6月19日　金曜1限　情報Ⅰ課題　補足解説

情報というかパソコン関係は、わからないと本当にわからなくなってしまいがちです。それを少しでも理解してもらって、成績につながればと思って作りました。
かなり基礎の基礎から（おそらく）難度順に書いています。自分の理解に合わせて、必要なところから後を読んでいただけると良いと思います。

● やること
IC トレーナーで回路を作ります。配線を上手く組み合わせて、想定された回路が作れれば成功となります。

§1 そもそも……

コンピュータが計算をするとはどういうことでしょうか？では試しに人間で比較してみましょう。ただし、簡単のためにこのプリントの中では数字が全て10進法で1桁であるものとします。

3+5=？

この程の計算は、日本最高峰の東大の入試試験を突破してきた皆さんにとってみれば楽勝のはずですね。少し細かく見てみると、皆さんは「3」という数字を認識して、「5」という数字を認識してから、その和である「8」を算出したわけです。「算出」についてはもう少し後で説明しますが、ここで注意して欲しいのは、

「数字を数字そのものとして理解した」

ということです。至極当たり前のことを書いているので逆にわかりにくいかもしれません…。すみません
ところが一方コンピュータは機械なので、スイッチのONとOFF、すなわち1か0かの情報しか認識できないのです。「3」という数字が与えられても、「？？？」となるわけですね。

そこで登場するのが「2進法」という考え方です。1か0の数字を横に並べることで、1の位、2の位、4の位、8の位、…、2^nの位、…と位取りをします。そしてそれらの数の和として数を表現するのです。
例えば「6」だったら、4+2で表されるので、
4の位=1
2の位=1
1の位=0
として、

$$(6)_{10} = (110)_2$$

となります。ここで$(6)_{10}$のように表したのは、6というのが10進法表示における6であることを示すためです。6なら特に間違いませんが、110だったら「百十」なのか「2進法での110」なのかがわからなくなってしまいます。

ここでは、コンピュータが

「数字を 1 と 0 の組み合わせで理解している」

ということを書いておきました。

§2　計算をするということ

私たちが計算をするときは、どのように考えているのでしょうか。先ほどの 3+5=8 においては、暗算を使って考える間もなく答えが出てきました。しかし実際にはそのように計算を「理解（認識）」しているのでは無いはずです。

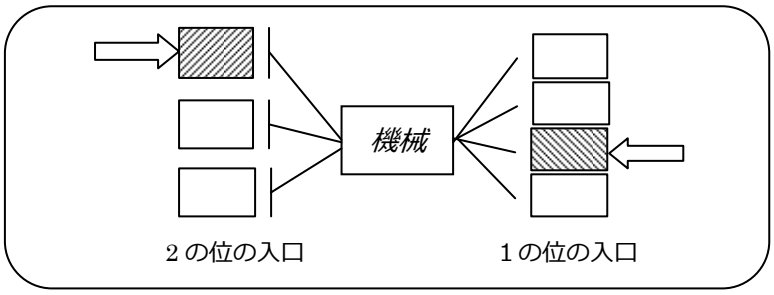
『3 個の「もの」と 5 個の「もの」を集めて、再び数え直すと 8 個になっている。』

このように、「個数を数える」ことで数字を認識しているのです（もちろん自然数の範囲で…）。

さて、当たり前のことをぐだぐだと述べましたが、これがコンピュータの中になるとどうなるかを考えます。そもそもコンピュータが勝手に「あ、ちょっと計算してみようかな…」などと考えて勝手に計算を始める訳はありません。外部の入力（人間の操作）を受け取って、正しい答えを出力するのです。

では人間が 3 と 5 をぶち込みましょうか。しかしちょっと待ってください、コンピュータは「3」や「5」をそのまま理解することができません。なので工夫が必要になってきます。3 は 2 進法で$(11)_2$と表せるので、1 を二回入力しましょうか。しかしそれでは二つの「1」が同等なものとして扱われて、$(11)_2$=3 が入力されたのか、純粋に 1 が二回入力されたのかわかりません。

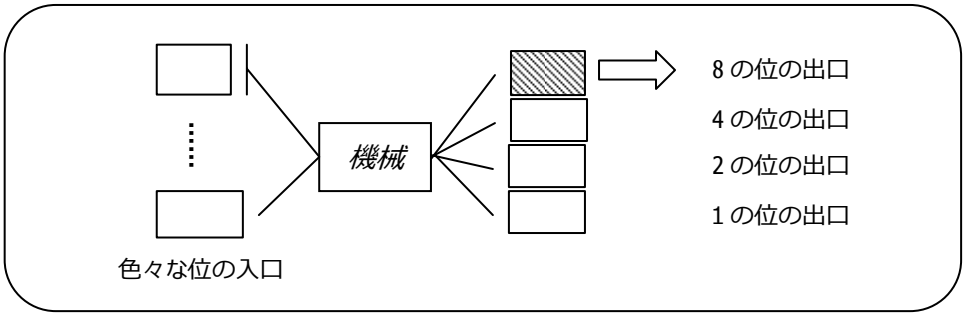
そこで行うのが、「2 の位を入力する入口」と「1 の位を入力する入口」を幾つかずつ用意しておいて、それらの入口にそれぞれ一つずつの「1（ON という情報）」を与えるという手法です。



3 を機械に入力するためには

これで 3 が機械に入力できたことになります。同様にして、4 の位の入口に一つ、1 の位の入口に 1 つの「1（ON という情報）」を入力すれば、機械に 5 を入力したことになります。

では、機械が計算をします。この過程は次の章で解説するので、今は計算をしてくれたことにしておいてください。計算をした結果、「8」つまり$(1000)_2$という ON と OFF の情報が出力されなければなりません。どうするかというと、もう予想がつくと思いますが、8、4、2、1 の位の出口を用意しておけばいいのです。



8 を出力するためには

出力するのが一つの数字（一つの ON 情報ではありません）の場合、出口はそれぞれの位に一つずつあれば十分です。入口の場合は……後で説明します。しかしここでは「ある位の入口に ON 情報を入力した結果、適切な出口から ON 情報が出てくる」ことがコンピュータにおける計算だということが理解できれば十分です。人間のやっている「数え直し」とは全く違うということを確認しておいてください。

§3　実際の計算のために（論理ゲート）

「計算とは何か」がわかった以上、コンピュータに計算をさせるには、「入力された値に対して正しく答えが出てくる仕組み」があれば良いわけです。簡単のために一旦足し算のことは忘れておきましょう。ここでは、「入口は 1 の位の入口が二つ、出口は 1 の位の出口が 1 つ」だけついている機械を考えます。ここに値を入力したとき何が出力されるかで、機械を分類してみます。

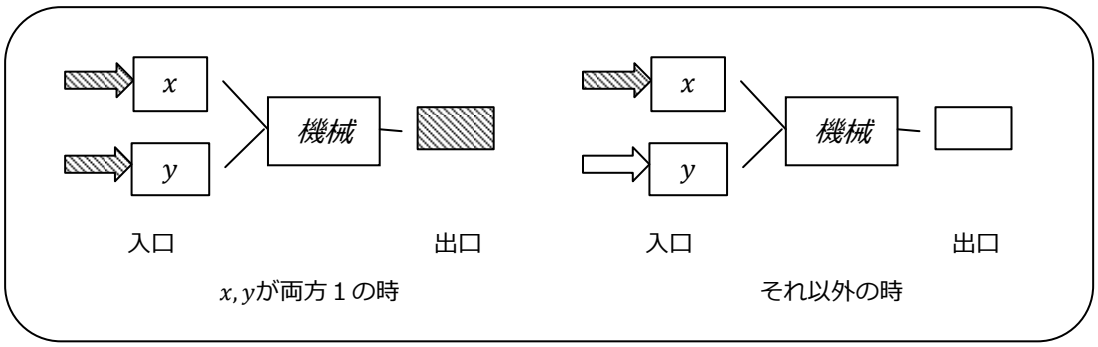
一の位の入口には、ON と OFF のどちらかを入力（OFF を入力、というのもおかしな話ですが…）することができます。その入口が二つあるので、入力の仕方は全部で 4 通りになります。例えば入口の片方をx、もう片方をyとすれば、下のような表が作れます。

x	0	0	1	1
y	0	1	0	1

xとyへの入力の組み合わせ

この 4 通りに対して、一つずつ出力の値（ON または OFF）が決まるような機械を考えます。

例えば、「xとyが両方とも ON（=1）の時だけ ON（=1）を出力する」という機械を考えてみましょう。



AND ゲートを表す図

この操作は、「xとyが『何かの機械』を通ることで違う値に変換された」と見ることができます。この『何かの機械』を**ゲート**と呼ぶことにしましょう。特に、上のように「xとyが両方とも 1 の時だけ 1 を出力する」ようなゲートを **AND ゲート**と呼びます。その他にも、「xとyの少なくともどちらか一つが 1 であれば 1 を出力する」ものを **OR ゲート**と呼びます。AND ゲートと OR ゲートは、上のxとyの表に対して、次の表のように出力をします。

x	0	0	1	1
y	0	1	0	1
AND	0	0	0	1
OR	0	1	1	1

AND ゲートと OR ゲートの出力結果

2 入力 1 出力があるなら、1 入力 1 出力や 0 入力のものがあってもおかしくないと考えるのは普通で、「1 が入力されれば 0 を、0 が入力されれば 1 を出力する」ゲートもあります。これは入力を否定するという意味で **NOT ゲート**と呼ばれます。後は、そもそも入力をしなくても 0 や 1 を出力する仕組み（これはゲートとは呼びません）もあります。

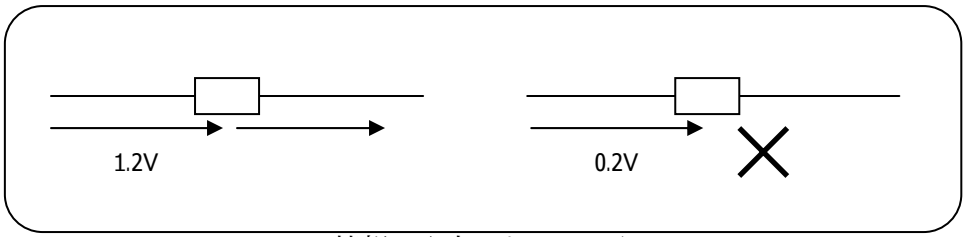
最後にゲートでないものを出しておいてまとめをするのは良くないのですが、これらのゲートを**論理ゲート**といいます。「何らかの論理（これまでカギ括弧で括ってきたものです）」に従ったゲートの働きをする、という意味です。

言葉の定義だけを述べておくと、上で AND ゲートと OR ゲートの出力結果を表した表がありますが、このように計算の入力と出力を表にしたものを**真理値表**と呼びます。

※補足　ゲートとは？

機械をゲートと呼ぶ、と簡単に言ってしまいましたが、ではその機械とは何なんだと問いたくなると思います。

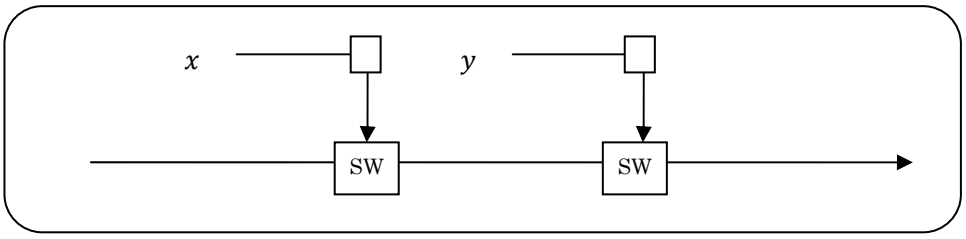
これを説明するためにまず知っておくべきことは、入力される ON や OFF の情報は、回路に掛かる電圧で操作されるということです。例えば、回路のある部分に、1V の電圧が掛かっていれば電流を流しそれ未満だったら流さない、という仕組みを付けておけば、電流は ON 情報として回路を流れることになります。ON 情報を流すには、何か増幅機を使って電圧を上げれば良いのです。



ON 情報が入力されるということ

時間が無くなってきたので AND ゲートだけを説明します。xとyの入口から ON 情報が入ったときに、スイッチを閉めるような仕組みを作っておきます。それを回路に二つ取り付ければ AND ゲートになります。

あっさり言ってしまいましたが、図にすると下のようになります



AND ゲートの中身

こうすれば、xとyの両方に ON 情報が入っているときに両方のスイッチが ON になり、回路（長い矢印）に電流が流れ、左からの ON 情報が右から出力されることになります。並列にすれば OR ゲートができるので考えてみてください。

§4　実際の計算のために（ブール代数）

話がぶっ飛んで参りました。なのでここはあっさり行きます。ここまでで、コンピュータに論理的な計算（この入力にこう出力しろ、というもの）をさせることを、論理ゲートを組み込むことで可能にしました。この章ではそれを表す記号を導入します。

xとyの入力を AND ゲートに通す、という指令を AND(x,y)と書きます。この記号は、xとyの値によって 0 か 1 を意味します。同様に、xを NOT ゲートに通して否定する、という指令を NOT(x)と書きます。ゲートに通すという操作は、関数にxやyを代入するような感覚と同じものであり、これらの AND、NOT、OR を関数としてみることが可能です。これらを**論理関数**と呼びましょう。

私たちが普段使うような関数は、例えば$f(x) = x^2$のように、xを代入したときの形として具体的に表現することができました。では論理関数は同じように表現できるのでしょうか。

二変数関数 AND(x,y)について考えると、xとyが両方とも 1 の時だけ 1 になり、片方でも 0 なら 0 になれば良いので、簡単に AND(x,y)= $x \cdot y$と表すことができます。

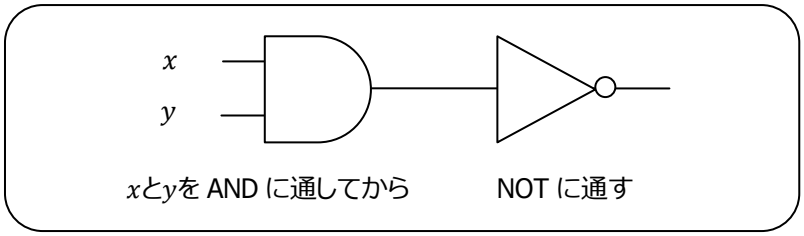
一方 NOT については、1 と 0 を上手く反転させるような実数関数は見あたらないので、新しく記号を定義してしまうことにします。NOT(x)=$\overline{x}$と表現します。xは 0 か 1 のどちらかしか取らないので、xが 0 なら$\overline{x}$は 1 です。このことから、$x + \overline{x} = 1$ や$x \cdot \overline{x} = 0$ などの性質がわかります。

このように、論理関数を数式で表したものを**ブール代数**と呼びます。この辺りの表現は教科書に上手く紹介されているのでそこを参考してください(p.161)。

同時に、NAND や NOR、XOR などの論理関数も把握しておきましょう。頭に N の付くものは、ある関数の値をさらに NOT ゲートに通したものです。合成関数というとわかりやすいかもしれません。NAND(x,y)=NOT(AND(x,y))となります。XOR については、特に合成というわけではなく、真理値表に表されたように、xとyのどちらかが 1 で、なおかつどちらも 1 の場合を除いた時に 1 を返すような関数を新しく定義しただけです。EQ も同様で、xとyの値が等しいときに 1 を返します。ここら辺は約束事なので、理解できないということは無いはずです。

さらにさらに、関数を箱のようなものと考えて、xとyを文字通りぶち込んだ場合に何か値が出てくる、というイメージを表すために、ある決められた「絵」を使います。先ほどはゲートを一般に「*機械*」と表現していましたが、それを絵にして描きやすく区別しやすくしたものです。教科書 p162 にその絵が紹介してあるので見ておいてください（パソコンだと作るのが面倒なんです…ごめんなさい）。

例えば、NAND ゲートを作るために AND と NOT ゲートを組み合わせることを考えます。



AND ゲートと NOT ゲートを組み合わせる

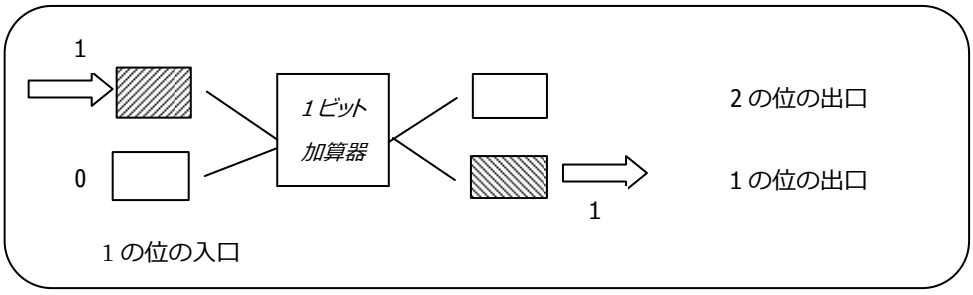
この章では新しく関数を紹介しつつ、関数を数式で表す方法を紹介しました。細かいことをいちいち覚えるのでなく、理解しながら進めると良いと思います。

§5　2 進数の 1 桁同士の足し算

ようやく足し算に戻ってきました。ここでは簡単な 1 桁の数二つの足し算、つまり 0+0=0、1+0=1、0+1=1、1+1=2 の三種類を考えてみましょう。

これらの計算ができるとはどういうことかは先ほど紹介しました。1 の位の入口が二つあり、そこには 1 か 0 の数を入力することができます。その入力された情報が、何らかの機械（ゲートの組み合わせ＝関数）を通して、正しい答えとして出力されれば良いのです。このとき出力される可能性のある数は 0、1、2（＝$(10)_2$）の三つであり、出口としては 1 の位の出口と 2 の位の出口が一つずつあれば十分です。

ここで言う機械は、**1 ビット半加算器**と呼ばれます。なぜ「半」なのかはよくわからないので考えないようにします。



1 ビット加算器（1+0=1 の例）

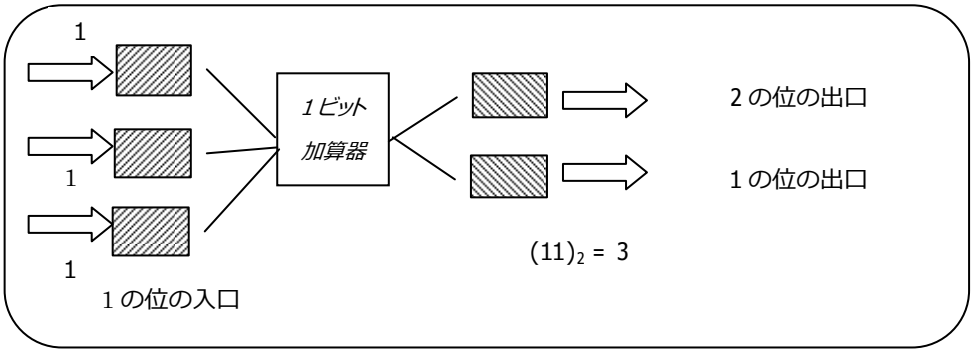
この 1 ビット加算器の仕組みが作れれば、この計算をコンピュータにさせられたことになりますが、

幸運なことにこの組み方は教科書にも授業ページにも載っていて、実習をする際に使う IC トレーラーの配線のつなぎ方も、どこかの神が用意してくれてあります。なのであとすべきことは、正しく配線をつないで、正しく 1+1＝2 が計算できるかを確かめるだけです。

§6　1 ビット全加算器

そろそろ 4 時を過ぎてきたので終わりにしたいと思いますが、最後に全加算器を紹介しておきます。半加算機から変わったことは、入力できる 1 桁の数が二つから三つに増えたことです。これによって、1+0+0=1、0+1+0=1、0+0+1=1、1+1+0=2、0+1+1=2、1+0+1=2、1+1+1=3、の 7 通りの入力に対して 2 桁の出力を返す仕組みを作ることになります。半加算機と同様に組み方は載っているので、それに従ってください。

その他、発展的な 2 ビット加算器や減算器、7 セグメント LED（これはまた別の話です）等については、別途相談してください。わかる範囲で教えます。



1 ビット全加算器（1+1+1=3 の例）

§7　まとめ

以上ざっくりと説明しましたが、説明が不適切だったりわかりにくいところがあるかもしれません。そのときは伝えてくれると助かります。コンピュータ内部における本当に基礎的な演算については一応解説したつもりです。この辺りの範囲は結構難しいので先に進むとまた混沌の世界が待っているかもしれません。でもあまりに複雑なので試験には出ません（そもそも半加算機以降が試験範囲外です）。なのでご安心を。一年の範囲の情報だったら、本当に細かい知識よりも「いかに論理的に考えるか」が問われると思うので、内容を理解することを優先すると良いかもしれません。もちろん覚えないといけないことは多々あります。