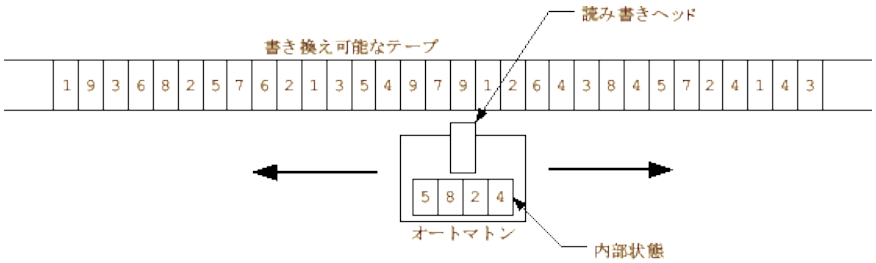
■#03 CPU の都合 データとプログラム

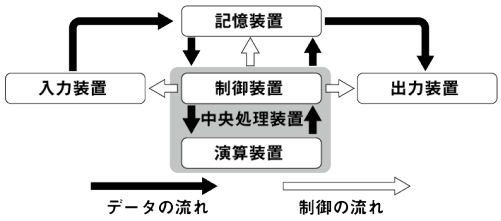
前回、2 進数の組合せに、コンピュータの命令を対応させ実際に隣の人を動かしてみました。これは、命令の最初から順番に読み取り、順番に実行しています。

この基礎的な理論として、“チューリングマシン” と言うものがあります。チューリングマシンは、理論的な機構で、命令の書かれた書き換え可能な情報テーブルの上を “オートマトン” と呼ばれる情報テーブルの上を移動可能な機械で読み取り処理していきます。



この機構を、現実に実現した機構に “ノイマン型コンピュータ” というものがあり、今日では基本的なコンピュータ・アーキテクチャのひとつとされています。

ノイマン型コンピュータは、プログラム内蔵方式のディジタルコンピュータで、制御装置と演算装置とアドレス付けされた記憶装置とそれらをつなぐバスを要素に構成され、命令（プログラム）とデータを区別せず記憶装置に記憶します。情報処理の勉強でよく出てきたものですね。



21 世紀初頭におけるコンピュータのほとんどはノイマン型と言えますが、現在では ニューラルコンピュータや GPU などに利用される、”データフロー型アーキテクチャ” など様々に改良されています。

前回は、命令コードを読み取り、命令表をもとに、実際の命令に変換していました。これは、今あるコンピュータが “0” か”1”のデジタルデータを利用するためであることを説明しましたが、実は、前回の授業はノイマン型コンピュータの基本的な仕組みを体験したものでした。

4bit $2^4 = 16$ 通り

0000	右手をあげる	0001	左手をあげる	0010	右手を下げる	0011	左手を下げる
0100	右手を開く	0101	左手を開く	0110	右手を閉じる	0111	左手を閉じる
1000	体を右に回す	1001	体を左に回す	1010	体を右に傾ける	1011	体を左に傾ける
1100	原点復帰	1101	笑う	1110	“OK” と言う	1111	何もしない (nop)

前回の内容を少し振り返ってみます。前回困ったことありませんでしたか？ “右手をあげる” という動作ですが人によってかなり動作が違ったと思います。手を上げる角度が少しだったり、元気よく真上にあげたり様々ですね。これをみんなが同じような動作をするために “どのくらい” 手を上げるのか指定できるようにすればいいようです。

そこで、”どのくらい” を数値化して、指定できるようにしてみましょう。先ほどの命令表を少し書き換えて・・・

4bit $2^4 = 16$ 通り命令表

0000 ＋ xxxx	右手をあげる ＋ 上げる角度	0001 ＋ xxxx	左手をあげる ＋ 上げる角度	0010 ＋ xxxx	右手を下げる ＋ 下げる角度	0011 ＋ xxxx	左手を下げる ＋ 下げる角度
0100 ＋ xxxx	右手を開く ＋ 開く角度	0101 ＋ xxxx	左手を開く ＋ 開く角度	0110 ＋ xxxx	右手を閉じる ＋ 閉じる角度	0111 ＋ xxxx	左手を閉じる ＋ 閉じる角度
1000 ＋ xxxx	体を右に回す ＋ 回す角度	1001 ＋ xxxx	体を左に回す ＋ 回す角度	1010 ＋ xxxx	体を右に傾ける ＋ 回す角度	1011 ＋ xxxx	体を左に傾ける ＋ 回す角度
1100	原点復帰	1101	笑う	1110	“OK” と言う	1111	何もしない (nop)

前回の命令例に、角度を足してみました。次のようになります。太枠がデータを含めた 1 命令になります

右手をあげる	15度	体を右に回す	10度	体を左に回す	10度	体を右に回す	15度	体を左に回す	15度
0000	1111	1000	1010	1001	1010	1000	1111	1001	1111
0	F	8	A	9	A	8	F	9	F

0x0F8A9A8F9F ですね。これで、命令＋角度(データ)の指定もできるようになりました。

しかし、問題があります。

ここで先ほどの命令を、2 進数の部分だけ抜き出してみます。

0000	1111	1000	1010	1001	1010	1000	1111	1001	1111

もしかすると

右手をあげ、何もしなくて、体を右に回して、何もしない・・・となる可能性もあります。

一見するとどれが命令かデータ部分なのか分かりづらいですね。これを “バイナリデータ” といいます。

実は、コンピュータにとって、先ほどのプログラムはデータの塊としか理解できていません、そのため bit エラーや、情報の欠落、実行場所の喪失などが起こるとむちゃくちゃな動きになってしまいます。そもそも人間にとって見理解ができません。

そこで考えられたのが “アセンブルコード” になります。

アセンブルコードって？

ちなみにこんなものです。

アドレス	アセンブルコード	生成される 2 進数データ(16 進表記)
: 0040159B 004015A0 004015A7 :	: MOV EAX,1 DD DWORD PTR DS:[401234],64 CMP DWORD PTR DS:[401234],3E8 :	: B8 01000000 8305 34124000 64 813D 34124000 E8030000 :

ちなみに、“MOV” はデータの移動、”DWORD PTR” は対象の値（やレジスタ）を何バイトとして扱うかを設定して、”CMP” は比較命令になります。

マイコン自体の命令を直接指定するため、単純な命令の組合せを処理通りに記述しなければならないため、なかなかこれでも分かりづらいですが、昔はこのアセンブルコードを直接書いている方も多く、そのことを ”ハンドアセンブル” すると言ってました。

今でも、高速化が必要な部分はアセンブラで書かれていることがあります。

〇チェックポイント・キーワード

・チューリングマシン
・現在のコンピュータの基本となる ノイマン型アーキテクチャ
・ニューラルネットワーク、GPU で用いられるデータフロー型アーキテクチャ
・”データ” と “命令” は CPU から見ればただの 2 進数。
・アセンブルコード