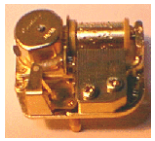
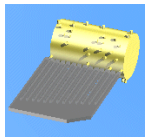
サイバースペースに飛び込もう
- マシン語/アセンブラに挑戦

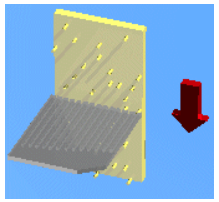
映画やアニメで近未来のサイバースペースを扱っているものは、何か1と0がいっぱいある画面が出てきませんか。実はこの0と1だけがコンピュータの中の世界にあるものです。この授業ではコンピュータのこの0と1の世界は何かを見ていきましょう。

コンピュータはどうして動くの(1)? コンピュータに近いもの





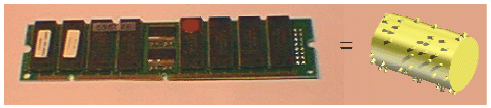
以外かもしれませんが、コンピュータの動きに近いものはオルゴールです。オルゴールは円筒(シリンダー)に突起があって、それが回って櫛状の金属版を弾いて音がでますね。
また、円筒の突起のパターンによって演奏される曲が違いますね。

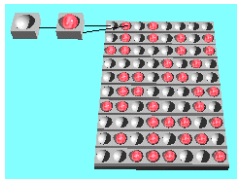


もう少しコンピュータに近いオルゴールを考えてみましょう。左の写真はディスクオルゴールといって、円柱の盤を使います。盤には、やはり突起がありますが、盤自体を交換することができます。また、右の図は円筒を板のようにしたオルゴールです。これでは1回しか演奏できませんが、板を動かすと曲が流れますね。

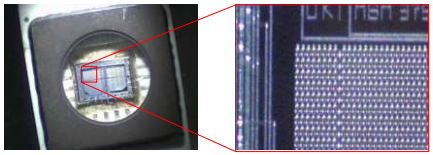
ソース:ディスクオルゴール www.youtube.com/watch?v=wD2NkvlSdnY

コンピュータはどうして動くの(2)? コンピュータの円筒





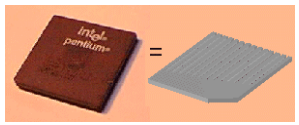
スイッチがオフ
(ｵﾙｺﾞｰﾙでてっぱり無)
スイッチがオン
(ｵﾙｺﾞｰﾙでてっぱり有)
ﾒﾓﾘ中の小さなｽｲｯﾁ状態

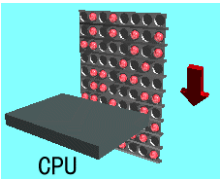


メモリを拡大して見ると

オルゴールの円筒に対応したものはコンピュータではメモリになります。
メモリはオン・オフの状態を保持する小さなスイッチの集まりです。
実際のメモリを拡大して見ると、小さなスイッチがびっしり入っています。
携帯電話なんかに使用するSDカードもメモリの一種で、このような小さなスイッチが、小さなカードの中に数十億から数百億個入っています。

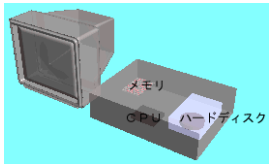
コンピュータはどうして動くの(3)? コンピュータの櫛(くし)の金属の板

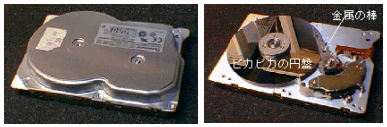




オルゴールの櫛状の金属の板に対応するものは、コンピュータではCPU(シーピーユー:中央処理装置)になり、これがコンピュータの脳になります。
CPUはオルゴールと同じように、メモリの中を1行づつ、読み込んでいて、オルゴールが音楽を演奏するように、そのオン・オフのパターンに対応した動作をします。パターンはいろいろあって、基本は四則演算ですが、例えばキーボードのどれが押されたか取り込むような動作パターンもあります。
前に示したディスクオルゴールが盤を変更すると演奏する音楽が変わるように、、スイッチのオン・オフを変更することでメモリ上のパターンの集まりを変更することができます。あるパターンではCPUがワープロと動作したり、また他のパターンでは映画を再生したりします。
プログラムは、CPUにある特定の仕事をさせるための、このメモリ上のパターンということになります。

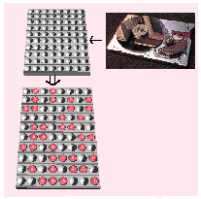
コンピュータはどうして動くの(4)？ プログラムの入れ替え





=

ハードディスクは磁気でスイッチのON/OFFを記録

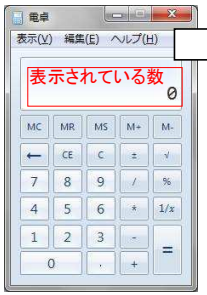


パソコンの中には、今まで説明してきたCPUとメモリがあります。また、それ以外に重要な部品としてハードディスクがあります。ハードディスクはピカピカの円盤が中に入っていて、ビデオテープや昔使っていたカセットテープのように磁気を記録できます。ハードディスクにはいろいろなプログラムのパターンが磁気的にオン・オフで記録されていて、必要な時にメモリにコピーされます。
パソコンで例えば、Wordのプログラムを起動すると、ワードのプログラムがメモリ上にコピーされて、それをCPUが読み取って仕事をすることになります。

プログラムはどう作る(1)？ 人間が計算する

コンピュータのCPUがメモリのパターンに従って動くことを今まで説明してきました。
では、望みにあった仕事をCPUにさせるためにメモリのパターン=プログラムはどのように作るのでしょうか?
一番簡単にコンピュータ動作させるために、電卓を使ってみましょう。
上図のように、4つのボタンを押していくと、3と答えが出ます。今ではあたりまえのことですね。

プログラムはどう作る(2)？ 人間がプログラム的に操作する。



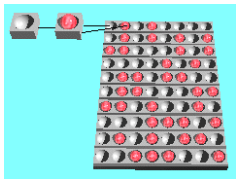
M箱

MC	M箱の中をクリヤー(0)する。
MR	M箱の中を表示する。
MS	表示されている数をM箱に入れる。
M+	表示されている数をM箱に足す。
M-	表示されている数をM箱から引く。

操作	表示されている数	M箱の中身
1 MS	1	1
2 M+	2	3
MR	3	3

もう少し、コンピュータのプログラム的に電卓を操作してみましょう。
電卓は直接見えませんが、Mという一時的に数字を記録できる箱みたいなものを内部に持っています、Mxというキーでこの箱に対して操作できます。
上表のように操作してみましょう。このM箱を使って、前と同様に1+2の結果を得ることができます。
今は人間が手動で足し算の操作をしましたが、同様なことを自動的にCPUにやらせるものがプログラムになります。

ワンポイントICT: 2進数



コンピュータはメモリのスイッチのオンオフの情報しかありません。どうやって数値を表しているのでしょうか?
コンピュータではスイッチのオンを1, オフを0としてそれらが表す二進数として数値を扱います

10進数の意味 1956は物が何個ありますか?

1	9	5	8	
1000 x 1	100 x 9	10 x 5	1 x 8	= 1000 + 900 + 50 +8 =1958

2進数の意味 1011(スイッチが●○●●の状態) は物が何個ありますか?

●	○	●	●	
1	0	1	1	
8 x 1	4 x 0	2 x 1	1 x 1	= 8 + 0 + 2 + 1 = 13

日常にも10進数以外のn進数があります。代表的なものが時間で60進数を使っています、例えばビデオの時間など1:15:25のような表示です。これを秒に換算する時は、3600*1 + 15* 60 + 25 と計算しますね。

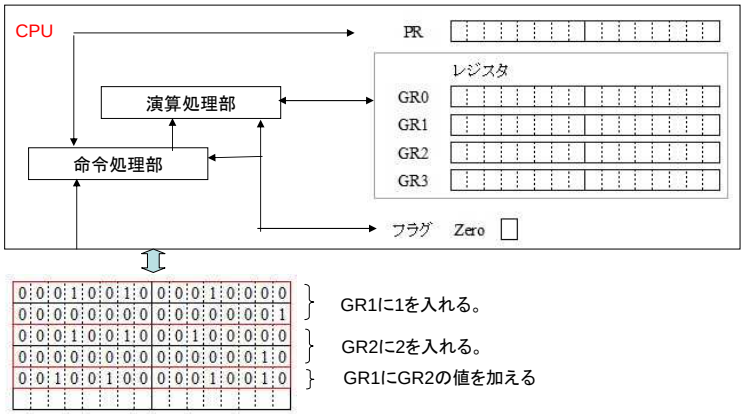
ワンポイントICT: 2進数とビット

10進数	2進数	
値	ビット数	値
0	4ビット	0000
1		0001
2		0010
3		0011
4		0100
5		0101
6		0110
7		0111
8		1000
9		1001
10		1010

10進数	2進数	
値	ビット数	値
11	4ビット	1011
12		1100
13		1101
14		1110
15		1111
0～255	8ビット	00000000～11111111
0 ～ 65536	16ビット	0000000000000000 ～ 1111111111111111

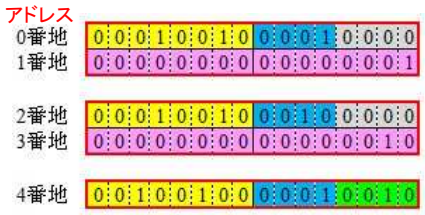
1個のスイッチのオンオフで表す情報の単位をビットといいます。コンピュータの最小の単位です。また4ビットあれば、10進数に変換した場合の0～15までの数値を表せます。また8ビットならば255まで、16ビットなら65536までの数値が表せます。Windowsの電卓はこの10進数と2進数が簡単に変換できるので確認してみましょう。

プログラムはどう作る(3)? CPUの構造とプログラム。



では、先ほど電卓の操作を自動的に行うプログラムを考えます。
まず、CPUの中には電卓のM箱と同じように計算結果などを記録するレジスタという箱があります。また命令処理部はメモリの中を読み取ってどのように動くか判断します。
上記のようなメモリのパターンがあると、最終的にGR1に3が自動的にはいります。

プログラムはどう作る(4)? マシン語(機械語)



レジスタに数値を入れる
1 -> GR1(0001)

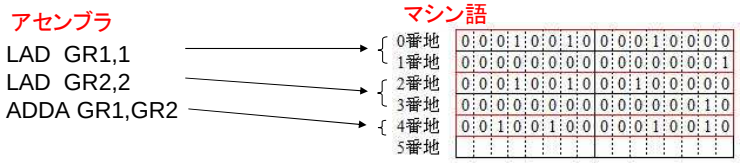
レジスタに数値を入れる
2(10) -> GR2(0010)

レジスタaにレジスタbを加える
GR1(01)+GR2(10) -> GR1

メモリのパターンになっているプログラムをもう少し細かくみてみましょう。
このプログラムはCOMET2というCPUのもので、CPUという機械が理解できるものでマシン語(機械語)と呼ばれています。どんなコンピュータでも、結局動いている時はこのマシン語だけが理解できます。
また、このCPUの場合は、メモリを16ビットごとに区切り、0から番号を振ってワードという単位で管理しています。個々のマシン語の命令は1ワード又は2ワードの中に入っています。そしてCPUはメモリ上のマシン語を1個づつ読み取って実行していきます。この番号をアドレスと言っています。

このようなコンピュータはプログラム内蔵方式でありノイマン型コンピュータと呼んでいます。世の中にある、ほとんどすべてのコンピュータは小さなスマートフォンからスーパーコンピュータまで、すべてこのノイマン型コンピュータです。

プログラムはどう作る(5)? アセンブラとマシン語



コンピュータを動かすためには、0と1のパターンのマシン語のプログラムを作る必要があります。これって難しいと思いませんか?
そこで、少しわかりやすいアセンブラ言語というのが考えられました。マシン語をある程度、意味のわかる言葉におきかえたものです。マシン語に比べるとずっとプログラミングしやすくなりました。

ただし、アセンブラ言語のままだとCPUは理解できないので、アセンブラ言語の文字からマシン語の0,1に変換する必要があります。この変換自体をコンピュータにさせることにして、その変換するプログラムをアセンブラーと呼んでいます。

アセンブラ言語の1行はマシン語の一つの命令に対応し、
CPUに一つの動作を指定することになります。

アセンブラ・プログラミング(1)　アセンブラの命令(1)

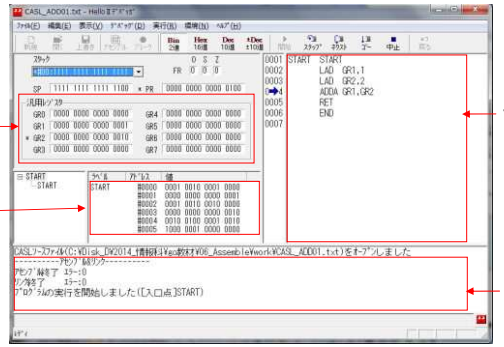
命令	一般形式	意味	例
数値を入れる	LAD レジスタ,数値	数値を指定したレジスタに入れる	LAD GR1,7
加算	ADDA レジスタ1, レジスタ2	レジスタ1にレジスタ2を加える	ADDA GR1, GR2
減算	SUBAレジスタ1, レジスタ2	レジスタ1からレジスタ2を引く	SUBA GR0, GR2
	RET	授業ではプログラムの終了を示します	RET

```
ADR0  START
      LAD  GR1,1
      LAD  GR2,2
      ADDA GR1,GR2
      RET
      END
```

これからアセンブラによるプログラムを学習していきます。覚える命令の数は全部で10個ですが、始めに上の4つを使います。
また、実際のプログラムは左のようにSTARTとENDでくくってつくります。

ソース:独立行政法人 情報処理推進機構 情報処理技術者試験 出題範囲

アセンブラ・プログラミング(2) 開発環境(デバッガー)



アセンブル	アセンブラのプログラムをマシン語に変換する。
Bin 2進	レジスタ・メモリの内容を2進数表示にする。
開始	プログラムを開始する。
ステップ	開始後、1行づつ実行する。
中止	プログラムを終了する。

SUBA命令を使って5-3を計算するようにプログラムを変更してみましょう

ソース: Hello Ⅱ シミュレーター http://www.aobasoft.co.jp/casl/

アセンブラ・プログラミング(3)　アセンブラの命令(2) データを使う

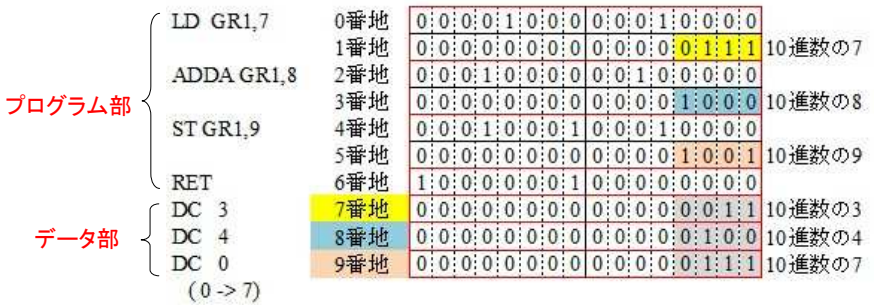
命令	一般形式	意味	例
メモリの値を入れる	LA レジスタ，アドレス	指定したアドレスの内容を指定したレジスタに入れる	LA GR1, DATA1
メモリへ値を入れる	ST レジスタ1, アドレス	指定したレジスタの内容を指定したアドレスに入れる	ST GR1, 7
加算	ADDA レジスタ, アドレス	指定したアドレスの内容を指定したレジスタに加える	ADDA GR1, DATA1
減算	SUB　Aレジスタ, アドレス	指定したアドレスの内容を指定したレジスタから引く	SUBA GR0, 7

機能	一般形式	意味	例
メモリへの数値の確保	DC 数値	メモリを確保し、値をいれてやく	DC 3

```
ADR0    START
        LD  GR1,7
        ADDA GR1,8
        ST GR1,9
        RET
        DC  3
        DC  4
        DC  0
        END
```

今までは、レジスタだけで計算していましたが、メモリにあるデータを処理することを考えてみます。左のプログラムはメモリに入ってる2つの値を計算して、別のメモリに入れるプログラムです。
次のスライドではメモリの中をもっと細かくみていきましょう。

アセンブラ・プログラミング(4)　データを使う



実際にプログラムのメモリの状態をみてみましょう。
まず、メモリの中では特にプログラムとデータがわかれているわけでは無く、混在しています。
そしてDCで定義した数値は、それぞれアドレスの7,8,9番地のメモリの中に格納されています。
今回のプログラムでは、アドレスを指定して、その中に入っている数値を処理の対象にしています。

違いは?
LD GR1,7
LDA GR1,7

プログラムはどう作る(5)？ 番地の代わりにラベルに使う

```
ADR0    START
        LD  GR1,7
        ADDA GR1,8
        ST GR1,9
        RET
        DC  3
        DC  4
        DC  0
        END
```

アドレスの数値の番地の代わりにラベルを使う

```
ADR0    START
        LD  GR1,DATA1
        ADDA GR1,DATA2
        ST GR1,DATA3
        RET
DATA1   DC  3
DATA2   DC  4
DATA3   DC  0
        END
```

プログラムの中でメモリのアドレスを指定することで、そこに記録されている数値を扱えます。但し、メモリのアドレスが何番になるかなかなかわかりませんね。又はプログラムを変更するとずれてしまいます。
そこでアセンブラ言語では、ラベルという機能をサポートしています。ラベル(名前)をプログラムの各行の初めにつけているとプログラムの中でそのラベルを実際のアドレスの数値の代わりに使えます。
上のプログラムではDATA1, DATA2, DATA3はそれぞれアドレスの7,8,9番地の代わりに使用できます。但しラベルはプログラム上の便宜的なもので、メモリに展開されたマシン語はどちらも同じになります。

プログラムはどう作る(6)？ 判断をする。

命令	一般形式	意味	例
比較する	CPA レジスタ1, レジスタ2 CPA レジスタ、アドレス	指定した2つの数値比較する。もし等しい場合はZeroフラグが1, それ以外は0	CPA GR1, GR2 CPA GR1, DATA1
等しい場合ジャンプ	JZE アドレス	等しい場合(Zeroフラグが1)の時指定したアドレスから実行	ST GR1, 7
無条件にジャンプ	JUMP アドレス	無条件に指定したアドレスから実行	JUMP L1

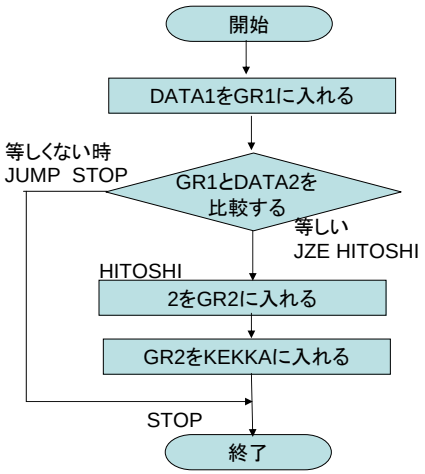
```
ADR0    START
        LD  GR1,DATA1
        CPA GR1,DATA2
        JZE HITOSHI
        JUMP STOP
HITOSHI LAD GR2,1
        ST  GR2,KEKKA
STOP    RET
DATA1   DC  3
DATA2   DC  3
KEKKA   DC  0
        END
```

左のプログラムは何をしているのでしょうか

コンピュータと他の道具・機械の違いは何でしょうか?
他の機械は予め決められた順番の動作又は、人間の操作によってしか動きません、これに対してコンピュータはいろいろな条件を判断して動作を変更することができます。マイン語レベルでこの判断を見てみましょう。

プログラムとフローチャート(判断をする)

```
ADR0    START
        LD  GR1,DATA1
        CPA GR1,DATA2
        JZE HITOSHI
        JUMP STOP
HITOSHI LAD GR2,1
        ST  GR2,KEKKA
STOP    RET
DATA1   DC  3
DATA2   DC  3
KEKKA   DC  0
        END
```



問題: 次のプログラムは何をしていますか(1)

```
ADR0    START
        LAD  GR0,0
        LD   GR1,DATA1
        LAD  GR2,1
        LAD  GR4,0
LSTART  CPA  GR1,GR4
        JZE  LEND
        ADDA GR0,GR1
        SUBA GR1,GR2
        JUMP LSTART
LEND    ST  GR0,KEKKA
        RET
DATA1   DC  5
KEKKA   DC  0
        END
```

問題: 次のプログラムは何をしていますか(2)

```
ADR0    START
        LAD  GR0,0
        LD   GR1,DATA2
        LAD  GR2,1
        LAD  GR4,0
LSTART  CPA  GR1,GR4
        JZE  LEND
        ADDA GR0,DATA1
        SUBA GR1,GR2
        JUMP LSTART
LEND    ST  GR0,KEKKA
        RET
DATA1   DC   3
DATA2   DC   4
KEKKA   DC   0
        END
```


ワンポイントICT: マシン語,アセンブラ,高級言語

アセンブラ

```
ADR0    START
        LD  GR1,DATA1
        CPA GR1,DATA2
        JZE HITOSHI
        JUMP STOP
HITOSHI LAD GR2,1
        ST  GR2,KEKKA
STOP    RET
DATA1   DC   3
DATA2   DC   3
KEKKA   DC   0
        END
```

高級言語(Basic)

```
Data1 = 3
Data2 = 3
If Data1 = Data2 Then
  Kekka = 1
End If
```

アセンブラという言葉は今日の授業で初めてかもしれません。但し、JavaやBasicという言葉は聞いたことがあるかもしれませんな。
上の二つは同じことをするプログラムをアセンブラとBasicで作ったものです。
アセンブラはほぼマシン語に近いもので、プログラムを作るときに非常に手間がかかります。これに対してJavaやBasicは高級言語と呼ばれより人間の理解しやすいものになっています。
ただし、高級言語でも実行される前にはマシン語に変換されます。

1と0がいっぱいある画面が意味は分かりましたか、これはコンピュータの中にあるデータとプログラムを示すものでした。