SIMPLE BINARY ADDER/SUBTRACTER

We carry out our arithmetic in the decimal system, which has ten symbols, 0 to 9. In order to set down numbers greater than 9 we have tens, hundreds, and thousands columns. This "positional" concept, where a 5 in the tens column for example means fifty, is an important part of our system of numbers. The Romans had a decimal system, but no positional notation. Thus they had different symbols for 1, 10, 100 and 1000 (I, X, C, and M respectively), and if they wished to write down an infinitely great number would have needed an infinitely great number of different symbols.

We tend to accept ten as the basis of our system of numbers without question. Other bases are possible, the duodecimalists believe that it should be twelve, in which case there would be twelve different symbols and dozens and gross columns instead of tens and hundreds. The duodecimal system is logically preferable to decimals, twelve having more factors than ten, but it has little chance of acceptance in a world widely committed to the base of ten.

One different system which is widely accepted for internal use by computers is the Binary, with a base of two. Only two symbols are used, 0 and 1, greater numbers being catered for by using the "positional" idea mentioned above. Valve and transistor circuits are easily designed to work as switches, that is, they are either in an "off" or "on" condition. Having two possible states for each "trigger" as it is called, we can let one of these states represent 0, the other 1. By connecting the triggers in chains to form a "register" we can store a binary number of any size, we would, for instance, require twenty triggers to be able to store any number up to a million. Because each trigger is working at opposite extremes, the effects of component tolerances, valve ageing etc., are not important and may be allowed for. The accuracy of calculation depends entirely on the number of digits processed, i.e. on the size of the registers. This is in marked contrast to the analogue type of computer, where the exactness of the answer depends on the accuracy and stability of the components used.

In the decimal system with ten symbols, each succeeding column to the left is ten times the preceding one. Similarly in the binary system with two symbols, each succeeding column to the left has twice the value of that on its right. The columns could be labelled:-

Decimal	Thousands	Hundreds	Tens	Units
Binary	Eights	Fours	Twos	Units

Thus five is represented 101 (4 + 1), fifteen is 1111 (8 + 4 + 2 + 1), and one thousand is 1111101000, (512 + 256 + 128 + 64 + 32 + 8). This shows one disadvantage of the binary system; that it requires more columns than decimals.

The normal rules of arithmetic apply. To take a simple example first:-

```
  1011   (11)
+ 10100  (20)
  11111  (31)
```

Decimal equivalents shown in brackets

Now note the carry operation:-

$$\begin{array}{rr} 1001 & (9) \\ + \ 101 & (5) \\ \hline 1110 & (14) \end{array}$$

In the units column, 1 and 1 are 10 (2), write down 0, carry 1.

Or again, slightly more difficult:-

$$\begin{array}{rr} 1111 & (15) \\ + \ 111 & (7) \\ \hline 10110 & (22) \end{array}$$

In the units column, 1 and 1 are 10 (2), write down 0, carry 1. In the 10's (2's) column, 1 and 1 are 10 (2), add carry 1 makes 11 (3). Write down 1, carry 1, and so on.

In the simple machine to be described, there is an upper row of lamps (Row A), and a lower row (Row B). A lamp which is lit represents a 1, when it is unlit it is a 0. The lamps are controlled by switches which also have interconnected contacts to give carry outputs to the next stage and outputs to the answer row of lamps in accordance with the rules of arithmetic. The display panel of the computer appears in similar form to any of the sums set out above, with a light wherever there is 1, and an unlit lamp wherever there is 0.

To return to the rules of arithmetic, here we have two subtractions:-

$$\begin{array}{rr} 1111 & (15) \\ - \ 111 & (7) \\ \hline 1000 & (8) \end{array} \qquad \begin{array}{rr} 1100 & (12) \\ - \ 11 & (3) \\ \hline 1001 & (9) \end{array}$$

In the second example we have the usual "borrowing" operation, but in binary arithmetic it is 10 (2) we borrow, not ten.

In the practical circuit, however, subtraction is performed in a different way (subtraction by complement). The rules for addition and subtraction being different, it follows that the interconnections between switches would need to be changed around to perform the two operations. Subtraction by complement gets round this difficulty by doing an addition in the main body of the computer. How this is done will now be described, first in decimals, then in binary form.

Take some convenient number greater than any number we are going to work with (say one hundred). We call this the modulo (mod.). Now the complement (comp.) of a number is the difference between that number and the mod. So the complement of 37, for mod. 100, is 63. As you see, comp. + number = mod.

Now suppose we have:-

$$\begin{array}{r} 83 \\ - \ 37 \end{array}$$

We say

	83
Add comp. 37(mod 100)	63
	146
Subtract mod.	100
	46 which is the right answer.

This operation can be set down generally as A - B = A + (Mod - B) - Mod.

The reader will protest that we have two subtractions now, (100 - 37and 146 - 100) instead of one, so what of the alleged advantages?

The point is that when we come to the binary system we choose the mod. to be such that the subtractions are very easily performed even in a circuit which is designed for addition only, and has no "subtraction logic".

Consider the practical circuit. The rows A and B each hold a four bit number ("bit" being a contraction of "binary digit"). The greatest number we can put into A or B is 1111 (15). Let us take this as our mod.

Now to work out a few complements:-

Mod 1111	Mod 1111	Mod 1111
- 0011	- 1010	- 1100
Comp 1100	Comp 0101	Comp 0011

As can be seen from the above, the comp. may easily be found by changing the 1's in a number to 0's, and vice versa. This is accomplished in the machine by changing over the lamp connections in the B row, so that when the switch is "off", i.e. at the 0 position, the lamp lights, i.e. indicates 1, and vice versa. So although the lamps in the B row indicate a number, the switches are set to the complement, and it is this complement which is fed into the adding circuits.

The number in A, and the complement in B are now added, and all that remains is to subtract the mod. (1111). We do this by the simple expedient of deducting 10000 (16) and adding 0001. Although the machine works with four digits in rows A and B, a fifth column has to be provided in the answer to take any carry from the 1000 (8) column. It can be shown that where A is greater than B, that is where the answer is positive, a 1 always appears in the 10000 (16) column of the answer. We can easily deduct 10000 (16) by disconnecting the lead which feeds this particular lamp. We can go one better than this and use the lead we have just disconnected to feed a 1 into the units column which is then added into the answer like any other carry. This transfer of a 1 in the 10000 column to the 0001 column is called "end around carry".

We can now work out a couple of examples:-

1011 (11)	1110 (14)
- 111 (7)	- 11 (3)
100 (4)	1011 (11)

By complement:-

	1011	(11)	+ comp.0011	1110	(14)
+ comp. 0111	1000	(8)	+ comp.0011	1100	(12)
	10011	(19)		11010	(26)
-	10000	(16)	-	10000	(16)
	00011	(3)		1010	(10)
+	1	(1)	+	1	(1)
	00100	(4)		1011	(11)

In the machine, of course, all these operations take place simultaneously, there is no question of it have to work step by step.

Where B is greater than A, that is where the answer would be negative, there is no output to the 10000 (16) column of the answer, therefore there is no 1 to be carried round to be added into the units column. The method does not break down, but the answer shown is the complement of the true answer and is negative. Exactly the same happens if we try to subtract the greater number from the smaller in decimals, for example:-

	37
-	83
	9954

with as many 9's to the left as we care to put down. In this case we can take 10,000 as the mod., and the true answer is 9954 - 10,000 = - 46. We usually avoid these complications by putting the greater number in A, but if desired, a "tell-tale" lamp may be connected to the "Not Carry" (denoted $\bar{C}$) line feeding back to the units column. It should be mentioned here that each carry (C) line in the machine has a partner, a "Not Carry" ($\bar{C}$) line. When the C line is energised, $\bar{C}$ is not energised, and vice versa. As we said at the beginning of this paragraph, when the number in B exceeds A, there is no end around carry, the corresponding $\bar{C}$ line is therefore energised and may be used to light a lamp labelled "Answer a complement and negative."

The reader will probably be tempted to ask what happens when A = B. The machine shows the right answer, which is zero, although logically it should show 1111 and light the tell-tale lamp.

This concludes the description of the operations which can be carried out on the machine, but in order to keep all the background material together, and to prepare the way for a fairly simple demonstration binary multiplier, a description of the multiplication process will follow.

After following the tortuous method of subtraction by complement the reader will realise that the method is always adapted to the machine, not the other way round.

A way of multiplying which is suitable for computers is the doubling and halving method. In its simplest form we might want to know 8 × 37. Now 8 × 37 is obviously the same as 4 × 74, which equals 2 × 148, which is the same as

1 × 296. The answer is then 296, found by doubling and halving.

Not all numbers keep on halving until they reach 1, like the 8 in the example just given, but for any two numbers we proceed as follows:-

Write down the numbers side by side. Then, underneath start doubling one number, and halving the other discarding fractions of 1, until we have reduced the diminishing number to 1. We add into a total all those doubled numbers which are opposite an odd halved number. (It is quicker to strike out all those opposite an even halved number, then add the rest). The total is the required answer.

A couple of examples should make this clear:-

9	×	17
4		~~34~~
2		~~68~~
1		136
		153

49	×	126
24		~~252~~
12		~~504~~
6		~~1008~~
3		2016
1		4032
		6174

Note that when we halved 9, we dropped the $\frac{1}{2}$ and wrote 4.

This is a perfectly general method, and involves only the ability to multiply and divide by two, and add. As a matter of interest the method can be adapted to deal separately with 100's, 10's and units columns, as, for example:-
To multiply 946 × 1312

9	4	6	131200
			~~13120~~
			~~1312~~
4	2	3	~~262400~~
			~~26240~~
			2624
2	1	1	~~524800~~
			52480
			5248
1	-	-	1049600
			1241152

Now a method which depends on doubling, halving, and adding, is highly suitable for a binary digital computer. We can modify the registers which store the numbers by connecting a "shifting gate" between neighbouring triggers, and whenever a "shift pulse" is applied to the gates each trigger will discharge its contents (0 or 1) into its neighbour, in one direction. The whole number is thus despatched down the chain of triggers, one step for each pulse. If the number travels towards the units end of the register this is equivalent to a division by two at each step, if towards the other end then it is a multiplication by two. Thus, in a dividing shifting register capable of storing a four bit number, a number 1100 (12) would be 0110 (6) after the first shift pulse, 0011 (3) after the second.

Let us have a look at a multiplication done in binary form using the doubling and halving method.

1001	(9)	×	10001	(17)
100	(4)		~~100010~~	(34)
10	(2)		~~1000100~~	(68)
1	(1)		10001000	(136)
			10011001	(153)

Now we need two shifting registers to hold the numbers to be operated on, one dividing, the other multiplying by two at each shift pulse. An adding circuit operates on the multiplying register, and adds its contents into a third register called an Accumulator, which holds a running total. This only happens when the number in the dividing register is odd, when it is even the adder is paralysed (inhibited). Odd/Even detection takes place at the units column on the dividing register. Since all even numbers end in 0, and all odd ones in 1, it is a simple matter to derive a control voltage to operate on the adder. In passing it will be noted that as we keep on dividing by two the number "drops off" the end of the register and is lost. If we apply n shift pulses to a register with n triggers, the whole number is lost and the register is cleared. If we connect the units end to the highest value column end, however, instead of dropping off, each digit will be fed back, so that after n pulses the number is in the same position as it was originally. Similarly we can transfer a number from one register to another by connecting together and applying shift pulses, or transfer a number to a storage or read-out device.

It is hoped that it will be possible at a later date to publish a practical circuit to carry out a multiplying function.

The practical adding/subtracting circuit may be considered as a series of four stages, one for each column in the addition. Each stage consists of a lamp and switch in the A and B rows, and a third lamp to show the sum or difference. The inputs to each stage, speaking in the logical not the electrical sense, are the state of each switch and the carry from the previous stage. The logical outputs are the 0 or 1 in the sum/difference lamp and the carry to the next stage. We can construct a "truth table" showing the outputs required for various input conditions. In the table shown here all the possible combinations of A, B and the Input carry are shown in eight rows. Alongside are shown the outputs required, which we derive by using the normal rules for addition. Any circuit which will give these outputs for the input conditions noted is capable of adding. This particular circuit has been designed to use switch contacts and lamps only, but the variations are legion. All four stages are identical, and the circuit has been shown for one stage only. An Add/Subtract switch operates only on the "end around carry" C and

A	B	Cin	Out	Cout
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
0	0	1	1	0
1	1	0	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

¢ lines, and on the lamps in the ~~output~~ row. With certain operations, say 1010 + 0101 = 1111, the current for a maximum of eight lamps flows through a single pair of contacts. It is wise therefore not to use too high a current rating for each lamp, 0.15A is probably the best compromise. The number of stages used may be increased beyond four, but there seems little point. If a "tell-tale" lamp is fitted to signal when B exceeds A in subtraction it should have a coloured lens to avoid confusion with the digits (clear or opal lenses).

PARTS LIST

4 Switches, rotary wafer, 6 pole 2 way (A switches)

5 " " " 4 " 2 " (B switches and Add/Subtract Switch)

9 Short pointer knobs

13 MES Signal lamps, Bulgin D180, less bulb, Clear or opal.

1 " " " " " " " Red or Green.

14 Bulbs, MES, 6.3V 0.15A.

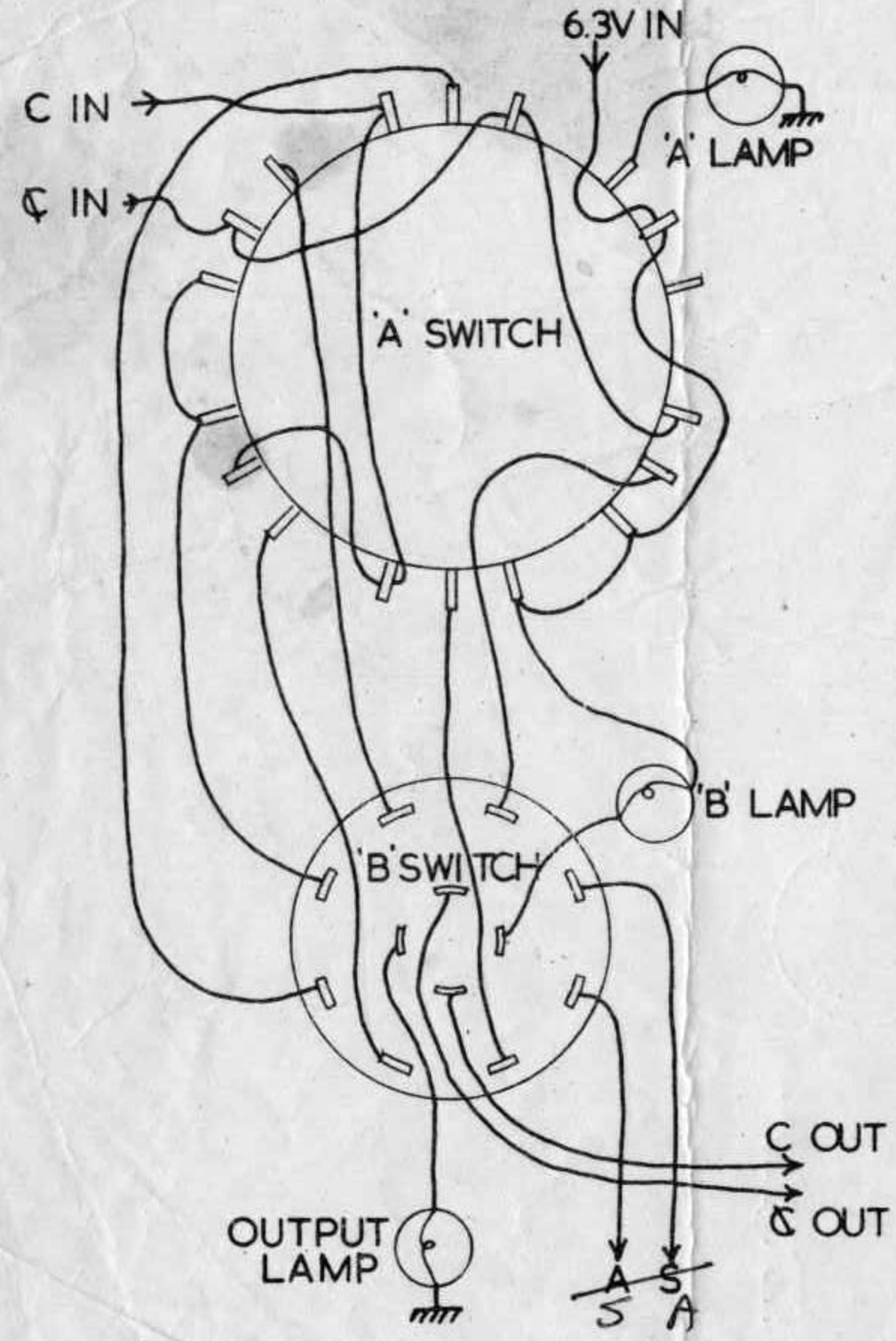
1 Mains toggle switch

1 " transformer, 6.3v 3A out.

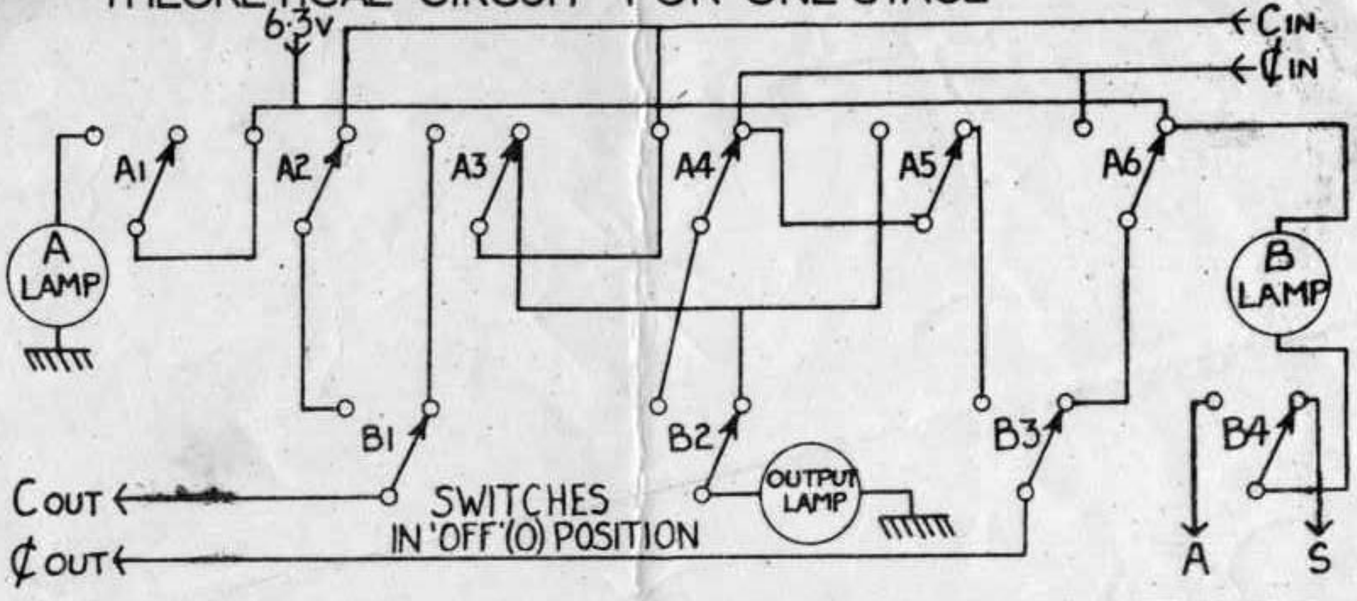
Panel as required.

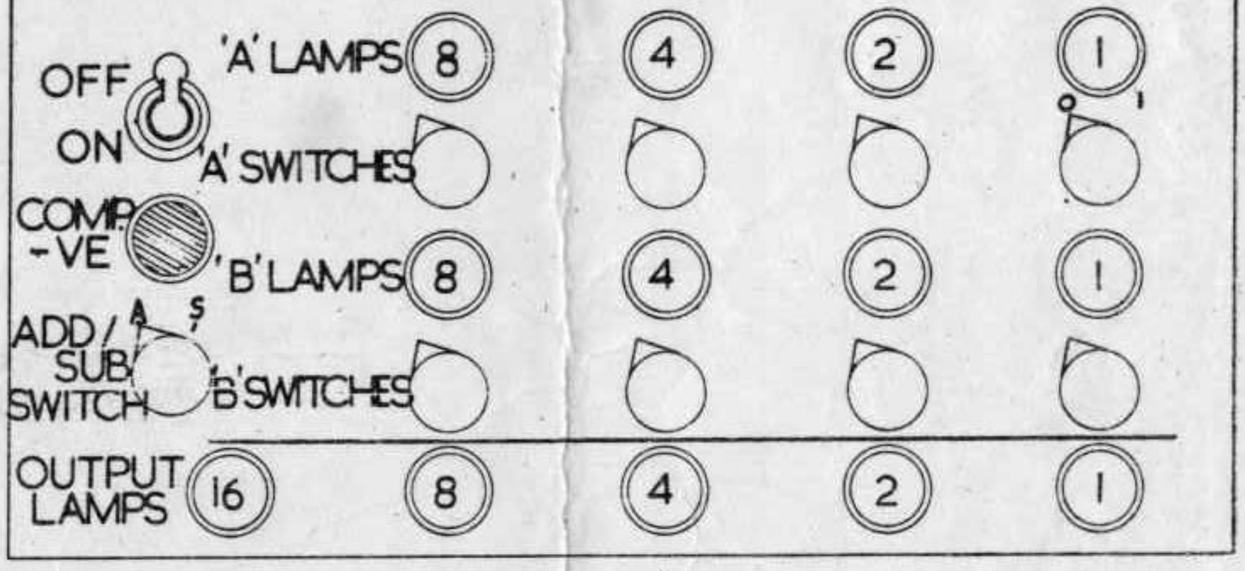
WIRING DIAGRAM FOR ONE STAGE

AS VIEWED FROM REAR OF PANEL



THEORETICAL CIRCUIT FOR ONE STAGE





INTER - STAGE AND ADD/SUBTRACT SWITCH CONNECTIONS