A GENUINE BOOTSTRAP

We began our course with an account of how to start up a computer from scratch. Now, at the end, it is very satisfying to find that the wheel has come full circle, and we are once again discussing the same topic. Perhaps the ideal way to conclude would be with an example of instructions for stopping a computer, but we haven't found any interesting examples. Instead, here's another example[MAN5] of a bootstrap process, taken from an even earlier system than the IBM 1620 with which we started.

The old Deuce computer was brought to mind by David Leigh's reminiscences (March l986 issue), and in my own case it was already there as I had just agreed to send my Deuce manual to the Computer Museum in Boston, Mass. A feature I have found particularly intriguing is the means provided for reading-in a user program and starting it running. It was arranged that this was done from a totally-cleared delay-line memory. The only thing that could be termed an 'operating system' was an initiating key on the panel, with the effect of clearing the delay-line storage and starting a card reader. The means of achieving operation on this basis is so ingenious it deserves to be recorded in some accessible place.

The main storage was in delay lines in which 32 words of 32 bits each followed each other round in a 'major cycle' of about a millisecond. There were also a number of shorter delay lines, including five temporary stores holding one word each, one of them, called TS COUNT, holding the current instruction word. The program was held in the twelve 32-word long delay lines.

Two fields of the 32-bit instruction word indicated a source and a destination, and the effect of the instruction was to transfer one or more words between the two. Each of the delay lines (except TS COUNT) could be a source or a destination, and there were also functional sources and destinations which introduced the main computing operations. For example, single-length addition was achieved by sending one of the operands to destination 25, where it was added to the existing contents of the temporary store TS13. For delay lines holding more than one word, the word for which the instruction was effective (or the first word of a block on which it acted) depended on the time at which it was obeyed; the instruction word contained in one field a wait number W, and the number of minor cycles between the selection of the instruction word and commencement of its execution was W+2. Also there was a timing number T and the number of minor cycles till the next instruction was selected was T+2.

Another field of the instruction word indicated the delay line from which the next instruction would be taken, and yet another indicated whether the transfer would be of a single word, or a pair of words, or a block beginning at the minor cycle W+2 ahead and ending at that at T+2. Another one-bit field indicated whether the instruction would be obeyed as soon as its appropriate minor cycle came round or whether it would act as a 'stopper' and wait for operation of the single-shot key, or alternatively for a ready signal from the card reader or card punch if one of these was active.

Standard punch cards were used for input, with each of the 12 lines representing a 32-bit word in its binary form (so only 32 of the 80 columns were utilised). A triad of cards (36 rows) held the contents of a long delay line, with the first four rows providing a header. The initiating key cleared the location (termed TS COUNT) holding the current instruction, except that the W and T values were not necessarily cleared. This meant that the contents of TS COUNT corresponded to a transfer from source 0 to destination 0, with uncertain W and T values, and the instruction constituted a stopper since the relevant one-bit field was blank.

There was no delay line numbered zero, and source zero referred either to the card reader or to delay line 8, the choice depending in a rather complicated way on the values of W and T. Destination zero had the effect that anything sent to it became contents of TS COUNT. The instruction initially in TS COUNT, therefore, was one that would read from the first row of the first card, or from one of the words in the (initially cleared) DL8, after receiving the ready signal from the card reader. The first row was in fact left blank, so either way the effect was to place the same instruction again in TS COUNT, except that this time the W and T values were safely set to zeros. The effect of obeying this as an instruction was to read into TS COUNT the second line in the heading.

The four-line heading is represented as follows, in terms of the conventions for writing down instruction words. This is for the case where the subsequent 32 lines are to be read into delay line 2, with DLI still blank (because it was always the last to be filled), and the triad of cards is followed by another similar triad referring to another delay line:

```
blank
2,    0-2,  1,    26,   25    X
2,    0-2,        30,   31    X
1,    0-2,        30,   31    X
```

The first field in the instruction word shows the number of the delay line from which the following instruction will be taken. The next two, conventionally separated by a dash, indicate the source and destination respectively. The next number, usually omitted if it is zero, is a modifier indicating the duration of the transfer, being zero for a single-word transfer and one for a transfer lasting from the minor cycle W+2 ahead to that T+2 ahead. The next two numbers are the values of W and T already mentioned, and the final X appears if the last field is blank to make the instruction a stopper.

As already mentioned, obeying the blank line as a 0-0 transfer reads into TS COUNT the second line of the header. When this is obeyed in turn it produces a transfer from the card reader to delay line 2, starting in the minor cycle which is 26+2 ahead, and ending with that 25+2 ahead (but in fact 25+2+32 ahead, since W>T). The effect of this is to fill DL2 with copies of the third line of the header, and to take one of these as the next instruction. Obeying this copies the fourth line of the header into a location in DL2, which is thereby defined as minor cycle 31 if this is the first delay line to be filled. The next instruction is taken from the position in DL2 which is 31+2 minor cycles ahead, or effectively one ahead and hence cycle zero, and its effect is to overwrite itself with a word coming from the card reader and to select the next instruction in sequence in DL2. Finally the instruction in minor cycle 31 is reached and after overwriting itself it causes the next instruction to come from DLI. The effect has been to read the 32 rows following the header into the locations numbered 0 to 31 in DL2.

If DLI is still clear, the selection of the next instruction from it resets TS COUNT to the blank state, ready to begin again on another triad of cards filling another delay line. If DLI is filled, running of the user program begins with the instruction in its minor cycle zero. I have no idea whether this arrangement has any possible relevance to modern practice, but it seems sufficiently ingenious to be worth adding to one's mental kit of parts!

REFERENCE.

MAN5 : A.M. Andrew : "The smallest operating system", *Computer Bulletin Series 3,* **2#4**, 40 (December 1986).