進数	00	01	10	11
2ビットまとめたら	0	1	2	3

のようになります。ということは、

01　01　10　10　00　10　00　00　00　00　00　00　00　00　10　00 は、

1122020000000020 となります。これは 4 進数といいます。でもまだ、数字が多いですね。

じゃ、3 ビットはどうでしょうか？これは 8 進数で 0～7 の数字を使って表しますが、3 ビットは少し半端で、ここに例として挙げた 32 ビット命令では、3 ビットずつまとめていくと最後に 2 ビット余ってしまい、使いにくいです。じゃ、4 ビットにしましょう。4 ビットなら割り切れますね。4 ビットで表される 2 進数は、0000、0001、0010、0011、0100、0101、0110、0111、1000、1001、1010、1011、1100、1101、1110、1111 の 16 種類です。さきほどのレジスター番号が 16 種類という説明がここでやっとできました、これが 16 進数ですね。では対応表を作りましょう！

2進数	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1
4ビットまとめたら	0	1	2	3	4	5	6	7	8	9	?	?	?	?	

問題です。1010 以降は数字がありません。我々はそもそも 10 進数の世界で生活していますから、16 進数と言われても、数字が足りないわけです。そこで、仕方がないので、アルファベットに登場してもらいます。以下のようにしました。

2進数	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	111
4ビットまとめたら	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

これで、先ほどの命令を表すと、

5A200008

です。何ともシンプルになりました。それでは、5 ビットまとめたらもっとシンプルになるのでは？いえいえ半端です。6 ビット、7 ビットも半端！

じゃ 8 ビットでは？

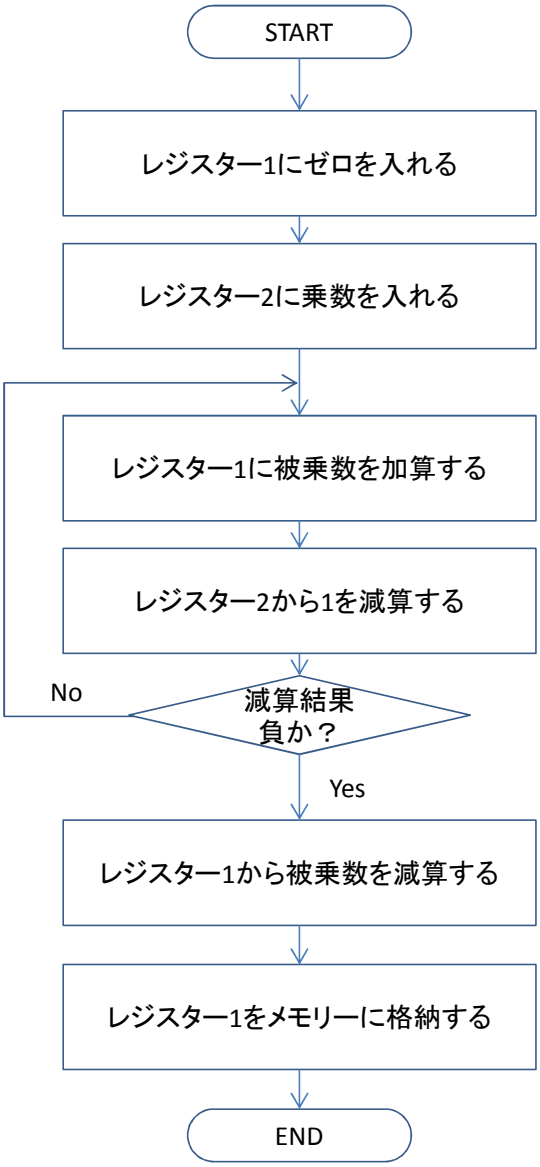
さっきの命令を 4 ケタの数字で表せますよ！

でもあまり調子に乗ってはいけません。8 ビットというのは 256 進数になりますね。そんなに文字はありません。アルファベットも 26 文字です。日本語のかなだって 50 文字程度、じゃ、漢字は？・・・まあ、妥協しましょう！

ということで、コンピュータの世界では、2 進数を 16 進数で表すようになりました。

余談ですが、インターネットなどでよく聞かれる IP アドレス、たまに目にされるのではないかと思いますが、これまでは IPv4 という 32 ビットのアドレスが使われてきました。が、10 進数で表されていました。2 進数を 10 進数に変換するのは大変で、私などはずいぶん苦労しましたが、IP アドレスがついに枯渇し、IPv6 という 128 ビットのアドレスに移行しつつあります。さすがに私のように苦労していた人たちが多かったのでしょう！16 進数にて表記されるようです。16 進数は慣れてくると 2 進数への変換が簡単で、IP アドレスのサブネットマスクのビット数がどうなっているかなどを 10 進数では、わかりづらかったものがわかり易くなると思います。余談でした。

さて、先ほども言いましたように何かプログラムのようなものを作ってみたいですね。5 つしか命令がないので、プログラムを組むと言っても、かなり単純な処理しかできませんが、上にあげた、乗算処理というのをプログラムで実現してみましょう。プログラムを組むときは、フローチャートというものを使います。下のようなものです。乗算プログラムの考え方としては、単純ですが「被乗数を乗数回繰り返して加算する」です。下のフローチャートはその処理を表しています。



このフローチャートの説明をしようと思いましたが、ここでの説明の目的は命令を並べてプログラムが作られるという例を紹介することなので、あまり詳しい説明はしないつもりでした。が、「最後何で、レジスター1 から被乗数を減算するのか？」ということが気になる人もいるかと思いますので、少しだけ、付け加えます。

レジスター2 の内容回数だけ加算を繰り返すという処理ですが、レジスター2 から 1 減算した結果を負かどうか聞いているけど、ゼロかどうかを聞けば、減算をすることはないのではないか！

とお思いになると思います。しかし、そうすると乗数（レジスター2 の初期値）がゼロの場合、いきなりゼロから 1 を引くと、次にゼロになるまで大変な時間がかかります。これはプログラムの暴走といい、しばらくウンともスンとも言わなくなります。このような

ことは、16 進数表記で命令コードを作り、コンピュータのメモリーに直接手で書き込んでプログラムを動かそうとするとよくあるバグですが、一応そのようなバグだけは無いようにフローチャートを作りました。ただ、負の数の場合の処理は考慮されていませんので、プログラムとしてはまだまだ失格ですが、ここでは、命令の組み合わせによるプログラム例の紹介・・・言い訳ばかりですみません。

話をもどします。さて、フローチャートができても、実際にプログラムをメモリーに書き込み、動かさないと処理ができません。ではメモリーにプログラムを書き込みましょう。実際に書き込むためには、メモリーの何番地に書くかなども考えないといけません。500 番地からこのプログラムを書くことにしましょう。ここでいう 500 という数字は 16 進数としたいと思います。10 進数と区別するために 16 進数であるという意味で、0x500 などと書いたり、500h と書いたりいろいろ 16 進数の表し方は方法がありますが、ここではよくプログラミング言語などで使われている 0x500 番地という 16 進数の表し方を使用します。プログラムとともに、いくつかのデータもメモリーに書いておかないといけません。被乗数、乗数、それからレジスター1 を最初ゼロにし、レジスター2 から 1 を減算するので、ゼロも 1 も用意しておきましょう。では、0x600 番地に被乗数、0x610 番地に乗数、0x620 番地にゼロ、0x630 番地に 1 を書きましょう。下のようになります。

分岐命令のメモリーアドレスが 0x508 となっていることだけは説明を付け加えておきます。メモリーのアドレスはバイト単位（8 ビット）につけられていることが多いので、この説明の中でもそのようにバイト単位でアドレスを付けることにしています。最初のロード命令（0x58100620）は 0x500 番地、次のロード命令（0x58200610）は 0x0504 番地、そして、次の加算命令（0x5A100600）は 0x508 となっているため、分岐命令の飛び先はこの加算命令の 0x508 番地が指定されます。

0x500 番地	0x58100620
	0x58200610
	0x5A100600
	0x5B200630
	0x48700508
	0x5B100600
	0x59100640
0x600	0x00000005
0x610	0x00000003

0x620　　0x00000000

0x630　　0x00000001

0x640　　積が格納されます。

いかがですか？面倒くさいということだけ、感じていただければ結構です。コンピュータの CPU を開発するハードウェアエンジニアであれば、このようなことをする必要もあるのですが、実際にプログラムを組むためにはもっと楽な方法があります。

そこで、アセンブリー言語というのを紹介します。これは 0 と 1 で命令を表現するのではなく、文字で記号にして表現をします。例えば下のように、です。

```
        L  R1,ZERO
        L  R2,JOUSUU
LOOP    A  R1,HIJOUSUU
        S  R2,ONE
        BC P,LOOP
        S  R1,HIJOUSUU
        ST R1,SEKI
JOUSUU  0x00000003
HIJOUSUU 0x00000005
ZERO    0x00000000
ONE     0x00000001
SEKI    積が格納されます
```

プログラムを作る人は、命令のコードがどうかとか、レジスター番号を 16 進数にするとコードがどうなるかとか、何番地からプログラムを書き込んであるとか、データが何番地に書かれているかなどということをあまり意識せずに作ることができます。このアセンブリー言語で書かれたプログラムは、アセンブラーというアセンブリー言語を翻訳するプログラムによって、コード、レジスター番号、メモリーアドレスの命令コード内への埋め込みなどがなされて、コードが作られます。

例えば、L はロード命令で、R1 とか R2 という表記でレジスター番号が指定できます。メモリーのアドレスは、ラベルを指定したいデータの前に書き、それを命令の記述の中で指定します、例えば、R1 にゼロを加算したいときは、ZERO というラベルを書いたところに、データとして 0x00000000 を書いておけば、0x00000000 というゼロのデータがメモリーに書かれ、ZERO というラベルで指定した L 命令では、命令コードの中にそこの番地が

埋め込まれます。A は加算、S は減算、BC は条件分岐、ST はストア命令で、同様に命令のコード、レジスター番号、分岐条件、メモリーアドレス、分岐先アドレスなどがアセンブラーによって、16 進数（2 進数）のコードとして作られます。

いかがですか？命令、プログラムとそれらを実行したり、格納したりする、CPU とメモリーの役割がわかりましたか？今回は命令とプログラムの説明を中心にしましたので、CPU の中の ALU やメモリーデータレジスターについては詳しくは説明をしませんでした。次回はこの CPU の中のしくみに話を進めてみたいと思います。