A large routine

Freek Wiedijk

Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 212, 6525 EC Nijmegen, The Netherlands

Abstract. This is a short text that I wrote for the *Liber Amicorum* for Jan Bergstra [1], presented at his retirement in December 2016. It speculates about a program that Alan Turing wrote in 1949.
The version for the *Liber* used a tiny font and very long lines to fit everything on the two pages I had available. This is a version with a more reasonable layout. It also has references, and an added appendix with the source code of the program.

It is not widely known, but already in June 1949[1] Alan Turing had the main concepts that still are the basis for the best approach to program verification known today, and for which Tony Hoare developed a logic in 1969 [3]. In his three page paper *Checking a large routine* [8,15], Turing describes how to verify the correctness of a program that calculates the factorial function by repeated additions. He both shows how to use *invariants* to establish the correctness of this program, as well as how to use a *variant* to establish termination.

In the paper the program is only presented as a flow chart. A modern C rendering is:

```
int fac (int n)
{
  int s, r, u, v;
  for (u = r = 1; v = u, r < n; r++)
    for (s = 1; u += v, s++ < r; )
      ;
  return u;
}
```

Turing's paper seems not to have been properly appreciated at the time. At the EDSAC conference where Turing presented the paper, Douglas Hartree criticized that the correctness proof sketched by Turing should not be called inductive. From a modern point of view this is clearly absurd.

Marc Schoolderman first told me about this paper (and in his master's thesis [12] used the modern Why3 tool [2] of Jean-Christophe Filliâtre to make Turing's paper fully precise; he also wrote the above C version of Turing's program). He

[1] Two years before Jan Bergstra was born.

then challenged me to speculate what specific computer Turing had in mind in the paper, and what the actual program for that computer might have been.

My answer is that the 'EPICAC'[2] of this paper probably was the Manchester Mark 1. Given that Turing at that time was Deputy Director of the Computing Laboratory in Manchester, this is not very surprising. Further arguments for this are that the paper talks about 40-bit words, and that it uses storage locations 27–31 called 'lines' for the variables.

Of the computers at that time only the ones in development in Manchester and at the IAS in Princeton had 40-bit words. Turing had written his PhD thesis in Princeton, and undoubtedly knew about the IAS machine, but he was much more closely involved with the Manchester machines. Also, in 1949 the Manchester machines already were operational, while the IAS machine only became operational in 1951. The Manchester machines had three generations: the **Manchester SSEM** (Small-Scale Experimental Machine) 'baby' (32-bit, operational June 1948), the **Manchester Mark 1** (40-bit, operational April 1949) and the commercial **Ferranti Mark 1** (40-bit, operational February 1951). Turing wrote in 1950 a very nice and surprisingly modern manual [9,16,17] for this last machine (which unlike Ferranti he calls 'Mark II', as in his opinion 'Mark I' already had been used). This manual also describes the earlier Manchester Mark 1 in an appendix, where it is called the 'Pilot machine'.

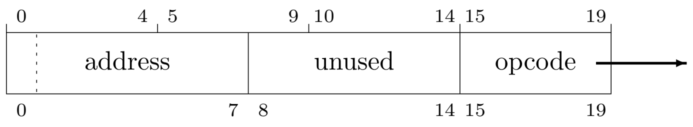
The question then is which of the two 40-bit Manchester computers Turing had in mind in his 'large routine' paper. These machines are very similar, but in 1949 the newer machine was not yet operational. There is another argument for the earlier machine: in both machines the addresses are *twice* the 'storage locations' that Turing uses in his paper (there is no space to fit a full word between two consecutive addresses), which means that Turing is *not* talking

[2] EPICAC is a fictional computer from the short story *'EPICAC'* [18] by Kurt Vonnegut, which was published November 1950, only one year after Turing's paper (and filmed for television in 1974 [10], 1992 [20] and 2008 [14]). The EPICAC computer also occurs in Vonnegut's 1952 debut novel *Player Piano* [19], which is about the effects of robotisation on society. The name EPICAC is a play on ENIAC (the first general-purpose electronic computer, operational February 1946; computers of that era had names in this style: EDSAC, BINAC, CSIRAC, SEAC, SWAC, EDVAC, UNIVAC, ILLIAC, MANIAC – the suffix 'AC' generally stands for 'Automatic Computer') and ipecac (a syrup with roots of the *ipecacuanha* plant, used to induce vomiting in case of poisoning). In the story, a man and a woman get romantically involved while working late at night with the EPICAC computer, and at the end of the story the computer provides the man with a lifetime support of love poems. This story matches real life in some respects: Conway Berners-Lee and Mary Lee Woods worked late at night with the Ferranti Mark 1 computer in 1953 and married in 1954 (their son invented the World Wide Web in 1991), while for this same computer in 1952 a program was developed by Christopher Strachey that produces love letters [13]. (Strachey also wrote the first program for the game of draughts [7], and the first program for computer music; in 1965 he became the first director of the Programming Research Group in Oxford, where he developed denotational semantics in collaboration with Dana Scott.) The 'love letters' program has been revived as an art project by David Link in 2009 [4,5,6].

about addresses. The storage of the Ferranti machine on a Williams tube consists of two columns next to each other, where 'words' are two successive half-lines in a single column. But the older machine has the 40-bit words spanning the whole width of the tube, which means that in that case Turing just numbered the five bottom-most of those 'lines' in his paper. For this reason it seems more natural to consider a program for the earlier Manchester Mark 1.

The architecture of the Manchester Mark 1 has 40-bit words, but the accumulator has 80 bits, and the memory is addressed in 20-bit 'bytes' (when fetching a full word from memory the lowest address bit is ignored, hence addresses are always aligned). The 'nibbles' of the machine are 5-bit, and are written using the ITA2 variant from 1924 of the Baudot punched tape code patented in 1874 (where additional symbols `/`, `@`, `:`, $\frac{1}{4}$, `"` and `£` represent the 'non-printing characters'). Turing does not use symbolic syntax for machine code and just writes each 20-bit instruction as four ITA2 characters. See for nice examples of programs written in this style the listings linked from [11].

The Manchester Mark 1 instruction set is very simple. An instruction has an opcode – of the 32 opcodes, our reconstructed program only uses seven – and an address field. The instruction layout (in little endian bit order) is:

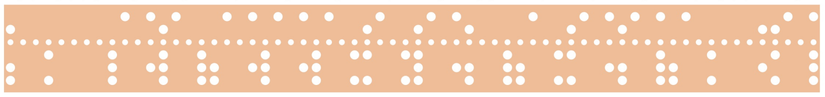


`H`	`00101`	$A \leftarrow [a]$
`M`	`00111`	$A \leftarrow A + [a]$
`X`	`10111`	$A \leftarrow A - [a]$
`D`	`10010`	$[a] \leftarrow A$
`R`	`01010`	skip next instruction if $A \geq 0$
`L`	`01001`	go to instruction at address $a + 2$
`N`	`00110`	halt

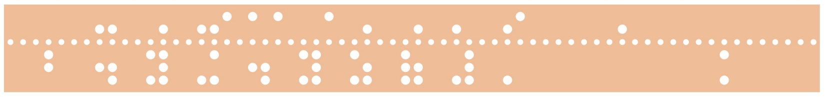
In this, A is the accumulator, a is the address, and $[a]$ the 40-bit word at that address in memory. After an instruction is executed, the program counter (where the lower bit is *not* ignored) is increased by *two*, so execution will proceed on the same side on the tube.

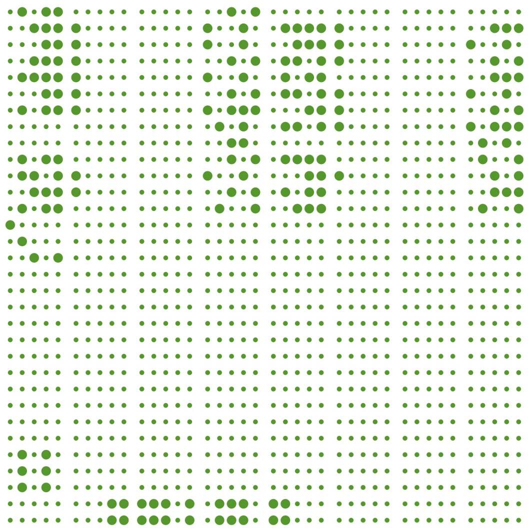
Here then, is a reconstruction of Turing's 'large routine' for the Manchester Mark 1, both the source code in paper tape format, and the resulting state of the machine as seen on a Williams tube:

```
G//H////ME/DVE/MOE/DME/DME/HPE/HVE/DG//MOE/HPE/DGE/XOE/H///RPE/X
```



```
///N///RG//HV//LPE/DOE/HME/HG//MG//LM//LE///////@///////H///////
```





The program is in the top half of the tube, in two columns, and the variables are at the bottom. The input $n = 10 = 1010_2$ is in line 29, while the output $u = 10! = 1101110101111100000000_2$ is in line 30.

References

1. Inge Bethke, Bert Bredeweg, and Alban Ponse, editors. *Liber Amicorum voor / for Jan A. Bergstra.* Informatics Institute, University of Amsterdam, 2016.
2. François Bobot, Jean-Christophe Filliâtre, Claude Marché, Guillaume Melquiond, and Andrei Paskevich. *The Why3 platform*, 2016. `http://why3.lri.fr/`.
3. Cliff B. Jones. Turing and Software Verification. Technical Report CS-TR-1441, Computing Science, Newcastle University, 2014.
4. David Link. There Must Be an Angel. On the Beginnings of the Arithmetics of Rays. In Siegfried Zielinski and David Link, editors, *Variantology 2: On Deep Time Relations of Arts, Sciences and Technologies.* Walter König, Cologne, 2006.
5. David Link. LoveLetters_1.0, 2009. `http://www.alpha60.de/art/love_letters/`.
6. David Link. *Das Herz der Maschine.* dOCUMENTA (13): 100 Notizen - 100 Gedanken, No. 037. Hatje Cantz Verlag, Berlin, 2011.
7. David Link. Programming ENTER: Christopher Strachey's Draughts Program. *Resurrection*, 2012.
8. F. L. Morris and C. B. Jones. An Early Program Proof by Alan Turing. *IEEE Ann. Hist. Comput.*, 6(2):139–143, 1984.
9. Brian Napper. Computer 50: The University of Manchester Celebrates the Birth of the Modern Computer, 2010. Website. `http://www.computer50.org/`.
10. Liam O'Brien (writer) and John Badham (director). Rex Harrison Presents Stories of Love: segment 'Epicac', 1974.

11. Huw Owen Pritchard and Frank Summer. Ferranti Mark I computer programmes, 1951–1953. Website. `http://curation.cs.manchester.ac.uk/computer50/www.computer50.org/mark1/Huw-Pritchard/FERRANTI/ferranti.html`.
12. Marc Schoolderman. Verification of Goroutines using Why3. Master's thesis, Institute for Computing and Information Sciences, Radboud University Nijmegen, 2016.
13. Christopher Strachey. The 'Thinking' Machine. *Encounter*, pages 25–31, October 1954.
14. Will Tully (writer, director). Epicac, 2008.
15. Alan Turing. Checking a large routine. In *Report of a Conference on High Speed Automatic Calculating Machines*, pages 67–69. University Mathematical Laboratory, Cambridge, 1949. `http://www.turingarchive.org/browse.php/b/8`.
16. Alan Turing. *Programmers' Handbook for the Manchester Electronic Computer Mark II*, 1950. `http://www.turingarchive.org/browse.php/B/32`, transcribed as [17].
17. Alan Turing and Robert S. Thau (transcriber). *Alan Turing's Manual for the Ferranti Mk. I*, 2000. `http://www.panix.com/~rst/turing.pdf`.
18. Kurt Vonnegut. 'EPICAC'. In *Welcome to the Monkey House*. Dial Press, New York, 1950.
19. Kurt Vonnegut. *Player Piano*. Charles Scribner's Sons, New York, 1952.
20. Kurt Vonnegut (writer). 'Monkey House', Season 2, Episode 1: Epicac, 1992.

Source code of the routine

For this paper, I wrote an emulator for a relevant small fragment of the Manchester Mark 1. The source code for this emulator can be found on the web at:

`http://www.cs.ru.nl/~freek/pilot/pilot.c`

Marc Schoolderman extended this with the feature that if it is compiled with the symbol `VT100` defined, then the emulator will insert terminal escape codes to make the output look like a Williams tube.

The input for this emulator corresponding to Turing's program as reconstructed in this paper, is:

```
//   G/ /H  // //       ;     A' = [1]                           E/
@/   ME /D  VE /M       ;     [u]' = A         F: A' = A + [v]     A/
:/   OE /D  ME /D       ;  B: [r]' = A            [u]' = A'        S/
I/   ME /H  PE /H       ;     A' = [u]            A' = [s]         U/
8/   VE /D  G/ /M       ;     [v]' = A            A' = A + [1]     D/
R/   OE /H  PE /D       ;     A' = [r]            [s]' = A         J/
N/   GE /X  OE /H       ;     A' = A - [n]        A' = [r]         F/
C/   // /R  PE /X       ;     skip if A < 0       A' = A - [s]     K/
T/   // /N  // /R       ;     halt                skip if A < 0    Z/
L/   G/ /H  V/ /L       ;     A' = [1]            C' = [E]         W/
H/   PE /D  OE /H       ;     [s]' = A            A' = [r]         Y/
P/   ME /H  G/ /M       ;  E: A' = [u]            A' = A + [1]     Q/
O/   G/ /L  M/ /L       ;     C' = [F]            C' = [B]         B/

G/   E/ //  // //       ;    1 = F - 2
M/   @/ //  // //       ;    2 = B - 2
V/   H/ //  // //       ;   20 = E - 2

PE   // //  // //       ;   s
OE   // //  // //       ;   r
GE   R/ //  // //       ;   n = 10
ME   // //  // //       ;   u
VE   // //  // //       ;   v
```

This input also can be found on the web at:

`http://www.cs.ru.nl/~freek/pilot/fac.pi`

Finally the two pictures from this paper (processed versions of the input and output described here) are on the web at:

`http://www.cs.ru.nl/~freek/pilot/tape.pdf`
`http://www.cs.ru.nl/~freek/pilot/tube.pdf`