第４話（出力装置）

キツネ、あと、モニターへの出力が残っているが、演算装置が計算した結果［0000000000001010］（＋１０）が主記憶装置の［８００９］（アドレス）に格納されたので、これを出力すれば終わりだろ。

あたりきしゃりきのコンコンチキよ！
タヌキのいうようにそのまま出力したら、モニターに空白が表示されるか、ゴミが表示されるだけよ。印刷ならば何も印刷されずに、なにこれ、という感じかな。第２話で話したと思うけど、モニターに表示するには、数値型ではなく、文字型にしなければならない。さらにゾーン形式にしなければ表示されないんだよ。

表示のさせ方だけでも大変なんだ。

高級言語を使えば、print や write という命令だけで表示や印字できるけど、アセンブラ言語の場合は、数値を文字型にし、それにゾーン部を付け、キーボードのキーと同じコードにする作業をしないと表示されないんだ。ただ、高級言語の場合もプログラマーが変換作業をしなくて良いだけで、同じことがコンパイル作業中に自動的に行われているんだ。人間の言葉に近いコードを使う高級言語は扱いやすいが、機械語に近いアセンブラ言語は人間にとって扱いにくいんだ。そういう意味では低級言語になるかな。アセンブラ言語は CPU に直接働きかけるので処理速度が格段に速かったが、今はコンピュータの性能が飛躍的に向上したので、高級言語を使っても遅いと感じなくなってしまったね。

結果の数値にゾーン部を付加しないと表示されないことは理解できたが、さらに高級言語とか低級言語とかコンパイラとか勉強しなければならない次の課題が出てきたのね。

そうなんだけれども、まず＋１０という計算結果にゾーン部を付けて表示させることを考えよう。これが意外と大変なんだ。１０を１と０を取り出して［00000001］と［00000000］にしなければならない。それにゾーン部の［00110000］を加算すれば［00110001］と［00110000］になり、16進数の 31 と 30 でキーボードのキー 1 と 0 と同じコードになり、表示や印字ができます。

１０÷１０とか具体的な例で教えてくれないか。

１０÷１０＝１余り０で１桁目は０で２桁目は１という形で簡単に取り出せるが、これをコンピュータの演算装置でやらせるとなると多くの知識が必要なんだ。でも各桁を取り出すのに１０で割るという基本的な考え方は同じだよ。
演算装置は、加算・減算・乗算・除算・比較演算しかできないことは理解しておいてね。さらに、乗算と除算はシフト演算と加算の組み合わせで結果を出しているんだ。それでシフト演算を理解しないと乗算と除算は理解できないんだ。
シフト演算の説明をするか！

［シフト演算］

乗算は左シフトで行います。
７×５＝３５をＣＰＵの演算装置をイメージして計算してみましょう。
７を８ビットの２進数で表わすと以下のようになります。

0	0	0	0	0	1	1	1

1 ビット左にシフトすると２倍の 14 になります。

0	0	0	0	1	1	1	0

さらに１ビット左にシフトすると４倍の 28 になります。

0	0	0	1	1	1	0	0

さらに１ビット左にシフトすると８倍の 56 になってしまいます。ここで、シフト演算は使えなくなり、７を加算することになります。

```
  00011100
 ＋00000111
 ----------
  00100011
```

35(10 進数)

0	0	1	0	0	0	1	1

除算は右シフトで行います。
７÷５＝1・・２（余り）をＣＰＵの演算装置をイメージして計算してみましょう。
７を８ビットの２進数で表わすと以下のようになります。

0	0	0	0	0	1	1	1

1 ビット右にシフトすると 1/2 倍の 3.5 になるはずですが。

0	0	0	0	0	0	1	1

→ 1　ビット落ちで 3 になってしまいます

さらに１ビット右にシフトすると 1/4 倍の 1.75 になるはずですが。

0	0	0	0	0	0	0	1

→ 1　ビット落ちで１になってしまいます

2 で割り切れる整数の割り算ならば、シフト演算でもできそうですが、他の方法と組み合わせないと 7÷5 はできそうにありません。
タヌキ、知識的興味があれば、シフト演算の詳細は、自分で勉強してね。

ＯＫ！何を勉強すればいいかわかったよ！

基礎的なシフト演算の説明を終えたので、最初に戻って＋１０÷１０＝１・・・０の説明をしますよ！
アセンブラ言語を使って説明すると複雑で混乱するからアウトラインを図解することにするよ。

［＋１０の２桁目の１と１桁目の０を取り出す考え方］符号は無視

＋１０　÷　１０　＝　１（商）・・・・０（余り）
　Ａ　　Ｂ　　Ｓ　　Ｍ　　と置きます。

（初期設定）ＣＰＵ内のレジスタを以下のように用意します。

Ａ

0	0	0	0	1	0	1	0

Ｓ

0	0	0	0	0	0	0	0

Ａ＜Ｂになるよう１ビット左にシフト

Ｂ

0	0	0	1	0	1	0	0

Ｍ

0	0	0	0	0	0	0	0

Ｌは操作対象の桁を指示、Ｂと連動

Ｌ

0	0	0	0	0	0	1	0

（手順①）Ａ　＜　Ｂならば

・Ｂを右に１ビットシフトします。

Ｂ

0	0	0	0	1	0	1	0

Ｓ

0	0	0	0	0	0	0	0

・Ｌも右に１ビットシフトします。

Ｌ

0	0	0	0	0	0	0	1

Ｍ

0	0	0	0	0	0	0	0

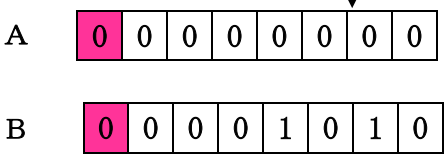
Ａ

0	0	0	0	1	0	1	0

（手順②）Ａ　＞＝　Ｂならば

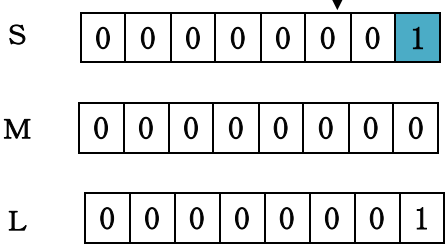
・Ａ　－　Ｂ

```
  00001010
- 00001010
----------
  00000000
```



・Ｓ　＋　Ｌ

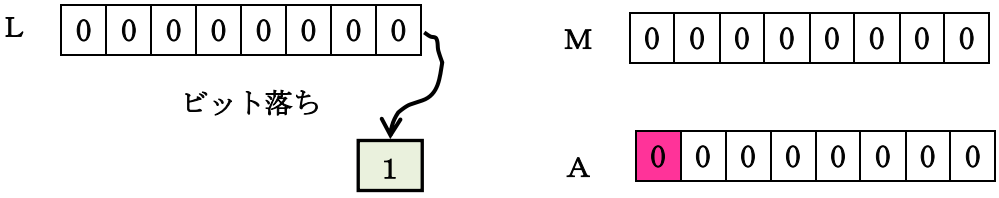
```
  00000000
+ 00000001
----------
  00000001
```



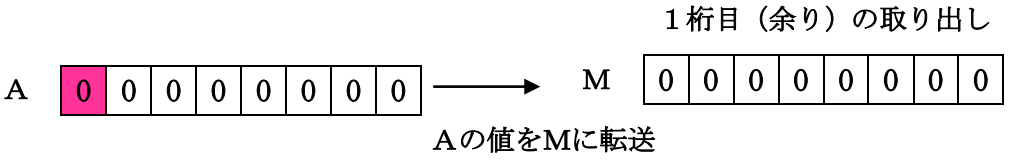
（手順③）Ａ　＜　Ｂならば

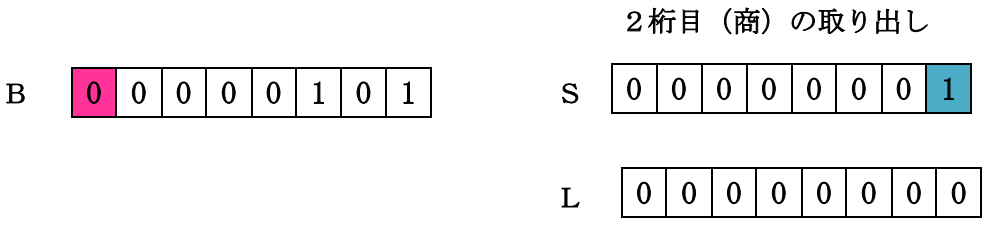
・Ｂを右に１ビットシフトします。

・Ｌも右に１ビットシフトします。



（手順④）Ｌ　＝　０　ならば





（手順⑤）ゾーン部［００１１］の加算

２桁目（商）の取り出し

S	0	0	0	0	0	0	0	1
＋	0	0	1	1	0	0	0	0
	0	0	1	1	0	0	0	1

３１（16 進数のキーコード）

１桁目（余り）の取り出し

M	0	0	0	0	0	0	0	0
＋	0	0	1	1	0	0	0	0
	0	0	1	1	0	0	0	0

３０（16 進数のキーコード）

※３１３０で出力が可能になります。

0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0

最も簡単な例で説明したよ。
上記のやりかたで、「２７　÷　１０」で２と７が分離できることを自分で確かめてみてよ。練習課題というところかな。
ＣＰＵから記憶装置に送られた結果は、ざっと、こんな感じで出力装置に送られるんだ。これが基本さ。
以上のように、コンピュータの５大機能の入力（データ）から出力（データ）までをイメージできるようになることが、コンピュータを学ぶ上での基本であり、これから勉強していく上でのポイントなのだ。森をみたら、これからは木（詳細）を見て行けばいいんだ。

キツネが言いたかった事が良くわかった。
難しい部分もあったが、これが基本なのか。オイラ、すぐに拳闘ゲームのプログラムが作れれば良いと思っていた。

基本を理解してないと、プログラミングができても結局、部品（ライブラリやクラス）をくっ付けるだけで全体像が見えなくなってしまうぞ。ちょうどチャップリンのモダンタイムスの世界に入り込むことになる。
ポンポコ大学の情報のポンキチ先生は、計算結果を主記憶装置を経由させないで、CPU の演算装置から直接モニターに出力する矢印を描いて平然としていたが、これは記憶装置を単なる記憶をつかさどる働き、とだけ理解し、パックとゾーンの変換過程の記憶という重要な部分を理解していないから生じる誤りなんだ。それぐらい基本は大切なのさ。
タヌキは、基本ができたから第５話に進もうか。第５話からは木の部分（詳細）に入って行くよ。

第５話へ