MERAC Evaluator

Rainer Glaschick, Paderborn, Germany
email: rainer@glaschick.de
http://rclab.de/
2019-01-14

1. Summary

MERAC stands for *Mechano-Electrical Retrograde Automatic Computer* and is the name for a family of simple programmable computers, that have been inspired by the *Analytical Engine* (AE), the ENIAC and other early computers. MERAC is useful for me in explaining these.

The available test machine has two decimal registers with 3 digits each, just as a proof of concept. An instruction has 7 bits, and in the first version, the sequencing is done by the user. (A card reader is under preparation.) A concept for automatic programme sequencing including structured conditional execution has been designed, but not yet built.

This model shows that latest at the begin of the 20th century, when telephnone relays were build in volumes, a programmable electro-mechanical (decimal) computer may have been built with relatively small effort.

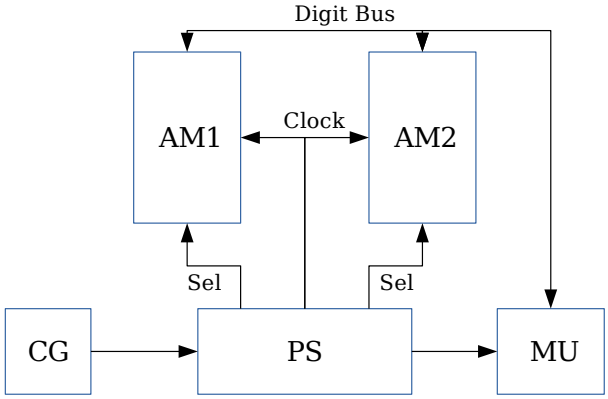
Comparisons with the above and other early computers are collected at the end.

2. Architecture of the decimal MERAC

The machine has a number of decimal registers, and a programme can add the contents of one register to a different one. The decimal digits are stored as rotation angle of a disc.

2.1. Components

The components are shown in the following schema for two registers:



A clock generator (`CG`) sends clock pulses to the program sequencer (`PS`). The `PS` selects (signal `Sel`) the (additive memory) registers `AM1` und `AM2` according to the information in a (micro-) programme line, sends common signals (`Clock`) and controls the modifier unit (`MU`); the latter couples the input and output part of the digit bus, which has three lines each for input and output, if the registers are three digits wide.

A number read from a source register sends a number of pulses corresponding to its state over the bus to the target register. As the digits are stored in rotational discs, it is only natural to add these pulses by advancing the disc. A carry has to be done if a digit disc changes from 9 to 0 or farther.

Reading a number is done by a full rotation of the disc in forward direction, so that the content is not destroyed, and the disc finally left at the same rotational angle. The pulses are sent after the transition from 9 to 0 by a contact that is required for the carry mechanism anyhow.

There is no operation that replaces the value of a register independent of its previous value, because this would require to first rotate the disc to the zero position, which would slow down the machine significantly. Instead, any read can erase the source register; in this case, it is not avanced further when the zero position is reached. As only the change from 9 to 0 is sensed, the current contents is stil sent by the remaining clock pulses. Note that the contents of a register is not sent starting with the first clock pulse, but started during a clock cylcle once the remaining clock pulses represent the number.

The digit pulses are transmitted over a bus with one line per digit. If a register is selected as source, it sends the digit pulses to a bus line, and the register selected as target adds these to the current state. All not active registers are disconnected logically, i.e. ignore the bus pulses.

In contrast to other machines, the digit bus in split into an input- and output bus segment, connected by the modification unit (`MU`). During an addition, both bus segments are connected directly. For a subtraction, the pulses are inverted, i.e. a pulse is sent by the `MU` if there is no pulse received, and vice versa, so that the 9s complement is effected. If the source register sends 00021, the `MU` sends 99978 instead. A target register with 00123 becomes then 00101; the overflow of the highest order digit is ignored. If the 10s complement is used, the result is one less than the desired result. To correct this, in case of a subtraction the carry input of the least significant digit is activated, that would normally not be used, as during an addition there is no carry from the non-existant place before the lowest digit. So the result is 00102.

The `MU` will also be used to determine if the transmitted number was zero or negative, and allow conditional branches.

To allow multiplicaion via software, the MU can also shift the numbers in decimal places just by connecting the bus lines exchanged cyclically.

If source and target registers are binary coded in the instruction word, and all are present, another bit sent to the MU can inhibit the connection of the bus lines and thus send a zero. This is useful just to increment a register or in conditional jumps.

A digit is not transmitted physically as a number of pulses; the bus line is a gate signal and the receiving register uses a global clock to generate the pulses advancing the disc.

This means also that the drive needs not to be a stepper magnet; the discs could reside on a common shaft and coupled depending on a signal, which could be done either by a magnet or mechanically by a bus rail.

The MU can determine if the last digit is odd with a simple flip-flop, that counts the pulses modulo 2.

To determine if a number is zero, there is just a flag that is set if any digit pulse is transmitted.

The determination if a number is negative requires a modulo 5 counter, which could be derived from the normal store disc by just expanding the carry contact from 5 to 9.

2.2. Automatic program control

Except for the evaluation setup, a sequential programme carrier is assumed for the following configuration: The machine has 16 decimal register; the number of decimal digits is determinde by the width of the bus including the modification unit; there is no correlation to the number of bits in an instruction. At least 5 decimal digits should be appropriate.

One bit (the first one) in an instruction selects either data transfer or control instructions; the function of the other bits is dependent from the first one.

Data transfer instructions have two fields for the register numbers, binary coded, thus 4 bit each. The two bits L (clear) and E (unit carry) are also required and are sent to all registers, but only used if the register is selected as source (L) or destination (E).

Anoter set of bits controls the modification unit:

```
Z  Zero suppress data transfer
N  Ninescomplement
M  Multiply by 10
D  Divide by 10
```

Thus the instuction has at least 1+4+4+2+4 = 15 bits.

Control instructions have 6 control bits:

```
V  Forward (otherwise backward) advance
I  Invert condition
Z  Zero flag set
N  Negative flag set
H  High-order digit was zero
L  Low-order digit was zero
```

This leaves 8 bits to determine the target. Because the programme carrier is sequential anyhow (paper tape or punched cards), the tartget is not a distance by counting instructions — which is fine for random access — but simply a bit pattern, a label.

If the conditions trigger a (forward or backward) advance, then the programme carrier will be moved until another control instruction with the same pattern (label) is found; where a forward jump stops at a backward jump, and vice versa. The found control instruction is not executed; the next instruction is the instuction in forward direction.

The carrier can be advanced with the digit clock, i.e. much quicker than a single instruction execution, but clearly slower than a direct jump in a random access memory.

Application should use structured loops and conditions, i.e. each loop begins with a forward move instruction and ends with a backward move instruction. The label bit pattern is just the binary coded nesting level. Note that the above pattern allow the conditions *always* and *never*.

2.3. Input and Output

Output of results could be done with a Register where the number discs have elevated digits which are pressed onto paper through an ink ribbon, as known from desk calculators.

Input may come from a register, that is set a punched card. A selector relay could scan ten lines during one cycle, and set the output strobe if a hole is found. This could be done in parallel for all digits. An 80 colums card could contain 8 digits, if coding is unary, i.e. 10 holes per digit. Scan could be binary, in which case 32 groups of 4 bits would be required for 8 digits. However, advancing cards along the shorter side is not easy.

If paper tape is used, the tape reader has to read as many digits as required autonomously, and send them digitwise to a register.

In all cases, the programm must be stopped until the i/o device is ready.

The modification unit can also be used for input and output.

Of course, with enough program memory and subroutines, the data can be read in digitwise.

2.4. Subroutines

Subroutines are not available due to the lack of random access program memory. A (unsigned) multiplication requires about 10 instructions and can be inserted as a prepared instruction sequence. (Not in the evaluation machine, which has no step-down — divide by 10 — in the MU.)

2.5. Memoy requirements

A typcial task for a computer of this size is a linear regression computation (best line fit).

Let the value pairs (x_i, y_i) provided on punched cards, then the following sums have to be calculated:

$\Sigma\ x_i$

$\Sigma\ y_i$

$\Sigma\ x_i * x_i$

$\Sigma\ x_i * y_i$

So number of required registers is:

Input values x and y:
2
Multiplication:
3
Sums:
4

totalling 9 registers. As y can be in the multiplicand register, because it is only added to a sum and then consumed by the final multiplication, 8 registers are sufficient.

The calculation of the final results from the above sums can be done by a desk calculator, if the programming effort is not justified.

Even if the values were entered manually, if the programme could be stored on a card and sequenced automatically, such a desk calculator would have been a significant advantage, as the first pocket calculators show.

3. The Evaluator

The MERAC evaluator is the smallest machine, that allows a first check of the components and to demonstrate the elements of the concept.

Furthermore, I found it helpful to first explain the MERAC Evaluator as a simple machine, and then talk about commonalities and differences to the AE, ENIAC etc, as this allows to highlight features without explaining the historical machines in sufficient detail.

3.1. Setup

The evaluator has:

- two registers with three digits each
- an elementary modification unit
- manually controlled sequence of operations

Three digits per register are necessary and sufficient to test the carry mechanism.

The manually controlled sequencer has a (changeable) matrix, where the rows can be selected by a rotary switch and activated once each time a pushbutton is pressed. The sequence thus is controlled by the operator.

Each (micro-) programme instruction has 7 bits:

- A1: add to register 1
- R1: read from register 1
- A2: add to register 2
- R2: read from register 2
- L: clear register(s) during read
- E: set carry for increment by 1
- N: change to 9s complement

For better overview, bits that are not set are indicated by a point, and sets bits by a letter.

To add register 1 to register 2, the row has to be set as:

```
.R A. ...
```

To subtract in opposite direction:

```
A. .R .EN
```

To clear a register, it is read out, but not added to the other one:

```
.R .. L..
```

To increment the first register:

```
A. .. .E.
```

The E-bit is not coupled to the N-bit, to allow addition of 1 without using a (auxiliary) register.

If no source register is activated, zero is sent; by setting the N-Bit thus the number 999, and this decrements the target register:

```
A. .. ..N
```

The elementary modification unit is only capable for the 9s complement, and can neither do a digit shift nor a sign detection.

3.2. Negative Numbers

The machine is best used with 10s complements numbers, although the readout of negative numbers is more complicated, see the appendix.

3.3. Fixed-point decimal fractions

Instead of using integral numbers, fixed-point numbers with the decimal point after the highest digit might be used. i.e. from -5.000 to +4.999.

Additions and subtractions are, as is well known, not affected; just multiplications and divisions must know the position of the decimal point.

Simple examples for fixed-point calculations without multiplication have not yet been identified. If dividing a number by 2 or 10 were available, the sum of 2^{-i} or 10^{-i} would be a possible example.

3.4. Punched card programming

The sequencer could be replaced by a reader for small punched cards or paper tape instead of the matrix and the rotary switch. This device would advance the card automatically to the next row once one is executed. Single step as well as continuous mode can be selected.

Instead of repeating a sequence, the instructions could be duplicated, as any example will be repeated only a few times.

Because one of the first four bits must be set to do something useful, the card reader could support a simple loop control similar to the Z3:

If the 4 leftmost bits are not set, and the `L` bit is set, the card is moved backwards to a row where the `E` or `N` bit is set.

Rows with no bits set are skipped.

Instead of adding the cards

```
.. .. .E.
.. .. L..
```

In the following examples, just a blank line before the section to be repeated is used.

While encoding of three levels of nested loops with the `E` and `N` bits is possible, this is not useful here.

3.5. Programming examples

The examples are, as only two registers are assumed, very simple. However, using standard desktop calculators, the effort is higher.

The two registers are denoted by `a` and `b`.

3.5.1. Triangle numbers

Triangle numbers are the sum of the first natural numbers, i.e. $b_{n+1} = b_n + (n+1)$

Starting with `a=0` and `b=0`, the current argument is in `a`, the result in `b`:

```
A. .. .E.       increment a by 1
.R A. ...       add a to b (result)
```

By executing each line in turn, the second line produces the next triangle number in b with the argument in a.

To clear the registers initially, a single instruction line is used that is not repeated:

```
.R .R L..       clear a and b simultaneously

A. .. .E.       increment a by 1
.R A. ...       add a to b (result)
```

The lines to be repeated are delimited by a blank line.

3.5.2. Fibonacci numbers

Fibonacci numbers are the sum of the two previous ones:

$$x_{n+1} = x_n + x_{n-1}$$

The first two lines set the initial values, and then the last two lines are executed alternatingly, producing the result also in alternating registers:

```
.R .R L..       a=0, b=0
A. .. .E.       a=1

.R A. ...       b =+ a
A. .R ...       a =+ b
```

3.5.3. Square numbers

To enumerate squares, no multiplications are required.

Using the first binominal formula:

$$(x+1)^2 = x^2 + 2x + 1$$

The argument is in `a`, the squares are in `b`. The first two lines set the initial values:

```
.R .R L..       a=0, b=0
A. A. .E.       a=1, b=1

.R A. ...       b =+ a
.R A. ...       b =+ a
A.      . ..1   a =+ 1
.. A. ..1       b =+ 1
```

Lines 3 to 6 are to be repeated, and each time, in `b` is the square of `a`.

Sometimes it is useful to increment the argument first and decrement instead of increment:

$$(x+1)^2 = x^2 + 2(x+1) - 1$$

The example is left to the reader.

A shorter program is obtained if the difference method is used, because the differece of two adjacent square numbers is always odd and increases by 2 each time:

$$(x+1)^2 - x^2 = 2x + 1$$

The result is:

```
.R .R L..       a =: b, b =: 0
A. A. ..1       a =+ 1, b =+ 1

A. .. ..1       a =+ 1
A. .. ..1       a =+ 1
.R A. ...       b =+ a          result in b
```

3.5.4. Echternachian Hopping

The numbers are advanced twice and more, and gone back one less. The distances increase, thus *super*.

```
.R .R L..       a=0, b=0
A. A. .E.       a=1, b=1

.R A. ...       b =+ a
```

```
A. .. .EN       a =- 1
.R A. .EN       b =- a
A. .. .E.       a =+ 1
A. .. .E.       a =+ 1
```

Alternatively, it could be +1, -2, +3, -4 etc, which will enumerate all numbers from 1 on, alternating the positive and negative ones:

```
.R .R L..       a=0, b=0
A. .. .E.       a=1

.R A. ...       b =+ a          positive number
A. .. .E.       a =+ 1
.R A. .EN       b =- a          negative number
A. .. .E.       a =+ 1
```

This will run serveal hours, and would benefit from a card reader with loops.

3.5.5. Square Root

To determine a square root, the square numbers may be enumerated and the process stopped, when the next larger square is found.

For enumerating, the difference method is used, so that 1, 3, 5, 7 etc. are subtracted, until the result is zero or negative. The number of rounds is the root. The argument has to be entered manually after the second line:

```
.R .R L..       a=0, b=0
A. .. .E.       a=1         enter argument in b

.R A. .EN       b =- a
A. .. .E.       a =+ 1
A. .. .E.       a =+ 1
```

If a third register is available, it is incremented each round. As a substitute, a simple counter can count the number of times register `b` is changed.

4. Comparison to other early computers

One motivation for MERAC was to show that a useful computer can be built with minimal effort. Because the machine is so easy to understand, and because it was designed with the Analytical Engine and the ENIAC in mind, comparing it to both and others might help to understand those better.

4.1. Analytical Engine (1843)

The mathematician Charles Babbage wanted to have tables (logarithms, nautical tables etc) error-free calculated and printed. So already the earliest sketches for a computing engine showed a printing mechanism, unlike any other calculating machine at his time. Also, already the first drawings show a mechanism to normalise the rotating angles, which represented the digits, so his was a truely digital machine. (Other efforts to build a machine like his difference engine did not have this mechanism and failed to run reliable.) The whole machine was mechanical, as electromagnetic relays were not yet established.

At that time, tables were calculated in tabular form. In the first columns, the arguments were noted; then, line by line from left to right, the next colunm was calculated as an arithmetic operation using the colums before. So it is clear in the hindsight that Babbage planned a machine with a large number of variables (columns) and the ability to calculate the contents of a variable as an operation from the other variables. As the number of rows in a table is normally limited, there is even not a loop necessary, as the cards for a row can be simply duplicated. Nevertheless, a simple loop for the number of rows would be sufficient.

As numbers were stored in sign and magnitude format, the effort for correctly adding and

subtracting numbers was large. To provide hardware multiplication and division was even more effort. Thus, it is not astonishing that Babbage wanted to concentrate all operations in the mill, in particular as this allowed a large number of variables.

As it is mechanically easy to turn a disc backwards, Babbage had always assumed that the discs had to be turned back to read out a number, and erased by this way, and the number read had to be saved and re-written if it were to be retained. So he neither realized that turning forward one full rotation, or continue to turn backwards for one full rotation, would restore the value. In contrast, the ENAIAC enigneers would have doubled the effort to provide a shift register in both directions, and had a strong incentive to find a solution with turning forwards only.

An iterative programme (in the sense that the iteration count is data dependent) is not needed for many applications. Very often, function values were calculated by power series (taylor series), where the maximum number of necessary steps could be determined in advance. All functions with one variable to be tabulated in equidistant steps that are calculated by polynoms are suitable for the DE, although the preparation, i.e. the calculation of the input numbers, is not simple and still a source of errors.

The probably simplest type of problems where the analytical engine is required is the solution of linear equations, even if the solution is analyticially known. Thus it is only logical that this is a commonly used example.

The MERAC evaluator moves the digits intermittent, while the AE did a continuous movement by coupling the digit wheels to a shaft or bus bar. As the signal on the digit bus is an enable signal, instead of a magnet advancing the disc stepwise, the signal could drive a magnet that couples the digit disc to a common shaft which rotates once per 10 digits. This might have been a quicker and more reliable option, but because of the extra effort less appropriate on a proof-of-concept level.

4.2. ENIAC (1946)

The ENIAC was planned to take over manual calculations which were done in computing rooms by help of mechanical desktop calculators. The latter could add or subtract a value in a settable register to or from an accumulating register. For multiplications and divisions a counting register was provided.

Thus it is not surprising that the ENIAC had twenty registers with 10 decimal digits each, to which the value of another register could be added (or subtracted from). Because the desktop calculators could neither multiply nor divide immediately, multiplication or division were part of the basic operation of the machine. Nevertheless, some registers were equipped with additional circuitry to aid multiplications and divisions.

Digit memory was provided electronically with 10 vacuum valves (double triodes) resulting in ring counters (shift registers with exactly one bit set), i.e. the electronic analogy of a rotatable disc.

Due to the fact that a bidirectional shift register required much more effort than to turn a mechanical disc back, the inventors had observerd that during readout neither a backward move is necessary nor the contents is unrecoverable. For this purpose, the ring counter is cycled one full cycle (10 clocks), and the number of counts after the transition from 9 to 0 will be sent as contents. This is mechanically easy too, and would have reduced the size of the store of the AE to about the half.

As the shift register consisted of binary elements, it could be reset immediatly and in short time; thus, as opposed to the AE, the accumulators could be overwritten on the fly.

The ENIAC uses an additional clock pulse to add a one for the lowest digit, if a subtraction took place. The MERAC uses instead the unused carry input for the lowest digit.

Programming the ENIAC was done by routing pulses to the accumulator controls, which issued such pulses after the end of a — repeatable — operation. Routing was done similar to later electronic analog computers by wander plugs. Thus they shared the same drawback that a new programme was a considerable effort, and the expensive machine often stood still while being programmed. Here we see the clear vision of Babbage, who emphasized that such a machine shall have a programme library to be able to provide solutions easily. (As the pulsed were routed

using another bus, it would have been possible and even not more costly to set up the connections by using cards for each of the 20 accumulators.)

If and how Bernoulli's numbers could be calculated with the ENIAC by the method given by A.A.L., and with what degree of automatism, has not yet been evaluated.

Note that Hewlett-Packard produced ten years later a decimal counter (AC-4) with only four valves by internally counting binary and resetting to zero if ten was reached. The decimal display used glow lamps in a resistor decoding matrix from binary to decimal. As three double triodes are needed for 3 bits, the loss in using decimal instead of binary digits is less than expected at that time.

4.3. Harwell Decatron Computer (HDC, WHICH, 1952)

The decatron is a decimal counting valve with glow discharge segements, which was announced in 1950. It was significantly less expensive than 10 double triodes, but also significantly slower. With this device, a computer with 20 (later 90) Registers (with 8 decimal digits each) was built, that is still in operation in the TNMOC (The National Museum Of Computing).

As with the AE and the ENIAC, the digits were transmitted in parallel, but each digit as a pulse count serially. 9s complement was used, which made the accumulator more complex, but was probably easier for print.

Readout was done, as in the ENIAC, by cyclic rotation of one round of 10 pulses.

Similar to the AE, the register contents could not be replaced: *At present a store can only be cleared while transferring a number out of it.* (NPL report 1953).

The great advantage of the HDC was that instructions (5 decimal digits) were defined and could be obtained from paper tape or the calculation registers. Six characters on paper tape were needed for one instruction, so using paper tape as input slowed down the machine. It was reported that with 40 registers this feature was used heavily.

The logic uses relays, and thus the machine is rather slow, although the decatrons would allow much quicker operation. While a human computer with a mechanical desktop calclulator could be quicker for short times, but less reliable and needed breaks, so in the long run was slower. And the machine was very reliable and could work upto 10 days unattended.

Each decatron is equivalent to a ring counter with 10 neon lamps; perhaps one valve is needed to detect the transition from 9 to 0, unless very sensitive (polarised) relays were used. The major problem is the dispersal of characteristics, which is less nowadays than at that time.

4.4. Zuse Z3 (1941)

Zuse's Z3 was a binary 22-bit relay based computer with 64 number registers and used already 1941 floating-point numbers, that were commercially available with the IBM 704 since 1954.

The Z3 is a one-address machine with an accumulator and provided multiplication and division in hardware, i.e. as a single instruction. Programming was by 8 channel punched paper tape. As an instruction required just 8 bits, reading the paper tape did not slow down the machine as with the HDC.

Programme branches and loops were not available; Zuse recommended to glue together the end and begin of a paper tape to repeatedly caclculate the same formula. As there was no reader nor printer for numbers, the results had to be entered and read out at the console anyhow, as the machine was intended as a useful proof-of-concept machine.

It successor, the Z4, had conditional instructions and branches, where the programme paper tape was moved to tape mark, not by a count of instructions to be moved over.

4.5. Atanasoff-Berry-Computer (1942)

The ABC was a serial binary computer, that used a rotating drum with contacts to capacitors, i.e. a dynamic memory, that was refreshed during each revolution.

Each track had its arithmetic unit; it was thus the first vector computer. Programming was via cards.

It had been built and tested, but was not recognised as useful during WW II, and was discovered during the patent disputes on the ENIAC patents.

4.6. Pilot ACE (1950)

The (Pilot) ACE was a serial 32-bit binary computer with 1 MHz bit clock, where programme instructions were only fetched from electronic memory; so it is the most advanced and significantly different to the other computers.

In the first ideas, Alan Turing planned a three address machine like the AE. As this seemed to be to expensive, a two address machine was designed and constructed.

It has only one type of instructions: transfer of a number from a source port to a destination port. Calculation and control was done by side-effects of the destination and source ports: one destination added the number to a register, while another destination replaced the register contents, etc.

The data transfer bus is explicit part of the concept: Each instruction just opens two ports: one that sends a chain of 32 bits on the bus, and another one that receives that bit sequence.

The MERAC uses a similar concept, but also has a modification unit inserted into the bus to provide some operations for all registers, where the ACE required different ports for this.

Main memory for instuctions and data were 384 registers with 32 bits each, by using ultrasonic delay lines.

As with the ENIAC, punched cards were used as input and output media, which could be prepared and the results printed with commercially available equipment offline. However, instead of 80 colums of 12 bits, less than half of the card was used for 32 bits in 12 words, which would provide 20 numbers of about 10 decimal digits per second, and thus was ten times quicker than paper tape. This allowed to save intermediate results to punched cards and allowed to do matrix calculations.

Decimal to binary conversion, when required for input and output, could be done during the waiting time for the next card row due to the high speed of the machine.

Also, the use of subroutines (not recursive) was planned from the beginning.

5. Appendix

Negative Numbers

For negative numbers, the 9s as well as the 10s complement could be used; in the latter case by using the 9s complement and an increment. Note that in 9s complement, the number 999 has to be treated as zero, also called *minus zero*, etc.

The selection concerns input and output, here shown by the example of digit discs.

The first (most significant) place is fully used (not alternating with + and -) and has the following inscription (optional red color instead of a minus sign):

```
0  1  2  3  4  -4  -3  -2  -1  -0
```

The other discs then show:

```
0|-9  1|-8  2|-7  3|-6  4|-5  5|-4  6/-3  7|-2  8|-1  9|-0
```

This is valid also for the last digit in the 9s complement; in the 10s complement, the last digits are:

```
0|-9+  1|-9  2|-8  3|-7  4|-6  5|-5  6|-4  7|-3  8|-2  9|-1
```

This results in the display of the following negative numbers in the 10s complement the black digits in the second and the red digits in the third column:

```
-001  999   001
-011  989   011
-010  990   009+
-456  544   456
-499  501   499
-500  500   499+
```

A + in the last colum indicates that the whole number has to be incremented by 1 for readout.

This is, however, a severe obstacle for a printing mechanism.

The choice of complement also concerns multiplication and division and the modification unit, as there two variants for the number zero (`000` und `999`).

Mechanical desktop calculating machines often use a movable window to show either the black or the red digits if the number is negative. This would not be impossible, but require a significant effort.

Notes

It is reported that the ENIAC was purposly designed as decimal calculator, because the effort of 280 valve functions per register with 10 decimal digits would be less than the estimated 450 valve functions for a 30-Bit accumulator (*First Computers*, p. 130). This evidently assumes a 30-bit parallel computer, which would have been about 10 times quicker. On the other hand, a 30-bit binary bit-serial computer with a serial adder requires less than 100 valve functions, but is 3 times slower. Thus the speed loss is the same factor 3 as the gain in reduced effort. Then, building three computers with the same money would have been more useful. (The later produced counter to 10 with 8 valve functions not considered.) A serial 21-bit calculator (appr. 6 decimal digits) requires only 60 valve functions, thus half the speed will be bought by a quarter of the effort.

As the ENIAC used 20 clock times at 100kHz clock frequency, it required 200µs per addition, i.e. could perform 5000 additions per second. Alan Turing's ACE had a 1MHz bit clock and could perform (optimally) about 30´000 additions per second.

The major influence was and is the available memory technology. By using delay lines, the ACE could provide larger memory for data and instructions. (Alan Turing wrote in his specifications that instructions must be sourced with the same speed as data, to be effective; thus his decision to put the program in the same type of memory.)

Core memory provided economicial larger memory and allowed the commercial use; it started the real eara of programmable computers. Any significant advances were coupled to new memory technologies.

Literature

John Manley, Elery Buckley: ”Neon Ring Counter. Electronics, January 1950

(to be extended)